Estratégia Concursos

Aula 01

Banco do Brasil (Escriturário - Agente de Tecnologia) Desenvolvimento de Software - 2023 (Pós-Edital)

Autor:
Raphael Henrique Lacerda, Paolla Ramos e Silva

02 de Janeiro de 2023

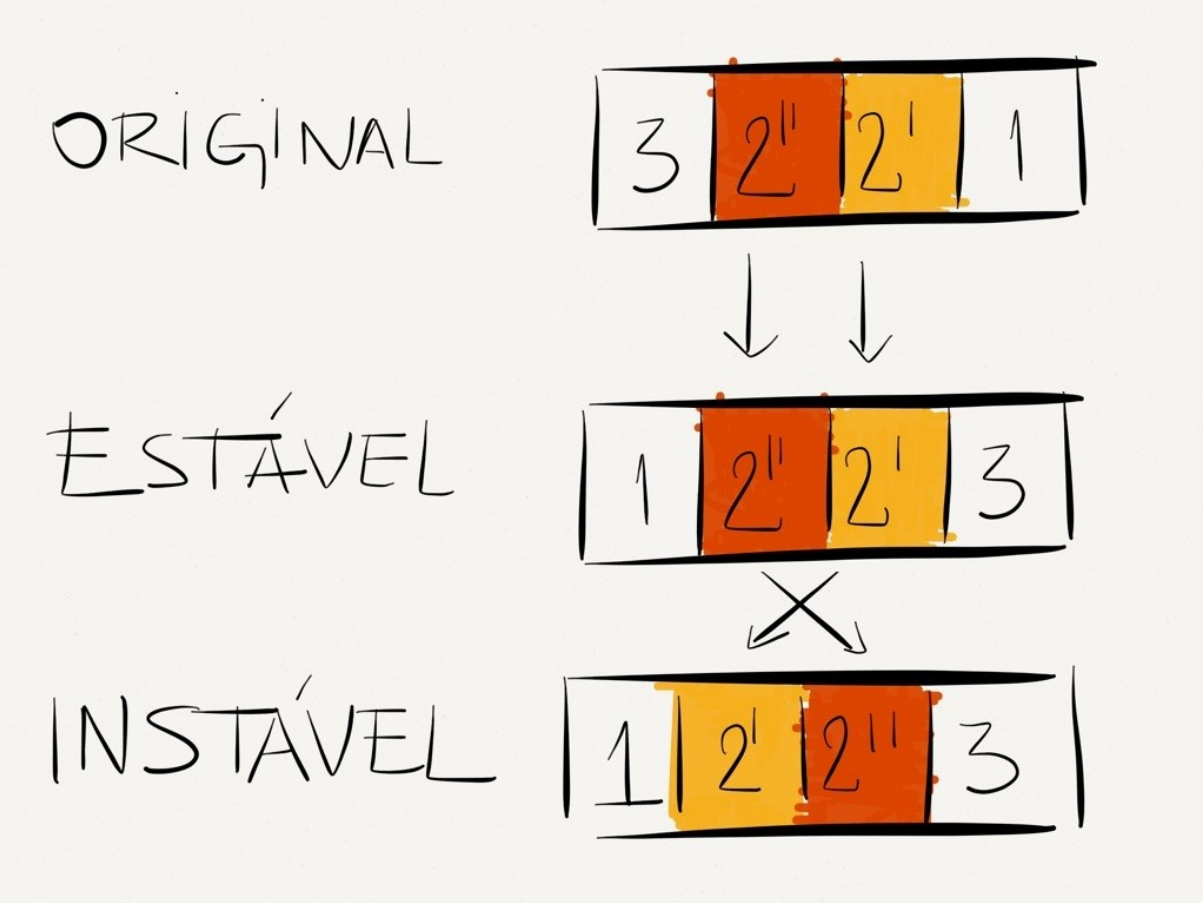
Índice

Índice

MÉTODOS DE ORDENAÇÃO

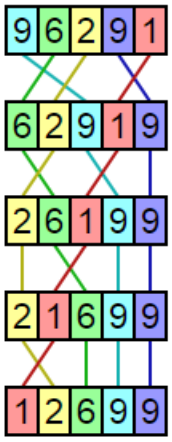
Métodos de Ordenação são algoritmos que têm o objetivo principal de posicionar os elementos de uma estrutura de dados em uma determinada ordem. Para que, professor? Ora, isso possibilita o acesso mais rápido e eficiente aos dados. Existem dezenas de métodos, todavia nessa aula veremos apenas os mais importantes: BubbleSort, QuickSort, InsertionSort, SelectionSort, MergeSort, ShellSort e HeapSort.

Antes de iniciar, vamos falar sobre o conceito de Estabilidade! Um método estável é aquele em que os itens com chaves iguais mantêm-se com a posição inalterada durante o processo de ordenação, i.e., preserva-se a ordem relativa dos itens com chaves duplicadas ou iguais. Métodos Estáveis: Bubble, Insertion e Merge; Métodos Instáveis: Selection, Quick, Heap e Shell. Vejamos um exemplo:



Na imagem acima, foi colocado um sinal de aspas simples e duplas apenas para diferenciá-los, mas trata-se do mesmo número. Um algoritmo estável ordena todo o restante e não perde tempo trocando as posições de elementos que possuam chaves idênticas. Já um algoritmo instável ordena todos os elementos, inclusive aqueles que possuem chaves idênticas (sob algum outro critério).

BubbleSort (Troca)



Esse algoritmo é o primeiro método de ordenação aprendido na faculdade, porque ele é bastante simples e intuitivo. Nesse método, os elementos da lista são movidos para as posições adequadas de forma contínua. Se um elemento está inicialmente em uma posição i e, para que a lista fique ordenada, ele deve ocupar a posição j, então ele terá que passar por todas as posições entre i e j.

Em cada iteração do método, percorremos a lista a partir de seu início comparando cada elemento com seu sucessor, trocando-se de posição se houver necessidade. É possível mostrar que, se a lista tiver n elementos, após no máximo (n-1) iterações, a lista estará em ordem (crescente ou decrescente). Observem abaixo o código para a Ordenação em Bolha:

```
ALGORITMO BOLHA
  ENTRADA: UM VETOR L COM N POSIÇÕES
  SAÍDA: O VETOR L EM ORDEM CRESCENTE

  PARA i = 1 até n - 1
    PARA j = 0 até n - 1 - i
      SE  L[j] >  L[j+1]
        AUX = L[j]          // SWAP
        L[j] = L[j+1]
        L[j+1] = AUX
FIM {BOLHA}
```

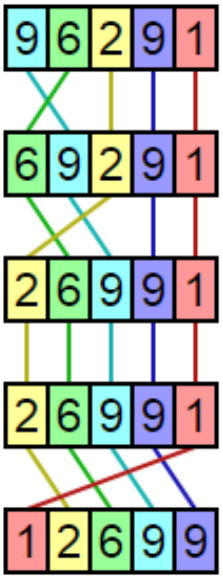
Melhor Caso	Caso Médio	Pior Caso
O(n)	$O(n^2)$	$O(n^2)$

InsertionSort (Inserção)

Esse algoritmo, também conhecido como Inserção Direta, é bastante simples e apresenta um desempenho significativamente melhor que o BubbleSort, em termos absolutos. Além disso, ele é extremamente eficiente para listas que já estejam substancialmente ordenadas e listas com pequeno número de elementos. Observem abaixo o código para a Ordenação de Inserção:

```
ALGORITMO INSERÇÃO
  ENTRADA: UM VETOR L COM N POSIÇÕES
  SAÍDA: O VETOR L EM ORDEM CRESCENTE

  PARA i = 1 até n - 1
    PIVO = L[i]
    j = i - 1
    ENQUANTO j ≥ 0 e L[j] > PIVO
      L[j+1] = L[j]
      j = j - 1
    L[j+1] = PIVO
FIM {INSERÇÃO}
```

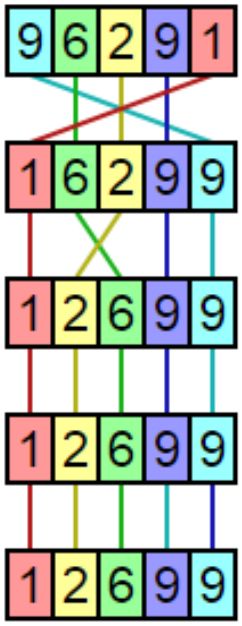


Nesse método, consideramos que a lista está dividida em parte esquerda, já ordenada, e parte direita, em possível desordem. Inicialmente, a parte esquerda contém apenas o primeiro elemento da lista. Cada iteração consiste em inserir o primeiro elemento da parte direta (pivô) na posição adequada da parte esquerda, de modo que a parte esquerda continue ordenada.

É fácil perceber que se a lista possui n elementos, após (n-1) inserções, ela estará ordenada. Para inserir o pivô, percorremos a parte esquerda, da direita para a esquerda, deslocando os elementos estritamente maiores que o pivô uma posição para direita. O pivô deve ser colocado imediatamente à esquerda do último elemento movido.

Melhor Caso	Caso Médio	Pior Caso
O(n)	$O(n^2)$	$O(n^2)$

SelectionSort (Seleção)



Esse algoritmo consiste em selecionar o menor1 elemento de um vetor e trocá-lo (swap) pelo item que estiver na primeira posição, i.e., inseri-lo no início do vetor. Essas duas operações são repetidas com os itens restantes até o último elemento. Tem como ponto forte o fato de que realiza poucas operações de swap. Seu desempenho costuma ser superior ao BubbleSort e inferior ao InsertionSort.

Assim como no InsertionSort, considera-se que a lista está dividida em parte esquerda, já ordenada, e parte direita, em possível desordem. Ademais, os elementos da parte esquerda são todos menores ou iguais aos elementos da parte direita. Cada iteração consiste em selecionar o menor elemento da parte direita (pivô) e trocá-lo com o primeiro elemento da parte direita.

Assim, a parte esquerda aumenta, visto que passa a incluir o pivô, e a parte direita diminui. Note que o pivô é maior que todos os demais elementos da parte esquerda, portanto a parte esquerda continua ordenada. Ademais, o pivô era o menor elemento da direita, logo todos os elementos da esquerda continuam sendo menores ou iguais aos elementos da direita. Inicialmente, a parte esquerda é vazia.

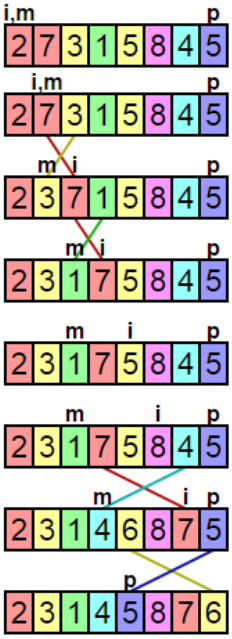
```
ALGORITMO SELEÇÃO
  ENTRADA: UM VETOR L COM N POSIÇÕES
  SAÍDA: O VETOR L EM ORDEM CRESCENTE

  PARA i = 0 ate n - 2
    MIN = i
    PARA j = i + 1 até n - 1
      SE L[j] < L[MIN]
        MIN= j
    TROCA(L[i], L[MIN])
FIM {SELEÇÃO}
```

Melhor Caso	Caso Médio	Pior Caso
$O(n^2)$	$O(n^2)$	$O(n^2)$

QuickSort (Troca)

[1] A definição formal afirma que é o maior valor; a maioria das implementações utiliza o menor valor. As questões de prova cobram algumas vezes o maior, outras vezes o menor.



Esse algoritmo divide um conjunto de itens em conjuntos menores, que são ordenados de forma independente, e depois os resultados são combinados para produzir a solução de ordenação do conjunto maior. Trata-se, portanto, de um algoritmo do tipo Divisão-e-Conquista, i.e., repartindo os dados em subgrupos, dependendo de um elemento chamado pivô.

Talvez seja o método de ordenação mais utilizado! Isso ocorre porque quase sempre ele é significativamente mais rápido do que todos os demais métodos de ordenação baseados em comparação. Ademais, suas características fazem com que ele, assim como o MergeSort, possa ser facilmente paralelizado. Ele também pode ser adaptado para realizar ordenação externa (QuickSort Externo).

Neste método, a lista é dividida em parte esquerda e parte direita, sendo que os elementos da parte esquerda são todos menores que os elementos da parte direita. Essa fase do processo é chamada de partição. Em seguida, as duas partes são ordenadas recursivamente (usando o próprio QuickSort). A lista está, portanto, ordenada corretamente!

Uma estratégia para fazer a partição é escolher um valor como pivô e então colocar na parte esquerda os elementos menores ou iguais ao pivô e na parte direita os elementos maiores que o pivô – galera, a escolha do pivô é crítica! Em geral, utiliza-se como pivô o primeiro elemento da lista, a despeito de existirem maneiras de escolher um "melhor" pivô.

Esse algoritmo é um dos métodos mais rápidos de ordenação, apesar de às vezes partições desequilibradas poderem conduzir a uma ordenação lenta. A eficácia do método depende da escolha do pivô mais adequado ao conjunto de dados que se deseja ordenar. Alguns, por exemplo, utilizam a mediana de três elementos para otimizar o algoritmo.

Alguns autores consideram a divisão em três subconjuntos, sendo o terceiro contendo valores iguais ao pivô. O Melhor Caso ocorre quando o conjunto é dividido em subconjuntos de mesmo

tamanho; o Pior Caso ocorre quando o pivô corresponde a um dos extremos (menor ou maior valor). Alguns o consideram um algoritmo frágil e não-estável, com baixa tolerância a erros.

```
PROCEDIMENTO PARTIÇÃO
  ENTRADA: UM VETOR L E AS POSIÇÕES INICIO E FIM
  SAÍDA:j, tal que L[INICIO]..L[j-1] ≤ L[j] e
        L[j+1]..L[FIM] > L[j]
     // j é a posição onde será colocado o pivô, como
     // ilustrado na figura abaixo
       PIVO = L[INICIO], i = INICIO + 1, j = FIM
       ENQUANTO i ≤ j
         ENQUANTO i ≤ j e L[i] ≤ PIVO
           i = i + 1
         ENQUANTO L[j] > PIVO
           j = j - 1
         SE i ≤ j
           TROCA(L[i],L[j])
           i = i + 1, j = j - 1
       TROCA(L[INICIO], L[j])
       DEVOLVA j
     FIM {PARTIÇÃO}
```

Melhor Caso	Caso Médio	Pior Caso
O(n log n)	O(n log n)	$O(n^2)$

ShellSort (Inserção)

Esse algoritmo é uma extensão ou refinamento do algoritmo do InsertionSort, contornando o problema que ocorre quando o menor item de um vetor está na posição mais à direita. Ademais, difere desse último pelo fato de, em vez de considerar o vetor a ser ordenado como um único segmento, ele considera vários segmentos e aplica o InsertionSort em cada um deles.

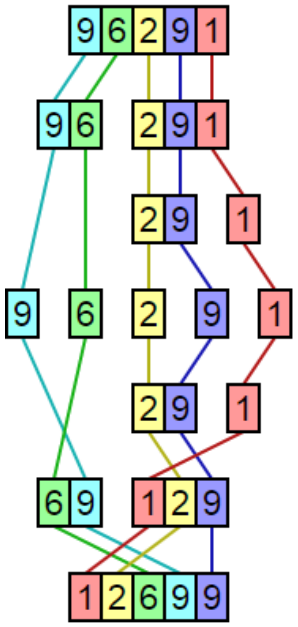
É o algoritmo mais eficiente dentre os de ordem quadrática. Nesse método, as comparações e as trocas são feitas conforme determinada distância (gap) entre dois elementos, de modo que, se gap = 6, há comparação entre o 1° e 7° elementos ou entre o 2° e 8° elementos e assim sucessivamente, repetindo até que as últimas comparações e trocas tenham sido efetuadas e o gap tenha chegado a 1.

```
ALGORITMO SHELLSORT
  ENTRADA: UM VETOR L COM N POSIÇÕES
  SAÍDA: O VETOR L EM ORDEM CRESCENTE

  H = 1
  ENQUANTO H < n FAÇA H = 3 * H + 1
  FAÇA
    H = H / 3              // divisão inteira
    PARA i = H até n - 1 // Inserção adaptado para h-listas
      PIVO = L[i]
      j = i - H
      ENQUANTO j ≥ 0 e L[j] > PIVO
        L[j + H] = L[j]
        j= j - H
      L[j + H] = PIVO
  ENQUANTO H > 1
     FIM {SHELLSORT}
```

Melhor Caso	Caso Médio	Pior Caso
O(n log n)	Depende do gap	$O(n^2)$

MerqeSort (Intercalação)



Esse algoritmo é baseado na estratégia de resolução de problemas conhecida como divisão-e-conquista. Essa técnica consiste basicamente em decompor a instância a ser resolvida em instâncias menores do mesmo tipo de problema, resolver tais instâncias (em geral, recursivamente) e por fim utilizar as soluções parciais para obter uma solução da instância original.

Naturalmente, nem todo problema pode ser resolvido através de divisão e conquista. Para que seja viável aplicar essa técnica a um problema, ele deve possuir duas propriedades estruturais. O problema deve ser decomponível, i.e., deve ser possível decompor qualquer instância não trivial do problema em instâncias menores do mesmo tipo de problema.

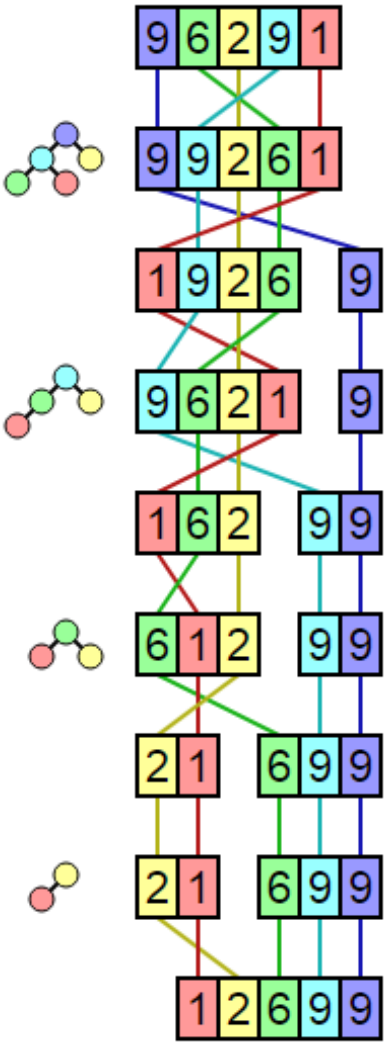
Além disso, deve ser sempre possível utilizar as soluções obtidas com a resolução das instâncias menores para chegar a uma solução da instância original. No MergeSort, divide-se a lista em duas metades. Essas metades são ordenadas recursivamente (usando o próprio MergeSort) e depois são intercaladas. Abaixo segue uma possível solução:

```
ALGORITMO MERGESORT
  ENTRADA: UM VETOR L E AS POSIÇÕES INICIO E FIM
  SAÍDA: O VETOR L EM ORDEM CRESCENTE DA POSIÇÃO INICIO ATÉ
         A POSIÇÃO FIM

  SE inicio < fim
    meio = (inicio + fim)/2          // divisão inteira
    SE inicio < meio
      MERGESORT(L, inicio, meio)
    SE meio + 1 < fim
      MERGESORT(L, meio + 1, fim)
    MERGE(L, inicio, meio, fim)
FIM {MERGESORT}
```

Melhor Caso	Caso Médio	Pior Caso
O(n log n)	O(n log n)	O(n log n)

HeapSort (Seleção)



Esse algoritmo utiliza uma heap, para ordenar os elementos estrutura de dados chamada à medida que os insere na estrutura.

Assim, ao final das inserções, os elementos podem ser sucessivamente removidos da raiz da heap, na ordem desejada.

Essa estrutura pode ser representada como uma árvore ou como um vetor. Entenderam? Inicialmente, insere-se os elementos da lista em um heap.

Em seguida, fazemos sucessivas remoções do menor elemento do heap, colocando os elementos removidos do heap de volta na lista – a lista estará então em ordem crescente.

O heapsort é um algoritmo de ordenação em que a sua estrutura auxiliar de armazenamento fora do arranjo de entrada é constante durante toda a sua execução.

```
ALGORITMO HEAP SORT
  ENTRADA: UM VETOR L COM N POSIÇÕES
  SAÍDA: O VETOR L EM ORDEM CRESCENTE

inicialize um HBC H com n posições
PARA i = 0 até n - 1
  INSERE_HBC(H, L[i])
PARA i = 0 até n - 1
  L[i] = REMOVE_MENOR(H)
FIM {HEAP SORT}
```

Melhor Caso	Caso Médio	Pior Caso
O(n log n)	O(n log n)	O(n log n)

Resumão de Complexidades

ALGORITMO	MELHOR CASO	CASO MÉDIO	PIOR CASO
BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$
QuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do gap	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)
HeapSort	O(n log n)	O(n log n)	O(n log n)

COMPLEXIDADE DE ALGORITMOS

Galera, por que estudamos a complexidade de algoritmos? Para determinar o custo computacional (tempo, espaço, etc) para execução de algoritmos. Em outras palavras, ela classifica problemas computacionais de acordo com sua dificuldade inerente. É importante entender isso para posteriormente estudarmos a complexidade de métodos de ordenação e métodos de pesquisa.

Nosso estudo aqui será bastante superficial por duas razões: primeiro, concursos cobram pouco e, quando cobram, querem saber as complexidades dos métodos de ordenação mais conhecidos; segundo, essa é uma disciplina absurdamente complexa que envolve Análise Assintótica, Cálculo Diferencial, Análise Polinomial (Linear, Exponencial, etc). Logo, vamos nos ater ao que cai em concurso público!

Só uma pausa: passei dias sem dormir na minha graduação por conta dessa disciplina! Na UnB, ela era ministrada pelo Instituto de Matemática e era considerada a disciplina mais difícil do curso :-(. Continuando: lá em cima eu falei sobre custo computacional! Ora, para que eu escolha um algoritmo, eu preciso definir algum parâmetro.

Podemos começar com Tempo! Um algoritmo que realiza uma tarefa em 20 minutos é melhor do que um algoritmo que realiza uma tarefa em 20 dias? Não é uma boa estratégia, porque depende do computador que eu estou utilizando (e todo o hardware correspondente), depende das otimizações realizadas pelo compilador, entre outras variáveis.

Vamos analisar, então, Espaço! Um algoritmo que utiliza 20Mb de RAM é melhor do que um algoritmo que utiliza 20Gb? Seguem os mesmos argumentos utilizados para o Tempo, ou seja, não é uma boa opção! E agora, o que faremos? Galera, eu tenho uma sugestão: investigar a quantidade de vezes que operações são executadas na execução do algoritmo!

Essa estratégia independe do computador (e hardware associado), do compilador, da linguagem de programação, das condições de implementação, entre outros fatores – ela depende apenas da qualidade inerente do algoritmo[1] implementado. Utilizam-se algumas simplificações matemáticas para se ter uma ideia do comportamento do algoritmo. Prosseguindo...

Dada uma entrada de dados de tamanho N, podemos calcular o custo computacional de um algoritmo em seu pior caso, médio caso e melhor caso! Como assim, professor? Para entender isso, vamos utilizar a metáfora de um jogo de baralho! Imaginem que eu estou jogando contra vocês. Vocês embaralham e me entregam 5 cartas, eu embaralho novamente e lhes entrego 5 cartas.

Quem joga baralho sabe que uma boa alternativa para grande parte dos jogos é ordenar as cartas em ordem crescente de modo a encontrar mais facilmente a melhor carta para jogar. Agora observem... vocês receberam as seguintes cartas (nessa ordem): 4, 5, 6, 7, 8. Já eu recebi as

[1] Pessoal, é claro que nossa visão sobre a complexidade dos algoritmos é teórica. Na prática, depende de diversos outros fatores, mas nosso foco é na visão analítica e, não, empírica.

seguintes cartas (também nessa ordem): 8, 7, 6, 5, 4 – nós queremos analisar a complexidade de ordenação dessas cartas.

Ora, convenhamos que vocês possuem o melhor caso, porque vocês deram a sorte de as cartas recebidas já estarem ordenadas. Já eu peguei o pior caso, porque as cartas estão ordenadas na ordem inversa. Por fim, o caso médio ocorre caso as cartas recebidas estejam em uma ordem aleatória. Com isso, espero que vocês tenham entendido o sentido de pior, médio e melhor casos.

Vamos partir agora para o estudo da Notação Big-O (ou Notação Assintótica)! Isso é simplesmente uma forma de representar o comportamento assintótico de uma função. No nosso contexto, ela busca expressar a quantidade de operações primitivas executadas como função do tamanho da entrada de dados. Vamos ver isso melhor!

A Notação Big-O é a representação relativa da complexidade de um algoritmo. É relativa porque só se pode comparar maçãs com maçãs, isto é, você não pode comparar um algoritmo de multiplicação aritmética com um algoritmo de ordenação de inteiros. É uma representação porque reduz a comparação entre algoritmos a uma simples variável por meio de observações e suposições.

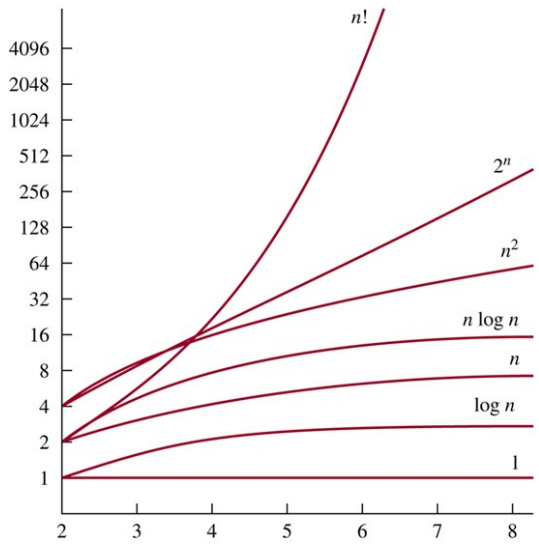
E trata da complexidade porque se é necessário 1 segundo para ordenar 10.000 elementos, quanto tempo levará para ordenar 1.000.000? A complexidade, nesse exemplo particular, é a medida relativa para alguma coisa. Vamos ver isso por meio de um exemplo: soma de dois inteiros! A soma é uma operação ou um problema, e o método para resolver esse problema é chamado algoritmo!

Transporte →	1	1	1	1	0
Parcela 1 →		1	1	0	1
Parcela 2 →	+	1	0	1	1
Soma →	1	1	0	0	0

Vamos supor que no algoritmo de somar (mostrado acima), a operação mais cara seja a adição. Observem que se somarmos dois números de 100 dígitos, teremos que fazer 100 adições. Se somarmos dois números de 10.000 dígitos, teremos que fazer 10.000 adições. Perceberam o padrão? A complexidade (aqui, número de operações) é diretamente proporcional ao número n de dígitos, i.e., O(n).

Quando dizemos que um algoritmo é $O(n^2)$, estamos querendo dizer que esse algoritmo é da ordem de grandeza quadrática! Ele basicamente serve para te dizer quão rápido uma função cresce, por exemplo: um algoritmo O(n) é melhor do que um algoritmo $O(n^2)$, porque ela cresce mais lentamente! Abaixo vemos uma lista das classes de funções mais comuns em ordem crescente de crescimento:

Notação	Nome
O(1)	Constante
O(log n)	Logarítmica
$O[(\log n)^c]$	Polilogarítmica
O(n)	Linear
O(n log n)	–
$O(n^2)$	Quadrática
$O(n^3)$	Cúbica
$O(n^c)$	Polinomial
$O(c^n)$	Exponencial
O(n!)	Fatorial



Quando dizemos que o Shellsort é um algoritmo $O(n^2)$, estamos querendo dizer que a complexidade (nesse caso, o número de operações) para ordenar um conjunto de n dados com o Algoritmo Shellsort é proporcional ao quadrado do número de elementos no conjunto! Grosso modo, para ordenar 20 números, é necessário realizar 400 operações (sem entrar em detalhes sobre a operação em si, nem sobre as simplificações matemáticas que são realizadas).

Entender como se chega a esses valores para cada método de ordenação e pesquisa é extremamente complexo! Galera, apesar de eu nunca ter visto isso em prova, é bom que vocês saibam que existem outras notações! Utiliza-se Notação Big-O (O) para pior caso; Notação Big-Ômega para melhor caso (Ω); e Notação Big-Theta (Θ) para caso médio.

Como na prática utiliza-se Big-O para tudo, o que eu recomendo (infelizmente, porque eu sei que vocês têm zilhões de coisas para decorar) é memorizar o pior caso dos principais métodos. Dessa forma, é possível responder a maioria das questões de prova sobre esse tema. Eventualmente, as questões pedem também caso médio e melhor caso, mas é menos comum. Bacana? :-)

Por último, uma pergunta muito frequente: Professor, já vi questões cobrando Logaritmo na Base 10, Logaritmo na Base 2, Logaritmo Neperiano, etc... isso não está errado? Galera, suponha que um algoritmo levou um tempo Θ(log10 n). Você também poderia dizer que levou um tempo Θ(lg n) (ou seja, Θ(log2 n)). Você pode utilizar qualquer base.

Sempre que a base do logaritmo é uma constante, não importa a base que usamos na notação assintótica. Por que não? Porque, para a notação assintótica, isso é completamente irrelevante. Beleza? Então, não se prendam a base do logaritmo, qualquer uma pode ser utilizada na representação de complexidade assintótica de algoritmos. Bacana? Exercícios...

PESQUISA DE DADOS

Uma das tarefas de maior importância na computação é a pesquisa de informações contidas em coleções de dados. Em geral, desejamos que essa tarefa seja executada sem que haja a necessidade de inspecionar toda a coleção de dados. Existem algumas maneiras de realizar pesquisas, tais como: Pesquisa Sequencial, Pesquisa Binária, Tabelas de Dispersão (Hashing), Árvores AVL, Árvores B e Árvores B+.

Busca Sequencial

Galera, imaginem que eu estou à procura de um valor X em um vetor L[ ]! Para tal, posso inspecionar as posições sequenciais de L[ ] a partir da primeira posição: se eu encontrar X, minha busca tem êxito; se eu alcanço a última posição e não encontro X, concluímos que esse valor não ocorre no vetor L[ ]. Como é chamada essa busca em que eu inspeciono uma estrutura posição por posição? Sequencial ou Linear.

Considerando que o vetor L[ ] contém N elementos, ordenados ou não, é fácil verificar que a busca sequencial requer tempo linearmente proporcional ao tamanho do vetor, i.e., da ordem O(n). Por conta disso, é comum dizer que a busca sequencial é uma Busca Linear. Entenderam? Quanto maior o vetor, maior o tempo em média para buscar um elemento! Quanto mais ao final, mais demorado.

A Busca Sequencial é muito lenta para grandes quantidades de dados, mas aceitável para listas pequenas e que mudam constantemente. Observa-se que no Melhor Caso, X está na primeira posição, logo necessita apenas de uma comparação; no Pior Caso, X está na última posição, logo necessita de N comparações; e no Caso Médio, X é encontrado após (n+1)/2 comparações.

A seguir, encontram-se dois algoritmos para realização de uma Busca Sequencial: o primeiro ocorre de forma simples e o segundo ocorre de forma recursiva. Galera, para vetores de médio ou grande porte, o tempo de busca sequencial é considerado completamente inaceitável, dado que existem técnicas mais eficientes! Professor, me dá um exemplo? Claro, veremos adiante: Busca Binária!

```
PROCEDIMENTO BUSCA_SEQUENCIAL ( L , X , POS )

   ENTRADA: UM VETOR L e UM VALOR X
   SAÍDA: POS = i, SE X OCORRE NA POSIÇÃO i DE L // SUCESSO
          POS = 0, CASO CONTRÁRIO.

   POS = 0

   PARA i = 1 até N
       SE L[i] = X
          POS = i
          ESCAPE
   // fim para
   DEVOLVA POS
FIM {BUSCA_SEQUENCIAL}
```

```
PROCEDIMENTO BUSCA_SEQUENCIAL_REC ( L , N, X )

   ENTRADA: UM VETOR L DE TAMANHO N e UM VALOR X
   SAÍDA: i, SE X OCORRE NA POSIÇÃO i DE L    // SUCESSO
          0, CASO CONTRÁRIO.

 SE N = 1
       SE L[1] = X
            DEVOLVA 1
       SENÃO
            DEVOLVA 0
         SENÃO
       SE L[N] = X
          DEVOLVA N
 SENÃO
       BUSCA_SEQUENCIAL_REC ( L , N-1, X )

FIM {BUSCA_SEQUENCIAL_REC}
```

Busca Binária

A Busca Binária é um algoritmo de busca em vetores que segue o paradigma de divisão-e-conquista. Parte-se do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca, comparando o elemento chave com o elemento do meio do vetor. Possui complexidade da ordem de $O(\log_2 n)$, em que N é o tamanho do vetor de busca.

Quando o Vetor L[ ] estiver em ordem crescente, podemos determinar se X ocorre em L[ ] de forma mais rápida da seguinte forma: inspeciona-se a posição central do vetor! Se essa posição já contiver X, a busca para! Por que, professor? Porque nós já encontramos X! Se X for menor que esse elemento central, passamos a procurar X, recursivamente, no intervalo de L[ ] que se encontra à esquerda da posição central.

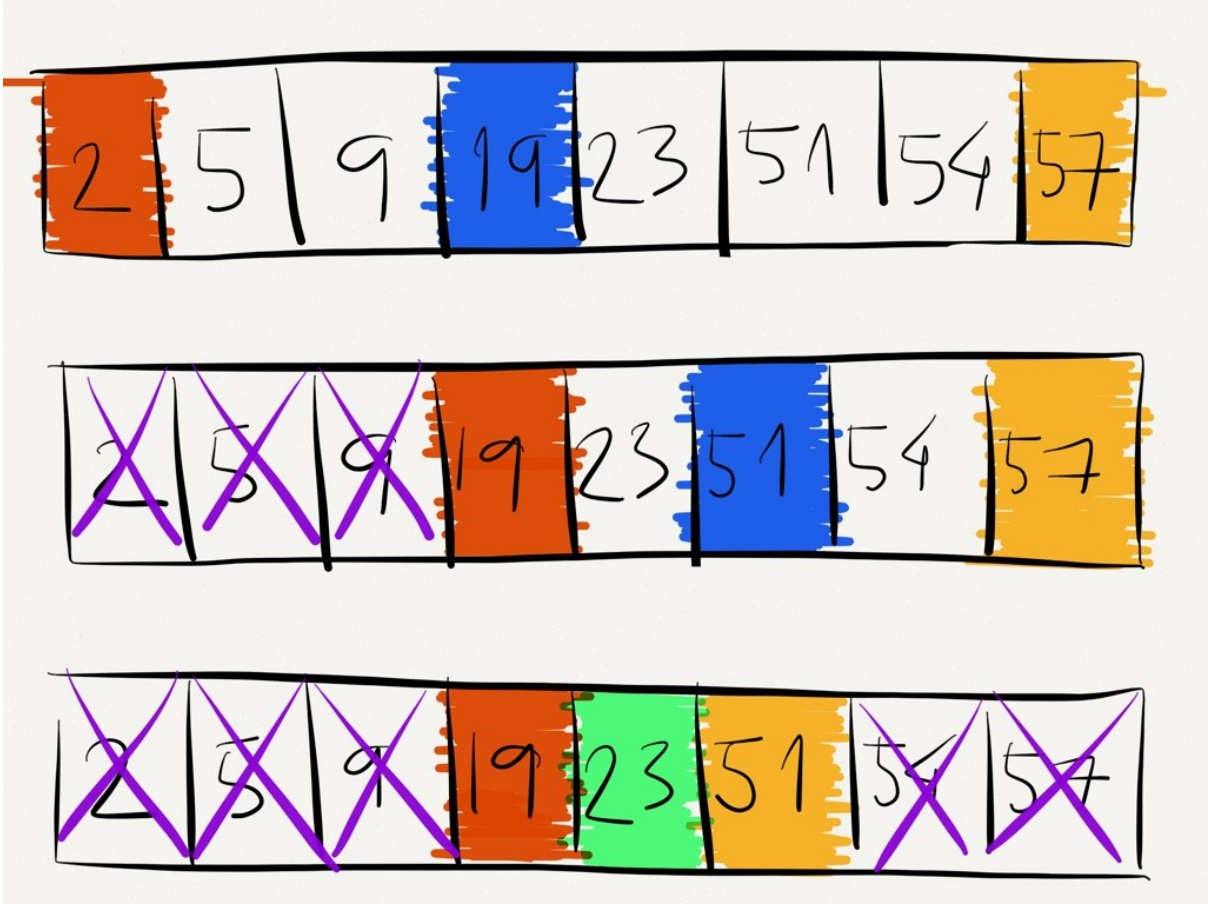
Se X for maior do que o elemento central, continuamos a procurar X, recursivamente, no intervalo de L que está à direita da posição central. Se o intervalo se tornar vazio, a busca para, tendo sido malsucedida. Esse procedimento é conhecido como Busca Binária e, facilmente, pode-se adaptar a busca em ordem decrescente. Segue abaixo um possível algoritmo:

```
PROCEDIMENTO BUSCA_BINARIA
  ENTRADA: UM VETOR L EM ORDEM CRESCENTE, UM VALOR X, E AS
           POSIÇÕES INICIO E FIM.
  SAÍDA: SIM, SE X OCORRE ENTRE AS POSIÇÕES INICIO E FIM DE
         L; NÃO, CASO CONTRÁRIO.

  SE INICIO > FIM
    DEVOLVA NÃO E PARE
  MEIO = (INICIO+FIM)/2              // DIVISÃO INTEIRA
  SE X = L[MEIO]
    DEVOLVA SIM E PARE
  SE X < L[MEIO]
    DEVOLVA BUSCA_BINARIA(L, X, INICIO, MEIO - 1)
  SENAO
    DEVOLVA BUSCA_BINARIA(L, X, MEIO + 1, FIM)
FIM {BUSCA_BINARIA}
```

Na imagem abaixo, estamos à procura do valor 23! Em vermelho, encontra-se o elemento inicial L[0] = 2 e, em amarelo, encontra-se o elemento final L[N-1] = 57. Procuramos, então, o elemento central! Como? Ele é o elemento de índice [0 + (N-1)]/2 = 7/2 = 3,5 = 3 (Arredonda-se para baixo). Ora, L[3] = 19! Encontramos? Não, 23>19! Sendo assim, L[0] = 19 e L[4] = 57.

Procuramos, então, o **elemento central!** Como? Ele é o elemento de índice [0 + (N-1)]/2 = 4/2 = 2. Ora, L[2] = 51! Encontramos? Não, 23<51! Sendo assim, L[0] = 19 e L[2] = 51. Procuramos, então, o elemento central! Como? Ele é o elemento de índice [0+ (N-1)]/2 = 2/2 = 1. Ora, L[1] = 23! Encontramos? Sim! Então, nossa busca obteve êxito e encontramos o que buscávamos.



QUESTÕES COMENTADAS – MÉTODOS DE ORDENAÇÃO - MULTIBANCAS

1. (Instituto Cidades - 2012 - TCM-GO - Auditor de Controle Externo - Informática) São exemplos de algoritmos de ordenação, exceto:

   a) Bubble Sort

   b) Select Sort

   c) Shell Sort

   d) Busca Sequencial;

   e) Quick Sort;

Comentários:

Conforme vimos em aula, a Busca Sequencial não é um algoritmo de ordenação! Na verdade, ele é um método de pesquisa sobre estruturas de dados. Gabarito: D

2. (FUMARC - 2012 - TJ-MG - Técnico Judiciário - Analista de Sistemas – I) Quicksort divide um conjunto de itens em conjuntos menores, que são ordenados de forma independe, e depois os resultados são combinados para produzir a solução de ordenação do conjunto maior.

Comentários:

Conforme vimos em aula, está perfeito! Sendo um algoritmo do tipo Dividir Para Conquistar, ele reparte o conjunto de dados em conjuntos menores, que são ordenados independentemente e depois combinados em uma solução maior. Gabarito: C

3. (CESPE - 2012 - MPE-PI - Analista Ministerial - Informática - Cargo 6) O heapsort é um algoritmo de ordenação em que a quantidade de elementos armazenada fora do arranjo de entrada é constante durante toda a sua execução.

Comentários:

Inicialmente, insere-se os elementos da lista em um heap. Em seguida, fazemos sucessivas remoções do menor elemento do heap, colocando os elementos removidos do heap de volta na lista – a lista estará então em ordem crescente. O heapsort é um algoritmo de ordenação em que a sua estrutura auxiliar de armazenamento fora do arranjo de entrada é constante durante toda a sua execução.

Essa questão é polêmica. O arranjo tem tamanho constante, mas a quantidade de elementos é variável. Diferente de outros algoritmos de ordenação que tem uma estrutura auxiliar de tamanho variável (assim como seus elementos), o Heap Sort tem uma estrutura auxiliar de tamanho fixo (porém a quantidade de elementos é variável). Como é dito por Neil Dale: "A heapsort is just as

efficient in terms of space; only one array is used to store the data. The heap sort requires only constante extra space". No entanto, a questão foi dada como correta! Gabarito: C

4. (CESPE - 2010 - ABIN - Oficial Técnico de Inteligência - Área de Suporte a Rede de Dados) A eficácia do método de ordenação rápida (quicksort) depende da escolha do pivô mais adequado ao conjunto de dados que se deseja ordenar. A situação ótima ocorre quando o pivô escolhido é igual ao valor máximo ou ao valor mínimo do conjunto de dados.

Comentários:

Alguns autores consideram a divisão em três subconjuntos, sendo o terceiro contendo valores iguais ao pivô. O Melhor Caso ocorre quando o conjunto é dividido em subconjuntos de mesmo tamanho; o Pior Caso ocorre quando o pivô corresponde a um dos extremos (menor ou maior valor). Alguns o consideram um algoritmo frágil e não-estável, com baixa tolerância a erros.

Conforme vimos em aula, a questão se refere ao pior caso! Gabarito: E

5. (CESPE - 2010 - ABIN - Oficial Técnico de Inteligência - Área de Suporte a Rede de Dados) A estabilidade de um método de ordenação é importante quando o conjunto de dados já está parcialmente ordenado.

Comentários:

Na imagem acima, foi colocado um sinal de aspas simples e duplas apenas para diferenciá-los, mas trata-se do mesmo número. Um algoritmo estável ordena todo o restante e não perde tempo trocando as posições de elementos que possuam chaves idênticas. Já um algoritmo instável ordena todos os elementos, inclusive aqueles que possuem chaves idênticas (sob algum outro critério).

Conforme vimos em aula, a estabilidade é irrelevante com dados parcialmente ordenados ou não! A estabilidade é importante quando se deseja ordenar um conjunto de dados por mais de um critério (Ex: primeiro pelas chaves e segundo por índices). Se esse não for o caso (e a questão não disse que era!), a estabilidade "não fede nem cheira". O fato de os dados estarem parcialmente ordenados não fará diferença em termos de ordenação – ambos serão ordenados da mesma maneira. Gabarito: E

6. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) A classificação interna por inserção é um método que realiza a ordenação de um vetor por meio da inserção de cada elemento em sua posição correta dentro de um subvetor classificado.

Comentários:

Conforme vimos em aula, trata-se do InsertionSort! Gabarito: C

7. (FCC - 2009 - TRT - 15ª Região - Analista Judiciário - Tecnologia da Informação) São algoritmos de classificação por trocas apenas os métodos:

a) SelectionSort e InsertionSort.

b) MergeSort e BubbleSort.

c) QuickSort e SelectionSort.

d) BubbleSort e QuickSort.

e) InsertionSort e MergeSort.

Comentários:

Conforme vimos em aula, BubbleSort e QuickSort são métodos de troca; InsertionSort é um método de Inserção; SelectionSort e HeapSort são métodos de Seleção; e MergeSort é um método de Intercalação. Gabarito: D

8. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – I) O tempo de pior caso do algoritmo QuickSort é de ordem menor que o tempo médio do algoritmo Bubblesort.

Comentários:

Algoritmo	Melhor Caso	Caso Médio	Pior Caso
BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$
QuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do *gap*	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)
HeapSort	O(n log n)	O(n log n)	O(n log n)

Conforme vimos em aula, está incorreto! São iguais: $O(n^2)$. Gabarito: E

9. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – II) O tempo médio do QuickSort é O(nlog2n), pois ele usa como estrutura básica uma árvore de prioridades.

Comentários:

Algoritmo	Melhor Caso	Caso Médio	Pior Caso
BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$
cQuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do *gap*	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)
HeapSort	O(n log n)	O(n log n)	O(n log n)

Conforme vimos em aula, de fato, ele tem tempo médio O(n log n), mas ele usa como estrutura básica uma lista ou um vetor! Gabarito: E

10.(CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – III) O tempo médio do QuickSort é de ordem igual ao tempo médio do MergeSort.

Comentários:

Algoritmo	Melhor Caso	Caso Médio	Pior Caso
BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$
QuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do *gap*	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)
HeapSort	O(n log n)	O(n log n)	O(n log n)

Conforme vimos em aula, ambos têm tempo O(n log n). Gabarito: C

11.(CESGRANRIO - 2012 - CMB – Analista de Sistemas – III) Em uma reunião de análise de desempenho de um sistema WEB, um programador apontou corretamente que a complexidade de tempo do algoritmo bubblesort, no pior caso, é:

a) O(1)

b) O(log n)

c) O(n)

d) O(n log n)

e) O(n2)

Comentários:

Algoritmo	Melhor Caso	Caso Médio	Pior Caso
BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$
QuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do *gap*	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)
HeapSort	O(n log n)	O(n log n)	O(n log n)

Conforme vimos em aula, trata-se de $O(n^2)$! Gabarito: E

12.(CESPE - 2010 – INMETRO – Analista de Sistemas) Se f é uma função de complexidade para um algoritmo F, então O(f) é considerada a complexidade assintótica ou o comportamento assintótico do algoritmo F. Assinale a opção que apresenta somente algoritmos que possuem complexidade assintótica quando f(n) = O(n log n).

a) HeapSort e BubbleSort

b) QuickSort e InsertionSort

c) MergeSort e BubbleSort

d) InsertionSort

e) HeapSort, QuickSort e MergeSort

Comentários:

Algoritmo	Melhor Caso	Caso Médio	Pior Caso
BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$

QuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do *gap*	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)
HeapSort	O(n log n)	O(n log n)	O(n log n)

Conforme vimos em aula, trata-se da última opção. Vejam que ele não utilizou, nessa questão, o Pior Caso. Gabarito: E

13.(FGV - 2013 – MPE/MS – Analista de Sistemas) Assinale a alternativa que indica o algoritmo de ordenação capaz de funcionar em tempo O(n) para alguns conjuntos de entrada.

a) Selectionsort (seleção)

b) Insertionsort (inserção)

c) Merge sort

d) Quicksort

e) Heapsort

Comentários:

Algoritmo	Melhor Caso	Caso Médio	Pior Caso
BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$
QuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do *gap*	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)
HeapSort	O(n log n)	O(n log n)	O(n log n)

Conforme vimos em aula, trata-se da segunda opção. Gabarito: B

14.(CESGRANRIO - 2010 – BACEN – Analista de Sistemas) Uma fábrica de software foi contratada para desenvolver um produto de análise de riscos. Em determinada funcionalidade desse software, é necessário realizar a ordenação de um conjunto formado por muitos números inteiros. Que algoritmo de ordenação oferece melhor complexidade de tempo (Big O notation) no pior caso?

a) Merge sort

b) Insertion sort

c) Bubble sort

d) Quick sort

e) Selection sort

Comentários:

Algoritmo	Melhor Caso	Caso Médio	Pior Caso
BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$
QuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do *gap*	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)
HeapSort	O(n log n)	O(n log n)	O(n log n)

Conforme vimos em aula, trata-se da primeira opção! Gabarito: A

15.(CESPE - 2011 – FUB – Analista de Sistemas) Os métodos de ordenação podem ser classificados como estáveis ou não estáveis. O método é estável se preserva a ordem relativa de dois valores idênticos. Alguns métodos eficientes como shellsort ou quicksort não são estáveis, enquanto alguns métodos pouco eficientes, como o método da bolha, são estáveis.

Comentários:

Métodos Estáveis: BubbleSort, InsertionSort e MergeSort; Métodos Instáveis: SelectionSort, QuickSort, HeapSort e ShellSort. Portanto, item perfeito! Gabarito: C

16.(CESPE - 2012 – BASA – Analista de Sistemas) O método de classificação Shellsort iguala-se ao método Quicksort em termos de complexidade temporal, porém é mais eficiente para quantidades pequenas a moderadas de dados.

Comentários:

Algoritmo	Melhor Caso	Caso Médio	Pior Caso

BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$
QuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do *gap*	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)
HeapSort	O(n log n)	O(n log n)	O(n log n)

Não, a complexidade temporal é completamente diferente! A complexidade que nós estudamos em aula se trata do comportamento do algoritmo em termos de custo - ela fornece uma resposta digamos que "genérica". Por que? Porque são ignorados constantes, parâmetros, etc - perceba que nós dizemos, por exemplo, que o algoritmo x tem complexidade $O(n^2)$. O que isso significa? Significa apenas que esse algoritmo tem um custo polinomial, mas nós sabemos que funções polinomiais têm o formato $ax^2 + bx + c$. Onde está o a, bx, c? Nós ignoramos e ficamos apenas com o parâmetro mais importante. Além disso tudo, ainda temos que considerar que existe o pior caso, melhor caso e caso médio.

Para saber a complexidade temporal de um algoritmo de ordenação, nós temos que nos focar em diversos parâmetros. Além das constantes e parâmetros da função que representa a complexidade, nós temos que levar em consideração aspectos práticos (Por exemplo: o hardware em que será rodado o programa). Resumindo nosso papo: não há correlação direta entre complexidade de custo e complexidade temporal, logo eu não posso usar a função de um para calcular o outro. Gabarito: E

17.(CESPE - 2012 – BASA – Analista de Sistemas) O método de classificação Quicksort é estável e executado em tempo linearmente dependente da quantidade de dados que estão sendo classificados.

Comentários:

Algoritmo	Melhor Caso	Caso Médio	Pior Caso
BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$
QuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do *gap*	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)

HeapSort	O(n log n)	O(n log n)	O(n log n)

Conforme vimos em aula, QuickSort é instável e não possui complexidade temporal linear! Gabarito: E

18.(CESPE - 2012 – BASA – Analista de Sistemas) No método de ordenamento denominado shellsort, as comparações e as trocas são feitas conforme determinada distância entre dois elementos, de modo que, uma distância igual a 6 seria a comparação entre o primeiro elemento e o sétimo, ou entre o segundo elemento e o oitavo, e assim sucessivamente, repetindo-se esse processo até que as últimas comparações e trocas tenham sido efetuadas e a distância tenha diminuído até chegar a 1.

Comentários:

É o algoritmo mais eficiente dentre os de ordem quadrática. Nesse método, as comparações e as trocas são feitas conforme determinada distância (gap) entre dois elementos, de modo que, se gap = 6, há comparação entre o 1° e 7° elementos ou entre o 2° e 8° elementos e assim sucessivamente, repetindo até que as últimas comparações e trocas tenham sido efetuadas e o gap tenha chegado a 1.

Conforme vimos em aula, a questão descreveu perfeitamente o mecanismo de ordenação do Shellsort. Gabarito: C

19.(FGV - 2008 – PETROBRÁS – Analista de Sistemas) Sobre o algoritmo de ordenação heapsort, assinale a afirmação correta.

a) Utiliza ordenação por árvore de decisão, ao invés de ordenação por comparação.

b) A estrutura de dados que utiliza, chamada heap, pode ser interpretada como uma árvore binária.

c) Seu desempenho de pior caso é pior do que o do algoritmo quicksort.

d) Seu desempenho de pior caso é o mesmo da ordenação por inserção.

e) Seu desempenho de pior caso é menor do que o da ordenação por intercalação.

Comentários:

(a) Utiliza ordenação por seleção; (b) Perfeito; (c) Não, é melhor que o QuickSort; (d) Não, é melhor que o InsertionSort; (e) Não, é idêntico ao MergeSort. Gabarito: B

20.(CESGRANRIO – 2009 – BASA – Analista de Sistemas) Com relação aos algoritmos quicksort e mergsort, o tempo de execução para o:

a) pior caso do quicksort é (n lg n).

b) pior caso do mergesort é (n2).

c) pior caso do mergesort é (n lg n).

d) caso médio do mergesort é O(lg n).

e) caso médio do quicksort é O(n2).

Comentários:

Algoritmo	Melhor Caso	Caso Médio	Pior Caso
BubbleSort	O(n)	$O(n^2)$	$O(n^2)$
InsertionSort	O(n)	$O(n^2)$	$O(n^2)$
SelectionSort	$O(n^2)$	$O(n^2)$	$O(n^2)$
QuickSort	O(n log n)	O(n log n)	$O(n^2)$
ShellSort	O(n log n)	Depende do *gap*	$O(n^2)$
MergeSort	O(n log n)	O(n log n)	O(n log n)
HeapSort	O(n log n)	O(n log n)	O(n log n)

Conforme vimos em aula, trata-se da terceira opção. Gabarito: C

21.(CESPE – 2009 – UNIPAMPA – Analista de Sistemas) O algoritmo quicksort, que divide uma instrução em quatro blocos diferentes de busca, é um exemplo de estrutura de ordenação de dados.

Comentários:

Conforme vimos em aula, são dois blocos diferentes de busca. Gabarito: E

22.(CESPE - 2013 – CPRM – Analista de Sistemas) No algoritmo de ordenação denominado quicksort, escolhe-se um ponto de referência, denominado pivô, e separam-se os elementos em dois grupos: à esquerda, ficam os elementos menores que o pivô, e à direita ficam os maiores. Repete-se esse processo para os grupos de elementos formados (esquerda e direita) até que todos os elementos estejam ordenados.

Comentários:

Esse algoritmo divide um conjunto de itens em conjuntos menores, que são ordenados de forma independente, e depois os resultados são combinados para produzir a solução de ordenação do conjunto maior. Trata-se, portanto, de um algoritmo do tipo Divisão-e-Conquista, i.e., repartindo os dados em subgrupos, dependendo de um elemento chamado pivô.

Neste método, a lista é dividida em parte esquerda e parte direita, sendo que os elementos da parte esquerda são todos menores que os elementos da parte direita. Essa fase do processo é chamada de partição. Em seguida, as duas partes são ordenadas recursivamente (usando o próprio QuickSort). A lista está, portanto, ordenada corretamente!

Conforme vimos em aula, é exatamente assim! Gabarito: C

23.(CESPE - 2013 – MPU – Analista de Sistemas) Entre os algoritmos de ordenação e pesquisa bubble sort, quicksort e heapsort, o quicksort é considerado o mais eficiente, pois se caracteriza como um algoritmo de dividir-para-conquistar, utilizando operações de particionamento.

Comentários:

Conforme vimos em aula, o HeapSort é o mais eficiente no pior caso! Gabarito: E

24.(CESPE - 2013 – TRT/9 – Analista de Sistemas) No método Quicksort, o pivô é responsável pelo número de partições em que o vetor é dividido. Como o pivô não pode ser um elemento que esteja repetido no vetor, o Quicksort não funciona quando há elementos repetidos.

Comentários:

O pivô não é responsável pelo número de partições em que o vetor é dividido. Ademais, ele pode sim ser um elemento que esteja repetido no vetor! Gabarito: E

25.(FCC - 2011 - TRT - 14ª Região (RO e AC) - Analista Judiciário - Tecnologia da Informação) NÃO se trata de um método de ordenação (algoritmo):

a) inserção direta.

b) seleção direta.

c) inserção por meio de incrementos decrescentes.

d) direta em cadeias.

e) particionamento.

Comentários:

(a) Trata-se do InsertionSort; (b) Trata-se do SelectionSort; (c) Trata-se do ShellSort; (d) Trata-se de um método de busca; (e) Trata-se do QuickSort. Portanto, é a quarta opção! Gabarito: D

26.(FCC - 2016 - TRT - 23ª REGIÃO (MT) - Analista Judiciário - Tecnologia da Informação) Considere o método de ordenação abaixo.

```
void ordena(int m,int x[]) {
  int aux,j,i;
  for(i=0;i<m-1;i++) {
    for(j=0;j<m-i-1;j++)
      if (x[j] > x[j+1]) {
        aux=x[j];
        x[j]=x[j+1];
        x[j+1]=aux;
      }
  }
}
```

Utilizando este algoritmo de ordenação, percorre-se a lista dada da esquerda para a direita, comparando pares de elementos consecutivos, trocando de lugar os que estão fora da ordem. Em cada troca, o maior elemento é deslocado uma posição para a direita. Trata-se de um algoritmo de ordenação:

a) Select Sort.

b) Insert Sort.

c) Bubble Sort.

d) Shell Sort.

e) Quick Sort.

Comentários:

O algoritmo realiza trocas de dois em dois, percorrendo todos os elementos. Como vimos, esse algoritmo é o Bubble Sort. Gabarito: C

QUESTÕES COMENTADAS – COMPLEXIDADE DE ALGORITMOS - MULTIBANCAS

1. (VUNESP – 2012 – TJ/SP - Analista Judiciário - Tecnologia da Informação) Considerando o conceito de Complexidade de Algoritmos, representado por O(função), assinale a alternativa que apresenta, de forma crescente, as complexidades de algoritmos.

a) O(2n); O(n3); O(n2); O(log2 n); O(n.log2 n).

b) O(n2); O(n3); O(2n); O(log2 n); O(n.log2 n).

c) O(n3); O(n2); O(2n); O(n.log2 n); O(log2 n).

d) O(log2 n); O(n.log2 n); O(n2); O(n3); O(2n).

e) O(n.log2 n); O(log2 n); O(2n); O(n3); O(n2).

Comentários:

Notação	Nome
O(1)	Constante
O(log n)	Logarítmica
O[(log n)^c]	Polilogarítmica
O(n)	Linear
O(n log n)	–
O(n^2)	Quadrática
O(n^3)	Cúbica
O(n^c)	Polinomial
O(c^n)	Exponencial
O(n!)	Fatorial

Conforme vimos em aula, basta consultar a tabelinha! Gabarito: D

2. (FCC - 2010 - TRT - 8ª Região (PA e AP) - Analista Judiciário - Tecnologia da Informação) Numa competição de programação, ganhava mais pontos o time que apresentasse o algoritmo mais eficiente para resolver o pior caso de um determinado problema. A complexidade assintótica (notação Big O) dos algoritmos elaborados está ilustrada na tabela abaixo.

Time	Complexidade
Branco	$O(n^{20})$
Amarelo	O(nlogn)
Azul	O(1)
Verde	O(n!)
Vermelho	$O(2^n)$

O time que obteve a medalha de prata (2º algoritmo mais eficiente) é o:

a) Branco.

b) Amarelo.

c) Azul.

d) Verde.

e) Vermelho.

Comentários:

Notação	Nome
O(1)	Constante
O(log n)	Logarítmica
$O[(\log n)^c]$	Polilogarítmica
O(n)	Linear
O(n log n)	–
$O(n^2)$	Quadrática
$O(n^3)$	Cúbica
$O(n^c)$	Polinomial
$O(c^n)$	Exponencial
O(n!)	Fatorial

Conforme vimos em aula, basta consultar a tabelinha! O primeiro é o time Azul (O(1)) e o segundo é o time Amarelo (O(n log n)). Gabarito: B

QUESTÕES COMENTADAS – PESQUISA DE DADOS - MULTIBANCAS

1. (CESPE - 2013 – TRT/MS – Analista de Sistemas) Considerando que se deseje efetuar uma pesquisa de um valor sobre a chave primária de uma tabela de um banco de dados com uma chave primária com um tipo de campo que receba um valor inteiro e que se possa fazer essa pesquisa utilizando-se a busca sequencial ou a busca binária, assinale a opção correta.

   a) O método de busca binária requer, no máximo, ln(n) comparações para determinar o elemento pesquisado, em que n é o número de registros.

   b) O método de busca binária será sempre mais rápido que o método de busca sequencial, independentemente de a tabela estar ordenada com base no elemento pesquisado.

   c) O método de busca sequencial requererá, no máximo, n^2 comparações para determinar o elemento pesquisado, em que n será o número de registros.

   d) O método de busca binária sempre efetuará menos comparações que o método de pesquisa sequencial.

   e) O método de busca sequencial efetuará menos comparações para encontrar o elemento pesquisado quando a tabela estiver ordenada em comparação à situação quando a tabela estiver desordenada.

Comentários:

A Busca Binária é um algoritmo de busca em vetores que segue o paradigma de divisão-e-conquista. Parte-se do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca, comparando o elemento chave com o elemento do meio do vetor. Possui complexidade da ordem de O(log2 n), em que N é o tamanho do vetor de busca.

a) Conforme vimos em aula, não é logaritmo neperiano (na verdade, é Base 2), mas a banca considerou correto mesmo assim.

A Busca Sequencial é muito lenta para grandes quantidades de dados, mas aceitável para listas pequenas e que mudam constantemente. Observa-se que no Melhor Caso, X está na primeira posição, logo necessita apenas de uma comparação; no Pior Caso, X está na última posição, logo necessita de N comparações; e no Caso Médio, X é encontrado após (n+1)/2 comparações.

b) Conforme vimos em aula, isso não ocorre sempre! Se compararmos o Melhor Caso da Busca Sequencial com o Pior Caso da Busca Binária, a primeira será mais rápida.

Considerando que o vetor L[ ] contém N elementos, ordenados ou não, é fácil verificar que a busca sequencial requer tempo linearmente proporcional ao tamanho do vetor, i.e., da ordem O(n). Por conta disso, é comum dizer que a busca sequencial é uma Busca Linear. Entenderam? Quanto maior o vetor, maior o tempo em média para buscar um elemento! Quanto mais ao final, mais demorado.

c) Conforme vimos em aula, ele possui complexidade da ordem de O(n).

A Busca Sequencial é muito lenta para grandes quantidades de dados, mas aceitável para listas pequenas e que mudam constantemente. Observa-se que no Melhor Caso, X está na primeira

posição, logo necessita apenas de uma comparação; no Pior Caso, X está na última posição, logo necessita de N comparações; e no Caso Médio, X é encontrado após (n+1)/2 comparações.

d) Conforme vimos em aula, isso não ocorre sempre! Se compararmos o Melhor Caso da Busca Sequencial com o Pior Caso da Busca Binária, a primeira será mais rápida.

Considerando que o vetor L[ ] contém N elementos, ordenados ou não, é fácil verificar que a busca sequencial requer tempo linearmente proporcional ao tamanho do vetor, i.e., da ordem O(n). Por conta disso, é comum dizer que a busca sequencial é uma Busca Linear. Entenderam? Quanto maior o vetor, maior o tempo em média para buscar um elemento! Quanto mais ao final, mais demorado.

e) Conforme vimos em aula, não é necessário que a lista esteja ordenada. Logo, isso não fará diferença. Gabarito: A

2. (ESAF - 2001 – BACEN – Analista de Sistemas) Na pior hipótese, o número de comparações necessárias para pesquisar um elemento em um array de 2048 elementos pelo método de pesquisa binária será:

a) 8

b) 9

c) 10

d) 11

e) 12

Comentários:

A Busca Binária é um algoritmo de busca em vetores que segue o paradigma de divisão-e-conquista. Parte-se do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca, comparando o elemento chave com o elemento do meio do vetor. Possui complexidade da ordem de O(log2 n), em que N é o tamanho do vetor de busca.

Conforme vimos em aula, a complexidade da Busca Binária é O(log2 n), em que N é o tamanho do vetor de busca – no nosso caso, 2048! Quanto é log2 2048? 11! Por que, professor? Porque 211 = 2048! Gabarito: D

3. (CESPE - 2013 – TCE/RO – Analista de Sistemas) Considere uma tabela de um banco de dados com chave primária e tipo de campo que receba um valor inteiro. Ao se efetuar uma pesquisa de um valor sobre a chave primária dessa tabela, o método de busca binária requer, no máximo, lg(n) comparações para localizar o elemento pesquisado, em que n é o número de registros.

Comentários:

A Busca Binária é um algoritmo de busca em vetores que segue o paradigma de divisão-e-conquista. Parte-se do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do

espaço de busca, comparando o elemento chave com o elemento do meio do vetor. Possui complexidade da ordem de O(log2 n), em que N é o tamanho do vetor de busca.

Conforme vimos em aula, de fato a complexidade da Busca Binária é O(log2 n). Gabarito: C

4. (FCC – 2016 - TRT - 14ª Região (RO e AC) - Analista Judiciário - Tecnologia da Informação) Dada uma coleção de n elementos ordenados por ordem crescente, pretende-se saber se um determinado elemento x existe nessa coleção. Supondo que essa coleção está implementada como sendo um vetor a[0...n-1] de n elementos inteiros, utilizando-se um algoritmo de pesquisa binária, o número de vezes que a comparação x==a[i] será executada, no pior caso, é calculada por:

a) n/2.

b) n−1.

c) √n.

d) log2(n).

e) n−=2.

Comentários:

A Busca Binária é um algoritmo de busca em vetores que segue o paradigma de divisão-e-conquista. Parte-se do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca, comparando o elemento chave com o elemento do meio do vetor. Possui complexidade da ordem de O(log2 n), em que N é o tamanho do vetor de busca. Gabarito: D

LISTA DE QUESTÕES – MÉTODOS DE ORDENAÇÃO - MULTIBANCAS

1. (Instituto Cidades - 2012 - TCM-GO - Auditor de Controle Externo - Informática) São exemplos de algoritmos de ordenação, exceto:

   a) Bubble Sort

   b) Select Sort

   c) Shell Sort

   d) Busca Sequencial;

   e) Quick Sort;

2. (FUMARC - 2012 - TJ-MG - Técnico Judiciário - Analista de Sistemas – I) Quicksort divide um conjunto de itens em conjuntos menores, que são ordenados de forma independe, e depois os resultados são combinados para produzir a solução de ordenação do conjunto maior.

3. (CESPE - 2012 - MPE-PI - Analista Ministerial - Informática - Cargo 6) O heapsort é um algoritmo de ordenação em que a quantidade de elementos armazenada fora do arranjo de entrada é constante durante toda a sua execução.

4. (CESPE - 2010 - ABIN - Oficial Técnico de Inteligência - Área de Suporte a Rede de Dados) A eficácia do método de ordenação rápida (quicksort) depende da escolha do pivô mais adequado ao conjunto de dados que se deseja ordenar. A situação ótima ocorre quando o pivô escolhido é igual ao valor máximo ou ao valor mínimo do conjunto de dados.

5. (CESPE - 2010 - ABIN - Oficial Técnico de Inteligência - Área de Suporte a Rede de Dados) A estabilidade de um método de ordenação é importante quando o conjunto de dados já está parcialmente ordenado.

6. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) A classificação interna por inserção é um método que realiza a ordenação de um vetor por meio da inserção de cada elemento em sua posição correta dentro de um subvetor classificado.

7. (FCC - 2009 - TRT - 15ª Região - Analista Judiciário - Tecnologia da Informação) São algoritmos de classificação por trocas apenas os métodos:

   a) SelectionSort e InsertionSort.

b) MergeSort e BubbleSort.

c) QuickSort e SelectionSort.

d) BubbleSort e QuickSort.

e) InsertionSort e MergeSort.

8. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – I) O tempo de pior caso do algoritmo QuickSort é de ordem menor que o tempo médio do algoritmo Bubblesort.

9. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – II) O tempo médio do QuickSort é O(nlog2n), pois ele usa como estrutura básica uma árvore de prioridades.

10. (CESGRANRIO - 2011 - PETROBRÁS – Analista de Sistemas – III) O tempo médio do QuickSort é de ordem igual ao tempo médio do MergeSort.

11. (CESGRANRIO - 2012 - CMB – Analista de Sistemas – III) Em uma reunião de análise de desempenho de um sistema WEB, um programador apontou corretamente que a complexidade de tempo do algoritmo bubblesort, no pior caso, é:

a) O(1)

b) O(log n)

c) O(n)

d) O(n log n)

e) O(n2)

12. (CESPE - 2010 – INMETRO – Analista de Sistemas) Se f é uma função de complexidade para um algoritmo F, então O(f) é considerada a complexidade assintótica ou o comportamento assintótico do algoritmo F. Assinale a opção que apresenta somente algoritmos que possuem complexidade assintótica quando f(n) = O(n log n).

a) HeapSort e BubbleSort

b) QuickSort e InsertionSort

c) MergeSort e BubbleSort

d) InsertionSort

e) HeapSort, QuickSort e MergeSort

13. (FGV - 2013 – MPE/MS – Analista de Sistemas) Assinale a alternativa que indica o algoritmo de ordenação capaz de funcionar em tempo O(n) para alguns conjuntos de entrada.

a) Selectionsort (seleção)

b) Insertionsort (inserção)

c) Merge sort

d) Quicksort

e) Heapsort

14.(CESGRANRIO - 2010 – BACEN – Analista de Sistemas) Uma fábrica de software foi contratada para desenvolver um produto de análise de riscos. Em determinada funcionalidade desse software, é necessário realizar a ordenação de um conjunto formado por muitos números inteiros. Que algoritmo de ordenação oferece melhor complexidade de tempo (Big O notation) no pior caso?

a) Merge sort

b) Insertion sort

c) Bubble sort

d) Quick sort

e) Selection sort

15.(CESPE - 2011 – FUB – Analista de Sistemas) Os métodos de ordenação podem ser classificados como estáveis ou não estáveis. O método é estável se preserva a ordem relativa de dois valores idênticos. Alguns métodos eficientes como shellsort ou quicksort não são estáveis, enquanto alguns métodos pouco eficientes, como o método da bolha, são estáveis.

16.(CESPE - 2012 – BASA – Analista de Sistemas) O método de classificação Shellsort iguala-se ao método Quicksort em termos de complexidade temporal, porém é mais eficiente para quantidades pequenas a moderadas de dados.

17.(CESPE - 2012 – BASA – Analista de Sistemas) O método de classificação Quicksort é estável e executado em tempo linearmente dependente da quantidade de dados que estão sendo classificados.

18.(CESPE - 2012 – BASA – Analista de Sistemas) No método de ordenamento denominado shellsort, as comparações e as trocas são feitas conforme determinada distância entre dois elementos, de modo que, uma distância igual a 6 seria a comparação entre o primeiro elemento e o sétimo, ou entre o segundo elemento e o oitavo, e assim sucessivamente, repetindo-se esse processo até que as últimas comparações e trocas tenham sido efetuadas e a distância tenha diminuído até chegar a 1.

19.(FGV - 2008 – PETROBRÁS – Analista de Sistemas) Sobre o algoritmo de ordenação heapsort, assinale a afirmação correta.

a) Utiliza ordenação por árvore de decisão, ao invés de ordenação por comparação.

b) A estrutura de dados que utiliza, chamada heap, pode ser interpretada como uma árvore binária.

c) Seu desempenho de pior caso é pior do que o do algoritmo quicksort.

d) Seu desempenho de pior caso é o mesmo da ordenação por inserção.

e) Seu desempenho de pior caso é menor do que o da ordenação por intercalação.

20. (CESGRANRIO – 2009 – BASA – Analista de Sistemas) Com relação aos algoritmos quicksort e mergsort, o tempo de execução para o:

a) pior caso do quicksort é (n lg n).

b) pior caso do mergesort é (n2).

c) pior caso do mergesort é (n lg n).

d) caso médio do mergesort é O(lg n).

e) caso médio do quicksort é O(n2).

21. (CESPE – 2009 – UNIPAMPA – Analista de Sistemas) O algoritmo quicksort, que divide uma instrução em quatro blocos diferentes de busca, é um exemplo de estrutura de ordenação de dados.

22. (CESPE - 2013 – CPRM – Analista de Sistemas) No algoritmo de ordenação denominado quicksort, escolhe-se um ponto de referência, denominado pivô, e separam-se os elementos em dois grupos: à esquerda, ficam os elementos menores que o pivô, e à direita ficam os maiores. Repete-se esse processo para os grupos de elementos formados (esquerda e direita) até que todos os elementos estejam ordenados.

23. (CESPE - 2013 – MPU – Analista de Sistemas) Entre os algoritmos de ordenação e pesquisa bubble sort, quicksort e heapsort, o quicksort é considerado o mais eficiente, pois se caracteriza como um algoritmo de dividir-para-conquistar, utilizando operações de particionamento.

24. (CESPE - 2013 – TRT/9 – Analista de Sistemas) No método Quicksort, o pivô é responsável pelo número de partições em que o vetor é dividido. Como o pivô não pode ser um elemento que esteja repetido no vetor, o Quicksort não funciona quando há elementos repetidos.

25. (FCC - 2011 - TRT - 14ª Região (RO e AC) - Analista Judiciário - Tecnologia da Informação) NÃO se trata de um método de ordenação (algoritmo):

a) inserção direta.

b) seleção direta.

c) inserção por meio de incrementos decrescentes.

d) direta em cadeias.

e) particionamento.

26. (FCC - 2016 - TRT - 23ª REGIÃO (MT) - Analista Judiciário - Tecnologia da Informação)

Considere o método de ordenação abaixo.

```
void ordena(int m,int x[]) {
  int aux,j,i;
  for(i=0;i<m-1;i++) {
    for(j=0;j<m-i-1;j++)
      if (x[j] > x[j+1]) {
        aux=x[j];
        x[j]=x[j+1];
        x[j+1]=aux;
      }
  }
}
```

Utilizando este algoritmo de ordenação, percorre-se a lista dada da esquerda para a direita, comparando pares de elementos consecutivos, trocando de lugar os que estão fora da ordem. Em cada troca, o maior elemento é deslocado uma posição para a direita. Trata-se de um algoritmo de ordenação:

a) Select Sort.

b) Insert Sort.

c) Bubble Sort.

d) Shell Sort.

e) Quick Sort.

GABARITO

1. D
2. C
3. C
4. E
5. E
6. C
7. D
8. E
9. E
10. C
11. E
12. E
13. B
14. A
15. C
16. E
17. E
18. C
19. B
20. C
21. E
22. C
23. E
24. E
25. D
26. C

LISTA DE QUESTÕES – COMPLEXIDADE DE ALGORITMOS - MULTIBANCAS

1. (VUNESP – 2012 – TJ/SP - Analista Judiciário - Tecnologia da Informação) Considerando o conceito de Complexidade de Algoritmos, representado por O(função), assinale a alternativa que apresenta, de forma crescente, as complexidades de algoritmos.

a) O(2n); O(n3); O(n2); O(log2 n); O(n.log2 n).

b) O(n2); O(n3); O(2n); O(log2 n); O(n.log2 n).

c) O(n3); O(n2); O(2n); O(n.log2 n); O(log2 n).

d) O(log2 n); O(n.log2 n); O(n2); O(n3); O(2n).

e) O(n.log2 n); O(log2 n); O(2n); O(n3); O(n2).

2. (FCC - 2010 - TRT - 8ª Região (PA e AP) - Analista Judiciário - Tecnologia da Informação) Numa competição de programação, ganhava mais pontos o time que apresentasse o algoritmo mais eficiente para resolver o pior caso de um determinado problema. A complexidade assintótica (notação Big O) dos algoritmos elaborados está ilustrada na tabela abaixo.

Time	Complexidade
Branco	$O(n^{20})$
Amarelo	O(nlogn)
Azul	O(1)
Verde	O(n!)
Vermelho	$O(2^n)$

O time que obteve a medalha de prata (2º algoritmo mais eficiente) é o:

a) Branco.

b) Amarelo.

c) Azul.

d) Verde.

e) Vermelho.

GABARITO

1. D
2. B

LISTA DE QUESTÕES – PESQUISA DE DADOS - MULTIBANCAS

1. (CESPE - 2013 – TRT/MS – Analista de Sistemas) Considerando que se deseje efetuar uma pesquisa de um valor sobre a chave primária de uma tabela de um banco de dados com uma chave primária com um tipo de campo que receba um valor inteiro e que se possa fazer essa pesquisa utilizando-se a busca sequencial ou a busca binária, assinale a opção correta.

   a) O método de busca binária requer, no máximo, ln(n) comparações para determinar o elemento pesquisado, em que n é o número de registros.

   b) O método de busca binária será sempre mais rápido que o método de busca sequencial, independentemente de a tabela estar ordenada com base no elemento pesquisado.

   c) O método de busca sequencial requererá, no máximo, n^2 comparações para determinar o elemento pesquisado, em que n será o número de registros.

   d) O método de busca binária sempre efetuará menos comparações que o método de pesquisa sequencial.

   e) O método de busca sequencial efetuará menos comparações para encontrar o elemento pesquisado quando a tabela estiver ordenada em comparação à situação quando a tabela estiver desordenada.

2. (ESAF - 2001 – BACEN – Analista de Sistemas) Na pior hipótese, o número de comparações necessárias para pesquisar um elemento em um array de 2048 elementos pelo método de pesquisa binária será:

   a) 8

   b) 9

   c) 10

   d) 11

   e) 12

3. (CESPE - 2013 – TCE/RO – Analista de Sistemas) Considere uma tabela de um banco de dados com chave primária e tipo de campo que receba um valor inteiro. Ao se efetuar uma pesquisa de um valor sobre a chave primária dessa tabela, o método de busca binária requer, no máximo, lg(n) comparações para localizar o elemento pesquisado, em que n é o número de registros.

4. (FCC – 2016 - TRT - 14ª Região (RO e AC) - Analista Judiciário - Tecnologia da Informação) Dada uma coleção de n elementos ordenados por ordem crescente, pretende-se saber se um determinado elemento x existe nessa coleção. Supondo que essa coleção está implementada como sendo um vetor a[0...n-1] de n elementos inteiros, utilizando-se um

algoritmo de pesquisa binária, o número de vezes que a comparação x==a[i] será executada, no pior caso, é calculada por:

a) n/2.

b) n−1.

c) √n.

d) log2(n).

e) n−=2.

GABARITO

1. A
2. D
3. C
4. D

Estrutura de Dados

Pessoal, um programa pode ser visto como uma especificação formal da solução de um problema. Wirth expressa esse conceito por meio de uma **equação**:

PROGRAMA = ALGORITMO + ESTRUTURA DE DADOS

Nosso foco aqui é em Estruturas de Dados! Na evolução do mundo computacional, um fator extremamente importante trata da forma de **armazenar informações**. De nada adianta o enorme desenvolvimento de hardware e software se a forma de armazenamento e tratamento de dados não evoluir harmonicamente. E é por isso que as estruturas de dados são tão fundamentais.

As estruturas de dados, na maioria dos casos, baseiam-se nos tipos de armazenamento vistos dia a dia, i.e., nada mais são do que a transformação de uma forma de armazenamento já conhecida e utilizada no mundo real adaptada para o mundo computacional. Por isso, cada tipo de estrutura de dados possui **vantagens e desvantagens** e cada uma tem sua **área de atuação otimizada**.

Bem, não vou enrolar muito explicando o que é uma Estrutura de Dados! A melhor forma de saber é vendo exemplos. Antes disso, eu gostaria de falar sobre um conceito importante: **Dados Homogêneos e Heterogêneos.** Os primeiros são aqueles que possuem só um tipo básico de dados (Ex: Inteiros); os segundos são aqueles que possuem mais de um tipo básico de dados (Ex: Inteiros + Caracteres). Os tipos básicos de dados também são chamados de tipos primitivos.

Entenderam? Existem estruturas de dados que tratam de dados homogêneos, i.e., todos os dados são apenas de um tipo básico, tais como **Vetores!** Ora, em um vetor, todos os elementos são do mesmo tipo. Existem estruturas de dados que tratam de dados heterogêneos, i.e., os dados são de tipos básicos diferentes, tais como Listas! Ora, em uma lista, todos os elementos são, em geral, de tipos básicos diferentes.

Além dessa classificação, existe outra também importante: **Estruturas Lineares e Estruturas Não-Lineares.** As Estruturas Lineares são aquelas em que cada elemento pode ter um único predecessor (exceto o primeiro elemento) e um único sucessor (exceto o último elemento). Como exemplo, podemos citar Listas, Pilhas, Filas, Arranjos, entre outros.

Já as Estruturas Não-Lineares são aquelas em que cada elemento pode ter mais de um predecessor e/ou mais de um sucessor. Como exemplo, podemos citar Árvores, Grafos e Tabelas de Dispersão. Essa é uma classificação muito importante e muito simples de entender. Pessoal,

vocês perceberão que esse assunto é cobrado de maneira superficial na maioria das questões, mas algumas são nível doutorado!

Por fim, vamos falar sobre Tipos Abstratos de Dados (TAD). Podemos defini-lo como um modelo matemático (v,o), em que v é um conjunto de valores e o é um conjunto de operações que podem ser realizadas sobre valores. Eu sei, essa definição é horrível de entender! Para compreender esse conceito, basta lembrar de abstração, i.e., apresentar interfaces e esconder detalhes.

Os Tipos Abstratos de Dados são simplesmente um modelo para um certo tipo de estrutura de dados. Como assim, professor? Quando eu falo em pilha, eu estou falando de um tipo abstrato de dados que tem duas operações com comportamentos bem definidos e conhecidos: push (para inserir elementos na pilha); e pop (para retirar elementos da pilha).

E a implementação dessas operações? Isso não importa! Aliás, não importa implementação nem paradigma nem linguagem de programação. Não importa se a pilha é implementada com um paradigma orientado a objetos ou com um paradigma estruturado; não importa se a pilha é implementada em Java ou Pascal; não importa como é a implementação interna – isso serve para outras estruturas[1].

Podemos concluir, portanto, que um tipo abstrato de dados contém um modelo que contém valores e operações associadas, de forma que essas operações sejam precisamente independentes de uma implementação particular. Em geral, um TAD é especificado por meio de uma especificação algébrica que, em geral, contém três partes: Especificação Sintática, Semântica e de Restrições.

A Especificação Sintática define o nome do tipo, suas operações e o tipo dos argumentos das operações, definindo a assinatura do TAD. A Especificação Semântica descreve propriedades e efeitos das operações de forma independente de uma implementação específica. E a Especificação de Restrições estabelece as condições que devem ser satisfeitas antes e depois da aplicação das operações.

Em outras palavras, o nível semântico trata do comportamento de um tipo abstrato de dados; e o nível sintático trata da apresentação de um tipo abstrato de dados. Podemos dizer, então, que o TAD encapsula uma estrutura de dados com características semelhantes – podendo ser formado por outros TADs –, e esconde a efetiva implementação dessa estrutura de quem a manipula.

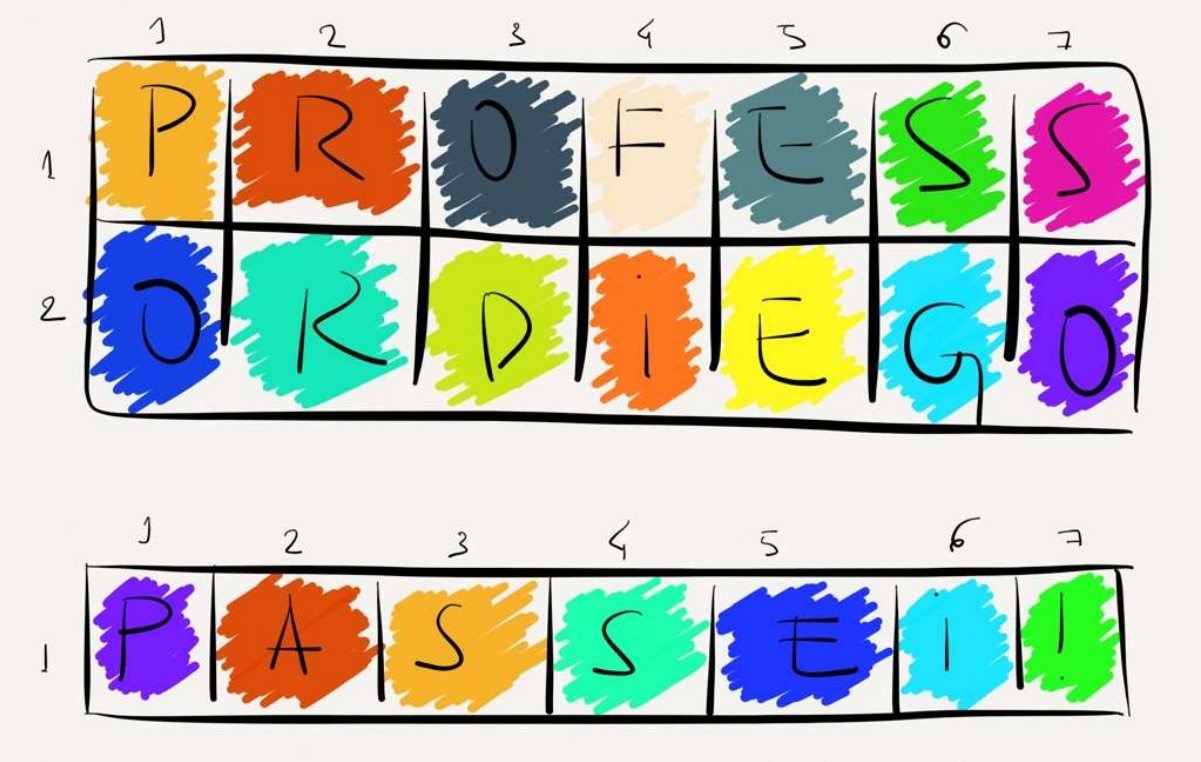
[1] Filas, Pilhas, Árvores, Deques, entre outros.

VETORES E MATRIZES

Um Vetor é uma estrutura de **dados linear** que necessita de somente **um índice** para que seus elementos sejam **indexados.** É uma estrutura homogênea, portanto armazena somente uma lista de valores do mesmo tipo. Ele pode ser estático ou dinâmico, com dados armazenados em posições contíguas de memória e permite o acesso direto ou aleatório a seus elementos.

Observem que, diferentemente das listas, filas e pilhas, ele vem praticamente **embutido** em qualquer **linguagem de programação.** *E a Matriz, professor?* Não muda muita coisa! Trata-se de um arranjo bidimensional ou multidimensional de alocação estática e sequencial. Ela necessita de um índice para referenciar a linha e outro para referenciar a coluna para que seus elementos sejam endereçados.

Da mesma forma que um vetor, uma matriz também pode ter tamanhos variados, todos os elementos são do mesmo tipo, cada célula contém somente um valor e os tamanhos dos valores são os mesmos. Os elementos ocupam posições contíguas na memória. A alocação dos elementos pode ser feita colocando os elementos **linha-por-linha ou coluna-por-coluna**.



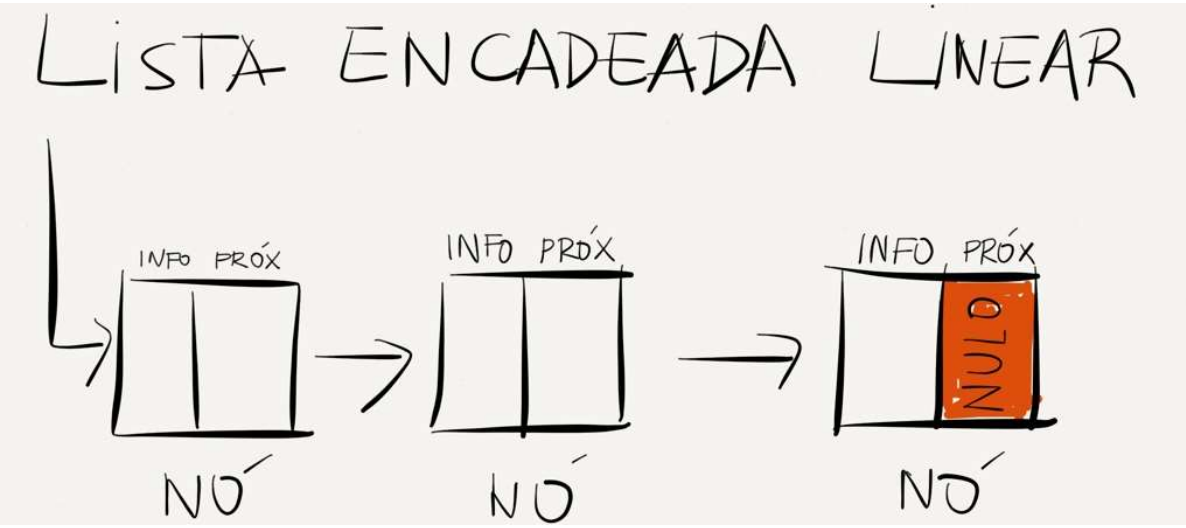
Matriz 2x7 e Vetor (7 Posições)

LISTA ENCADEADA

Também conhecida como Lista Encadeada Linear, Lista Ligada Linear ou Linked List, trata-se de uma estrutura de dados dinâmica formada por uma sequência encadeada de elementos chamados nós, que contêm dois campos: **campo de informação** e **campo de endereço**. O primeiro armazena o real elemento da lista e o segundo contém o endereço do próximo nó da lista.

Esse endereço, que é usado para acessar determinado nó, é conhecido como **ponteiro**. A lista ligada inteira é acessada a partir de um ponteiro externo que aponta para o primeiro nó na lista, i.e., contém o endereço do primeiro nó[1]. Por ponteiro "externo", entendemos aquele que não está incluído dentro de um nó. Em vez disso, seu valor pode ser acessado diretamente, por referência a uma variável.

O campo do próximo endereço do último nó na lista contém um valor especial, conhecido como **NULL**, que não é um endereço válido. Esse ponteiro nulo é usado para indicar o final de uma lista. Uma lista é chamada Lista Vazia ou Lista Nula caso não tenha nós ou tenha apenas um nó sentinela. O valor do ponteiro externo para esta lista é o ponteiro nulo. Uma lista pode ser inicializada com uma lista vazia.

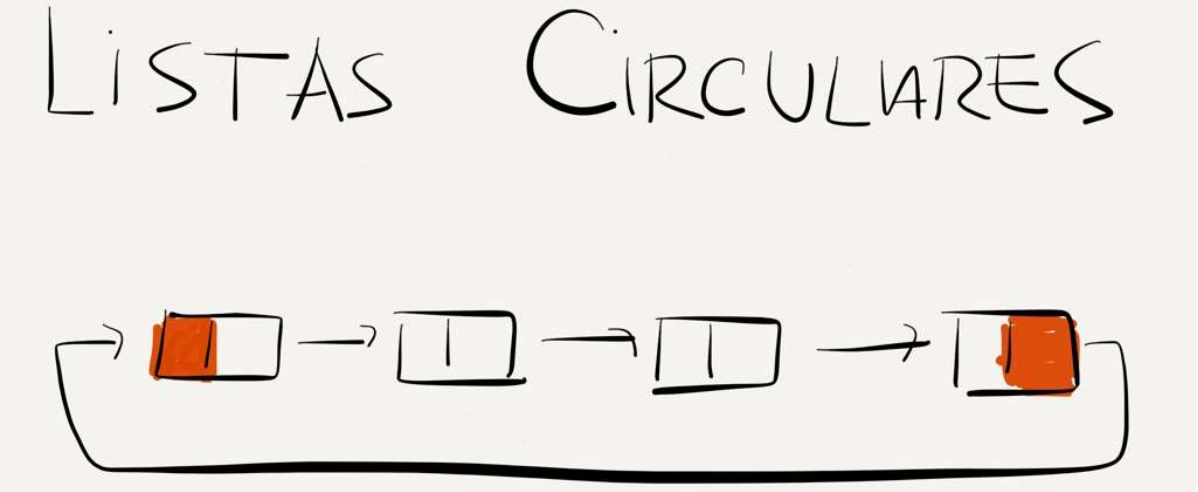


Suponha que seja feita uma mudança na estrutura de uma lista linear, de modo que o campo próximo no último nó contenha um ponteiro de volta para o primeiro nó, em vez de um ponteiro nulo. Esse tipo de lista é chamado **Lista Circular** (ou Fechada[2]), i.e., a

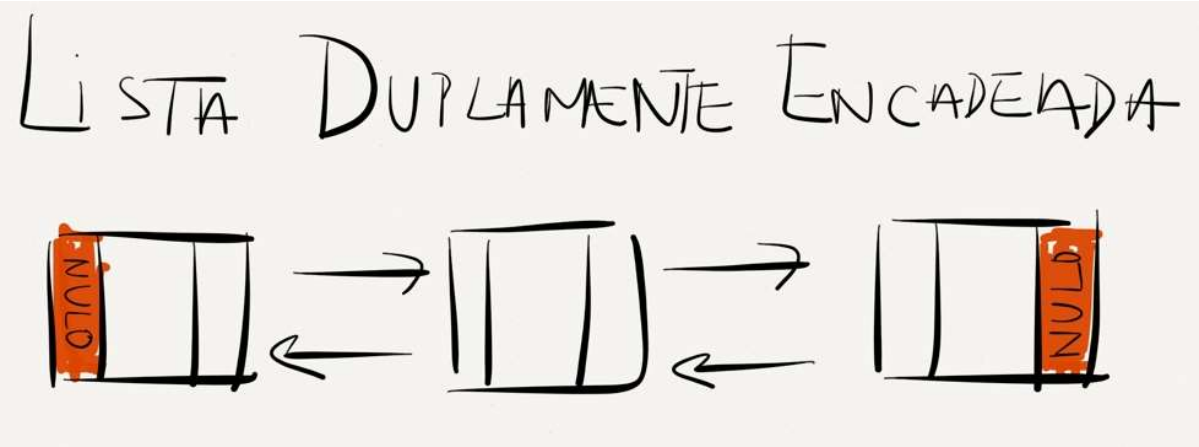
[1] O endereço do primeiro nó pode ser encapsulado para facilitar possíveis futuras operações sobre a lista sem a necessidade de se conhecer sua estrutura interna. O primeiro elemento e o último nós são muitas vezes chamados de *Sentinela*.

[2] Se Listas Circulares são conhecidas como Listas Fechadas, as Listas Abertas são todas aquelas que são Não-Circulares. Por fim: da mesma forma que há Listas Circulares Simples, há também Listas Circulares Duplas. Nesse caso, o ponteiro anterior do primeiro elemento aponta para o último elemento e o ponteiro posterior do último elemento aponta para o primeiro elemento.

partir de qualquer ponto, é possível atingir qualquer outro ponto da lista. Certinho, até agora?



Observe que uma Lista Circular não tem um primeiro ou último nó natural. Precisamos, portanto, estabelecer um **primeiro** e um **último nó** por convenção. Uma convenção útil é permitir que o ponteiro externo para a lista circular aponte para o último nó, e que o nó seguinte se torne o primeiro nó. Assim podemos incluir ou remover um elemento convenientemente a partir do início ou do final de uma lista.



Embora uma lista circularmente ligada tenha vantagens sobre uma lista linear, ela ainda apresenta várias deficiências. Não se pode atravessar uma lista desse tipo no sentido contrário nem um nó pode ser eliminado de uma lista circularmente ligada sem se ter um ponteiro para o nó antecessor. Nos casos em que tais recursos são necessários, a estrutura de dados adequada é uma **lista duplamente ligada**.

Cada nó numa lista desse tipo contém **dois ponteiros**, um para seu **predecessor** e outro para seu **sucessor.** Na realidade, no contexto de listas duplamente ligadas, os termos predecessor e sucessor não fazem sentido porque a lista é totalmente simétrica. As listas

duplamente ligadas podem ser lineares ou circulares e podem conter ou não um nó de cabeçalho.

Podemos considerar os nós numa lista duplamente ligada como consistindo em três campos: um campo info que contém as informações armazenadas no nó, e os campos left e right, que contêm ponteiros para os nós em ambos os lados. Dado um ponteiro para um elemento, pode-se acessar os elementos adjacentes e, dado um ponteiro para o último elemento, pode-se percorrer a lista em **ordem inversa**.

Existem **cinco** operações básicas sobre uma lista encadeada: Criação, em que se cria a lista na memória; Busca, em que se pesquisa nós na lista; Inclusão, em que se insere novos nós na lista em uma determinada posição; Remoção, em que se elimina um elemento da lista; e, por fim, Destruição, em que se destrói a lista junto com todos os seus nós.

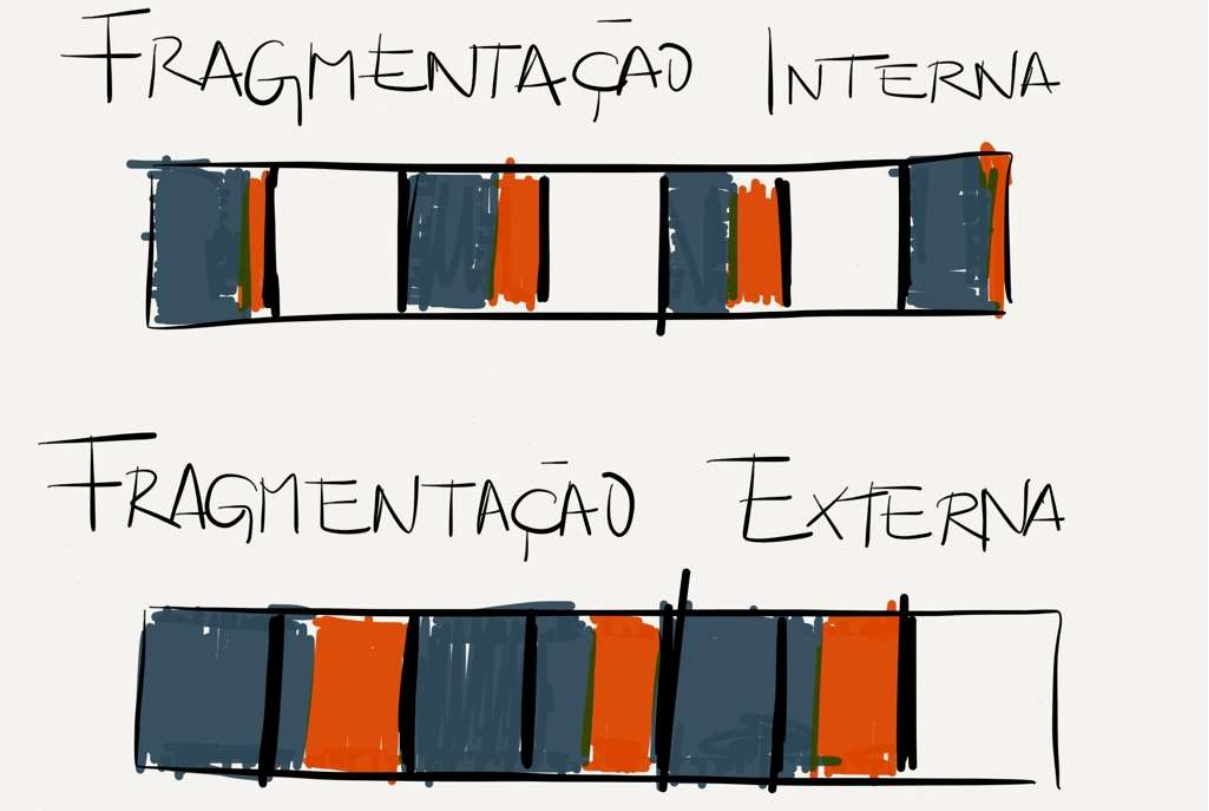
IMPORTANTE

Pilhas e Filas são subespécies de Listas. No entanto, cuidado na hora de responder questões! De maneira genérica, Pilhas e Filas podem ser implementadas como Listas. No entanto, elas possuem características particulares de uma lista genérica. Ok?

Precisamos falar um pouco sobre Fragmentação! O que é isso, professor? Galera, falou em **fragmentação**, lembrem-se de **desperdício** de espaço disponível de **memória.** O fenômeno no qual existem vários blocos disponíveis pequenos e não-contíguos é chamado fragmentação externa porque o espaço disponível é desperdiçado fora dos blocos alocados.

Esse fenômeno é o oposto da fragmentação interna, no qual o espaço disponível é desperdiçado dentro dos blocos alocados, como apresenta a imagem abaixo. Sistemas Operacionais possuem uma estrutura de dados que armazena informações sobre áreas ou blocos livres (geralmente uma lista ou tabela). Uma lista encadeada elimina o problema da **fragmentação externa.** Por que?

Porque mantém os arquivos, cada um, como uma lista encadeada de blocos de disco. Dessa forma, uma parte de cada bloco é usada como ponteiro para o próximo bloco e o restante do bloco é usado para dados. Uma vantagem desse tipo de alocação é que o tamanho do arquivo não precisa ser conhecido antes de sua criação, já que cada bloco terá um ponteiro para o próximo bloco.



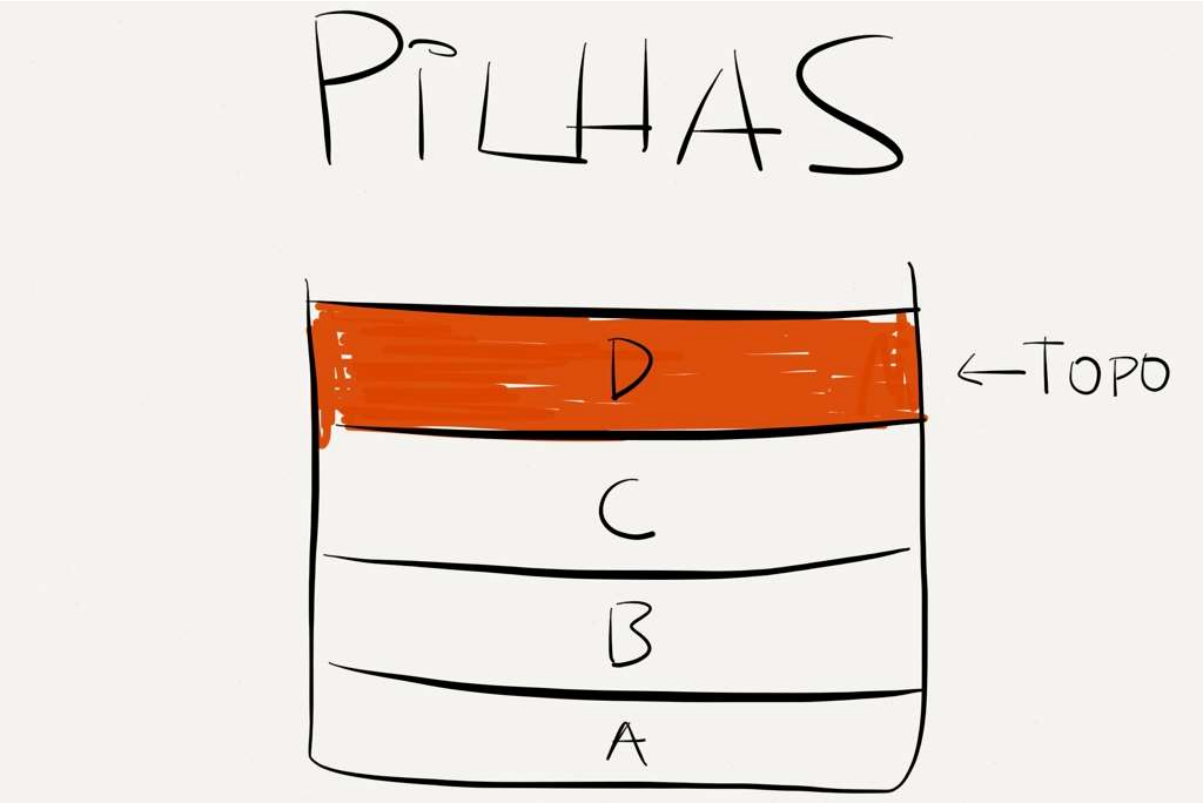
Galera... e o acesso a uma lista? A Lista é uma estrutura de **acesso sequencial, i.e.,** é preciso percorrer nó por nó para acessar um **dado específico**. Logo, é proporcional ao número de elementos – Acesso O(n). E os Vetores? Eles são uma estrutura de acesso direto, i.e., pode-se acessar um elemento diretamente. Portanto, não precisa percorrer elemento por elemento (Acesso O(1))[3].

[3] No Acesso Sequencial: quanto mais ao fim, maior o tempo para acessar; no Acesso Direto: todos os elementos são acessados no mesmo tempo.

PILHAS

A Pilha é um **conjunto ordenado** de itens no qual novos itens podem ser inseridos e eliminados em uma extremidade chamada **topo**. Novos itens podem ser colocados no topo da pilha (tornando-se o novo primeiro elemento) ou os itens que estiverem no topo da pilha poderão ser removidos (tornando-se o elemento mais abaixo o novo primeiro elemento).

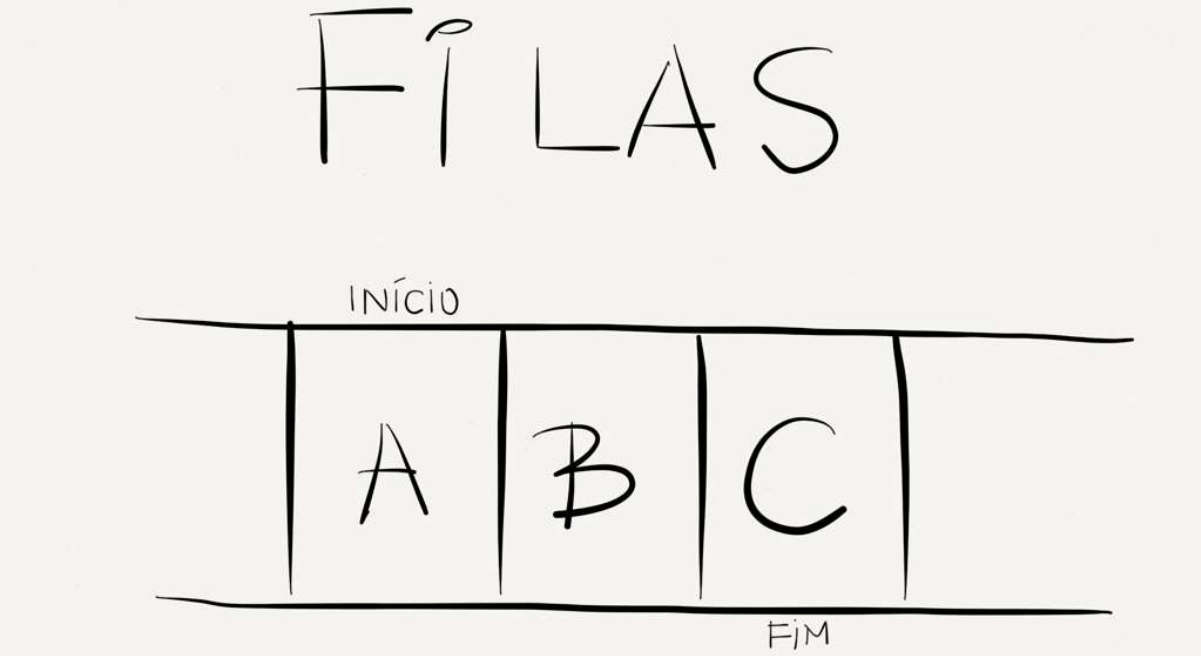
Também conhecida como **Lista LIFO** (Last In First Out), basta lembrar de uma pilha de pratos esperando para serem lavados, i.e., o último a entrar é o primeiro a sair. A ordem em que os pratos são retirados da pilha é o oposto da ordem em que eles são colocados sobre a pilha e, como consequência, apenas o prato do topo da pilha está acessível.



As Pilhas oferecem três operações básicas: push, que insere um novo elemento no topo da pilha; pop, que remove um elemento do topo da pilha; e top (também conhecida como check), que acessa e consulta o elemento do topo da pilha. Pilhas podem ser implementadas por meio de **Vetores** (Pilha Sequencial - Alocação Estática de Memória) ou **Listas** (Pilha Encadeada - Alocação Dinâmica de Memória).

FILAS

Uma fila é um conjunto **ordenado** de itens a partir do qual podem-se **eliminar** itens numa extremidade (chamada início da fila) e no qual podem-se **inserir** itens na outra extremidade (chamada final da fila). Também conhecida como Lista FIFO (First In First Out), basta lembrar de uma fila de pessoas esperando para serem atendidas em um banco, i.e., o primeiro a entrar é o primeiro a sair.



Quando um elemento é colocado na fila, ele ocupa seu lugar no fim da fila, como um aluno recém-chegado que ocupa o final da fileira. O elemento retirado da fila é sempre aquele que está no início da fila, como o aluno que se encontra no começo da fileira e que esperou mais tempo. As operações básicas são **Enqueue** (Enfileirar) e **Dequeue** (Desenfileirar). As Filas possuem **início** (ou cabeça) e **fim** (ou cauda).

É bom salientar outro conceito importante: Deque (Double Ended Queue)! É também conhecida como Filas Duplamente Encadeadas e permite a **eliminação e inserção de** itens em ambas as extremidades. Ademais, elas permitem algum tipo de priorização, visto que é possível inserir elementos de ambos os lados. Assim sendo, é comum em sistemas distribuídos!

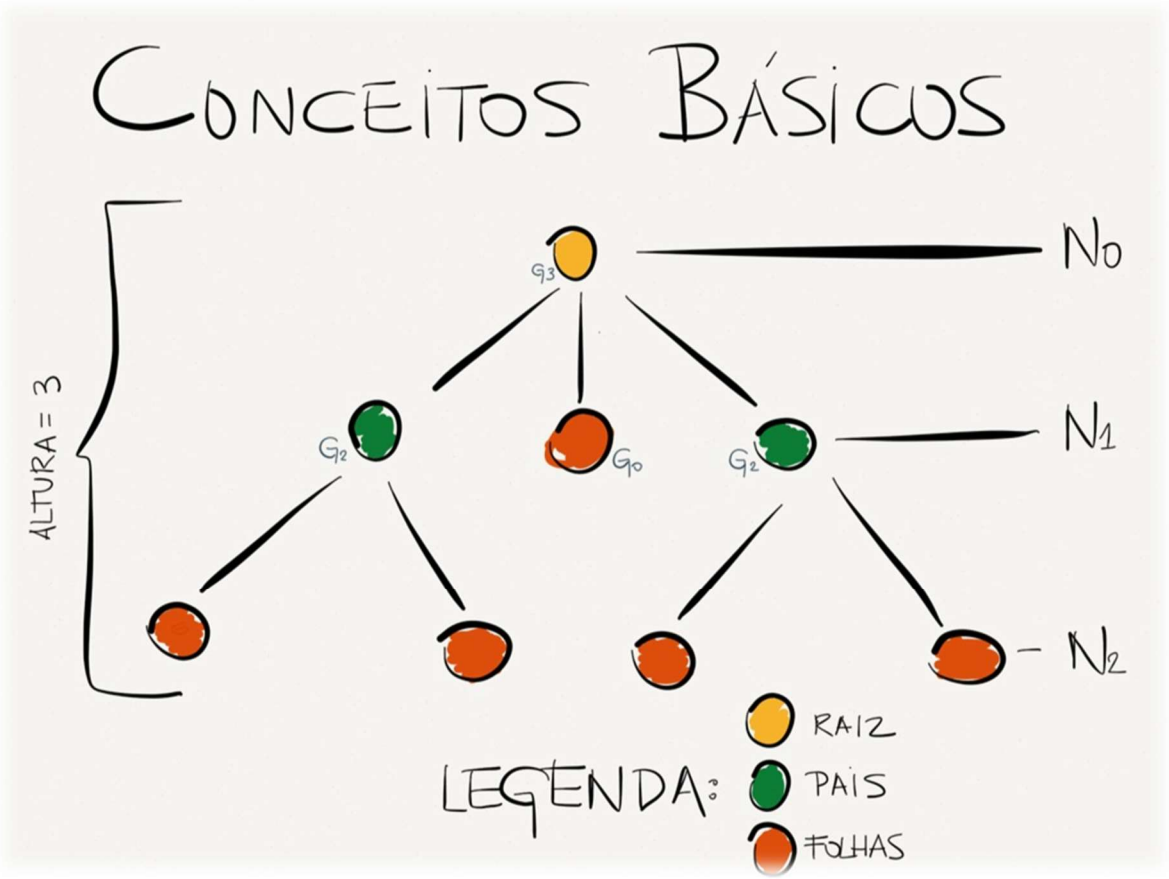
Sistemas distribuídos sempre necessitam que algum tipo de processamento seja mais rápido, por ser mais prioritário naquele momento, deixando outros tipos mais lentos ou em fila de espera, por não requerem tanta pressa. Ele pode ser entendido como uma **extensão da estrutura** de dados Fila. Agora uma pergunta: Qual a diferença entre uma lista duplamente encadeada e um deque?

Pessoal, um deque gerencia elementos como um vetor, fornece **acesso aleatório** e tem quase a mesma interface que um **vetor.** Ele se diferencia de uma lista duplamente encadeada, entre outras coisas, por essa não fornecer acesso aleatório aos elementos, i.e.,

para acessar o quinto elemento, você deve navegar pelos quatro primeiros elementos – logo a lista é mais lenta nesse sentido. Bacana?

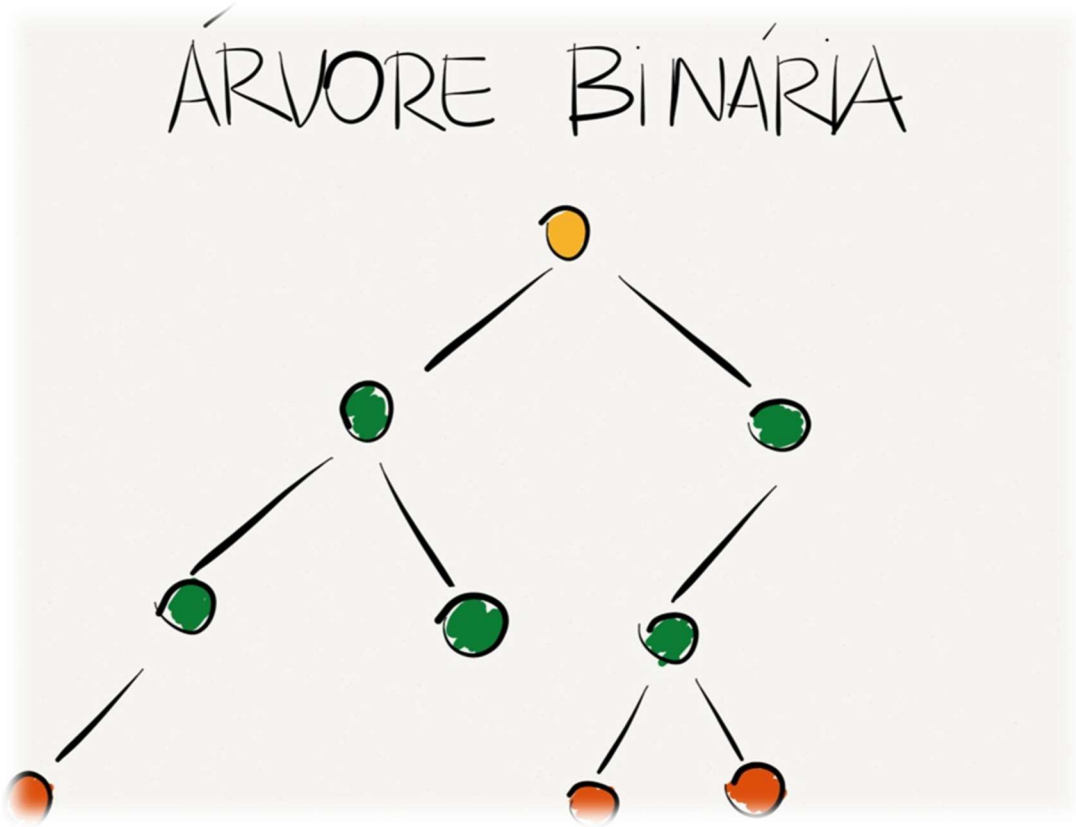
ESTRUTURAS DE DADOS: ÁRVORE

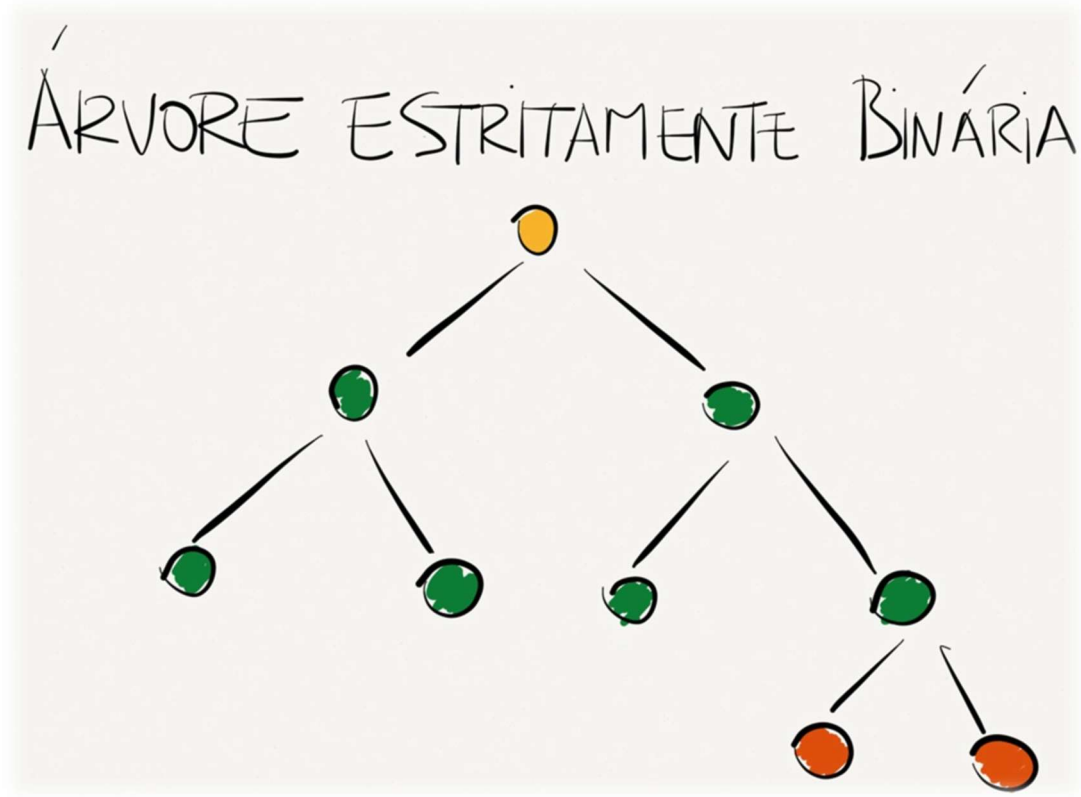
Uma árvore é uma estrutura de **dados hierárquica** (não-linear) composta por um **conjunto finito** de elementos com um único elemento raiz, com zero ou mais sub-árvores ligadas a esse elemento raiz. Como mostra a imagem abaixo, há uma única raiz, em amarelo. Há também nós folhas, em vermelho e seus pais, em verde. Observem ainda os conceitos de Altura, Grau e Nível de uma árvore.

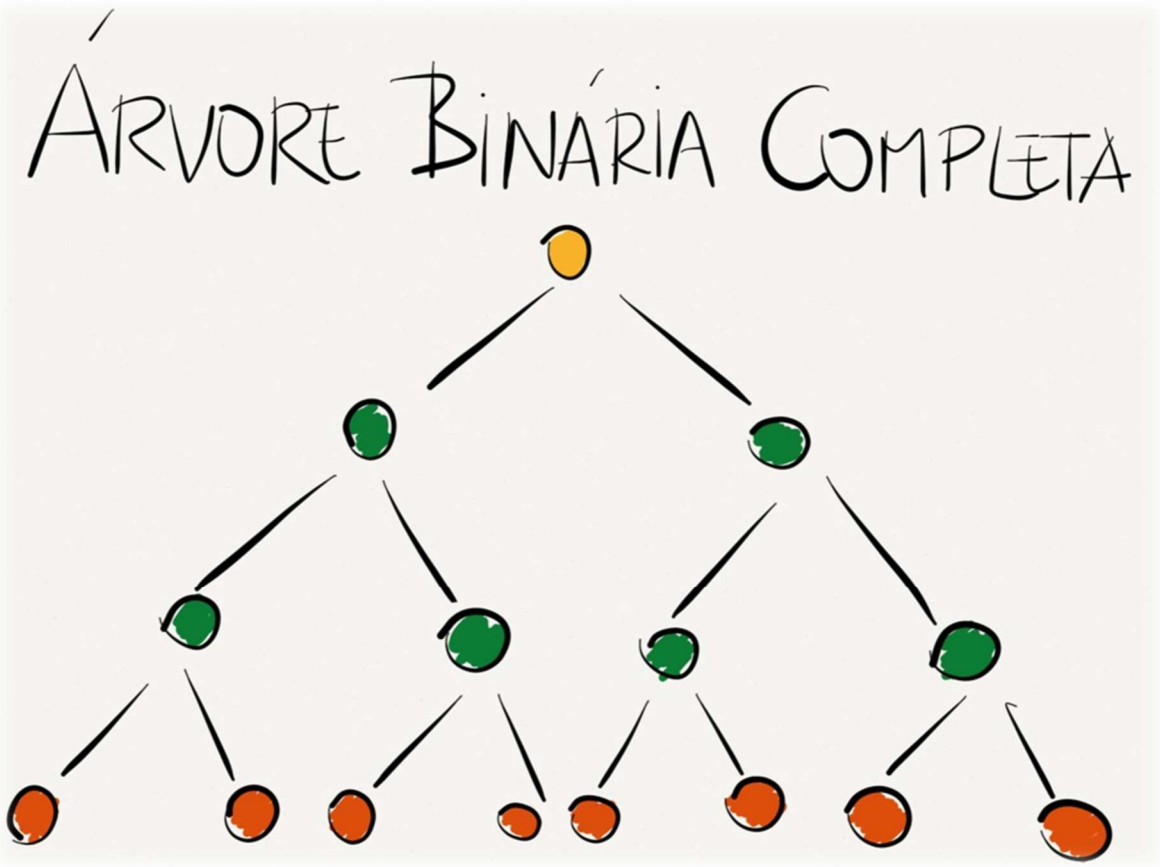


O Grau informa a quantidade de filhos de um determinado nó! A Raiz tem Nível 0 (excepcionalmente, alguns autores consideram que tem Nível 1) e o nível de qualquer outro nó na árvore é um nível a mais que o nível de seu pai. Por fim, a Altura é a distância entre a raiz e seu descendente mais afastado. Dessas informações, podemos concluir que toda folha tem **Grau 0.**

Existe um tipo particular de árvore chamado: **Árvore Binária!** O que é isso? É uma estrutura de dados hierárquica em que todos os nós têm grau 0, 1 ou 2. Já uma Árvore Estritamente Binária é aquela em que todos os nós têm grau **0 ou 2.** E uma Árvore Binária Completa é aquela em que todas as folhas estão no mesmo nível, como mostram as imagens abaixo.


ÁRVORE BINÁRIA


ÁRVORE ESTRITAMENTE BINÁRIA



Uma Árvore Binária Completa com x folhas conterá sempre (2x – 1) nós. Observem a imagem acima e façam as contas: 2*8 – 1 = 15 nós! Uma árvore binária completa de **altura h e nível n** contém **(2^h-1)** ou **$(2^{n+1}-1)$** nós e usa-se **(2^n),** para calcular a quantidade de nós em determinado nível. Na imagem acima, há uma árvore de h = 4 e n = 3; logo, existem 2^{3+1} -1 = 15 nós no total; e no Nível 3, existe 2^3 = 8 nós.

Vamos falar um pouco agora sobre Árvore de Busca Binária! Trata-se de uma Uma estrutura de dados vinculada, baseada em nós, onde cada nó contém uma chave e duas subárvores à esquerda e a direita. Para todos nós, a chave da subárvore esquerda deve ser menor que a chave desse nó, e a chave da subárvore direita deve ser maior. Beleza?

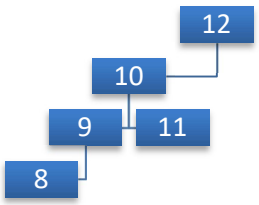
Todas estas subárvores devem qualificar-se como árvores binárias de busca. O pior caso de tempo de complexidade para a pesquisa em uma árvore binária de busca é a altura da árvore, isso pode ser tão pequeno como O(log n) para uma árvore com n elementos. Galera, abaixo nós vamos ver como árvores podem ser representadas e o conceito de árvore binária de busca ficará mais clara!

Como representamos árvores? Podemos representar uma árvore como um conjunto de **parênteses aninhados.** Nessa notação, (P (F_1)(F_2)) significa que P, F_1, F_2 são nós e que F_1, F_2, são filhos do pai P. Ao transcrever isso para o desenho hierárquico de uma árvore, lemos da esquerda para a direita. Agora suponhamos que F_1 tem dois filhos N_1 e N_2. Logo, reescrevemos a subárvore de F_1 como (F_1 (N_1)(N_2)).

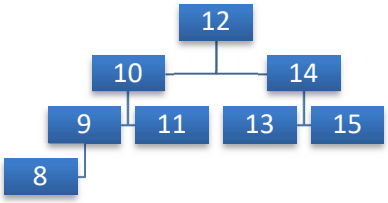
Podemos, então, substituir F_1 por $(F_1(N_1)(N_2))$. Ao final, ficará $(P(F_1(N_1)(N_2))(F_2))$. E assim por diante. Temos então que o primeiro elemento é a raiz e sempre que tivermos um parêntese, teremos uma nova **subárvore**. Vamos exemplificar: (12(10(9(8))(11))(14(13)(15))). Como ficaria, professor? Sabemos que 12 será a raiz dessa árvore e, a partir daí, criamos a árvore da esquerda para direita.

Observem o parêntese após o 12! Notem que ele só é fechado após o 11: (12(10(9(8))(11)). Isso significa que tudo que está em vermelho é subárvore da esquerda da raiz 12 – e o restante (14(13)(15)) é subárvore da direita da raiz 12. Observem o parêntese após o 10! Notem que ele só é fechado após o 8: 10(9(8)). Isso significa que tudo que está em amarelo é subárvore da esquerda da raiz 10.

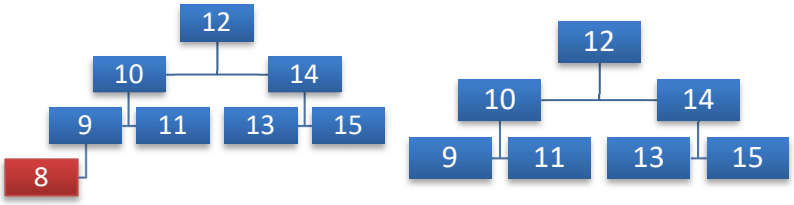
E o restante (11) é subárvore da direita da raiz 10. Observem o parêntese após o 9! Notem que ele só é fechado após o 8: 9(8). Isso significa que tudo que está em verde é subárvore da esquerda da raiz 9 – e o restante... que restante, professor? Pois é, não tem restante! Logo, 9 não tem subárvore da direita. Vejam abaixo como ficou e vamos analisar o outro lado.



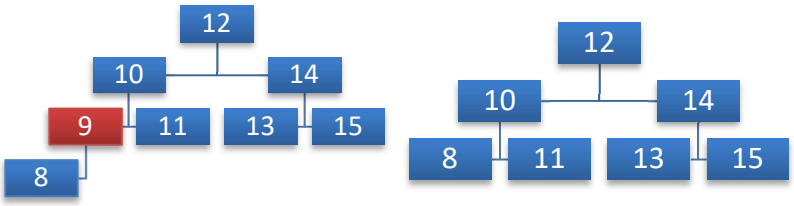
A subárvore da direita da raiz 12 tem raiz 14 e tem dois filhos: na esquerda, 13 e na direita 15. Fim! Galera, eu sei que parece complicado, mas leiam e pratiquem umas três vezes – de preferência no papel - que vocês internalizam tranquilamente esse conteúdo. É chatinho, mas não é difícil! Vejam abaixo como ficou o resultado final e vamos seguir em frente...



Bem, pessoal! Dito isso, vamos analisar agora como ficaria para remover um nó desta árvore. Existem três possibilidades para realizar essa operação: (1) remover um nó que não tem filhos; (2) remover um nó que tem apenas um filho; (3) e remover um nó que tenha dois filhos. O primeiro caso é muito simples: basta retirar o nó desejado e ponto final. Vejamos como fica:



No segundo caso, basta retirar o nó da árvore e conectar seu único filho (e sua subárvore, se houver) diretamente ao pai do nó removido. Vejamos:



Já o último caso, nós podemos utilizar duas estratégias. Você pode escolher qual deseja utilizar em uma situação específica. Vejamos:

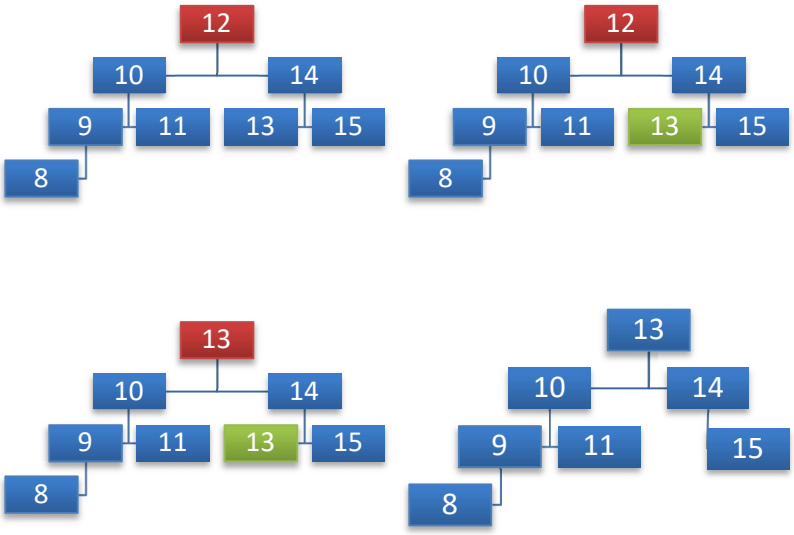
ESTRATÉGIA 1

PASSO 1: IDENTIFIQUE O ELEMENTO QUE VOCÊ DESEJA RETIRAR DA ÁRVORE (EM VERMELHO)

PASSO 2: IDENTIFIQUE O MENOR ELEMENTO DE TODA SUBÁRVORE À DIREITA DO NÓ IDENTIFICADO NO PASSO 1 (EM VERDE)

PASSO 3: COPIE O VALOR DO NÓ IDENTIFICADO NO PASSO 2 PARA O NÓ IDENTIFICADO NO PASSO 1

PASSO 4: REMOVA O ELEMENTO IDENTIFICADO NO PASSO 2.



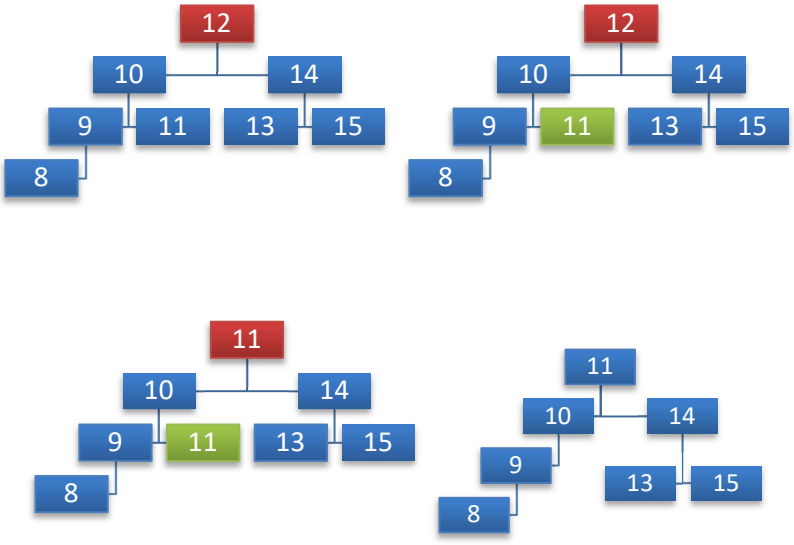
ESTRATÉGIA 2

PASSO 1: IDENTIFIQUE O ELEMENTO QUE VOCÊ DESEJA RETIRAR DA ÁRVORE (EM VERMELHO)

PASSO 2: IDENTIFIQUE O MAIOR ELEMENTO DE TODA SUBÁRVORE À ESQUERDA DO NÓ IDENTIFICADO NO PASSO 1 (EM VERDE)

PASSO 3: COPIE O VALOR DO NÓ IDENTIFICADO NO PASSO 2 PARA O NÓ IDENTIFICADO NO PASSO 1

PASSO 4: REMOVA O ELEMENTO IDENTIFICADO NO PASSO 2.



Por fim, como seria a representação por parênteses aninhados? Na primeira estratégia, temos: (13(10(9(8))(11))(14(15))); na segunda, temos (11(10(9(8)))(14(13)(15))).

Galera, em uma Árvore de Busca Binária, podemos fazer três percursos: **pré-ordem, in-ordem e pós-ordem** (esses prefixos são em relação a raiz). É interessante notar que, quando se faz um percurso em ordem em uma árvore binária de busca, os valores dos nós aparecem em ordem crescente. A operação "Percorre" tem como objetivo percorrer a árvore numa dada ordem, enumerando os seus nós.

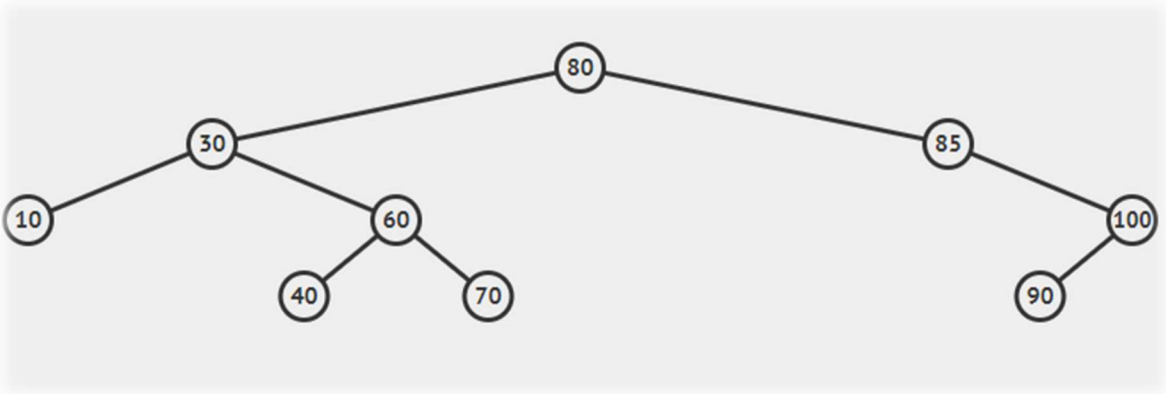
Quando um nó é enumerado, diz-se que ele foi **"visitado"**. Vamos ver agora esses três percursos:

- **Pré-Ordem (ou Profundidade):** visita a raiz; percorre a subárvore esquerda em pré-ordem; percorre a subárvore direita em pré-ordem.

- **In-Ordem (ou Simétrica):** percorre a subárvore esquerda em in-ordem; visita a raiz; percorre a subárvore direita em in-ordem.

- **Pós-Ordem:** percorre a subárvore esquerda em pós-ordem; percorre a subárvore direita em pós-ordem; visita a raiz.

Vamos ver um exemplo:



Como ler essa árvore em Pré-Ordem? Vamos tentar...

```
//Percorrendo a Árvore em Pré-Ordem:


1. Visite a Raiz: {80};
```

2. Percorra a subárvore esquerda em pré-ordem:
    1. Visite a Raiz: {30};
    2. Percorra a subárvore esquerda em pré-ordem:
        1. Visite a Raiz: {10};
        2. Percorra a subárvore esquerda em pré-ordem: {Vazio}
        3. Percorra a subárvore direita em pré-ordem: {Vazio}
    3. Percorra a subárvore direita em pré-ordem:
        1. Visite a Raiz: {60};
        2. Percorra a subárvore esquerda em pré-ordem:
            1. Visite a Raiz: {40}
            2. Percorra a subárvore esquerda em pré-ordem: {Vazio}
            3. Percorra a subárvore direita em pré-ordem: {Vazio}
        3. Percorra a subárvore direita em pré-ordem:
            1. Visite a Raiz: {70}
            2. Percorra a subárvore esquerda em pré-ordem: {Vazio}
            3. Percorra a subárvore direita em pré-ordem: {Vazio}
3. Percorra a subárvore direita em pré-ordem:
    1. Visite a Raiz: {85};
    2. Percorra a subárvore esquerda em pré-ordem: {Vazio}
    3. Percorra a subárvore direita em pré-ordem: {Vazio}
        1. Visite a Raiz: {100};
        2. Percorra a subárvore esquerda em pré-ordem:
            1. Visite a Raiz: {90};
            2. Percorra a subárvore esquerda em pré-ordem: {Vazio}
            3. Percorra a subárvore direita em pré-ordem: {Vazio}

3. Percorra a subárvore direita em pré-ordem: {Vazio}

RESULTADO: 80, 30, 10, 60, 40, 70, 85, 100, 90

//Percorrendo a Árvore em In-Ordem:

1. Percorra a subárvore esquerda em in-ordem:
    1. Percorra a subárvore esquerda em in-ordem:
        1. Percorra a subárvore esquerda em in-ordem: {Vazio}
        2. Visite a Raiz: {10}
        3. Percorra a subárvore direita em in-ordem: {Vazio}
    2. Visite a Raiz: {30}
    3. Percorra a subárvore direita em in-ordem:
        1. Percorra a subárvore esquerda em in-ordem:
            1. Percorra a subárvore esquerda em in-ordem: {Vazio}
            2. Visite a Raiz: {40}
            3. Percorra a subárvore direita em in-ordem: {Vazio}
        2. Visite a Raiz: {60}
        3. Percorra a subárvore direita em in-ordem:
            1. Percorra a subárvore esquerda em in-ordem: {Vazio}
            2. Visite a Raiz: {70}
            3. Percorra a subárvore direita em in-ordem: {Vazio}
2. Visite a Raiz: {80}
3. Percorra a subárvore direita em in-ordem:
    1. Percorra a subárvore esquerda em in-ordem: {Vazio}

2. Visite a Raiz: {85}

3. Percorra a subárvore direita em in-ordem:

1. Percorra a subárvore esquerda em in-ordem:

1. Percorra a subárvore esquerda em in-ordem: {Vazio}

2. Visite a Raiz: {90}

3. Percorra a subárvore direita em in-ordem: {Vazio}

2. Visite a Raiz: {100}

3. Percorra a subárvore direita em in-ordem: {Vazio}

RESULTADO: 10, 30, 40, 60, 70, 80, 85, 90, 100.

//Percorrendo a Árvore em Pós-Ordem:

1. Percorra a subárvore esquerda em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}

2. Percorra a subárvore direita em pós-ordem: {Vazio}

3. Visite a Raiz: {10}

2. Percorra a subárvore direita em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}

2. Percorra a subárvore direita em pós-ordem: {Vazio}

3. Visite a Raiz: {40}

2. Percorra a subárvore direita em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}

2. Percorra a subárvore direita em pós-ordem: {Vazio}

3. Visite a Raiz: {70}

3. Visite a Raiz: {60}

3. Visite a Raiz: {30}

2. Percorra a subárvore direita em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}

2. Percorra a subárvore direita em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem:

1. Percorra a subárvore esquerda em pós-ordem: {Vazio}

2. Percorra a subárvore direita em pós-ordem: {Vazio}

3. Visite a Raiz: {90}

2. Percorra a subárvore direita em pós-ordem: {Vazio}

3. Visite a Raiz: {100}

3. Visite a Raiz: {85}

3. Visite a Raiz: {80}

RESULTADO: 10, 40, 70, 60, 30, 90, 100, 85, 80.

Agora vamos falar um pouquinho sobre três tipos especiais e árvores: **Árvore B, Árvore B+ e Árvore AVL.** E aí, eu preciso bastante da atenção de vocês agora! Eu coloco esse assunto porque os editais pedem apenas Árvore e não especificam a profundidade do assunto. Com exceção da CESGRANRIO, é raro outras bancas cobrarem esse assunto na profundidade que veremos agora.

É um assunto mais difícil e que cai pouco em prova, portanto só recomendo seguir caso queiram realmente cercar todas as possibilidades. Galera, época de faculdade, segundo semestre, disciplina de Estrutura de Dados! O trabalho final da disciplina era construir um compactador! Isso mesmo! Uma espécie de WinZip, WinRar, etc. E a estrutura usada para compactar arquivos de índices era uma Árvore B.

Uma árvore B é uma maneira de armazenar grandes quantidades de dados de tal forma que você pode procurá-los e recuperá-los muito rapidamente. As árvores B são a base da

maioria dos bancos de dados **modernos**. Como eles funcionam é um bocado complicado, mas eu vou contar uma historinha que talvez os ajude a entender melhor! Vamo lá...

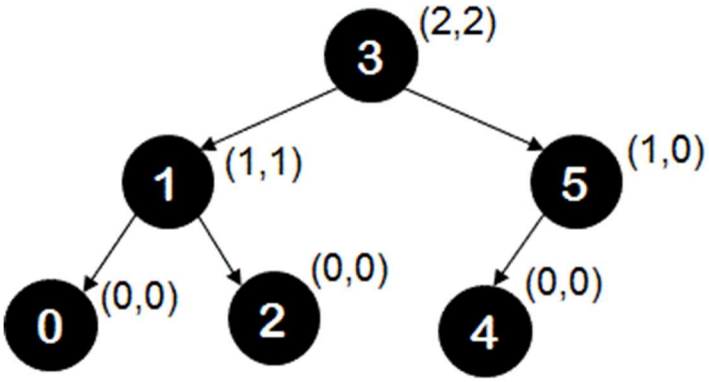
Imagine que você está procurando um par de novos fones de ouvido. Você tem algumas abordagens. Certo? Você poderia ir a todas as lojas do mundo até encontrar o produto que você procura. Como você pode imaginar, esta seria uma maneira horrível de fazer compras. Em vez disso, você poderia ir em uma FNAC, porque você sabe que eles vendem eletrônicos.

Uma vez que chegou à FNAC, você pode observar cada uma das descrições de corredor para ver onde os fones de ouvido são armazenados. Depois de encontrar o corredor correto, você pode escolher os fones de ouvido que você deseja. Ponto final! Observe como o objetivo desse processo é restringir o foco de uma pesquisa cada vez mais...

É assim que funcionam as Árvores B! Ao organizar os dados de forma específica, podemos aproveitá-las para que não desperdicemos nosso tempo buscando dados que tenham chance zero de armazenar os dados que estamos procurando. E a árvore já é construída de maneira a **organizar** os dados da melhor forma possível. Bacana, pessoal?

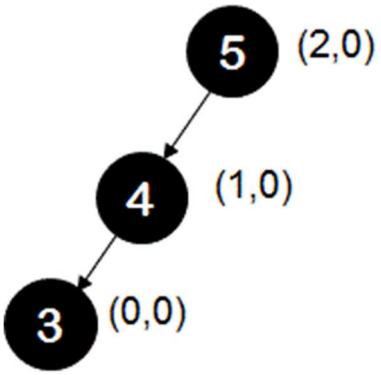
E qual a diferença de uma Árvore B para uma Árvore B+? Galera, a principal diferença é que, em uma Árvore B, as chaves e os dados podem ser armazenados tanto nos nós internos da árvore quanto nas folhas da árvore, enquanto que em uma Árvore B+ as chaves podem ser armazenadas em qualquer nó, mas os dados só podem ser armazenados nas **folhas**.

Por fim, vamos falar um pouco sobre Árvores AVL! Uma Árvore AVL (Adelson-Vesky e Landis) é uma Árvore Binária de Busca em que, para qualquer nó, a altura das subárvores da esquerda e da direita não podem ter uma diferença maior do que 1, portanto uma Árvore AVL é uma Árvore Binária de Busca autobalanceável. Calma que nós vamos ver isso em mais detalhes...

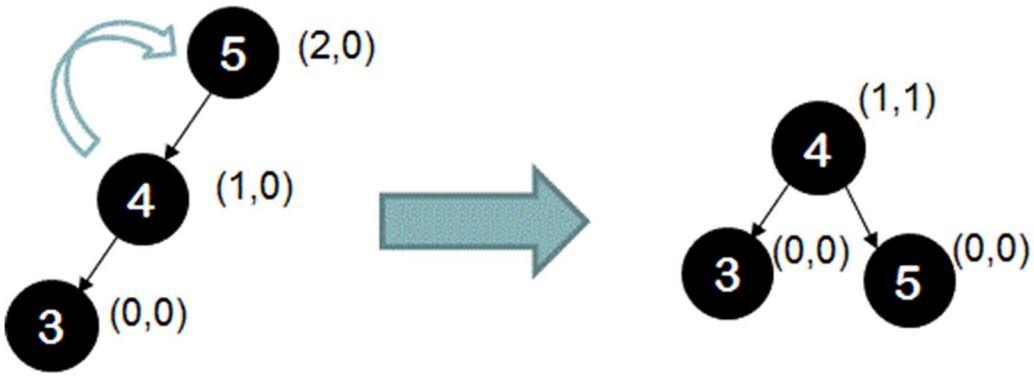


Observem o Nó 3: a altura da subárvore da esquerda é 2 e da subárvore da direita também é 2. Vejamos agora o Nó 1: a altura da subárvore da esquerda é 1 e da subárvore da direita também é 1. E o Nó 5: a altura da subárvore da esquerda é 1 e da subárvore da direita é 0 (visto que ela não existe). Vocês podem ver todos os nós e não encontrarão subárvore da esquerda e direita com diferença **maior que 1.**

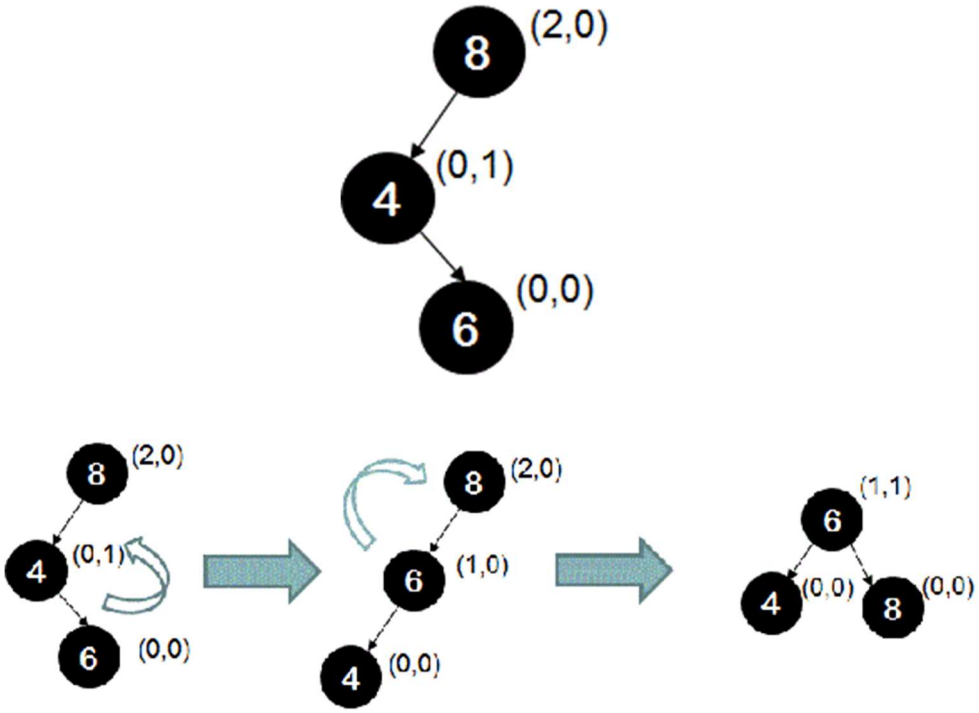
Uma Árvore AVL mantém seu equilíbrio de altura executando operações de rotação se algum dos nós violar a propriedade da diferença maior que 1. Exemplo 1: para a seguinte árvore, a propriedade da árvore AVL é violada no Nó 5, porque a subárvore esquerda possui altura 2, mas a subárvore da direita tem altura 0, então eles diferem em 2. Entenderam?



Essa diferença de 2 é construída nos dois ramos esquerdos - Ramo 5-4 e Ramo 4-3. Concordam? Se a diferença de 2 for construída por dois ramos esquerdos, chamamos esse caso de um Caso Esquerdo-Esquerdo (Genial, né?). Neste caso, realizamos uma rotação à direita no Ramo 5-4 como mostrado na imagem abaixo de forma a **rebalancear a árvore.**



Já na imagem abaixo, a Árvore AVL tem sua propriedade violada no Nó 8. A altura da subárvore esquerda é maior do que a altura da subárvore direita em 2. Essa diferença de 2 é construída por um ramo esquerdo (Ramo 8-4) e um ramo direito (Ramo 4-6), logo é um Caso Esquerdo-Direito. Para restaurar o equilíbrio de altura, executamos uma rotação esquerda seguida de uma rotação direita. Vejam...



Então, galera, caso a árvore não esteja balanceada, é necessário seu balanceamento através de rotações. No Caso Esquerda-Esquerda, basta fazer uma rotação simples para direita no nó desbalanceado. No caso Esquerda-Direita, temos que fazer uma rotação dupla, i.e., faz-se uma rotação para esquerda no nó filho e uma rotação para direita no nó desbalanceado.

No caso Direita-Direita, basta fazer uma rotação simples para a esquerda no nó desbalanceado. No caso Direita-Esquerda, temos que fazer uma rotação dupla, i.e., faz-se uma rotação para direita no nó filho, seguida de uma rotação para esquerda no nó desbalanceado. Bacana? Vocês entenderão isso melhor nos exercícios. Vamos ver agora a complexidade logarítmica dessas estruturas.

ÁRVORE BINÁRIA DE BUSCA		
ALGORITMO	CASO MÉDIO	PIOR CASO
Espaço	O(n)	O(n)
Busca	O(log n)	O(n)
Inserção	O(log n)	O(n)
Remoção	O(log n)	O(n)

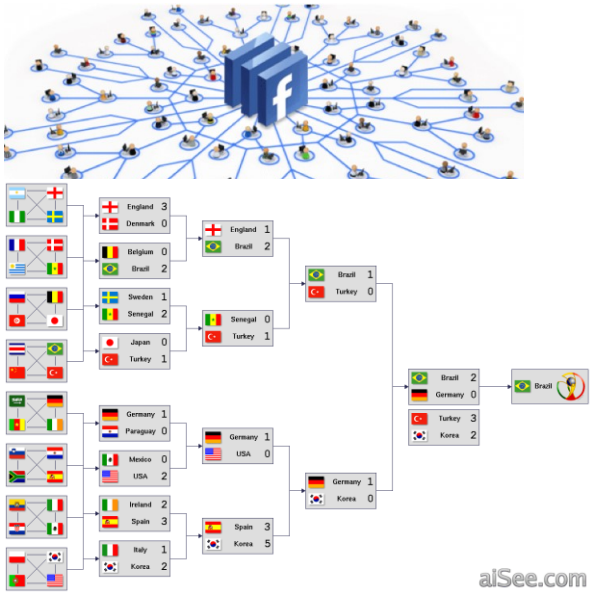
ÁRVORE B / ÁRVORE AVL		
ALGORITMO	CASO MÉDIO	PIOR CASO
Espaço	O(n)	O(n)
Busca	O(log n)	O(log n)
Inserção	O(log n)	O(log n)
Remoção	O(log n)	O(log n)

Quem quiser brincar de Árvore Binária de Busca https://www.cs.usfca.edu/~galles/visualization/BST.html Quem quiser brincar de Árvore AVL https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

GRAFOS

Fiz uma disciplina na faculdade chamada Teoria dos Grafos! Aquilo era absurdamente complexo, mas para concursos a teoria é beeeem mais tranquila e muito rara de cair. Portanto fiquem tranquilos, bacana? Uma definição de grafo afirma que ele é uma estrutura de dados que consiste em um **conjunto de nós** (ou vértices) e um **conjunto de arcos** (ou arestas).

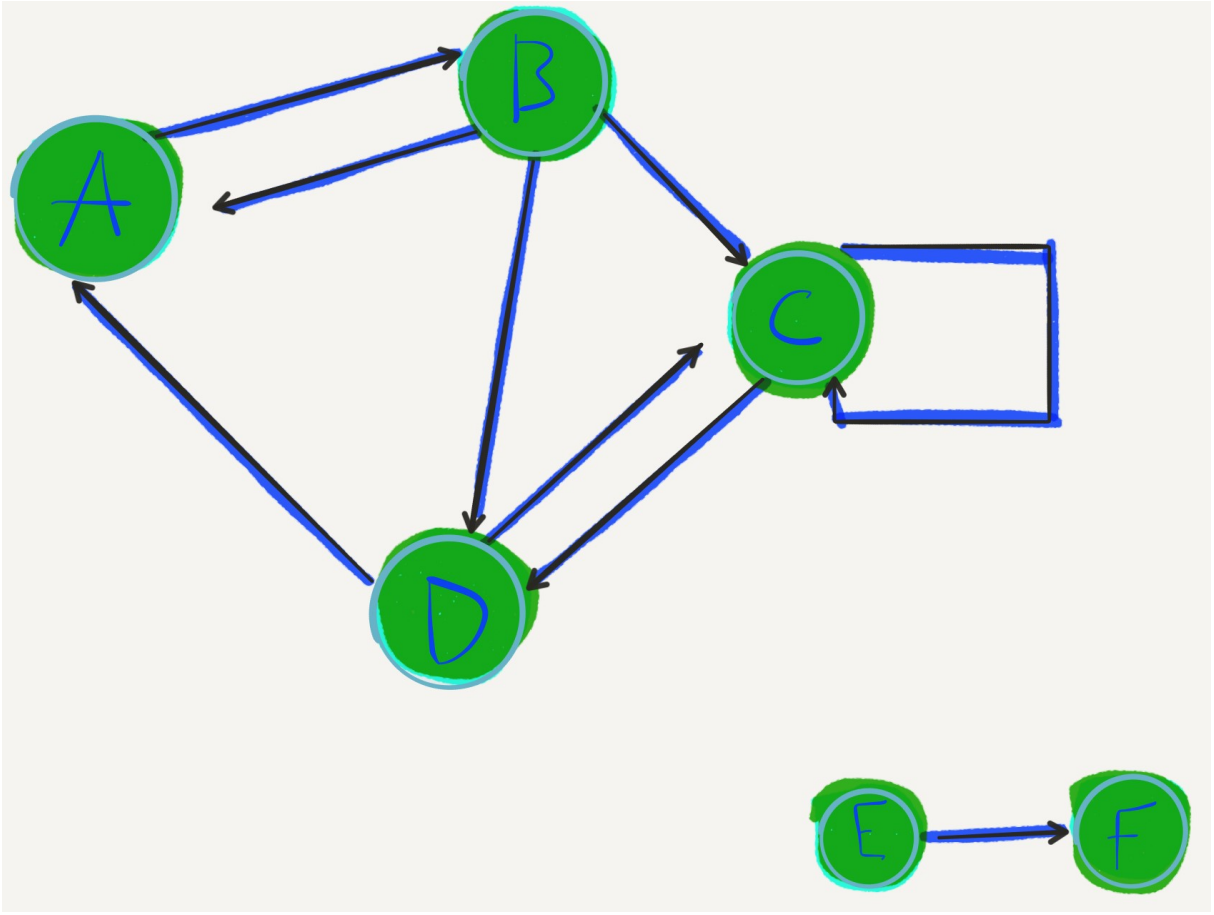
Em outras palavras, podemos dizer que é simplesmente um conjunto de pontos e linhas que conectam vários pontos. Ou também que é uma representação abstrata de um conjunto de objetos e das relações existentes entre eles. Uma grande variedade de estruturas do mundo real podem ser representadas **abstratamente** através de **grafos**. Professor, pode me passar um exemplo? Claro!



Vejam só como se trata de um conceito bastante abrangente! O Facebook pode ser considerado um grafo (pessoas se interligam através de amizades); um mapa rodoviário pode ser considerado um grafo (cidades se relacionam através de estradas); as eliminatórias de um campeonato de futebol também podem ser consideradas um grafo (times jogam entre si para ganhar o campeonato).

Abaixo temos um exemplo de um grafo que eu desenhei e que nos ajudará a entender melhor a terminologia utilizada. Seguem as premissas:

- N = {A, B, C, D, E, F}
- G = {(A,B), (B,A), (B,C), (B,D), (C,C), (D,A), (D,C), (E,F)}



- **Nó, Nodo ou Vértice:**

Qualquer elemento de um conjunto N.

- **Aresta ou Arco:**

Qualquer elemento de um conjunto A.

- **Nós Adjacentes** (Relação de Adjacência):

São aqueles nós ligados por um arco (Ex: A é adjacente a D).

- **Arco Incidente** (Relação de Incidência):

São aqueles arcos que levam a um nó (Ex: (C,D) é incidente em D).

- **Grau:**

É a quantidade de arcos que partem ou chegam em/de um nó.

- **Caminho:**

Sequência de arcos que levam de um nó a outro (Ex: A→C <(A,B), (B,D), (D,C)>).

- **Circuito ou Ciclo:**

É o caminho que leva ao mesmo nó do qual saiu (Ex: <(A,B), (B,D), (D,A)>).

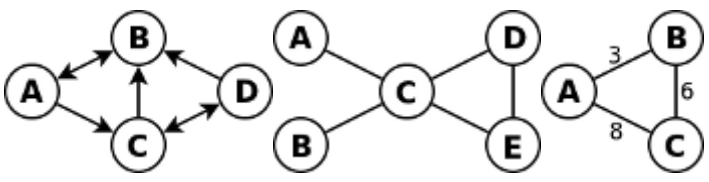
- **Laço:**

É o circuito de um único arco (<(C,C)>).

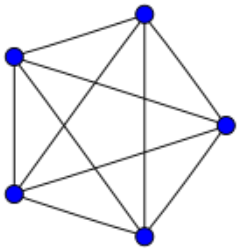
- **Ordem:**

É a quantidade de vértices totais do grafo.

Vamos ver agora alguns conceitos importantes! Um grafo pode ser dirigido – também chamado direcionado, orientado ou dígrafo –, se as arestas tiverem uma direção (imagem da esquerda), ou não dirigido, se as arestas não tiverem direção (imagem central). Se as arestas tiverem associado um peso ou custo, o grafo passa a ser chamado **ponderado, valorado ou pesado** (imagem da direita).



Um grafo simples é aquele que não contém laços. Um grafo vazio é aquele que contém exclusivamente vértices (não contém arcos). Um grafo misto é aquele que possui arestas dirigidas e não-dirigidas. Um grafo trivial é aquele que possui somente um vértice. Um grafo é denso se contém muitos arcos em relação ao número de vértices e esparso se contém poucos arcos. Como assim, professor?

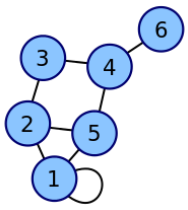


Um grafo é denso se o seu número de arcos é da mesma ordem que o quadrado do número de vértices; e é esparso se o número de arcos for da mesma ordem que o número de vértices. Um grafo é **regular** se todos os vértices têm o mesmo grau; e é **completo** se todo vértice é adjacente a todos os outros vértices (o grafo ao lado é regular e completo).

Um grafo é dito **conexo ou conectado** quando existir pelo menos um caminho entre cada par de vértices. Caso contrário, ele será dito desconexo, isto é, se há pelo menos um par de vértices que não esteja ligado a nenhuma cadeia (caminho). Vejam a imagem do grafo que eu desenhei lá em cima! Ele é conexo ou desconexo? Ele é desconexo, há um par de vértices (E,F) que não está ligado a nenhum caminho.

Se o grafo for **direcionado/orientado**, um grafo é dito fortemente conexo se existir um caminho de A → B e de B → A, para cada par (A,B) de vértices de um grafo. Em outras palavras, o grafo é fortemente conexo se cada par de vértices participa de um circuito. Isto significa que cada vértice pode ser alcançável partindo-se de qualquer outro vértice do grafo.

Galera, existem outras maneiras de representar um grafo. Uma das maneiras mais comuns é por meio de uma **Matriz de Adjacências.** A definição precisa das entradas da matriz varia de acordo com as propriedades do grafo que se deseja representar, porém de forma geral o valor a_{ij} guarda informações sobre como os vértices v_i e v_j estão relacionados (isto é, informações sobre a adjacência de v_i e v_j).



Para representar um grafo não direcionado, simples e sem pesos, basta que as entradas aij da Matriz A contenham 1 se vi e vj são adjacentes e 0, caso contrário. Se as arestas do grafo tiverem pesos, aij pode conter, ao invés de 1 quando houver uma aresta entre v_i e v_j, o peso dessa mesma aresta.

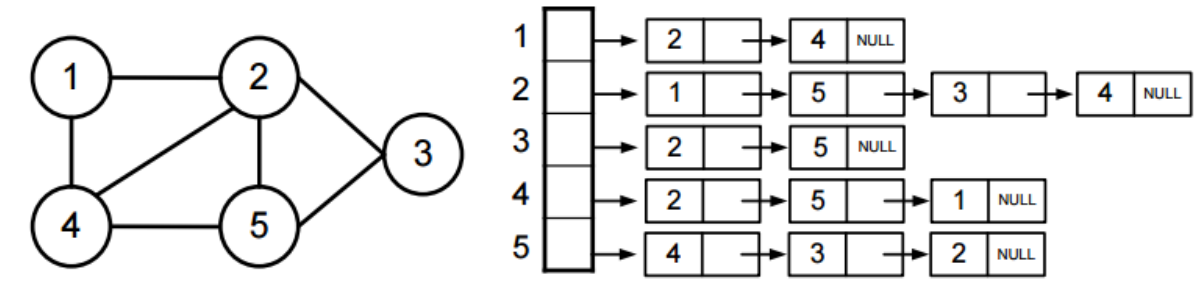
$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

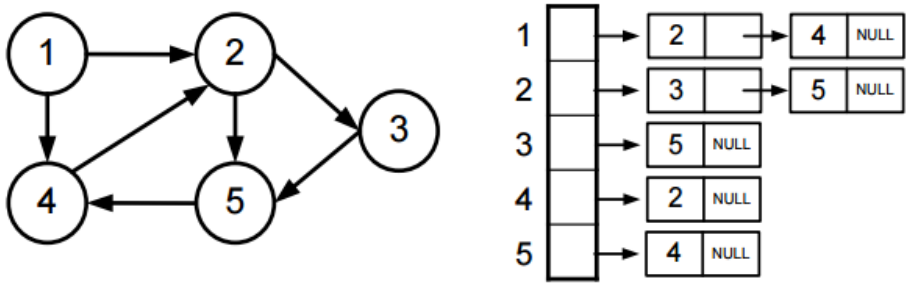
Vamos entender? O elemento A11 = 1, significando que o Nó 1 tem adjacência com o Nó 1 (ele mesmo, basta ver a figura); o elemento A12 = 1, significando que o Nó 1 tem adjacência com o Nó 2; elemento A13 = 0, significando que o Nó 1 não tem adjacência com o Nó 3; o elemento A_{14} = 0, significando que o Nó 1 não tem adjacência com o Nó 4.

O elemento A_{15} = 1, significando que o Nó 1 tem adjacência com o Nó 5; o elemento A_{16} = 0, significando que o Nó 1 não tem adjacência com o Nó 3. Galera, agora ficou Fácil de entender, concordam? Lembrando que, se fosse um grafo ponderado, bastaria colocar o peso, em vez de colocar 1 na Matriz de Adjacências. Existem só mais alguns detalhes.

Em grafos não direcionados, as matrizes de adjacência são simétricas ao longo da diagonal principal - isto é, a entrada a_{ij} é igual à entrada a_{ji}. **Matrizes de Adjacência de grafos direcionados**, no entanto, não são assim. Num dígrafo sem pesos, a entrada aij da matriz é 1 se há um arco de vi para vj e 0, caso contrário. Há outra maneira de representar também chamada Lista de Adjacências.

Se o grafo é não direcionado, cada entrada é um conjunto de dois nós contendo as duas extremidades da aresta correspondente; se ele for dirigido, cada entrada é uma tupla de dois nós, um indicando o nó de origem e o outro denotando o nó destino do arco correspondente. É bem simples, para cada nó, eu escrevo suas adjacências. No exemplo anterior: Nó 1 = 1, 2, 5; Nó 2 = 1, 3, 5; e assim por diante.





Professor, qual é melhor? Cara, cada representação tem suas **vantagens e desvantagens!** É fácil encontrar todos os vértices adjacentes a um vértice dada em uma representação lista de

adjacência; você simplesmente lê a sua lista de adjacência. Com uma matriz de adjacência em vez disso se tem que pesquisar mais de uma linha inteira, gastando tempo O(n).

Se, pelo contrário, deseja realizar um teste de vizinhança em dois vértices (isto é, determinar se eles têm uma aresta entre eles), uma matriz de adjacência proporciona isso na hora. No entanto, este teste de vizinhança em uma lista de adjacências requer **tempo** proporcional ao **número de arestas** associado com os dois **vértices**. Há diversos outros aspectos também a considerar, como tamanho.

Pensem comigo! Para um grafo com uma Matriz de Adjacência esparsa (não densa), uma representação de Lista de Adjacências do grafo ocupa **menos espaço**, porque ele não usa nenhum espaço para representar as arestas que não estão presentes. Lembram-se da lista? Nós só representamos os nós adjacentes, em contraste com a Matriz de Adjacência. No entanto, quanto mais denso, isso pode mudar.

HASHING

As Tabelas de Dispersão, também conhecidas como Tabelas de Espelhamento ou Tabelas Hashing, armazenam uma coleção de valores, sendo que cada valor está associado a uma chave. Tais chaves têm que ser todas distintas e são usadas para mapear os valores na tabela. Esse mapeamento é feito por uma **a função de hashing**, que chamaremos de **h**.

Por exemplo, podemos implementar uma tabela de dispersão usando um vetor com oito posições e utilizar h(x) = x MOD 8 como função de hashing[1]. Dizemos que h(k) é a posição original da chave k e é nessa posição da tabela que a chave k deve ser inserida. A imagem abaixo ilustra a inserção de uma coleção de valores com suas respectivas chaves numa Tabela de Dispersão.

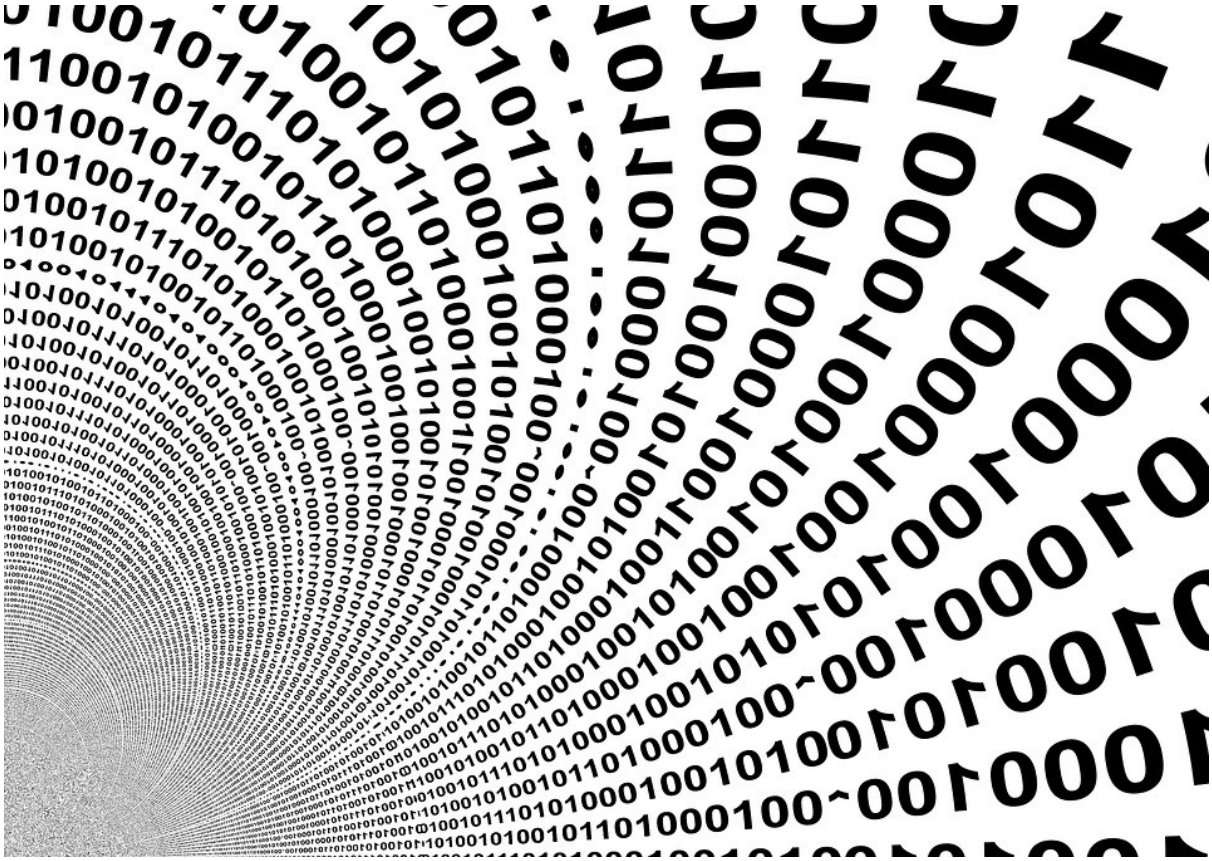
Observe que o valor Gil foi colocado na posição 7 da tabela, pois a chave associada a Gil é 31 e h(31) = 7. O que aconteceria se tratássemos de inserir nessa tabela o valor Ivo com chave 33? Observe que h(33) = 1, mas a posição 1 da tabela já está ocupada. Dizemos que as chaves 25 e 33 colidiram. Mais precisamente, duas chaves x e y colidem se h(x) = h(y).

Note que o problema da colisão de chaves ocorre porque, na maioria dos casos, o domínio das chaves é maior que a quantidade de posições da tabela. Sendo assim, pelo princípio da casa de pombos, qualquer função do conjunto das chaves para o conjunto das posições da tabela **não é injetora**.

Não é aceitável recusar a inserção de uma chave que colida com outra já existente na tabela se ela ainda tiver posições livres. Precisamos de alguma estratégia para lidar com as colisões de chaves. Há diversas técnicas para lidar com as colisões, tais como Hashing Fechado ou Hashing Aberto. O Método de Hashing é recomendado para um **grande número de dados** que possuam faixas de **valores variáveis.**

[1] Galera... o MOD representa o resto de uma divisão. Ex: 10 mod 8 = 2, porque 10/8 possui quociente 1 e resto 2. Em outras palavras, 8*1 + 2 = 10.

BITMAPS



Nesta aula vamos aprender um pouco mais sobre indexação utilizando **mapas de bits, os bitmaps.** Não estamos falando do formato de imagem que todos devem conhecer, bpm, muito utilizado antes de haver algoritmos de compressão, mas de uma estrutura de indexação que facilita o acesso a informações disponibilizadas em bancos de dados.

Antes de apresentar os bitmaps e como são utilizados para recuperar informações, temos que relembrar alguns conceitos de bancos de dados. Uma tabela é definida por suas colunas e por suas entradas (linhas) com os dados armazenados, chamadas de tuplas. Cada coluna apresenta uma característica da tabela, enquanto cada tupla apresenta uma observação da tabela. Para facilitar, vamos criar uma tabela de exemplo, representando "Banda".

BANDA

ROWID	NOME	PAIS	ESTILO
1	Radiohead	Inglaterra	Indie Rock
2	The Killers	EUA	Rock
3	Beach House	EUA	Indie Rock
4	The Cure	Inglaterra	Indie Rock
5	The Cardigans	Suécia	Rock
6	R.E.M.	EUA	Rock
7	The Hives	Suécia	Punk Rock
8	The Who	Inglaterra	Rock

9	Green Day	EUA	Punk Rock
10	Sex Pistols	Inglaterra	Punk Rock

Ainda relembrando um pouco de banco de dados, se quisermos buscar toda as bandas inglesas, podemos realizar a consulta abaixo:

SELECT * FROM BANDA WHERE PAIS = 'Inglaterra'

E se quiséssemos saber quais são dos EUA e da Inglaterra? Teríamos a consulta a seguir

SELECT * FROM BANDA WHERE PAIS = 'Inglaterra' OR PAIS = 'EUA'

As consultas funcionam normalmente, mas para realizá-las, o sistema gerenciador de banco de dados (SGDB) deve percorrer todas as tuplas (linhas/observações) e comparar 'Inglaterra' com o valor da coluna 'PAIS', na primeira consulta, e com 'Inglaterra' e 'EUA' na segunda. Essas "varreduras", chamadas de FULL TABLE SCAN, são muito custosas para o banco e, dependendo do tamanho da tabela, podem levar a um tempo excessivamente grande para serem executadas. Um modo de acessar de forma mais direta, sem necessidade de realizar pesquisas custosas é a criação de índices. Existem muitas técnicas para criar índices e uma delas é a utilização dos bitmaps, ou ainda, mapas de bits.

Um bitmap deve indicar quais tuplas possuem um certo valor para a coluna escolhida para se criar o índice. Se consultas como a que mostramos for muito comum, é válido criarmos um índice para a coluna 'PAIS' e facilitar o acesso.

O bitmap é uma tabela onde as linhas são os possíveis valores da coluna selecionada para criar o índice (nesse caso, todos os valores de 'PAIS') e as colunas são os números da tuplas (ROWID). Para cada linha do bitmap, serão preenchidos os valores 0 e 1 caso o valor de 'PAIS' da referida tupla da tabela 'Banda' ser ou não daquele país. Ou seja, para o caso específico, teremos três tuplas no bitmap, cada uma correspondente a um valor de 'PAIS': EUA, Inglaterra e Suécia. As colunas do bitmap são os números das tuplas da tabela 'Banda', ou seja: 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10. Professor, não entendi nadinha! Para visualizar melhor como um bitmap é construído, vamos criar um representando a coluna 'PAIS':

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EUA	0	1	1	0	0	1	0	0	1	0
Inglaterra	1	0	0	1	0	0	0	1	0	1
Suécia	0	0	0	0	1	0	1	0	0	0

Para entender o preenchimento temos que nos remeter à tabela original 'Banda'. Percebam que as bandas da Inglaterra são as de número 2, 3, 6, e 9. Desse modo, no bitmap, preenchemos com valor 1 as colunas correspondentes aos números (ROWID) das tuplas na tabela original. Da mesma forma se faz com EUA e Suécia. Percebam também que, uma vez uma linha preenchida com 1, as outras colunas serão obrigatoriamente 0, pois uma banda somente pode ser de um país e o índice deve refletir o mesmo.

Professor, e isso serve para... Se quisermos realizar as consultas anteriores utilizando os índices em bitmap temos que executar uma varredura completa, assim como antes de termos o índice, mas utilizando os bitmaos não haverá comparações de strings, mas

somente uma avaliação bit a bit, que é muito mais performática. Ademais, é possível compactar os bitmaps com algoritmos que reduzem bastante o espaço exigido dessas estruturas.

Os bitmaps são muito utilizados para criação de índices em SGDBs Oracle, PostgreSQL, Teradata, dentre outros. O seu uso é mais premente e performático em sistemas OLAP (Online Analytical Processing), pois nestes a necessidade de atualização dos dados é muito menor, algo que é relativamente custoso no uso de bitmaps, pois, qualquer alteração nas colunas onde há índices criados, deve-se atualizar os bits do mapa.

Somente como último exemplo, vamos criar um bitmap para outra coluna da tabela 'Banda', agora para a coluna 'Estilo'. Quantas linhas o bitmap dessa coluna terá? E quantas colunas?

	1	2	3	4	5	6	7	8	9	10
Rock	0	1	0	0	1	1	0	1	0	0
Indie Rock	1	0	1	1	0	0	0	0	0	0
Punk Rock	0	0	0	0	0	0	1	0	1	1

Se fizermos uma pesquisa para saber quais bandas são de punk rock ou indie rock, pelo bitmap podemos rapidamente verificar que são as tuplas 1, 3, 4, 7, 9 e 10. Um computador pode ler vários bits de uma vez ao vez de realizar a comparação um a um, aumentando muito a performance de pesquisas que utilizam o índice em bitmap.

Mostramos a criação de bitmaps para somente uma coluna, mas é possível criamos para mais de uma. Esses índices são interessantes quando consultas frequentes com mais de uma coluna. Um exemplo seria consultar bandas que são da Inglaterra que tocam o estilo indie rock. Para construir um índice com duas ou mais colunas, temos que representar todas as combinações possíveis entre os valores das colunas e preencher os bits com o valor 1 nas tuplas onde os dois valores ocorrem simultaneamente. Como exemplo, vamos criar um índice para PAISESTILO:

	1	2	3	4	5	6	7	8	9	10
EUARock	0	1	0	0	0	1	0	0	0	0
EUAIndieRock	0	0	1	0	0	0	0	0	0	0
EUAPunkRock	0	0	0	0	0	0	0	0	1	0
InglaterraRock	0	0	0	0	0	0	0	1	0	0
InglaterraIndieRock	1	0	0	1	0	0	0	0	0	0
InglaterraPunkRock	0	0	0	0	0	0	0	0	0	1
SuéciaRock	0	0	0	0	1	0	0	0	0	0
SuéciaIndieRock	0	0	0	0	0	0	0	0	0	0
SuéciaPunkRock	0	0	0	0	0	0	1	0	0	0

A consulta das bandas inglesas que tocam indie rock, portanto, são as tuplas 1 e 4.

Bitmpas, portanto, são estruturas utilizadas como índices em bancos de dados para facilitar consultas, principalmente com colunas com pouca cardinalidade, ou seja, poucos valores possíveis.

ESTRUTURA DE ARQUIVOS

É senso comum que há a necessidade de **salvar** ou **guardar** nossos **dados** ou **programas.** E onde podemos fazer isso? Em geral, em dispositivos persistentes (Discos, PenDrive, Fitas, etc). O armazenamento de pequenos volumes de dados, via de regra, não encerra grandes problemas no que diz respeito à distribuição dos registros dentro de um arquivo.

À medida que cresce o volume de dados, a frequência ou a complexidade dos acessos, crescem também os problemas de eficiência do armazenamento dos arquivos e do acesso a seus registros, sendo a sofisticação das técnicas de armazenamento e recuperação de dados uma consequência da necessidade de **acesso rápido** a grandes arquivos ou arquivos muito solicitados.

A maneira intuitiva de **armazenar** um arquivo consiste na **distribuição** dos seus registros em uma ordem arbitrária, um após o outro, dentro da **área destinada.** Esta ordem pode ser, por exemplo, aquela na qual os registros são gerados, causando dificuldade na localização e perda de eficiência, mas esta técnica é bastante usada, principalmente durante as fases preliminares da geração de um arquivo.

Quando escrevemos um programa para gerar um arquivo de dados, costumamos agrupar os campos que fornecem informações sobre um mesmo indivíduo em um registro. Um arquivo é formado por uma coleção de registros lógicos, cada um deles representando um objeto ou entidade. E o que seria um registro lógico? É uma sequência de itens, sendo cada item chamado **campo ou atributo.**

Cada item corresponde a uma **característica ou propriedade** do objeto representado. Existem três formas de acessar arquivos: sequencial, direta ou indexado. No primeiro caso, os registros são lidos em sequência, um após o outro. A cada registro lido o indicador de posição de arquivo é avançado para que a próxima leitura ocorra iniciando naquele ponto.

No segundo caso, pode-se acessar qualquer posição do arquivo, para tanto é necessário preciso ajustar o indicador de posição do arquivo para onde deseja-se realizar uma operação de leitura/escrita. No terceiro caso, o arquivo é visto como um conjunto de registros indexados por uma chave.

QUESTÕES COMENTADAS – ESTRUTURA DE DADOS - MULTIBANCAS

1. **(FGV – 2015 – DPE/MT – Analista de Sistemas) No desenvolvimento de sistemas, a escolha de estruturas de dados em memória é especialmente relevante. Dentre outras classificações, é possível agrupar essas estruturas em lineares e não lineares, conforme a quantidade de sucessores e antecessores que os elementos da estrutura possam ter. Assinale a opção que apresenta, respectivamente, estruturas de dados lineares e não lineares.**

a) Tabela de dispersão e fila.
b) Estrutura de seleção e pilha.
c) Pilha e estrutura de seleção.
d) Pilha e árvore binária de busca.
e) Fila e pilha.

Comentários:

Além dessa classificação, existe outra também importante: Estruturas Lineares e Estruturas Não-Lineares. As Estruturas Lineares são aquelas em que cada elemento pode ter um único predecessor (exceto o primeiro elemento) e um único sucessor (exceto o último elemento). Como exemplo, podemos citar Listas, Pilhas, Filas, Arranjos, entre outros.

Já as Estruturas Não-Lineares são aquelas em que cada elemento pode ter mais de um predecessor e/ou mais de um sucessor. Como exemplo, podemos citar Árvores, Grafos e Tabelas de Dispersão. Essa é uma classificação muito importante e muito simples de entender. Pessoal, vocês perceberão que esse assunto é cobrado de maneira superficial na maioria das questões, mas algumas são nível doutorado!

Conforme vimos em aula, trata-se de pilha e árvore respectivamente. **Gabarito: D**

2. **(CESPE – 2010 – DETRAN/ES – Analista de Sistemas) Um tipo abstrato de dados apresenta uma parte destinada à implementação e outra à especificação. Na primeira, são descritas, em forma sintática e semântica, as operações que podem ser realizadas; na segunda, os objetos e as operações são representados por meio de representação, operação e inicialização.**

Comentários:

A Especificação Sintática define o nome do tipo, suas operações e o tipo dos argumentos das operações, definindo a assinatura do TAD. A Especificação Semântica descreve propriedades e efeitos das operações de forma independente de uma implementação específica. E a Especificação de Restrições estabelece as condições que devem ser satisfeitas antes e depois da aplicação das operações.

Conforme vimos em aula, temos duas partes: Especificação e Implementação. Sendo que a especificação se divide em Especificação Sintática e Semântica e Implementação se divide em Representação e Algoritmos. Logo, a questão está invertida: na segunda, são descritas, em forma sintática e semântica, as operações que podem ser realizadas; na primeira, os objetos e as operações são representados por meio de representação, operação e inicialização. **Gabarito: E**

3. (CESPE – 2010 – TRT/RN – Analista de Sistemas) O tipo abstrato de dados consiste em um modelo matemático (v,o), em que v é um conjunto de valores e o é um conjunto de operações que podem ser realizadas sobre valores.

Comentários:

Por fim, vamos falar sobre Tipos Abstratos de Dados (TAD). Podemos defini-lo como um modelo matemático (v,o), em que v é um conjunto de valores e o é um conjunto de operações que podem ser realizadas sobre valores. Eu sei, essa definição é horrível de entender! Para compreender esse conceito, basta lembrar de abstração, i.e., apresentar interfaces e esconder detalhes.

Conforme vimos em aula, a questão está conforme nós descrevemos em nossa aula. Essa é a definição formal de Tipos Abstratos de Dados. **Gabarito: C**

4. (CESPE – 2010 – BASA – Analista de Sistemas) A escolha de estruturas internas de dados utilizados por um programa pode ser organizada a partir de TADs que definem classes de objetos com características distintas.

Comentários:

Em outras palavras, o nível semântico trata do comportamento de um tipo abstrato de dados; e o nível sintático trata da apresentação de um tipo abstrato de dados. Podemos dizer, então, que o TAD encapsula uma estrutura de dados com características semelhantes – podendo ser formado por outros TADs –, e esconde a efetiva implementação dessa estrutura de quem a manipula.

Conforme vimos em aula, definem classes de objetos com características semelhantes. **Gabarito: E**

5. (CESPE – 2010 – BASA – Analista de Sistemas) A descrição dos parâmetros das operações e os efeitos da ativação das operações representam, respectivamente, os níveis sintático e semântico em que ocorre a especificação dos TDAs.

Comentários:

A Especificação Sintática define o nome do tipo, suas operações e o tipo dos argumentos das operações, definindo a assinatura do TAD. A Especificação Semântica descreve propriedades e efeitos das operações de forma independente de uma implementação

específica. E a Especificação de Restrições estabelece as condições que devem ser satisfeitas antes e depois da aplicação das operações.

Conforme vimos em aula, realmente se trata respectivamente do nível sintático e semântico. **Gabarito: C**

6. (FCC – 2010 – TRE/AM – Analista de Sistemas) Em relação aos tipos abstratos de dados - TAD, é correto afirmar:

a) O TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações sobre esses dados.

b) Na transferência de dados de uma pilha para outra, não é necessário saber como a pilha é efetivamente implementada.

c) Alterações na implementação de um TAD implicam em alterações em seu uso.

d) Um programador pode alterar os dados armazenados, mesmo que não tenha conhecimento de sua implementação.

e) TAD é um tipo de dados que esconde a sua implementação de quem o manipula.

Comentários:

Em outras palavras, o nível semântico trata do comportamento de um tipo abstrato de dados; e o nível sintático trata da apresentação de um tipo abstrato de dados. Podemos dizer, então, que o TAD encapsula uma estrutura de dados com características semelhantes – podendo ser formado por outros TADs –, e esconde a efetiva implementação dessa estrutura de quem a manipula.

(a) Errado, ele encapsula a estrutura de dados; (b) Correto, não é necessário saber a implementação, porém a FCC marcou o gabarito como errado; (c) Errado, a implementação pode mudar livremente; (d) Errado, sem conhecimento da implementação, ele não poderá alterar dados armazenados; (e) Correto, esse item também está correto (Lembrem-se: Na FCC, muitas vezes tempos que marcar a mais correta ou a menos errada – infelizmente!). **Gabarito: E**

7. (FCC – 2009 – TRE/PI – Analista de Sistemas) Em relação a tipos abstratos de dados, é correto afirmar que:

a) o TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações disponibilizadas sobre esses dados.

b) algumas pilhas admitem serem declaradas como tipos abstratos de dados.

c) filas não permitem declaração como tipos abstratos de dados.

d) os tipos abstratos de dados podem ser formados pela união de tipos de dados primitivos, mas não por outros tipos abstratos de dados.

e) são tipos de dados que escondem a sua implementação de quem o manipula; de maneira geral as operações sobre estes dados são executadas sem que se saiba como isso é feito.

Comentários:

Em outras palavras, o nível semântico trata do comportamento de um tipo abstrato de dados; e o nível sintático trata da apresentação de um tipo abstrato de dados. Podemos dizer, então, que o TAD encapsula uma estrutura de dados com características semelhantes – podendo ser formado por outros TADs –, e esconde a efetiva implementação dessa estrutura de quem a manipula.

(a) Errado. Pelo contrário, ele encapsula a estrutura de dados de modo que o usuário não tem acesso a implementação das operações; (b) Correto. Todas elas admitem, porém a FCC marcou o gabarito como errado; (c) Errado. Elas permitem a declaração como tipos abstratos de dados; (d) Errado. Um TAD pode ser formado por outros TADs; (e) Correto. Escondem a implementação de quem os manipula. **Gabarito: E**

LISTA DE QUESTÕES – VETORES E MATRIZES - MULTIBANCAS

1. **(FCC - 2009 - TJ-PA - Analista Judiciário - Tecnologia da Informação) Considere uma estrutura de dados do tipo vetor. Com respeito a tal estrutura, é correto que seus componentes são, caracteristicamente,**

   a) heterogêneos e com acesso FIFO.

   b) heterogêneos e com acesso LIFO.

   c) heterogêneos e com acesso indexado-sequencial.

   d) homogêneos e acesso não indexado.

   e) homogêneos e de acesso aleatório por intermédio de índices.

2. **(CETAP - 2010 - AL-RR - Analista de Sistemas) Matrizes são estruturas de dados de n-dimensões. Por simplicidade, chamaremos de matrizes as matrizes bidimensionais numéricas (que armazenam números inteiros). Sendo assim, marque a alternativa INCORRETA.**

   a) Uma matriz de m linhas e n colunas contêm (m * n) dados.

   b) Uma matriz pode ser representada utilizando listas ligadas.

   c) A soma dos elementos de uma matriz pode ser calculada fazendo dois laços aninhados, um sobre as linhas e o outro sobre as colunas.

   d) A soma de duas matrizes de m linhas e n colunas resulta em uma matriz de 2*m linhas e 2*n colunas.

   e) O produto de duas matrizes de n linhas e n colunas resulta em uma matriz de n linhas e n colunas.

3. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Arquitetura de Tecnologia) Os dados armazenados em uma estrutura do tipo matriz não podem ser acessados de maneira aleatória.** Portanto, usa-se normalmente uma matriz quando o volume de inserção e remoção de dados é maior que o volume de leitura dos elementos armazenados.

4. **(CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) Entre alguns tipos de estrutura de dados, podem ser citados os vetores, as pilhas e as filas.**

5. **(CESPE - 2011 - EBC - Analista - Engenharia de Software) Vetores são utilizados quando estruturas indexadas necessitam de mais que um índice para identificar um de seus elementos.**

6. **(CESPE - 2010 - TRE-BA - Analista Judiciário - Análise de Sistemas) Vetores podem ser considerados como listas de informações armazenadas em posição contígua na memória.**

7. **(CESPE - 2010 - TRE-BA - Analista Judiciário - Análise de Sistemas) Uma posição específica de um vetor pode ser acessada diretamente por meio de seu índice.**

8. **(FCC - 2016 - Copergás - PE - Analista Tecnologia da Informação) Considere o algoritmo a seguir, na forma de pseudocódigo:**

```
Var n, i, j, k, x: inteiro
Var v: vetor[0..7] inteiro
Início
 v[0] ← 12
 v[1] ← 145
 v[2] ← 1
 v[3] ← 3
 v[4] ← 67
 v[5] ← 9
 v[6] ← 45
 n ← 8
 k ← 3
 x ← 0

 Para j ← n-1 até k passo -1 faça
  v[j] ← v[j - 1];
 Fim_para

 v[k] ← x;
Fim
```

Este pseudocódigo:

a) exclui o valor contido na posição x do vetor v.

b) insere o valor de x entre v[k-1] e v[k] no vetor v.

c) exclui o valor contido na posição k do vetor v.

d) tentará, em algum momento, acessar uma posição que não existe no vetor.

e) insere o valor de k entre v[x] e v[x+1] no vetor v.

GABARITO

1. E
2. D
3. E
4. C
5. E
6. C
7. C
8. B

QUESTÕES COMENTADAS – LISTA ENCADEADA - MULTIBANCAS

1. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) O tempo de busca de um elemento em uma lista duplamente encadeada é igual à metade do tempo da busca de um elemento em uma lista simplesmente encadeada.**

Comentários:

Não! Apesar de permitir que se percorra a lista em ambas as direções, em média ambas possuem o mesmo tempo de busca de um elemento. **Gabarito: E**

2. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) Em algumas implementações, uma lista vazia pode ter um único nó, chamado de sentinela, nó cabeça ou header. Entre suas possíveis funções, inclui-se simplificar a implementação de algumas operações realizadas sobre a lista, como inserir novos dados, recuperar o tamanho da lista, entre outras.**

Comentários:

Perfeito! Ele simplifica a implementação de algumas operações porque se guarda o endereço do primeiro e do último elemento de uma estrutura de dados de modo que o programador não precisa conhecer a estrutura de implementação da lista para realizar suas operações. **Gabarito: C**

3. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) Estruturas ligadas como listas encadeadas superam a limitação das matrizes que não podem alterar seu tamanho inicial.**

Comentários:

Perfeito! Listas Encadeadas admitem alocação dinâmica, em contraste com as matrizes. **Gabarito: C**

4. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As listas duplamente encadeadas diferenciam-se das listas simplesmente encadeadas pelo fato de, na primeira, os nós da lista formarem um anel com o último elemento ligado ao primeiro da lista.**

Comentários:

Não, a diferença é que Listas Duplamente Encadeadas possuem dois ponteiros, que apontam para o nó sucessor e para o nó predecessor e Listas Simplesmente Encadeadas possuem apenas um ponteiro, que aponta para o nó sucessor. **Gabarito: E**

5. **(FCC - 2012 - TRE-SP - Analista Judiciário - Análise de Sistemas - E) Numa lista singularmente encadeada, para acessar o último nodo é necessário partir do primeiro e ir seguindo os campos de ligação até chegar ao final da lista.**

Comentários:

Perfeito! Se é uma lista singularmente encadeada, é necessário percorrer cada elemento um-a-um até chegar ao final da lista. **Gabarito: C**

6. **(CESPE - 2011 - EBC - Analista - Engenharia de Software) Uma lista é uma coleção de elementos do mesmo tipo dispostos linearmente, que podem ou não seguir determinada organização. As listas podem ser dos seguintes tipos: de encadeamento simples, duplamente encadeadas e ordenadas.**

Comentários:

Uma lista é, por natureza, heterogênea, i.e., seus elementos são compostos por tipos de dados primitivos diferentes. A questão afirmou que a lista é uma coleção de elementos do mesmo tipo. E ela está certa, veja só o peguinha da questão:

Eu crio um tipo composto por dois tipos de dados diferentes: tipoEstrategia = {curso: caractere; duracao: inteiro}. Observe que o tipo tipoEstrategia é composto por tipos de dados primitivos diferentes (caractere e inteiro). Agora eu crio uma lista ListaEstrategia: tipoEstrategia. Veja que todos os elementos dessa lista terão o mesmo tipo (tipoEstrategia). Em outras palavras: a Lista Heterogênea é composta por elementos do mesmo tipo que, em geral, são compostos por mais de um tipo primitivo.

Além disso, as listas podem ser simplesmente encadeadas, duplamente encadeadas e ordenadas. E ainda podem ser circulares. Observem que alguns autores consideram Listas Ordenadas como um tipo de lista! Como, professor? Ela é uma lista em que seus elementos são ordenados (crescente ou decrescente). **Gabarito: C**

7. **(CESPE - 2009 - ANAC - Técnico Administrativo - Informática) Em uma lista circular duplamente encadeada, cada nó aponta para dois outros nós da lista, um anterior e um posterior.**

Comentários:

Perfeito! Há dois ponteiros: uma para o nó anterior e um para o nó posterior. **Gabarito: C**

8. **(CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) A principal característica de uma lista encadeada é o fato de o último elemento da lista apontar para o elemento imediatamente anterior.**

Comentários:

Não, o último elemento da lista não aponta para nenhum outro nó em uma lista não-circular. **Gabarito: E**

9. **(CESPE - 2009 - TCE-AC - Analista de Controle Externo - Processamentos de Dados) Uma lista encadeada é uma coleção de nodos que, juntos, formam uma ordem linear. Se é possível os nodos se deslocarem em ambas as direções na lista, diz-se que se trata de uma lista simplesmente encadeada.**

Comentários:

Se é possível os nodos se deslocarem em ambas as direções na lista, diz-se que se trata de uma lista duplamente encadeada. **Gabarito: E**

10. **(CESPE - 2008 – HEMOBRÁS – Técnico de Informática) Uma estrutura do tipo lista, em que é desejável percorrer o seu conteúdo nas duas direções indiferentemente, é denominado lista duplamente encadeada.**

Comentários:

Perfeito, é exatamente isso! **Gabarito: C**

11. **(CESPE - 2010 – TRE/MT – Analista de Sistemas – C) Uma lista duplamente encadeada é uma lista em que o seu último elemento referencia o primeiro.**

Comentários:

Não, isso se trata de uma Lista Circular! **Gabarito: E**

12. **(CESPE - 2006 – SGA/AC – Analista de Sistemas) O principal problema da alocação por lista encadeada é a fragmentação.**

Comentários:

Não! Em geral, a alocação por lista encadeada elimina a fragmentação. **Gabarito: E**

13. (CESPE - 2008 – MCT – Analista de Sistemas) O armazenamento de arquivos em disco pode ser feito por meio de uma lista encadeada, em que os blocos de disco são ligados por ponteiros. A utilização de lista encadeada elimina completamente o problema de fragmentação interna.

Comentários:

Não, ela elimina a fragmentação externa! **Gabarito: E**

14. (CESPE - 2009 – FINEP – Analista de Sistemas) Uma lista encadeada é uma representação de objetos na memória do computador que consiste de uma sequência de células em que:

a) cada célula contém apenas o endereço da célula seguinte.

b) cada célula contém um objeto e o tipo de dados da célula seguinte.

c) o último elemento da sequência aponta para o próximo objeto que normalmente possui o endereço físico como not null.

d) cada célula contém um objeto de algum tipo e o endereço da célula seguinte.

e) a primeira célula contém o endereço da última célula.

Comentários:

Cada célula contém um objeto de algum tipo e o endereço da célula seguinte! **Gabarito: D**

15. (CESPE - 2010 – BASA – Analista de Sistemas) Em uma lista encadeada, o tempo de acesso a qualquer um de seus elementos é constante e independente do tamanho da estrutura de dados.

Comentários:

Claro que não! Em uma busca sequencial, o tempo de acesso é proporcional ao tamanho da estrutura de dados, i.e., quanto mais ao final da lista, maior o tempo de acesso! Por que, professor? Porque a lista é acessada sequencialmente (ou seja, é preciso percorrer elemento por elemento) e, não, diretamente (ou seja, pode-se acessar de modo direto). Um vetor tem acesso direto, portanto seu tempo de acesso é igual independentemente do tamanho da estrutura. **Gabarito: E**

16. (CESPE - 2010 – INMETRO – Analista de Sistemas – C) Considere que Roberto tenha feito uso de uma lista encadeada simples para programar o armazenamento e o posterior acesso aos dados acerca dos equipamentos instalados em sua empresa. Considere, ainda, que, após realizar uma consulta acerca do equipamento X, Roberto precisou acessar outro equipamento Y que se encontrava, nessa lista, em posição anterior ao equipamento X. Nessa

situação, pela forma como os ponteiros são implementados em uma lista encadeada simples, o algoritmo usado por Roberto realizou a consulta ao equipamento Y sem reiniciar a pesquisa do começo da lista.

Comentários:

Não! Infelizmente, ele teve que reiniciar a pesquisa a partir do primeiro elemento da lista, na medida em que ele não pode voltar. Por que, professor? Porque se trata de uma lista encadeada simples e, não, dupla. Portanto, a lista só é percorrida em uma única direção. **Gabarito: E**

17. (FCC - 2003 – TRE/AM – Analista de Sistemas) Os dados contidos em uma lista encadeada estão:

a) ordenados seqüencialmente.

b) sem ordem lógica ou física alguma.

c) em ordem física e não, necessariamente, em ordem lógica.

d) em ordem lógica e, necessariamente, em ordem física.

e) em ordem lógica e não, necessariamente, em ordem física.

Comentários:

A Ordem Física é sua disposição na memória do computador e a Ordem Lógica é como ela pode ser lida e entendida. Ora, a ordem em que ela se encontra na memória pouco importa, visto que cada sistema operacional e cada sistema de arquivos tem sua maneira. Portanto, trata-se da ordem lógica e, não, necessariamente física. **Gabarito: E**

18. (FCC - 2010 – DPE/SP – Analista de Sistemas) Uma estrutura de dados que possui três campos: dois ponteiros e campo de informação denomina-se:

a) lista encadeada dupla.

b) Lista encadeada simples.

c) pilha.

d) fila.

e) vetor.

Comentários:

Trata-se da Lista Encadeada Dupla: dois ponteiros (Ant e Prox) e um campo de informação. **Gabarito: A**

19. (CESPE - 2010 – TRE/MT – Analista de Sistemas) O algoritmo para inclusão de elementos em uma pilha é usado sem nenhuma alteração para incluir elementos em uma lista.

Comentários:

Galera... uma pilha pode ser implementada por meio de uma lista! Ademais, o algoritmo para inclusão de elementos de ambas necessita do primeiro elemento (ou topo). Portanto, questão correta! **Gabarito: C**

QUESTÕES COMENTADAS - PILHAS - MULTIBANCAS

1. **(CESPE - 2011 - FUB - Analista de Tecnologia da Informação - Específicos) As pilhas são listas encadeadas cujos elementos são retirados e acrescentados sempre ao final, enquanto as filas são listas encadeadas cujos elementos são retirados e acrescentados sempre no início.**

Comentários:

Bem... o que é o final de uma Pilha? Pois é, não se sabe! O que existe é o Topo da Pilha, de onde sempre são retirados e acrescentados elementos. Em Filas, elementos são retirados do início e acrescentados no final. **Gabarito: E**

2. **(CESPE - 2013 - INPI - Analista de Planejamento - Desenvolvimento e Manutenção de Sistemas) Na estrutura de dados do tipo lista, todo elemento novo que é introduzido na pilha torna-se o elemento do topo.**

Comentários:

Que questão confusa! Vamos comigo: vocês sabem muito bem que filas e pilhas são considerados espécies de listas. A questão inicialmente fala de uma lista, mas depois ela menciona uma pilha – podemos inferir, então, que se trata de uma lista do tipo pilha. Em uma pilha, todo elemento novo que é introduzido torna-se o elemento do topo, logo... questão correta!

Bem, esse foi o Gabarito Preliminar, mas a banca mudou de opinião e, no Gabarito Definitivo, permaneceu como errada. E a justificativa dela foi: "A ausência de especificação do tipo de lista no item torna correta a informação nele apresentada, razão pela qual se opta pela alteração de seu gabarito". Vejam que bizarro: se torna correta a informação apresentada, o gabarito definitivo deveria ser C e, não, E.

Além disso, a questão informa em sua segunda parte que se trata de uma pilha. Logo, não há que se falar em "ausência de especificação do tipo de lista". Enfim, questão péssima, horrível e mal redigida :(**Gabarito: E**

3. **(CESPE - 2012 - TJ-RO - Analista Judiciário - Analista de Sistemas Suporte – E) Visitas a sítios armazenadas em um navegador na ordem last-in-first-out é um exemplo de lista.**

Comentários:

Não! Last-In-First-Out (LIFO) é um exemplo de Pilha! Cuidado, pilhas podem ser implementadas como listas, mas esse não é o foco da questão. **Gabarito: E**

4. (ESAF - 2013 - DNIT - Analista Administrativo - Tecnologia da Informação) Assinale a opção correta relativa às operações básicas suportadas por pilhas.

a) Push: insere um novo elemento no final da pilha.

b) Pop: adiciona elementos ao topo da pilha.

c) Pull: insere um novo elemento no interior da pilha.

d) Top: transfere o último elemento para o topo da pilha.

e) Top: acessa o elemento posicionado no topo da pilha.

Comentários:

(a) Não, é no topo; (b) Não, remove do topo; (c) Não, não existe; (d) Não, simplesmente acessa e consulta o elemento do topo; (e) Perfeito! **Gabarito: E**

5. (FCC - 2012 – TST - Analista de Sistemas – C) As pilhas e as filas são estruturas de dados essenciais para os sistemas computacionais. É correto afirmar que a pilha é conhecida como lista FIFO - First In First Out.

Comentários:

Não! Pilha é LIFO e Fila é FIFO. **Gabarito: E**

6. (FCC - 2012 – TRE/CE - Analista de Sistemas) Sobre pilhas é correto afirmar:

a) Uma lista LIFO (Last-In/First-Out) é uma estrutura estática, ou seja, é uma coleção que não pode aumentar e diminuir durante sua existência.

b) Os elementos na pilha são sempre removidos na mesma ordem em que foram inseridos.

c) Uma pilha suporta apenas duas operações básicas, tradicionalmente denominadas push (insere um novo elemento no topo da pilha) e pop (remove um elemento do topo da pilha).

d) Cada vez que um novo elemento deve ser inserido na pilha, ele é colocado no seu topo e, em qualquer momento, apenas aquele posicionado no topo da pilha pode ser removido.

e) Sendo P uma pilha e x um elemento qualquer, a operação Push(P,x) diminui o tamanho da pilha P, removendo o elemento x do seu topo.

Comentários:

(a) Não, é uma estrutura dinâmica; (b) Não, é na ordem inversa (último a entrar é o primeiro a sair); (c) Não, há também Top ou Check, que acessar e consulta o elemento do topo; (e) Push é a operação de inserção de novos elementos na pilha, portanto aumenta seu tamanho adicionando o elemento x no topo.

E a letra D? Vamos lá! Sabemos que uma pilha é um tipo de lista - o que muda é apenas a perspectiva. Como assim? Eu vejo um monte de elementos em sequência. Ora, se eu coloco uma regra em que o primeiro elemento a entrar é o primeiro a sair, chamamos essa lista de fila; se eu coloco uma regra em que o primeiro elemento a entrar é o último a sair, chamamos essa lista de pilha.

Ok! Dito isso, o algoritmo é exatamente o mesmo, eu vou simplesmente mudar a perspectiva e a minha visão sobre a estrutura. Bacana? **Gabarito: D**

7. (CESPE - 2011 - EBC - Analista - Engenharia de Software) As pilhas, também conhecidas como listas LIFO ou PEPS, são listas lineares em que todas as operações de inserção e remoção de elementos são feitas por um único extremo da lista, denominado topo.

Comentários:

Não! LIFO é similar a UEPS (Último a Entrar, Primeiro a Sair). PEPS refere-se a Primeiro a Entrar, Primeiro a Sair, ou seja, FIFO. **Gabarito: E**

8. (VUNESP - 2011 - TJM-SP - Analista de Sistemas - Judiciário) Lista do tipo LIFO (Last in, First Out) e lista do tipo FIFO (Firstin,First Out) são, respectivamente, características das estruturas de dados denominadas:

a) Fila e Pilha.

b) Pilha e Fila.

c) Grafo e Árvore.

d) Árvore e Grafo.

e) Árvore Binária e Árvore Ternária.

Comentários:

E aí, já está automático para responder? Tem que ser automática: Pilha (LIFO) e Fila (FIFO). **Gabarito: B**

9. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Arquitetura de Tecnologia) A definição da estrutura pilha permite a inserção e a eliminação de itens, de modo que uma pilha é um objeto dinâmico, cujo tamanho pode variar constantemente.

Comentários:

Essa questão é polêmica, porque é inevitável pensar em Pilhas Sequenciais (implementadas por vetores estáticos)! No entanto, é comum que as bancas tratem por

padrão Pilha como Pilha Encadeada (implementadas por listas dinâmicas). Dessa forma, a questão está perfeita! **Gabarito: C**

10. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) Na representação física de uma pilha sequencial, é necessário uso de uma variável ponteiro externa que indique a extremidade da lista linear onde ocorrem as operações de inserção e retirada de nós.

Comentários:

As Pilhas oferecem três operações básicas: push, que insere um novo elemento no topo da pilha; pop, que remove um elemento do topo da pilha; e top (também conhecida como check), que acessa e consulta o elemento do topo da pilha. Pilhas podem ser implementadas por meio de Vetores (Pilha Sequencial - Alocação Estática de Memória) ou Listas (Pilha Encadeada - Alocação Dinâmica de Memória).

Conforme vimos em aula, a questão trata de uma Pilha Sequencial (i.e., implementada por meio de Vetores). Dessa forma, não é necessário o uso de ponteiros – esse seria o caso de uma Pilha Encadeada. Eu posso realmente dizer que é suficiente, mas não posso afirmar que é necessária a utilização de um ponteiro externo. Eu até poderia dizer que é necessário o uso de um indicador, mas ele também não necessariamente será um ponteiro. Logo, discordo do gabarito! **Gabarito: C**

11. (CESPE - 2009 - ANAC - Técnico Administrativo - Informática) As operações de inserir e retirar sempre afetam a base de uma pilha.

Comentários:

Não, sempre afetam o topo da pilha! **Gabarito: E**

12. (FCC - 2009 - TRT - 16ª REGIÃO (MA) - Técnico Judiciário - Tecnologia da Informação) Pilha é uma estrutura de dados:

a) cujo acesso aos seus elementos segue tanto a lógica LIFO quanto a FIFO.

b) cujo acesso aos seus elementos ocorre de forma aleatória.

c) que pode ser implementada somente por meio de vetores.

d) que pode ser implementada somente por meio de listas.

e) cujo acesso aos seus elementos segue a lógica LIFO, apenas.

Comentários:

(a) Não, somente LIFO; (b) Não, somente pelo Topo; (c) Não, pode ser por listas; (d) Não, pode ser por vetores; (e) Perfeito, é exatamente isso. **Gabarito: E**

13. (CESPE - 2004 – STJ – Analista de Sistemas) Em geral, em uma pilha só se admite ter acesso ao elemento localizado em seu topo. Isso se adapta perfeitamente à característica das seqüências em que só o primeiro componente é diretamente acessível.

Comentários:

Perfeito, é exatamente isso – muda-se apenas a perspectiva! **Gabarito: C**

14. (CESPE - 2004 – STJ – Analista de Sistemas) A seguir, está representada corretamente uma operação de desempilhamento em uma pilha de nome p.

se p.topo = 0 então

nada {pilha vazia}

senão p.topo ← p.topo-1

Comentários:

Galera, a questão deu uma vaciladinha! O ideal seria dizer p.topo = null, mas vamos subentender que foi isso mesmo que ele quis dizer. Desse modo, se não tem topo, é porque a pilha está vazia. Se tiver topo, então o topo será o elemento anterior ao topo. O que ocorreu? Eu desempilhei a pilha! **Gabarito: C**

15. (CESPE - 2010 – TRE/MT - Analista de Sistemas – A) O tipo nó é inadequado para implementar estruturas de dados do tipo pilha.

Comentários:

Não! Uma pilha pode ser implementada por meio de um vetor ou de uma lista. Nesse último caso, temos tipos nós! **Gabarito: E**

16. (FGV – 2015 – DPE/MT – Analista de Sistemas) Assinale a opção que apresenta a estrutura de dados na qual o primeiro elemento inserido é o último a ser removido.

a) Árvore

b) Fila

c) Pilha

d) Grafo

e) Tabela de dispersão

Comentários:

Também conhecida como Lista LIFO (Last In First Out), basta lembrar de uma pilha de pratos esperando para serem lavados, i.e., o último a entrar é o primeiro a sair. A ordem em que os pratos são retirados da pilha é o oposto da ordem em que eles são colocados sobre a pilha e, como consequência, apenas o prato do topo da pilha está acessível.

Conforme vimos em aula, trata-se da Pilha. **Gabarito: C**

17. (FCC – 2012 – MPE/AP – Técnico Ministerial - Informática) Nas estruturas de dados,

a) devido às características das operações da fila, o primeiro elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como LIFO.

b) as pilhas são utilizadas para controlar o acesso de arquivos que concorrem a uma única impressora.

c) a fila é uma lista linear na qual as operações de inserção e retirada ocorrem apenas no início da lista.

d) a pilha é uma lista linear na qual as operações de inserção e retirada são efetuadas apenas no seu topo.

e) devido às características das operações da pilha, o último elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como FIFO.

Comentários:

(a) as filas não são LIFO, mas sim FIFO, ou seja, o primeiro elemento da fila será, na verdade, o primeiro a ser retirado. Só pensarmos numa fila de banco, se alguém chega por último e é atendido primeiro, ficaria bem bravo, e vocês?? :D Item errado; (b) os trabalhos que chegam a uma impressora devem ser do tipo FIFO, ou seja, o primeiro trabalho enviado deve ser o primeiro a ser impresso. Item errado; (c) na fila os elementos são incluídos numa das extremidades e retirados da outra. Item errado; (d) na pilha as operações de inclusão na pilha quanto de retirada acontecem numa mesma extremidade. A extremidade escolhida é o topo da pilha. Item certo; (e) na verdade essas características são das filas. Item errado. **Gabarito: D**

QUESTÕES COMENTADAS - FILAS - MULTIBANCAS

1. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Análise de Sistemas) Em um programa existe a necessidade de guardar todas as alterações feitas em determinado dado para que seja possível desfazer alterações feitas ao longo de toda a sua existência. Nessa situação, a estrutura de dados mais adequada para o armazenamento de todas as alterações citadas seria uma fila.**

Comentários:

Não! Pensem comigo: eu faço uma atividade, depois outra, depois mais uma e, por fim, mais outra. Se eu desejo desfazer a última atividade realizada para retornar a um estado anterior, eu preciso de uma pilha. Dessa forma, resgata-se o último estado válido e, não, o primeiro. **Gabarito: E**

2. **(CESPE - 2012 – TST – Analista de Sistemas – A) As pilhas e as filas são estruturas de dados essenciais para os sistemas computacionais. É correto afirmar que a fila é conhecida como lista LIFO - Last In First Out.**

Comentários:

Não, Fila é FIFO! **Gabarito: E**

3. **(CESPE - 2012 - TRE-RJ - Técnico Judiciário - Programação de Sistemas) As filas são estruturas com base no princípio LIFO (last in, first out), no qual os dados que forem inseridos primeiro na fila serão os últimos a serem removidos. Existem duas funções que se aplicam a todas as filas: PUSH, que insere um dado no topo da fila, e POP, que remove o item no topo da fila.**

Comentários:

Não, isso é uma Pilha (LIFO). **Gabarito: E**

4. **(FCC - 2012 - MPE-AP – Analista de Sistemas - A) Nas estruturas de dados, devido às características das operações da fila, o primeiro elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como LIFO.**

Comentários:

Não, será o primeiro a ser retirado – são do tipo FIFO! **Gabarito: E**

5. **(FCC - 2012 - MPE-AP – Analista de Sistemas - C) Nas estruturas de dados, a fila é uma lista linear na qual as operações de inserção e retirada ocorrem apenas no início da lista.**

Comentários:

Não, isso é a definição de Pilha! **Gabarito: E**

6. **(FCC - 2012 - TRE-SP - Analista Judiciário - Análise de Sistemas – D) Pela definição de fila, se os elementos são inseridos por um extremo da lista linear, eles só podem ser removidos pelo outro.**

Comentários:

Exato! Essa é a definição de fila: insere-se por um extremo e remove-se por outro. **Gabarito: C**

7. **(FCC - 2011 - TRT - 19ª Região (AL) - Analista Judiciário - Tecnologia da Informação) FIFO refere-se a estruturas de dados do tipo:**

a) fila.

b) árvore binária.

c) pilha.

d) matriz quadrada.

e) cubo.

Comentários:

Trata-se da Fila! **Gabarito: A**

8. **(ESAF - 2010 - CVM - Analista de Sistemas - prova 2) Uma fila é um tipo de lista linear em que:**

a) as inserções são realizadas em um extremo e as remoções no outro extremo.

b) as inserções e remoções são realizadas em um mesmo extremo.

c) podem ser realizadas apenas inserções.

d) a inserção de um elemento requer a remoção de outro elemento.

e) a ordem de saída não corresponde à ordem de entrada dos elementos.

Comentários:

As inserções são realizadas em um extremo e as remoções são realizadas no outro extremo, por isso é FIFO! **Gabarito: A**

9. (CESPE - 2010 - DETRAN-ES - Analista de Sistemas) No armazenamento de dados pelo método FIFO (first in - first out), a estrutura de dados é representada por uma fila, em cuja posição final ocorrem inserções e, na inicial, retiradas.

Comentários:

Perfeito! Basta lembrar de uma fila: o primeiro a entrar é o primeiro a sair. **Gabarito: C**

10. (CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) Entre alguns tipos de estrutura de dados, podem ser citados os vetores, as pilhas e as filas.

Comentários:

Perfeito, são todos exemplos de estruturas de dados! **Gabarito: C**

11. (CESPE - 2004 – SES/PA – Analista de Sistemas) Uma estrutura mais geral que as pilhas e filas é o deque, em que as inserções, retiradas e acessos são permitidos em ambas as extremidades.

Comentários:

Perfeito, deques permitem todas essas operações! **Gabarito: C**

12. (CESPE - 2009 – TCE/AC – Analista de Sistemas – D) Um deque (double ended queue) requer inserção e remoção no topo de uma lista e permite a implementação de filas com algum tipo de prioridade. A implementação de um deque, geralmente é realizada com a utilização de uma lista simplesmente encadeada.

Comentários:

Não, pode ser do início ou fim da lista! De fato, permite a implementação de filas com algum tipo de prioridade, mas geralmente é realizada com a utilização de filas duplamente encadeadas. **Gabarito: E**

13. (FCC - 2007 – TRT/23 – Analista de Sistemas) Uma estrutura de dados com vocação de FIFO de duplo fim e que admite a rápida inserção e remoção em ambos os extremos é:

a) uma pilha.

b) uma splay tree.

c) um deque.

d) uma lista linear.

e) uma árvore AVL.

Comentários:

Trata-se de um Deque – fila duplamente encadeada! **Gabarito: C**

14. (CESPE - 2004 – PBV/RR - Analista de Sistemas) As filas com prioridade são listas lineares nas quais os elementos são pares da forma (qi, pi), em que q é o elemento do tipo base e p é uma prioridade. Elas possuem uma política de fila do tipo FIFO (first in first out) entre os elementos de mesma prioridade.

Comentários:

Perfeito! É assim que funciona a prioridade em conjunto com filas. **Gabarito: C**

15. (CESPE - 2004 – STJ – Analista de Sistemas) A seguir, está representada corretamente uma operação de retirada em uma fila de nome f.

se f.começo = nil então

erro {fila vazia}

senão j ← f.começo.info

Comentários:

Não, o correto seria:

se f.começo = nil então

erro {fila vazia}

senão f.começo ← f.começo.anterior

Por que, professor? As duas primeiras linhas estão apenas dizendo que se o primeiro elemento da fila for Null (ou Nil), vai dar erro porque a fila está vazia, logo não há como retirar elementos de uma fila vazia. Agora vejam a última linha: ele atribui a uma variável j o valor f.começo.info. Na verdade, ele simplesmente está colocando em j os dados do primeiro elemento da fila, mas a questão pede o código para retirar um elemento da lista e não para capturar os dados do primeiro elemento. Na resposta, eu coloco que f.começo, i.e., o primeiro elemento da lista vai ser f.começo.anterior, ou seja, temos um novo primeiro elemento e eu retirei aquele elemento anterior. **Gabarito: E**

16. **(FCC - 2016 - Prefeitura de Teresina - PI - Analista Tecnológico - Analista de Suporte Técnico) Considerando uma estrutura de dados do tipo fila, e a seguinte sequência de comandos sobre essa fila (sendo que o comando Push representa uma inserção de elemento e o comando Pop representa uma exclusão de elemento) e considerando também que a fila estava inicialmente vazia:**

Push 3, Push 5, Pop 3, Push 7, Pop 5, Push 9, Push 8

Após a execução dessa sequência de comandos, o conjunto de elementos que resulta na fila é:

a) 3 – 5 – 7 – 9 – 8.

b) 7 – 9 – 8 – 3 – 5.

c) 3 – 3 – 5 – 5 – 7 – 9 – 8.

d) 7 – 9 – 8.

e) 3 – 5 – 3 – 7 – 5 – 9 – 8.

Comentários:

Como a questão nem pediu a ordem, ficou bem fácil. Push inclui e pop retira. Se há dois pop's, os elementos 3 e 5 são removidos da fila, sobrando 7, 9 e 8. **Gabarito: D**

17. **(FCC - 2016 – TRT - 23ª REGIÃO (MT) - Técnico Judiciário - Tecnologia da Informação) Estruturas de dados básicas, como as pilhas e filas, são usadas em uma gama variada de aplicações. As filas, por exemplo, suportam alguns métodos essenciais, como o:**

a) enqueue(x), que insere o elemento x no fim da fila, sobrepondo o último elemento.

b) dequeue(), que remove e retorna o elemento do começo da fila; um erro ocorrerá se a fila estiver vazia.

c) push(x), que insere o elemento x no topo da fila, sem sobrepor nenhum elemento.

d) pop(), que remove o elemento do início da fila e o retorna, ou seja, devolve o último elemento inserido.

e) top(), que retorna o elemento do fim da fila sem removê-lo; um erro ocorrerá se a fila estiver vazia.

Comentários:

Queue = fila! Stack = pilha! Sendo assim, o que seria "enqueue"? Para quem tem idade (assim como eu!! ☹) deve se lembrar do famoso Winamp (ah, minha época de ouvir mp3!). Quanto clicávamos no botão direito de uma música, sempre tinha a opção "enqueue in Winamp", ou seja, incluir na fila de reprodução. Enqueue, portanto, inclui na fila, enquanto dequeue remove! **Gabarito: B**

18. (FCC - 2017 – TRE/BA - Analista de Sistemas) A estrutura que, além de ser similar à fila, é apropriada para ampliar as características desta, permitindo inserir e retirar elementos tanto do início quanto do fim da fila, é o(a):

a) árvore.

b) lista duplamente encadeada.

c) deque.

d) fila circular.

e) pilha

Comentários:

Trata-se de um Deque (Double Ended Queue). Deque é uma estrutura de dados similar à fila e que permite que elementos possam ser adicionados ou removidos da frente (cabeça) ou de trás (cauda). Qual a diferença entre uma lista duplamente encadeada e um deque? Um deque gerencia elementos como um vetor, fornece acesso aleatório e tem quase a mesma interface que um vetor.

Uma lista duplamente encadeada se difere de um deque por não fornecer acesso aleatório aos elementos, i.e., para acessar o quinto elemento, você deve necessariamente navegar pelos quatro primeiros elementos – logo a lista é mais lenta nesse sentido. Existem outras diferenças, mas essa é a diferença fundamental entre essas duas estruturas. Bacana?
Gabarito: C

QUESTÕES COMENTADAS - ESTRUTURAS DE DADOS - ÁRVORE #42 - MULTIBANCAS

1. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As operações de busca em uma árvore binária não a alteram, enquanto operações de inserção e remoção de nós provocam mudanças sistemáticas na árvore.**

Comentários:

Perfeito! Operações de Busca não alteram nenhuma estrutura de dados. Já Operações de Inserção e Remoção podem provocar diversas mudanças estruturais. **Gabarito: C**

2. **(CETAP - 2010 - AL-RR - Analista de Sistemas - A) Uma árvore binária é aquela que tem como conteúdo somente valores binários.**

Comentários:

Não! Uma árvore binária é aquela que tem, no máximo, grau 2! **Gabarito: E**

3. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) O tipo de dados árvore representa organizações hierárquicas entre dados.**

Comentários:

Perfeito, observem que alguns autores tratam Tipos de Dados como sinônimo de Estruturas de Dados. **Gabarito: C**

4. **(CETAP - 2010 - AL-RR - Analista de Sistemas - B) Uma árvore é composta por duas raízes, sendo uma principal e a outra secundária.**

Comentários:

Não, uma árvore possui somente um nó raíz! **Gabarito: E**

5. **(CESPE - 2010 - DETRAN-ES - Analista de Sistemas) Denomina-se árvore binária a que possui apenas dois nós.**

Comentários:

Não, árvore binária é aquela em que cada nó tem, no máximo, dois filhos! **Gabarito: E**

6. **(FUNCAB - 2010 - SEJUS-RO - Analista de Sistemas - II) Árvores são estruturas de dados estáticas com sua raiz representada no nível um.**

Comentários:

Não! Árvores são estruturas dinâmicas e sua raiz, em geral, é representada no nível 0 (mas depende de autor para autor). **Gabarito: E**

7. **(CESPE - 2009 - ANAC - Especialista em Regulação - Economia) Considerando-se uma árvore binária completa até o nível 5, então a quantidade de folhas nesse nível será 24.**

Comentários:

Não! A quantidade de folhas em um determinado nível – considerando a raiz como nível 0 –, é dada pela fórmula 2d, portanto 25. **Gabarito: E**

8. **(CESPE - 2009 – ANAC - Analista de Sistemas) Uma árvore binária completa até o nível 10 tem 2.047 nós.**

Comentários:

Se possui 10 níveis, possui (2d+1 -1): 2047 nós! **Gabarito: C**

9. **(FGV – 2015 – DPE/MT – Analista de Sistemas) No desenvolvimento de sistemas, a escolha de estruturas de dados em memória é especialmente relevante. Dentre outras classificações, é possível agrupar essas estruturas em lineares e não lineares, conforme a quantidade de sucessores e antecessores que os elementos da estrutura possam ter. Assinale a opção que apresenta, respectivamente, estruturas de dados lineares e não lineares.**

a) Tabela de dispersão e fila.
b) Estrutura de seleção e pilha.
c) Pilha e estrutura de seleção.
d) Pilha e árvore binária de busca.
e) Fila e pilha.

Comentários:

Além dessa classificação, existe outra também importante: Estruturas Lineares e Estruturas Não-Lineares. As Estruturas Lineares são aquelas em que cada elemento pode ter um único predecessor (exceto o primeiro elemento) e um único sucessor (exceto o último elemento). Como exemplo, podemos citar Listas, Pilhas, Filas, Arranjos, entre outros.

Já as Estruturas Não-Lineares são aquelas em que cada elemento pode ter mais de um predecessor e/ou mais de um sucessor. Como exemplo, podemos citar Árvores, Grafos e

Tabelas de Dispersão. Essa é uma classificação muito importante e muito simples de entender. Pessoal, vocês perceberão que esse assunto é cobrado de maneira superficial na maioria das questões, mas algumas são nível doutorado!

Conforme vimos em aula, trata-se de pilha e árvore respectivamente. **Gabarito: D**

10. (CESPE - 2010 – TRE/MT – Analista de Sistemas – B) As listas, pilhas, filas e árvores são estruturas de dados que têm como principal característica a sequencialidade dos seus elementos.

Comentários:

Não! Árvores ntêm como principal característica a sequencialidade dos seus elementos.
Gabarito: E

11. (FCC - 2012 - MPE-AP - Analista Ministerial - Tecnologia da Informação – A) A árvore é uma estrutura linear que permite representar uma relação de hierarquia. Ela possui um nó raiz e subárvores não vazias.

Comentários:

Árvore é uma estrutura linear? Não, hierárquica! **Gabarito: E**

12. (CESPE - 2010 – TRE/MT - Analista de Sistemas – E) O uso de recursividade é totalmente inadequado na implementação de operações para manipular elementos de uma estrutura de dados do tipo árvore.

Comentários:

Pelo contrário, é fundamental para implementação de operações. **Gabarito: E**

13. (FCC - 2011 - TRT - 19ª Região (AL) - Técnico Judiciário - Tecnologia da Informação) Em uma árvore binária, todos os nós têm grau:

a) 2.
b) 0, 1 ou 2.
c) divisível por 2.
d) maior ou igual a 2.
e) 0 ou 1.

Comentários:

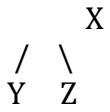
Olha a pegadinha! Todos os nós têm grau 0 (Folha), 1 (Único filho) ou 2 (Dois filhos).
Gabarito: B

14. (CESPE - 2011 – STM – Analista de Sistemas) Enquanto uma lista encadeada somente pode ser percorrida de um único modo, uma árvore binária pode ser percorrida de muitas maneiras diferentes.

Comentários:

Galera, pense em uma árvore bem simples com um pai (raiz) e dois filhos. O Modo Pré-fixado vai ler primeiro a raíz, depois a sub-árvore da esquerda e depois a sub-árvore da direita. O Modo In-fixado vai ler primeiro a sub-árvore da esquerda, depois a raiz e depois a sub-árvore da direita. O Modo Pós-fixado vai ler primeiro a sub-árvore da esquerda, depois a sub-árvore da direita e depois a raiz.

Vamos resumir: o modo de percorrimento de uma árvore pode obedecer três regras de acordo com a posição da raiz (pai): pré-fixado (raiz, esquerda, direita); in-fixado (esquerda, raiz, direita); e pós-fixada (esquerda, direita, raiz). Vamos agora ver uma árvore de exemplo:



- Percorrimento Pré-fixado: X Y Z
- Percorrimento In-fixado: Y X Z
- Percorrimento Pós-fixado: Y Z X

Logo, uma árvore pode ser percorrida de modo pré-fixado, in-fixado e pós-fixado.
Gabarito: C

15. (FCC - 2016 - Prefeitura de Teresina - PI - Analista Tecnológico - Analista de Suporte Técnico) Considerando a estrutura de dados denominada árvore,

a) a sua altura é definida como a profundidade média de todos os seus vértices.

b) um vértice com um ou dois filhos é denominado folha.

c) cada nó tem no mínimo dois filhos em uma árvore binária.

d) as folhas de uma árvore binária completa podem ter profundidades distintas entre si.

e) a profundidade de um vértice em uma árvore é definida como o comprimento da raiz da árvore até esse vértice.

Comentários:

(a) Errado! Altura é definida pela folha mais profunda; (b) Errado! Folhas não possuem filhos, do contrário não seriam folhas (as arves... somo nozes... péssimo, professor :P); (c) Errado! Os nós de uma árvore binaria podem ter NO MÁXIMO dois filhos, e não no

mínimo; (d) Errado! Uma árvore binária completa é aquele em que todos os nós internos possuem seus dois filhos (máximo). Desse modo, todas as folhas devem ter a mesma profundidade; (e) Certo! **Gabarito: E**

16. (CESPE – 2017 – TRE/BA - Analista de Sistemas) No estabelecimento de uma estrutura hierárquica, foi definida a seguinte árvore binária S:

S = (12(10(9(8))(11))(14(13)(15)))

Considerando o resultado da operação de exclusão do nó 12, assinale a opção que corresponde a nova estrutura da árvore S.

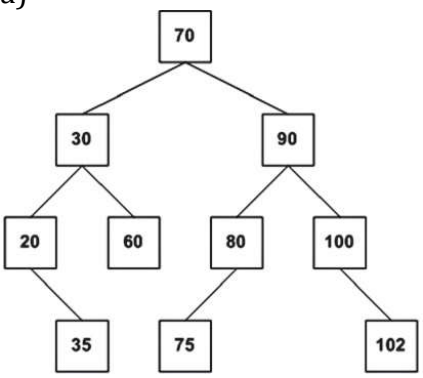
a) (10(9(8))(11(14(13)(15)))
b) (11(9(8)(10))(14(13)(15)))
c) (11(10(9(8))(14(13)(15)))
d) (13(10(9)(11))(14(15)))
e) (13(11(9)(10))(14(15)))

Comentários:

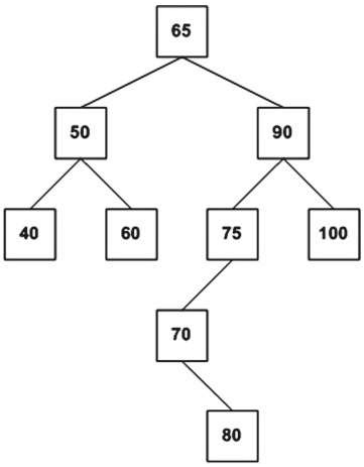
Conforme vimos em aula, a questão já inicia com um problema grave: ela não especifica qual tipo de árvore. Ainda assim, ela está errada e é fácil descobrir que não pode ser a Letra C (Gabarito Preliminar) porque a quantidade de parênteses abertos e fechados são diferentes – logo jamais poderia ser essa a resposta. Concluímos, então, que essa questão não possui resposta alguma, visto que falta fechar um parêntese. A Letra C (11(10(9(8))(14(13)(15))) está quase certa, mas faltou um parêntese: (11(10(9(8)))(14(13)(15))). **Gabarito: X**

17. (CESGRANRIO – 2012 – PETROBRÁS - Analista de Sistemas) Qual figura representa uma árvore AVL?

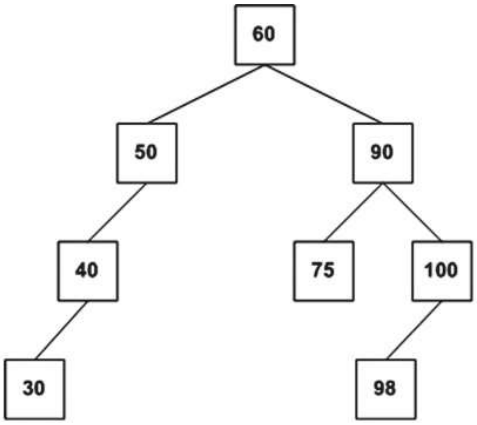
a)



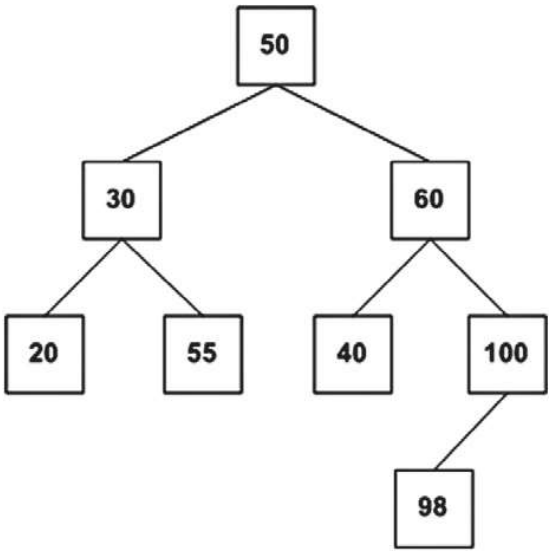
b)


65
50
90
40
60
75
100
70
80

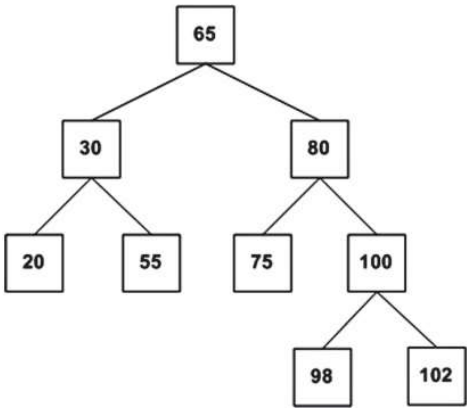
c)


60
50
90
40
75
100
30
98

d)


50
30
60
20
55
40
100
98

e)



Comentários:

Vamos relembrar dois conceitos: (1) Uma Árvore AVL é uma árvore binária de busca em que, para todos os seus nós, a diferença da altura da subárvore da esquerda para a altura da subárvore da direita deve ser no máximo 1. (2) Uma Árvore Binária de Busca é aquela em que, para todos os nós, a subárvore da esquerda possui um valor menor que a subárvore da direita. Dito isso, vamos analisar agora as opções:

(a) Errado. Não é uma árvore binária de busca, visto que o 35 é maior que 30 e está na subárvore da esquerda;

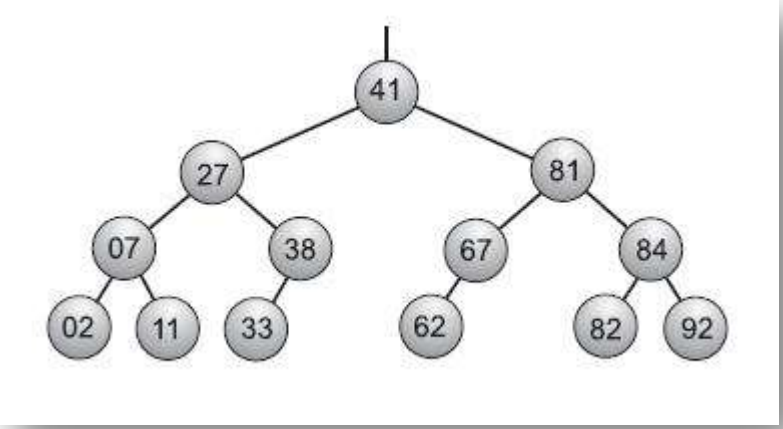
(b) Errado. Não é uma árvore binária de busca, visto que o 80 é maior que 75 e está na subárvore da esquerda;

(c) Errado. Observem que se trata realmente de uma árvore binária de busca, porém está desbalanceada. A diferença de altura da subárvore da esquerda de 50 e a subárvore da direita de 50 é 2 (que é maior do 1), portanto não é uma Árvore AVL.

(d) Errado. Não é uma árvore binária de busca, visto que o 55 é maior que 50 e está na subárvore da esquerda;

(e) Correto. Trata-se de árvore binária de busca e está completamente balanceada – requisitos para ser definida como uma Árvore AVL. **Gabarito: E**

18. (CESGRANRIO – 2006 – DECEA - Analista de Sistemas) Suponha a seguinte árvore AVL.



A inserção do elemento 30 nessa árvore:

a) aumenta a profundidade da árvore após uma rotação.
b) provoca uma rotação à direita.
c) deixa os nós 02 e 07 no mesmo nível.
d) altera a raiz da árvore (nó 41).
e) torna o nó 33 pai do nó 27.

Comentários:

Vamos lá! A questão quer inserir 30. Onde ele entraria? À esquerda do 33! No entanto, isso tornaria a árvore desbalanceada. Por que? Porque a árvore à esquerda do nó 38 teria altura 2 e o nó à direita do nó 38 teria altura 0 – a diferença seria 2, que é maior do que 1. Podemos notar que se trata de um Caso Esquerda-Esquerda, logo temos que fazer uma rotação simples para direita.

Rotacionamos o Ramo 38-33 para direita. Dessa forma, teríamos o 33 no lugar do 38 e o 38 à direita do 33. Pronto! Agora vamos analisar as opções: (a) Errado. A profundidade permanece a mesma; (b) Correto. Foi isso que nós fizemos; (c) Errado. Isso não faz nenhum sentido; (d) Errado. Isso também não faz nenhum sentido; (e) Errado. Também nenhum sentido. **Gabarito: B**

19. (CESPE – 2012 – TJ/RO - Analista de Sistemas) Assinale a opção em que é apresentado exemplo de estrutura de informação do tipo abstrata, balanceada, não linear e com relacionamento hierárquico.

a) lista duplamente encadeada
b) árvore binária
c) pilha
d) árvore AVL
e) deque

Comentários:

Tipo abstrato? Todos são! Não Linear? Árvore Binária e Árvore AVL! Com relacionamento hierárquico? Árvore Binária e Árvore AVL! Balanceada? Somente a Árvore AVL. **Gabarito: D**

20. (FCC – 2008 – TRT/18 - Analista de Sistemas) Árvore AVL balanceada em altura significa que, para cada nó da árvore, a diferença entre as alturas das suas sub-árvores (direita e esquerda) sempre será:

a) menor ou igual a 2.
b) igual a 0 ou -1.
c) maior que 1.
d) igual a 1.
e) igual a -1, 0 ou 1.

Comentários:

Por fim, vamos falar um pouco sobre Árvores AVL! Uma Árvore AVL (Adelson-Vesky e Landis) é uma Árvore Binária de Busca em que, para qualquer nó, a altura das subárvores da esquerda e da direita não podem ter uma diferença maior do que 1, portanto uma Árvore AVL é uma Árvore Binária de Busca autobalanceável. Calma que nós vamos ver isso em mais detalhes...

Conforme vimos em aula, trata-se da última opção. Lembrando que, na aula, nós falamos que era a diferença não podia ser maior que 1. Uma outra forma de escrever isso é dizer que o módulo da diferença entre as alturas não pode ser maior do 1. E outra forma de escrever isso é dizer que um nó só poder ter uma diferença de altura de suas subárvores de 1, 0 ou -1. Certinho? **Gabarito: E**

21. (CESGRANRIO – 2010 – PETROBRÁS - Analista de Sistemas) Uma árvore AVL é uma estrutura de dados muito usada para armazenar dados em memória. Ela possui algumas propriedades que fazem com que sua altura tenha uma relação muito específica com o número de elementos nela armazenados. Para uma folha, cuja altura é igual a um, tem-se uma árvore AVL com 6 nós.

Qual é a altura máxima que esta árvore pode ter?

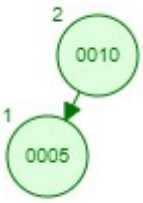
a) 6
b) 5
c) 4
d) 3
e) 2

Comentários:

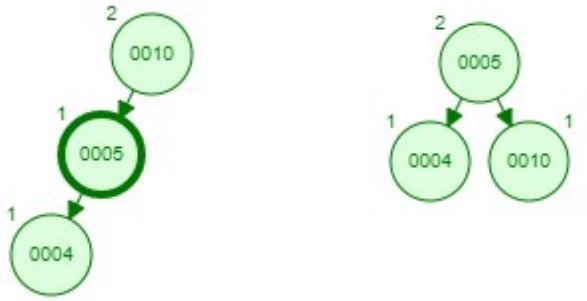
Galera, vocês precisam saber uma coisa: uma minoria dos autores considera que as folhas de uma árvore possuem altura 1 (sendo que a maioria considera que a altura de uma folha

é 0) – a questão afirma que, para uma folha, a altura é igual a 1. Dito isso, vamos analisar a questão! Ela te deu uma regra: você tem que construir uma Árvore AVL com 6.

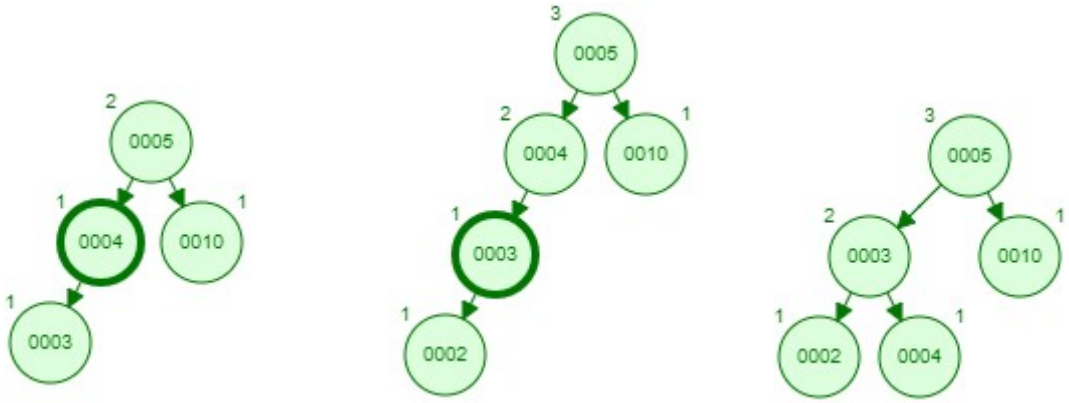
Em outras palavras, você tem que tentar construir a árvore com a maior altura possível, mas ela tem que estar balanceada. Então, eu começo com dois nós:



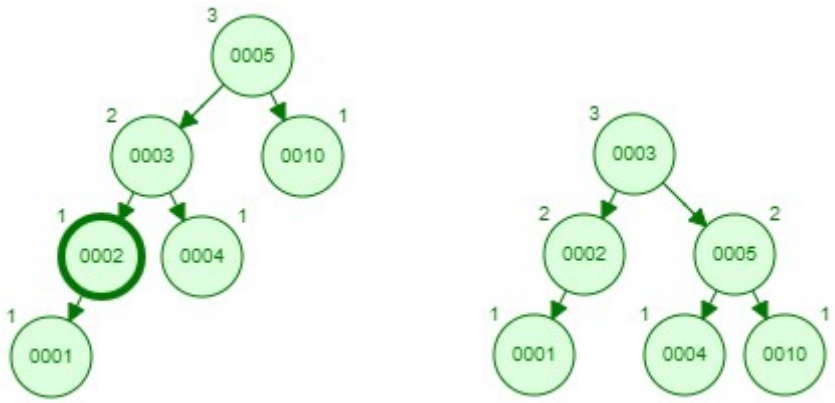
Como quero atingir a altura máxima, coloco mais um nó abaixo de 0005 e a árvore ficará com altura 3, mas ficará desbalanceada. Após balancear, fica como na direita:



Preciso encontrar a altura máxima, então coloco mais um nó (0003) e me sobrará mais duas tentativas. Se eu colocar mais um abaixo de 0003, ficará desbalanceado:



Vejam na imagem acima! Coloquei 0003 e não desbalanceou; coloquei 0002, desbalanceou e na última eu tive que rebalancear. Vamos inserir o último:



Pronto! Com seis nós, a altura máxima que atingiremos será 3. Portanto, a resposta para nossa questão é Letra D. **Gabarito: D**

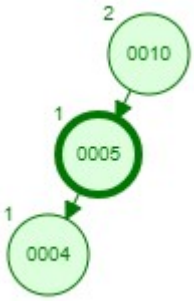
22. (CESGRANRIO – 2011 – PETROBRÁS - Analista de Sistemas) Uma árvore AVL é uma árvore binária de busca autobalanceada que respeita algumas propriedades fundamentais. Como todas as árvores, ela tem uma propriedade chamada altura, que é igual ao valor da altura de sua raiz.

Sabendo que a altura de uma folha é igual a um e que a altura de um nó pai é igual ao máximo das alturas de seus filhos mais um, qual estrutura NÃO pode representar uma árvore AVL?

a) Uma árvore vazia
b) Uma árvore com dois nós
c) Uma árvore com três nós e altura igual a dois
d) Uma árvore com três nós e altura igual a três
e) Uma árvore com seis nós e altura igual a três

Comentários:

(a) Errado, uma árvore vazia é uma Árvore AVL; (b) Errado, é impossível uma árvore com dois nós não ser uma Árvore AVL; (c) Errado, uma árvore com três nós e altura igual a dois é perfeitamente balanceada; (d) Correto, uma árvore com três nós e altura igual a três não pode estar balanceada (vide imagem abaixo); (e) Errado, uma árvore com seis nós e altura igual a três também está balanceada. Gabarito: D.



23. (CESGRANRIO – 2011 – PETROBRÁS - Analista de Sistemas) Após a inserção de um nó, é necessário verificar cada um dos nós ancestrais desse nó inserido, relativamente à consistência com as regras estruturais de uma árvore AVL.

PORQUE

O fator de balanceamento de cada nó, em uma árvore AVL, deve pertencer ao conjunto formado por {-2, -1, 0, +1, +2}.

Analisando-se as afirmações acima, conclui-se que:

a) as duas afirmações são verdadeiras, e a segunda justifica a primeira.
b) as duas afirmações são verdadeiras, e a segunda não justifica a primeira.
c) a primeira afirmação é verdadeira, e a segunda é falsa.
d) a primeira afirmação é falsa, e a segunda é verdadeira.
e) as duas afirmações são falsas.

Comentários:

Galera, a primeira frase está perfeita! Após inserir um novo nó, você tem que verificar os nós ancestrais para se certificar de que a Árvore AVL continua balanceada. No entanto, o fator de balanceamento de cada nó deve pertencer ao conjunto formado por {-1, 0, 1}. Lembrem-se: o módulo da diferença jamais pode ser maior do que 1, portanto a primeira está verdadeira e a segunda falsa. **Gabarito: C**

24. (CESGRANRIO – 2010 – EPE - Analista de Sistemas) Um programador decidiu utilizar, em determinado sistema de análise estatística, uma árvore AVL como estrutura de dados. Considerando-se n a quantidade de elementos dessa árvore, o melhor algoritmo de pesquisa, com base em comparações, possui complexidade de tempo, no pior caso, igual a:

a) O(1)
b) O(log n).
c) Ω(n)
d) Ω(n log n)
e) $\Omega(n^2)$

Comentários:

ÁRVORE B / ÁRVORE AVL		
ALGORITMO	CASO MÉDIO	PIOR CASO
Espaço	O(n)	O(n)
Busca	O(log n)	O(log n)
Inserção	O(log n)	O(log n)
Remoção	O(log n)	O(log n)

Questão tranquila! Trata-se do O(log n). **Gabarito: B**

25. (CESGRANRIO – 2012 – PETROBRÁS - Analista de Sistemas) Todos os N nomes de uma lista de assinantes de uma companhia telefônica foram inseridos, em ordem alfabética, em três estruturas de dados: uma árvore binária de busca, uma árvore AVL e uma árvore B.

As alturas resultantes das três árvores são, respectivamente,

a) O(Log(N)), O(Log(N)), O(1)
b) O(Log(N)), O(N), O(Log(N))
c) O(N), O(Log(N)), O(1)
d) O(N), O(Log(N)), O(Log(N))
e) O(N), O(N), O(Log(N))

Comentários:

ÁRVORE BINÁRIA DE BUSCA		
ALGORITMO	CASO MÉDIO	PIOR CASO
Espaço	O(n)	O(n)
Busca	O(log n)	O(n)
Inserção	O(log n)	O(n)
Remoção	O(log n)	O(n)

ÁRVORE B / ÁRVORE AVL		
ALGORITMO	CASO MÉDIO	PIOR CASO
Espaço	O(n)	O(n)
Busca	O(log n)	O(log n)
Inserção	O(log n)	O(log n)
Remoção	O(log n)	O(log n)

Questão tranquila! Trata-se do O(n), O(log n) e O(log n) respectivamente. **Gabarito: D**

26. (IBFC – 2014 – TRE/AM - Analista de Sistemas) Quanto ao Algoritmo e estrutura de dados no caso de árvore AVL (ou árvore balanceada pela altura), analise as afirmativas abaixo, dê valores Verdadeiro (V) ou Falso (F) e assinale a alternativa que apresenta a sequência correta de cima para baixo:

( ) Uma árvore AVL é dita balanceada quando, para cada nó da árvore, a diferença entre as alturas das suas sub-árvores (direita e esquerda) não é maior do que um.

( ) Caso a árvore não esteja balanceada é necessário seu balanceamento através da rotação simples ou rotação dupla.

Assinale a alternativa correta:

a) F-F

b) F-V
c) V-F
d) V-V

Comentários:

A primeira alternativa está impecável, assim como a segunda. Vimos exaustivamente em aula! **Gabarito: D**

27. (CESGRANRIO – 2010 – PETROBRÁS - Analista de Sistemas) Uma sequência desordenada de números armazenada em um vetor é inserida em uma árvore AVL. Após a inserção nesta árvore, é feito um percurso em ordem simétrica (em ordem) e o valor de cada nó visitado é inserido em uma pilha. Depois de todos os nós serem visitados, todos os números são retirados da pilha e apresentados na tela. A lista de números apresentada na tela está:

a) ordenada ascendentemente de acordo com os números.
b) ordenada descendentemente de acordo com os números.
c) na mesma ordem do vetor original.
d) na ordem inversa do vetor original.
e) ordenada ascendentemente de acordo com sua altura na árvore.

Comentários:

Essa questão é legal! Olha só...

Eu tenho um vetor com um bocado de valor desordenado. Esses valores são colocados em uma Árvore AVL. Ora, uma árvore AVL é uma árvore binária de busca, portanto segue suas propriedades. Logo, não importa se estava desordenado. À medida que são inseridos os valores desordenados na árvore, ela vai se balanceando e ordenando os dados.

Após isso, vamos retirar os dados da Árvore AVL. Como? Da esquerda para a direita! E vamos colocá-los em uma pilha. Se estamos lendo da esquerda para a direita, estamos retirando do menor valor para o maior valor, logo o maior valor da pilha será o maior valor da Árvore AVL. Por fim, ao retirar os elementos da pilha, retiramos do topo (maior) para a base (menor), logo em ordem descendente. **Gabarito: B**

28. (FGV – 2009 – MEC - Analista de Sistemas) Acerca das estruturas de dados Árvores, analise as afirmativas a seguir.

I. A árvore AVL é uma árvore binária com uma condição de balanço, porém não completamente balanceada.

II. Árvores admitem tratamento computacional eficiente quando comparadas às estruturas mais genéricas como os grafos.

III. Em uma Árvore Binária de Busca, todas as chaves da subárvore esquerda são maiores que a chave da raiz.

Assinale:

a) se somente a afirmativa I estiver correta.
b) se somente as afirmativas I e II estiverem corretas.
c) se somente as afirmativas I e III estiverem corretas.
d) se somente as afirmativas II e III estiverem corretas.
e) se todas as afirmativas estiverem corretas.

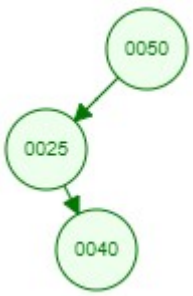
Comentários:

(I) Correto, ela é completamente balanceada; (II) Correto, isso é verdade! (III) Errado, são menores que a chave da raiz. **Gabarito: B**

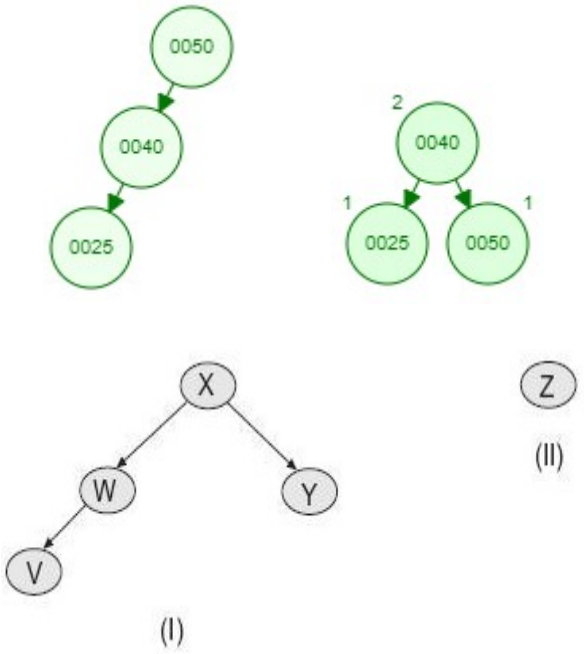
29. (CESPE – 2014 – TJ/SE - Analista de Sistemas) Em uma árvore AVL (Adelson-Velsky e Landis), caso a diferença de altura entre as sub-árvores de um nó seja igual a 2 e a diferença de altura entre o nó filho do nó desbalanceado seja igual a -1, deve-se realizar uma rotação dupla com o filho para a direita e o pai para a esquerda a fim de que a árvore volte a ser balanceada.

Comentários:

Vamos entender a questão! Nós temos um nó (0050) cujo fator de balanceamento é 2. Isso significa que a altura da subárvore à esquerda desse nó (2) menos a altura da subárvore à direita (0) é igual a 2 [2-0 = 2]. O nó filho (0025) desse nó pai (0050) está com balanceamento igual a -1, visto que a altura da subárvore à esquerda desse nó (0) menos a altura da subárvore à direita (1) é igual a -1 [0-1 = -1].



Logo, temos um Caso Esquerda-Direita. E sabemos que para balancear essa árvore, devemos fazer uma rotação dupla: primeiro à esquerda no Ramo 0025-0040 (primeira imagem) e depois à direita no Ramo 0025-0050 (segunda imagem), portanto é exatamente o oposto do que a questão diz. No entanto, o Gabarito Definitivo foi Correto! Sinceramente, não faço ideia de onde o CESPE tirou isso... **Gabarito: C.**



30. (CESPE – 2010 – PETROBRÁS - Analista de Sistemas) As árvores usadas como estruturas de pesquisa têm características especiais que garantem sua utilidade e propriedades como facilidade de acesso aos elementos procurados em cada instante. A esse respeito, considere as afirmações abaixo.

I - A árvore representada na figura (I) acima não é uma árvore AVL, pois as folhas não estão no mesmo nível.

II - A sequência 20, 30, 35, 34, 32, 33 representa um percurso sintaticamente correto de busca do elemento 33 em uma árvore binária de busca.

III - A árvore representada na figura (II) acima é uma árvore binária, apesar da raiz não ter filhos.

É (São) correta(s) APENAS a(s) afirmativa(s):

a) I.
b) II.
c) III.
d) I e II.
e) II e III.

Comentários:

(I) Errado. Trata-se, sim, de uma Árvore AVL; (II) Correto. Temos a sequência 20, 30, 35, 34, 32, 33 e estamos procurando o 33. 33>20, logo descemos à direita; 33>30, logo descemos à direita de novo; 33<35, logo descemos à esquerda; 33<34, logo descemos à esquerda de novo; 33>32, logo descemos à direita. Pronto, encontramos o 33; (III) Correto. Trata-se de um Árvore Binária sem filhos. **Gabarito: E**

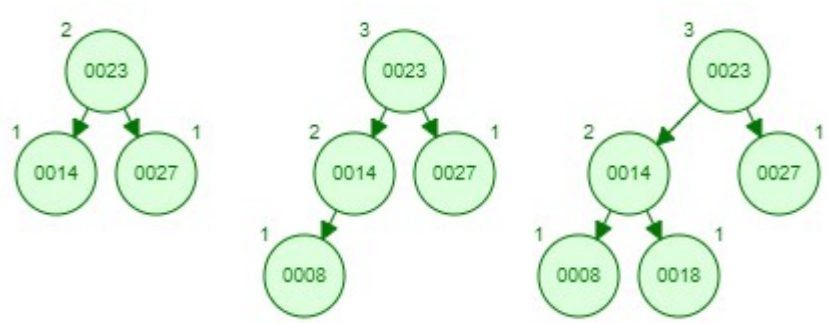
31. (CESPE – 2010 – PETROBRÁS - Analista de Sistemas) No sistema de dados do Departamento de Recursos Humanos de uma grande empresa multinacional, os registros de funcionários são armazenados em uma estrutura de dados do tipo árvore binária AVL, onde cada registro é identificado por uma chave numérica inteira. Partindo de uma árvore vazia, os registros cujas chaves são 23, 14, 27, 8, 18, 15, 30, 25 e 32 serão, nessa ordem, adicionados à árvore.

Dessa forma, o algoritmo de inserção na árvore AVL deverá realizar a primeira operação de rotação na árvore na ocasião da inserção do elemento:

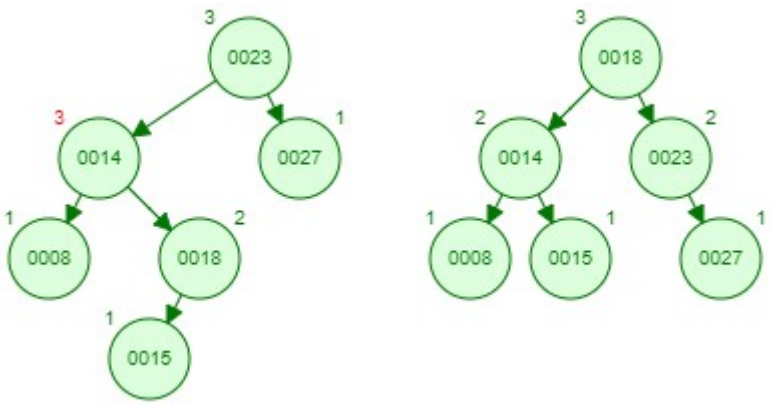
a) 30
b) 25
c) 18
d) 15
e) 8

Comentários:

Vamos simular! Vamos inserir logo os três primeiros: 23, 14 e 27 (primeira imagem); depois inserimos o número 8 (segunda imagem); e inserimos o número 18 (terceira imagem). Vejamos como ficou:



Agora vamos inserir o 15! Esse nó iria para a esquerda de 18 como mostra a primeira imagem abaixo. Nesse caso, o nó 23 ficaria desbalanceado. O que fazer? Temos um Caso Esquerda-Direita, portanto temos que fazer uma rotação dupla: primeiro à esquerda no Ramo 14-18 e depois à direita no Ramo 18-23. O resultado é mostrado a imagem abaixo e é na inserção do nó 15 que devemos fazer a primeira rotação. **Gabarito: D**



32. (CESPE – 2014 – TJ/SE - Analista de Sistemas) Existem dois vetores, chamados A e B, que estão ordenados e contêm N elementos cada, respeitando a propriedade A[N-1]<B[0], onde os índices de ambos os vetores vão de 0 a N-1. Retiram-se primeiro todos os elementos de A na ordem em que se apresentam e inserem-se esses elementos em uma árvore binária de busca, fazendo o mesmo depois com os elementos de B, que são inseridos na mesma árvore de busca que os de A. Depois, retiram-se os elementos da árvore em um percurso pós ordem, inserindo-os em uma pilha. Em seguida retiram-se os elementos da pilha, que são inseridos de volta nos vetores, começando pelo elemento 0 do vetor A e aumentando o índice em 1 a cada inserção, até preencher todas as N posições, inserindo, então, os N elementos restantes no vetor B da mesma maneira.

Ao final do processo, tem-se que os vetores:

a) estão ordenados e A[i] < B[i], para todo i=0,...., N-1.
b) estão ordenados e A[i] > B[i], para todo i=0,...., N-1.
c) estão ordenados e não existe mais uma propriedade que relacione A[i] e B[i].
d) não estão ordenados e A[i] < B[i], para todo i=0,...., N-1.
e) não estão ordenados e A[i] > B[i], para todo i=0,...., N-1.

Comentários:

Ele diz que nós temos dois vetores em que A[N-1] < B[0]. Então, vamos imaginá-los aqui:

- Vetor A = [1, 3, 5]
- Vetor B = [7, 9, 11]

Depois ele diz que são retirados todos os elementos de A na ordem que se apresentam e são inseridos em uma árvore binária de busca (lembre-se que uma árvore binária de busca é aquela em que todos os nós da subárvore esquerda possuem um valor numérico menor que o da raiz e os nós da subárvore direita possuem um valor numérico maior que o da raiz). Se você desenhar essa árvore, vai perceber que ela vai ficar em ordem toda para a direita - sem nenhum elemento para esquerda. Dito isso, ficou:

- 1, 3, 5, 7, 9, 11.

Depois ele disse que os elementos foram retirados da árvore em pós-ordem, ou seja, subárvore à esquerda, depois subárvore à direita e só depois raiz. Portanto, não tem elemento na esquerda, você retira o elemento da direita e depois a raiz. E são colocados em uma pilha, logo ficaria:

- 11, 9, 7, 5, 3, 1

Lembrando que numa pilha você insere sempre elementos no topo, logo 1 seria o topo da pilha. Depois ele diz que você retira os elementos da pilha (também sempre pelo topo) e coloca de volta nos vetores. Logo, ficaria:

- Vetor A = [1, 3, 5]
- Vetor B = [7, 9, 11]

Pronto! Note que fica exatamente a mesma coisa e os vetores ficariam ordenados e A[i] < B[i], para todo i=0,...., N-1. Bacana? **Gabarito: A**

QUESTÕES COMENTADAS - GRAFOS - MULTIBANCAS

1. (CESPE - 2010 – TJ/ES – Analista de Suporte) Considerando-se a implementação de um grafo denso, direcionado e ponderado, se o número de vértices ao quadrado tem valor próximo ao número de arcos, o uso de uma matriz de adjacência simétrica apresenta vantagens em relação ao uso de uma lista de adjacência.

Comentários:

Em grafos não direcionados, as matrizes de adjacência são simétricas ao longo da diagonal principal - isto é, a entrada aij é igual à entrada aji. Matrizes de Adjacência de grafos direcionados, no entanto, não são assim. Num dígrafo sem pesos, a entrada aij da matriz é 1 se há um arco de vi para vj e 0, caso contrário. Há outra maneira de representar também chamada Lista de Adjacências.

Vamos por partes! Ele afirma que o grafo é denso, direcionado e ponderado. Em seguida, ele afirma que o número de vértices ao quadrado tem valor próximo ao número de arcos. Precisava dizer isso? Não está errado, mas ele já havia afirmado que era um grafo denso. A Matriz de Adjacência simétrica é uma representação utilizada em grafos não-direcionados, logo a questão já está errada. Gabarito: E

2. (FCC - 2013 – MPE/SE – Analista do Ministério Público) Considere:

I. Estrutura de dados que possui uma sequência de células, na qual cada célula contém um objeto de algum tipo e o endereço da célula seguinte.

II. Podem ser orientados, regulares, completos e bipartidos e possuir ordem, adjacência e grau.

III. Possuem o método de varredura esquerda-raiz-direita (e-r-d).

Os itens de I a III descrevem, respectivamente,

a) árvores binárias, listas ligadas e arrays.

b) arrays, árvores binárias e listas ligadas.

c) grafos, árvores binárias e arrays.

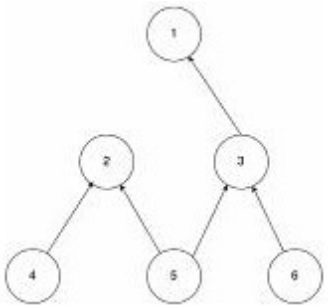
d) listas ligadas, grafos e árvores binárias.

e) grafos, listas ligadas e árvores binárias.

Comentários:

(a) Trata-se das Listas Ligadas, visto que falou de sequência, objeto e endereço da célula seguinte; (b) Trata-se dos Grafos, visto que falou dos tipos regular, bipartido, completo e orientado; (c) Trata-se das Árvores Binárias, visto que falou de método de varredura e raiz. Gabarito: D

3. (CESPE - 2013 – TCE/ES – Analista de Sistemas) Considerando o grafo ilustrado acima, assinale a opção em que é apresentada a descrição em vértices (V) e arestas (A).



a) V = {1, 2, 3, 4, 5, 6 }

A = {(2, 4), (2, 3), (2, 5), (3, 6), (1, 5)}

b) V = { 2, 4, 1, 3, 6, 5 }

A = {(4, 2), (1, 3), (5, 2), (6, 3), (5, 3)}

c) V = {1, 2, 3, 4, 5, 6 }

A = {(4, 2), (3, 4), (5, 2), (6, 3), (5, 3)}

d) V = {1, 2, 3, 4, 5, 6 }

A = {(4, 2), (3, 1), (5, 1), (6, 2), (5, 3)}

e) V = { 2, 4, 1, 3, 6, 5 }

A = {(4, 2), (3, 1), (5, 2), (6, 3), (5, 3)}

Comentários:

Galera, os vértices não precisam estar ordenados, logo todos os itens estão corretos. No entanto, as arestas precisam corresponder ao grafo. Vamos por eliminação: (a) A aresta (2,3) não existe; (b) Quase tudo certo, mas a ordem (1,3) está errada – seria (3,1); (c) A aresta (3,4) não existe; (d) A aresta (5,1) não existe; (e) Tudo perfeito! Gabarito: E

4. (CESPE - 2012 – TJ/RO – Analista de Suporte – ITEM B) Grafo corresponde a uma estrutura abstrata de dados que representa um relacionamento entre pares de objetos e que pode armazenar dados em suas arestas e vértices, ou em ambos.

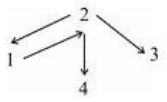
Comentários:

Fiz uma disciplina na faculdade chamada Teoria dos Grafos! Aquilo era absurdamente complexo, mas para concursos a teoria é beeeem mais tranquila e muito rara de cair. Portanto fiquem tranquilos, bacana? Uma definição de grafo afirma que ele é uma estrutura de dados que consiste em um conjunto de nós (ou vértices) e um conjunto de arcos (ou arestas).

Em outras palavras, podemos dizer que é simplesmente um conjunto de pontos e linhas que conectam vários pontos. Ou também que é uma representação abstrata de um conjunto de objetos e das relações existentes entre eles. Uma grande variedade de estruturas do mundo real podem ser representadas abstratamente através de grafos. Professor, pode me passar um exemplo? Claro!

Conforme vimos em aula, a definição está perfeita! Gabarito: C

5. (CESPE - 2012 – PEFOCE – Perito Criminal) Considere que um grafo G seja constituído por um conjunto (N) e por uma relação binária (A), tal que G = (N, A), em que os elementos de N são denominados nós (ou vértices) e os elementos de A são denominados arcos (ou arestas). Em face dessas informações e do grafo abaixo, é correto afirmar que esses conjuntos são N = {1,2,3,4} e A = {(1,2),(2,1),(2,4),(2,3)}.



Comentários:

Conforme vimos em aula, a questão está perfeita! Temos quatro nós: N = {1,2,3,4}; e quatro arestas: A = {(1,2),(2,1),(2,4),(2,3)} – observem que a ordenação do grafo ordenado está perfeita. Gabarito: C

6. (CESPE - 2012 – BASA – Analista de Sistemas) É misto o grafo com arestas não dirigidas que representam ruas de dois sentidos e com arestas dirigidas que correspondem a trechos de um único sentido, modelado para representar o mapa de uma cidade cujos vértices sejam os cruzamentos ou finais de ruas e cujas arestas sejam os trechos de ruas sem cruzamentos.

Comentários:

Um grafo simples é aquele que não contém laços. Um grafo vazio é aquele que contém exclusivamente vértices (não contém arcos). Um grafo misto é aquele que possui arestas dirigidas e não-dirigidas. Um grafo trivial é aquele que possui somente um vértice. Um grafo é denso se contém muitos arcos em relação ao número de vértices e esparso se contém poucos arcos. Como assim, professor?

Conforme vimos em aula, um grafo misto é aquele que possui arestas dirigidas e não-dirigidas. O caso citado na questão é um exemplo perfeito e foi retirado integralmente do livro Projeto de algoritmos: Fundamentos, análise e exemplos da internet de Michael T. Goodrich e Roberto Tamassia. Gabarito: C

7. (CESPE - 2012 – BASA – Analista de Sistemas) Para modelar a rede que conecta todos os computadores em uma sala de escritório com a menor metragem possível de cabos, é adequado utilizar um grafo G cujos vértices representem os possíveis pares (u, v) de computadores e cujas arestas representem o comprimento dos cabos necessários para ligar os computadores u e v, determinando-se o caminho mínimo, que contenha todos os vértices de G, a partir de um dado vértice v.

Comentários:

Galera, leiam devagar a questão! Vejam essa parte: "(...) cujos vértices representem os possíveis pares (u,v)". Vocês, é claro, se lembram do conceito de vértices e arestas. Ora, vértice representa um par de computadores? Não, vértices são os computadores! Quem representa pares são as arestas! Gabarito: E

8. (CESPE - 2012 – BASA – Analista de Sistemas) Um grafo completo contém pelo menos um subgrafo ponderado.

Comentários:

Essa questão não faz o menor sentido! Podemos dizer que um grafo completo contém pelo menos um subgrafo. No entanto, os conceitos de completude e ponderação são completamente independentes. Eu posso ter um grafo completo ponderado ou não; e posso ter um grafo ponderado completo ou não. Gabarito: E

9. (CESPE - 2012 – BASA – Analista de Sistemas) Um grafo não direcionado é dito conectado quando há pelo menos um caminho entre dois vértices quaisquer do grafo.

Comentários:

Um grafo é dito conexo ou conectado quando existir pelo menos um caminho entre cada par de vértices. Caso contrário, ele será dito desconexo, isto é, se há pelo menos um par de vértices que não esteja ligado a nenhuma cadeia (caminho). Vejam a imagem do grafo que eu desenhei lá em cima! Ele é conexo ou desconexo? Ele é desconexo, há um par de vértices (E,F) que não está ligado a nenhum caminho.

Conforme vimos em aula, a questão comete um minúsculo deslize! Seria mais correto dizer que um grafo não direcionado é dito conectado quando sempre há pelo menos um caminho entre dois vértices quaisquer do grafo. Por que, professor? Vejam o grafo que eu desenhei na teoria dessa aula.

Pela definição da questão, ele é conectado quando há pelo menos um caminho entre dois vértices quaisquer do grafo. Vamos pegar dois vértices quaisquer? Sim, vamos pegar o par A-C! Há pelo menos um caminho entre eles? Sim, logo podemos dizer – por essa definição – que ele é conectado.

No entanto, o par D-E não possui um caminho entre eles, mas a definição da questão não disse para verificar todos os pares de vértices. Entenderam a sutileza? Portanto, seria ideal dizer que ele é conectado quando há pelo menos um caminho entre cada par de vértices do grafo.

Dessa forma, cobrimos todos os pares. É tão sutil que nem eu havia percebido – quem me alertou foi um aluno. Gabarito: C

10. (CESPE - 2012 – TJ/AC – Analista de Sistemas) Define-se um grafo como fortemente conexo se todos os nós puderem ser atingidos a partir de qualquer outro nó.

Comentários:

Se o grafo for direcionado/orientado, um grafo é dito fortemente conexo se existir um caminho de A → B e de B → A, para cada par (A,B) de vértices de um grafo. Em outras palavras, o grafo é fortemente conexo se cada par de vértices participa de um circuito. Isto significa que cada vértice pode ser alcançável partindo-se de qualquer outro vértice do grafo.

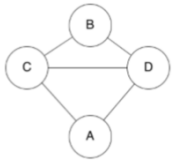
Conforme vimos em aula, um grafo é fortemente conexo se todos os nós puderem ser atingidos a partir de qualquer outro nó. Como vimos em na descrição, cada vértice pode ser alcançável partindo-se de qualquer outro vértice do grafo. Gabarito: C

11. (CESPE - 2013 – CRPM – Analista de Sistemas) Considere que o grafo não orientado representado na figura abaixo possua as seguintes características:

G1 = (V1, A1)

V1 = {A, B, C, D}

A1 = {(A, C), (A, D), (B, C), (B, D), (A,B)}.



Nesse caso, é correto afirmar que o grafo G1 possui quatro vértices, nomeados de A, B, C e D, e cinco arcos, que conectam pares de vértices, conforme especificado em A1.

Comentários:

A questão está quase perfeita, mas ela possui um deslize: não existe a aresta (A,B) – seria (C,D). Gabarito: E

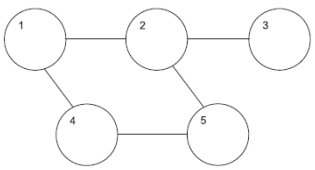
12. (CESPE - 2012 – BASA – Analista de Sistemas) A implementação de um grafo do tipo ponderado e direcionado na forma de uma matriz de adjacência utiliza menor quantidade de memória que a implementação desse mesmo grafo na forma de uma lista encadeada.

Comentários:

Pensem comigo! Para um grafo com uma Matriz de Adjacência esparsa (não densa), uma representação de Lista de Adjacências do grafo ocupa menos espaço, porque ele não usa nenhum espaço para representar as arestas que não estão presentes. Lembram-se da lista? Nós só representamos os nós adjacentes, em contraste com a Matriz de Adjacência. No entanto, quanto mais denso, isso pode mudar.

Conforme vimos em aula, para grafos esparsos (não densos), a Lista de Adjacência (que é uma lista encadeada) ocupa menos espaço em memória, na medida em que não necessita representar vértices não adjacentes – diferente da Matriz de Adjacência. Gabarito: E

13. (CESGRANRIO - 2014 – BASA – Analista de Sistemas) O grafo acima pode ser representado pela seguinte matriz:



a) $\begin{vmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{vmatrix}$

b) $\begin{vmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{vmatrix}$

c) $\begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{vmatrix}$

d) $\begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{vmatrix}$

e) $\begin{vmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{vmatrix}$

Comentários:

Vamos ver se vocês se lembram! O vértice A11 tem ser 0, porque não há uma aresta do Nó 1 para o Nó 1. O vértice A12 tem ser 1, porque há uma aresta do Nó 1 para o Nó 2. O vértice A13 tem ser 0, porque não há uma aresta do Nó 1 para o Nó 3. O vértice A14 tem ser 1, porque há uma aresta do Nó 1 para o Nó 4. O vértice A15 tem ser 0, porque não há uma aresta do Nó 1 para o Nó 5. Logo, a primeira linha deve ser: 0, 1, 0, 1, 0. Eliminamos todos os itens, exceto os dois primeiros. Vamos pegar um vértice específico agora para descobrir qual está certo. Observem o vértice A23 e percebam que ele deve ser 1, porque existe uma aresta do Nó 2 para o Nó 3. Descobrimos a resposta! Gabarito: A

14. (CESPE - 2012 – TJ/SE – Analista de Sistemas) Um grafo é formado por um par de conjuntos de vértices e arestas, não podendo o conjunto de vértices ser particionado em subconjuntos.

Comentários:

Podem, sim. O nome disso é: Grafo Bipartido! Um grafo é dito ser bipartido quando seu conjunto de vértices V puder ser particionado em dois subconjuntos V1 e V2. Portanto não há óbice quanto ao particionamento de um conjunto de vértices em subconjuntos. Gabarito: E

15. (CESPE - 2012 – TRT/AM – Analista de Sistemas) Um grafo é uma estrutura de dados consistida em um conjunto de nós (ou vértices) e um conjunto de arcos (ou arestas). O grafo em que os arcos possuem um número ou peso associados a eles, é chamado de grafo:

a) predecessor.

b) adjacente.

c) incidente.

d) ponderado.

e) orientado.

Comentários:

Vamos ver agora alguns conceitos importantes! Um grafo pode ser dirigido – também chamado direcionado, orientado ou dígrafo –, se as arestas tiverem uma direção (imagem da esquerda), ou não dirigido, se as arestas não tiverem direção (imagem central). Se as arestas tiverem associado um peso ou custo, o grafo passa a ser chamado ponderado, valorado ou pesado (imagem da direita).

Conforme vimos em aula, um grafo com arcos numerados ou com peso são chamados Grafos Ponderados ou Grafos Valorados ou Grafos Pesados. Gabarito: D

QUESTÕES COMENTADAS - HASHING - MULTIBANCAS

1. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) A pesquisa sequencial é aplicável em estruturas não ordenadas.**

Comentários:

Perfeito! Para fazer uma pesquisa sequencial, não é necessário que os dados estejam ordenados – diferentemente da pesquisa binária. **Gabarito: C**

2. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As colisões ocorrem na utilização de tabela hash porque várias chaves podem resultar na mesma posição.**

Comentários:

Perfeito! É raro, mas acontece... **Gabarito: C**

3. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) Ocorre o hashing quando não há o armazenamento de cada entrada de uma tabela em um específico endereço calculado a partir da aplicação de uma função chave da entrada.**

Comentários:

Pelo contrário, ocorre o hashing quando há o armazenamento de cada entrada de uma tabela em um específico endereço calculado a partir da aplicação de uma função chave da entrada. **Gabarito: E**

4. **(FCC - 2008 - METRÔ-SP - Analista Treinee - Análise de Sistemas) O objetivo de fazer uma busca rápida a partir de uma chave de pesquisa simples e obter o valor desejado é alcançado pela estrutura de dados especial denominada:**

a) array.

b) lista.

c) vetor.

d) árvore binária.

e) tabela de hashing.

Comentários:

Trata-se da Tabela de Hashing! **Gabarito: E**

5. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) A busca que utiliza uma tabela hash realiza comparação das chaves para encontrar a posição do elemento que está sendo buscado.**

Comentários:

Não, eles utilizam a chave para gerar resultados que, esses sim, são comparados. **Gabarito: E**

6. **(CESPE - 2010 - DETRAN-ES - Analista de Sistemas) No método de hashing, por meio de acesso sequencial, são utilizados tabelas e mapas para recuperar informações de endereço de arquivos de forma rápida e eficiente.**

Comentários:

Não, Método de Hashing não faz Acesso Sequencial. **Gabarito: E**

7. **(FCC - 2015 - DPE-SP - Programador) Um Programador da Defensoria Pública do Estado de São Paulo foi solicitado a propor uma solução para o problema: Há uma quantidade grande de dados classificáveis por chave e estes dados devem ser divididos em subconjuntos com base em alguma característica das chaves. Um método eficiente deve ser capaz de localizar em qual subconjunto deve-se colocar cada chave e depois estes subconjuntos bem menores devem ser gerenciados por algum método simples de busca para que se localize uma chave rapidamente. O Programador propôs como solução, corretamente, a implementação de:**

a) Deques.

b) Tabela e função hash.

c) Pilhas.

d) Fila duplamente encadeada.

e) Árvore Binária de Busca.

Comentários:

Como se fala em subdivisão em conjuntos com base nas chaves, podemos afirmar que se trata-se da Tabela de Hashing! As árvores de busca binária não dividem os elementos em subconjuntos, somente ordena a estrutura não-linear de forma a facilitar uma busca binária. **Gabarito: B**

QUESTÕES COMENTADAS - BITMAP - MULTIBANCAS

1. (FGV - 2015 – TJ-RO – Analista Judiciário - Análise de Sistemas)

ID	Nome	Curso
1210	A	Física
356	B	Química
23	C	Matemática
57	D	Física
45	E	Física
6	F	Matemática
210	G	Matemática

Se fosse construído um índice de banco de dados do tipo “bitmap" para essa tabela, tendo o campo Curso como chave, o conteúdo desse índice seria:

a)

6	3
23	3
45	1
57	1
210	3
356	2
1210	1

b)

Física	1001100
Química	0100000
Matemática	0010011

c)

6	001
23	001
45	100

57 100
210 001
356 010
1210 100

d)
0011001100
0100100000
0010010011

e)
00000000110 Matemática
00000010111 Matemática
00000101101 Física
00000111011 Física
00011010010 Matemática
00101100100 Química
10010111010 Química

Comentários:

A questão solicita a criação de um índice usando bitmap da coluna curso. Lembrando da aula teórica que a tabela bitmap terá como linhas os valores da coluna fonte do índice, ou seja, Física, Química e Matemática. As colunas do bitmap são os números das tuplas (ROWID) . O conteúdo do bitmap é o preenchimento de acordo com o valor de cada tupla original relativo à coluna fonte do índice. Construindo o bitmap para curso temos:

	1	**2**	**3**	**4**	**5**	**6**	**7**
Física	1	0	0	1	1	0	0
Química	0	1	0	0	0	0	0
Matemática	0	0	1	0	0	1	1

A resposta, portanto, é a letra B. **Gabarito: B**

2. (FGV - 2014 – DPE-RJ – Técnico Superior Especializado - Administração de Dados)

Candidato	
inscrição	**candidatoNome**
101	João
102	Maria
105	Gabriela
106	João

Prova	
provaNome	**data**
Português	12/01/2014
Matemática	12/01/2014
Prática	19/01/2014

Avaliação		
inscrição	**provaNome**	**nota**
101	Português	10
102	Português	8
105	Português	3
106	Português	5
101	Matemática	7
102	Matemática	4
105	Matemática	8
101	Prática	5
102	Prática	5
105	Prática	NULL
106	Prática	4

Índices do tipo Bitmap podem ser usados em tabelas de bancos de dados. O quadro abaixo representa o mapa de bits na indexação da tabela Avaliação quando a chave considerada é a concatenação dos atributos Inscrição e provaNome.

101Matemática	**00001000000**
101Português	**10000000000**
101Prática	**00000001000**
102Matemática	**00000100000**
102Português	**01000000000**
102Prática	**00000000100**
105Matemática	**00000010000**

105Português 00100000000
105Prática 00000000010
106Português 00010000000
106Prática 00000000001

Num mapa de bits para a mesma tabela, usando apenas o atributo Inscrição, o mapa de bits seria

a)
101 10001001000
102 01000100100
105 00100010010
106 00010000001

b)
101 1000
102 0100
105 0010
106 0001

c)
101 10000000000
102 01000100000
105 00100000000
106 00010000000

d)
101 00000001000
102 00000000100
105 00000000010
106 00000000001

e)
101 111

102 111

105 111

106 011

Conforme vimos na parte teórica da aula, os índices podem ser criadosa partir de uma coluna só, mas também para concatenação de colunas. A questão pede o índice mais simples, somente para a coluna Inscrição

	1	2	3	4	5	6	7	8	9	10	11
101	1	0	0	0	1	0	0	1	0	0	0
102	0	1	0	0	0	1	0	0	1	0	0
105	0	0	1	0	0	0	1	0	0	1	0
106	0	0	0	1	0	0	0	0	0	0	1

A resposta, portanto, é letra A. **Gabarito: A**

LISTA DE QUESTÕES – ESTRUTURA DE DADOS - MULTIBANCAS

1. **(FGV – 2015 – DPE/MT – Analista de Sistemas) No desenvolvimento de sistemas, a escolha de estruturas de dados em memória é especialmente relevante. Dentre outras classificações, é possível agrupar essas estruturas em lineares e não lineares, conforme a quantidade de sucessores e antecessores que os elementos da estrutura possam ter. Assinale a opção que apresenta, respectivamente, estruturas de dados lineares e não lineares.**

   a) Tabela de dispersão e fila.

   b) Estrutura de seleção e pilha.

   c) Pilha e estrutura de seleção.

   d) Pilha e árvore binária de busca.

   e) Fila e pilha.

2. **(CESPE – 2010 – DETRAN/ES – Analista de Sistemas) Um tipo abstrato de dados apresenta uma parte destinada à implementação e outra à especificação. Na primeira, são descritas, em forma sintática e semântica, as operações que podem ser realizadas; na segunda, os objetos e as operações são representados por meio de representação, operação e inicialização.**

3. **(CESPE – 2010 – TRT/RN – Analista de Sistemas) O tipo abstrato de dados consiste em um modelo matemático (v,o), em que v é um conjunto de valores e o é um conjunto de operações que podem ser realizadas sobre valores.**

4. **(CESPE – 2010 – BASA – Analista de Sistemas) A escolha de estruturas internas de dados utilizados por um programa pode ser organizada a partir de TADs que definem classes de objetos com características distintas.**

5. **(CESPE – 2010 – BASA – Analista de Sistemas) A descrição dos parâmetros das operações e os efeitos da ativação das operações representam, respectivamente, os níveis sintático e semântico em que ocorre a especificação dos TDAs.**

6. **(FCC – 2010 – TRE/AM – Analista de Sistemas) Em relação aos tipos abstratos de dados - TAD, é correto afirmar:**

   a) O TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações sobre esses dados.

b) Na transferência de dados de uma pilha para outra, não é necessário saber como a pilha é efetivamente implementada.

c) Alterações na implementação de um TAD implicam em alterações em seu uso.

d) Um programador pode alterar os dados armazenados, mesmo que não tenha conhecimento de sua implementação.

e) TAD é um tipo de dados que esconde a sua implementação de quem o manipula.

7. (FCC – 2009 – TRE/PI – Analista de Sistemas) Em relação a tipos abstratos de dados, é correto afirmar que:

a) o TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações disponibilizadas sobre esses dados.

b) algumas pilhas admitem serem declaradas como tipos abstratos de dados.

c) filas não permitem declaração como tipos abstratos de dados.

d) os tipos abstratos de dados podem ser formados pela união de tipos de dados primitivos, mas não por outros tipos abstratos de dados.

e) são tipos de dados que escondem a sua implementação de quem o manipula; de maneira geral as operações sobre estes dados são executadas sem que se saiba como isso é feito.

GABARITO

1. D
2. E
3. C
4. E
5. C
6. E
7. E

LISTA DE QUESTÕES – VETORES E MATRIZES - MULTIBANCAS

1. **(FCC - 2009 - TJ-PA - Analista Judiciário - Tecnologia da Informação) Considere uma estrutura de dados do tipo vetor. Com respeito a tal estrutura, é correto que seus componentes são, caracteristicamente,**

   a) heterogêneos e com acesso FIFO.

   b) heterogêneos e com acesso LIFO.

   c) heterogêneos e com acesso indexado-sequencial.

   d) homogêneos e acesso não indexado.

   e) homogêneos e de acesso aleatório por intermédio de índices.

2. **(CETAP - 2010 - AL-RR - Analista de Sistemas) Matrizes são estruturas de dados de n-dimensões. Por simplicidade, chamaremos de matrizes as matrizes bidimensionais numéricas (que armazenam números inteiros). Sendo assim, marque a alternativa INCORRETA.**

   a) Uma matriz de m linhas e n colunas contêm (m * n) dados.

   b) Uma matriz pode ser representada utilizando listas ligadas.

   c) A soma dos elementos de uma matriz pode ser calculada fazendo dois laços aninhados, um sobre as linhas e o outro sobre as colunas.

   d) A soma de duas matrizes de m linhas e n colunas resulta em uma matriz de 2*m linhas e 2*n colunas.

   e) O produto de duas matrizes de n linhas e n colunas resulta em uma matriz de n linhas e n colunas.

3. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Arquitetura de Tecnologia) Os dados armazenados em uma estrutura do tipo matriz não podem ser acessados de maneira aleatória.** Portanto, usa-se normalmente uma matriz quando o volume de inserção e remoção de dados é maior que o volume de leitura dos elementos armazenados.

4. **(CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) Entre alguns tipos de estrutura de dados, podem ser citados os vetores, as pilhas e as filas.**

5. **(CESPE - 2011 - EBC - Analista - Engenharia de Software) Vetores são utilizados quando estruturas indexadas necessitam de mais que um índice para identificar um de seus elementos.**

6. **(CESPE - 2010 - TRE-BA - Analista Judiciário - Análise de Sistemas) Vetores podem ser considerados como listas de informações armazenadas em posição contígua na memória.**

7. **(CESPE - 2010 - TRE-BA - Analista Judiciário - Análise de Sistemas) Uma posição específica de um vetor pode ser acessada diretamente por meio de seu índice.**

8. **(FCC - 2016 - Copergás - PE - Analista Tecnologia da Informação) Considere o algoritmo a seguir, na forma de pseudocódigo:**

```
Var n, i, j, k, x: inteiro
Var v: vetor[0..7] inteiro
Início
 v[0] ← 12
 v[1] ← 145
 v[2] ← 1
 v[3] ← 3
 v[4] ← 67
 v[5] ← 9
 v[6] ← 45
 n ← 8
 k ← 3
 x ← 0

 Para j ← n-1 até k passo -1 faça
  v[j] ← v[j - 1];
 Fim_para

 v[k] ← x;
Fim
```

Este pseudocódigo:

a) exclui o valor contido na posição x do vetor v.

b) insere o valor de x entre v[k-1] e v[k] no vetor v.

c) exclui o valor contido na posição k do vetor v.

d) tentará, em algum momento, acessar uma posição que não existe no vetor.

e) insere o valor de k entre v[x] e v[x+1] no vetor v.

GABARITO

1. E
2. D
3. E
4. C
5. E
6. C
7. C
8. B

LISTA DE QUESTÕES – LISTA ENCADEADA - MULTIBANCAS

1. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) O tempo de busca de um elemento em uma lista duplamente encadeada é igual à metade do tempo da busca de um elemento em uma lista simplesmente encadeada.**

2. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) Em algumas implementações, uma lista vazia pode ter um único nó, chamado de sentinela, nó cabeça ou header. Entre suas possíveis funções, inclui-se simplificar a implementação de algumas operações realizadas sobre a lista, como inserir novos dados, recuperar o tamanho da lista, entre outras.**

3. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) Estruturas ligadas como listas encadeadas superam a limitação das matrizes que não podem alterar seu tamanho inicial.**

4. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As listas duplamente encadeadas diferenciam-se das listas simplesmente encadeadas pelo fato de, na primeira, os nós da lista formarem um anel com o último elemento ligado ao primeiro da lista.**

5. **(FCC - 2012 - TRE-SP - Analista Judiciário - Análise de Sistemas - E) Numa lista singularmente encadeada, para acessar o último nodo é necessário partir do primeiro e ir seguindo os campos de ligação até chegar ao final da lista.**

6. **(CESPE - 2011 - EBC - Analista - Engenharia de Software) Uma lista é uma coleção de elementos do mesmo tipo dispostos linearmente, que podem ou não seguir determinada organização. As listas podem ser dos seguintes tipos: de encadeamento simples, duplamente encadeadas e ordenadas.**

7. **(CESPE - 2009 - ANAC - Técnico Administrativo - Informática) Em uma lista circular duplamente encadeada, cada nó aponta para dois outros nós da lista, um anterior e um posterior.**

8. **(CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) A principal característica de uma lista encadeada é o fato de o último elemento da lista apontar para o elemento imediatamente anterior.**

9. **(CESPE - 2009 - TCE-AC - Analista de Controle Externo - Processamentos de Dados) Uma lista encadeada é uma coleção de nodos que, juntos, formam uma ordem linear. Se é possível os nodos se deslocarem em ambas as direções na lista, diz-se que se trata de uma lista simplesmente encadeada.**

10. **(CESPE - 2008 – HEMOBRÁS – Técnico de Informática) Uma estrutura do tipo lista, em que é desejável percorrer o seu conteúdo nas duas direções indiferentemente, é denominado lista duplamente encadeada.**

11. **(CESPE - 2010 – TRE/MT – Analista de Sistemas – C) Uma lista duplamente encadeada é uma lista em que o seu último elemento referencia o primeiro.**

12. **(CESPE - 2006 – SGA/AC – Analista de Sistemas) O principal problema da alocação por lista encadeada é a fragmentação.**

13. **(CESPE - 2008 – MCT – Analista de Sistemas) O armazenamento de arquivos em disco pode ser feito por meio de uma lista encadeada, em que os blocos de disco são ligados por ponteiros. A utilização de lista encadeada elimina completamente o problema de fragmentação interna.**

14. **(CESPE - 2009 – FINEP – Analista de Sistemas) Uma lista encadeada é uma representação de objetos na memória do computador que consiste de uma sequência de células em que:**

    a) cada célula contém apenas o endereço da célula seguinte.

    b) cada célula contém um objeto e o tipo de dados da célula seguinte.

    c) o último elemento da sequência aponta para o próximo objeto que normalmente possui o endereço físico como not null.

    d) cada célula contém um objeto de algum tipo e o endereço da célula seguinte.

    e) a primeira célula contém o endereço da última célula.

15. **(CESPE - 2010 – BASA – Analista de Sistemas) Em uma lista encadeada, o tempo de acesso a qualquer um de seus elementos é constante e independente do tamanho da estrutura de dados.**

16. **(CESPE - 2010 – INMETRO – Analista de Sistemas – C) Considere que Roberto tenha feito uso de uma lista encadeada simples para programar o armazenamento e o posterior acesso aos dados acerca dos equipamentos instalados em sua empresa. Considere, ainda, que, após realizar uma consulta acerca do equipamento X, Roberto precisou acessar outro equipamento Y que se encontrava, nessa lista, em posição anterior ao equipamento X. Nessa situação, pela forma como os ponteiros são implementados em uma lista**

encadeada simples, o algoritmo usado por Roberto realizou a consulta ao equipamento Y sem reiniciar a pesquisa do começo da lista.

17. (FCC - 2003 – TRE/AM – Analista de Sistemas) Os dados contidos em uma lista encadeada estão:

a) ordenados seqüencialmente.

b) sem ordem lógica ou física alguma.

c) em ordem física e não, necessariamente, em ordem lógica.

d) em ordem lógica e, necessariamente, em ordem física.

e) em ordem lógica e não, necessariamente, em ordem física.

18. (FCC - 2010 – DPE/SP – Analista de Sistemas) Uma estrutura de dados que possui três campos: dois ponteiros e campo de informação denomina-se:

a) lista encadeada dupla.

b) Lista encadeada simples.

c) pilha.

d) fila.

e) vetor.

19. (CESPE - 2010 – TRE/MT – Analista de Sistemas) O algoritmo para inclusão de elementos em uma pilha é usado sem nenhuma alteração para incluir elementos em uma lista.

GABARITO

1. E
2. C
3. C
4. E
5. C
6. C
7. C
8. E
9. E
10. C
11. E
12. E
13. E
14. D
15. E
16. E
17. E
18. A
19. C

LISTA DE QUESTÕES - PILHAS - MULTIBANCAS

1. **(CESPE - 2011 - FUB - Analista de Tecnologia da Informação - Específicos) As pilhas são listas encadeadas cujos elementos são retirados e acrescentados sempre ao final, enquanto as filas são listas encadeadas cujos elementos são retirados e acrescentados sempre no início.**

2. **(CESPE - 2013 - INPI - Analista de Planejamento - Desenvolvimento e Manutenção de Sistemas) Na estrutura de dados do tipo lista, todo elemento novo que é introduzido na pilha torna-se o elemento do topo.**

3. **(CESPE - 2012 - TJ-RO - Analista Judiciário - Analista de Sistemas Suporte – E) Visitas a sítios armazenadas em um navegador na ordem last-in-first-out é um exemplo de lista.**

4. **(ESAF - 2013 - DNIT - Analista Administrativo - Tecnologia da Informação) Assinale a opção correta relativa às operações básicas suportadas por pilhas.**
   a) Push: insere um novo elemento no final da pilha.
   b) Pop: adiciona elementos ao topo da pilha.
   c) Pull: insere um novo elemento no interior da pilha.
   d) Top: transfere o último elemento para o topo da pilha.
   e) Top: acessa o elemento posicionado no topo da pilha.

5. **(FCC - 2012 – TST - Analista de Sistemas – C) As pilhas e as filas são estruturas de dados essenciais para os sistemas computacionais. É correto afirmar que a pilha é conhecida como lista FIFO - First In First Out.**

6. **(FCC - 2012 – TRE/CE - Analista de Sistemas) Sobre pilhas é correto afirmar:**

   a) Uma lista LIFO (Last-In/First-Out) é uma estrutura estática, ou seja, é uma coleção que não pode aumentar e diminuir durante sua existência.

   b) Os elementos na pilha são sempre removidos na mesma ordem em que foram inseridos.

   c) Uma pilha suporta apenas duas operações básicas, tradicionalmente denominadas push (insere um novo elemento no topo da pilha) e pop (remove um elemento do topo da pilha).

   d) Cada vez que um novo elemento deve ser inserido na pilha, ele é colocado no seu topo e, em qualquer momento, apenas aquele posicionado no topo da pilha pode ser removido.

   e) Sendo P uma pilha e x um elemento qualquer, a operação Push(P,x) diminui o tamanho da pilha P, removendo o elemento x do seu topo.

7. **(CESPE - 2011 - EBC - Analista - Engenharia de Software) As pilhas, também conhecidas como listas LIFO ou PEPS, são listas lineares em que todas as operações de inserção e remoção de elementos são feitas por um único extremo da lista, denominado topo.**

8. **(VUNESP - 2011 - TJM-SP - Analista de Sistemas - Judiciário) Lista do tipo LIFO (Last in, First Out) e lista do tipo FIFO (Firstin,First Out) são, respectivamente, características das estruturas de dados denominadas:**
   a) Fila e Pilha.
   b) Pilha e Fila.
   c) Grafo e Árvore.
   d) Árvore e Grafo.
   e) Árvore Binária e Árvore Ternária.

9. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Arquitetura de Tecnologia) A definição da estrutura pilha permite a inserção e a eliminação de itens, de modo que uma pilha é um objeto dinâmico, cujo tamanho pode variar constantemente.**

10. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) Na representação física de uma pilha sequencial, é necessário uso de uma variável ponteiro externa que indique a extremidade da lista linear onde ocorrem as operações de inserção e retirada de nós.**

11. **(CESPE - 2009 - ANAC - Técnico Administrativo - Informática) As operações de inserir e retirar sempre afetam a base de uma pilha.**

12. **(FCC - 2009 - TRT - 16ª REGIÃO (MA) - Técnico Judiciário - Tecnologia da Informação) Pilha é uma estrutura de dados:**
   a) cujo acesso aos seus elementos segue tanto a lógica LIFO quanto a FIFO.
   b) cujo acesso aos seus elementos ocorre de forma aleatória.
   c) que pode ser implementada somente por meio de vetores.
   d) que pode ser implementada somente por meio de listas.
   e) cujo acesso aos seus elementos segue a lógica LIFO, apenas.

13. **(CESPE - 2004 – STJ – Analista de Sistemas) Em geral, em uma pilha só se admite ter acesso ao elemento localizado em seu topo. Isso se adapta perfeitamente à característica das seqüências em que só o primeiro componente é diretamente acessível.**

14. **(CESPE - 2004 – STJ – Analista de Sistemas) A seguir, está representada corretamente uma operação de desempilhamento em uma pilha de nome p.**

```
se p.topo = 0 então
    nada {pilha vazia}
senão p.topo ← p.topo-1
```

15. (CESPE - 2010 – TRE/MT - Analista de Sistemas – A) O tipo nó é inadequado para implementar estruturas de dados do tipo pilha.

16. (FGV – 2015 – DPE/MT – Analista de Sistemas) Assinale a opção que apresenta a estrutura de dados na qual o primeiro elemento inserido é o último a ser removido.

a) Árvore
b) Fila
c) Pilha
d) Grafo
e) Tabela de dispersão

17. (FCC – 2012 – MPE/AP – Técnico Ministerial - Informática) Nas estruturas de dados,

a) devido às características das operações da fila, o primeiro elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como LIFO.

b) as pilhas são utilizadas para controlar o acesso de arquivos que concorrem a uma única impressora.

c) a fila é uma lista linear na qual as operações de inserção e retirada ocorrem apenas no início da lista.

d) a pilha é uma lista linear na qual as operações de inserção e retirada são efetuadas apenas no seu topo.

e) devido às características das operações da pilha, o último elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como FIFO.

GABARITO

1. E
2. E
3. E
4. E
5. E
6. D
7. E
8. B
9. C
10. C
11. E
12. E
13. C
14. C
15. E
16. C
17. D

LISTA DE QUESTÕES - FILAS - MULTIBANCAS

1. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Análise de Sistemas) Em um programa existe a necessidade de guardar todas as alterações feitas em determinado dado para que seja possível desfazer alterações feitas ao longo de toda a sua existência. Nessa situação, a estrutura de dados mais adequada para o armazenamento de todas as alterações citadas seria uma fila.**

2. **(CESPE - 2012 – TST – Analista de Sistemas – A) As pilhas e as filas são estruturas de dados essenciais para os sistemas computacionais. É correto afirmar que a fila é conhecida como lista LIFO - Last In First Out.**

3. **(CESPE - 2012 - TRE-RJ - Técnico Judiciário - Programação de Sistemas) As filas são estruturas com base no princípio LIFO (last in, first out), no qual os dados que forem inseridos primeiro na fila serão os últimos a serem removidos. Existem duas funções que se aplicam a todas as filas: PUSH, que insere um dado no topo da fila, e POP, que remove o item no topo da fila.**

4. **(FCC - 2012 - MPE-AP – Analista de Sistemas - A) Nas estruturas de dados, devido às características das operações da fila, o primeiro elemento a ser inserido será o último a ser retirado. Estruturas desse tipo são conhecidas como LIFO.**

5. **(FCC - 2012 - MPE-AP – Analista de Sistemas - C) Nas estruturas de dados, a fila é uma lista linear na qual as operações de inserção e retirada ocorrem apenas no início da lista.**

6. **(FCC - 2012 - TRE-SP - Analista Judiciário - Análise de Sistemas – D) Pela definição de fila, se os elementos são inseridos por um extremo da lista linear, eles só podem ser removidos pelo outro.**

7. **(FCC - 2011 - TRT - 19ª Região (AL) - Analista Judiciário - Tecnologia da Informação) FIFO refere-se a estruturas de dados do tipo:**

   a) fila.

   b) árvore binária.

   c) pilha.

   d) matriz quadrada.

   e) cubo.

8. **(ESAF - 2010 - CVM - Analista de Sistemas - prova 2) Uma fila é um tipo de lista linear em que:**

a) as inserções são realizadas em um extremo e as remoções no outro extremo.

b) as inserções e remoções são realizadas em um mesmo extremo.

c) podem ser realizadas apenas inserções.

d) a inserção de um elemento requer a remoção de outro elemento.

e) a ordem de saída não corresponde à ordem de entrada dos elementos.

9. **(CESPE - 2010 - DETRAN-ES - Analista de Sistemas) No armazenamento de dados pelo método FIFO (first in - first out), a estrutura de dados é representada por uma fila, em cuja posição final ocorrem inserções e, na inicial, retiradas.**

10. **(CESPE - 2008 - TRT - 5ª Região (BA) - Técnico Judiciário - Tecnologia da Informação) Entre alguns tipos de estrutura de dados, podem ser citados os vetores, as pilhas e as filas.**

11. **(CESPE - 2004 – SES/PA – Analista de Sistemas) Uma estrutura mais geral que as pilhas e filas é o deque, em que as inserções, retiradas e acessos são permitidos em ambas as extremidades.**

12. **(CESPE - 2009 – TCE/AC – Analista de Sistemas – D) Um deque (double ended queue) requer inserção e remoção no topo de uma lista e permite a implementação de filas com algum tipo de prioridade. A implementação de um deque, geralmente é realizada com a utilização de uma lista simplesmente encadeada.**

13. **(FCC - 2007 – TRT/23 – Analista de Sistemas) Uma estrutura de dados com vocação de FIFO de duplo fim e que admite a rápida inserção e remoção em ambos os extremos é:**

a) uma pilha.

b) uma splay tree.

c) um deque.

d) uma lista linear.

e) uma árvore AVL.

14. **(CESPE - 2004 – PBV/RR - Analista de Sistemas) As filas com prioridade são listas lineares nas quais os elementos são pares da forma (qi, pi), em que q é o elemento do tipo base e p é uma prioridade. Elas possuem uma política de fila do tipo FIFO (first in first out) entre os elementos de mesma prioridade.**

15. (CESPE - 2004 – STJ – Analista de Sistemas) A seguir, está representada corretamente uma operação de retirada em uma fila de nome f.

se f.começo = nil então

erro {fila vazia}

senão j ← f.começo.info

16. (FCC - 2016 - Prefeitura de Teresina - PI - Analista Tecnológico - Analista de Suporte Técnico) Considerando uma estrutura de dados do tipo fila, e a seguinte sequência de comandos sobre essa fila (sendo que o comando Push representa uma inserção de elemento e o comando Pop representa uma exclusão de elemento) e considerando também que a fila estava inicialmente vazia:

Push 3, Push 5, Pop 3, Push 7, Pop 5, Push 9, Push 8

Após a execução dessa sequência de comandos, o conjunto de elementos que resulta na fila é:

a) 3 – 5 – 7 – 9 – 8.

b) 7 – 9 – 8 – 3 – 5.

c) 3 – 3 – 5 – 5 – 7 – 9 – 8.

d) 7 – 9 – 8.

e) 3 – 5 – 3 – 7 – 5 – 9 – 8.

17. (FCC - 2016 – TRT - 23ª REGIÃO (MT) - Técnico Judiciário - Tecnologia da Informação) Estruturas de dados básicas, como as pilhas e filas, são usadas em uma gama variada de aplicações. As filas, por exemplo, suportam alguns métodos essenciais, como o

a) enqueue(x), que insere o elemento x no fim da fila, sobrepondo o último elemento.

b) dequeue(), que remove e retorna o elemento do começo da fila; um erro ocorrerá se a fila estiver vazia.

c) push(x), que insere o elemento x no topo da fila, sem sobrepor nenhum elemento.

d) pop(), que remove o elemento do início da fila e o retorna, ou seja, devolve o último elemento inserido.

e) top(), que retorna o elemento do fim da fila sem removê-lo; um erro ocorrerá se a fila estiver vazia.

18. (FCC - 2017 – TRE/BA - Analista de Sistemas) A estrutura que, além de ser similar à fila, é apropriada para ampliar as características desta, permitindo inserir e retirar elementos tanto do início quanto do fim da fila, é o(a):

a) árvore.

b) lista duplamente encadeada.

c) deque.

d) fila circular.

e) pilha

GABARITO

1. E
2. E
3. E
4. E
5. E
6. C
7. A
8. A
9. C
10. C
11. C
12. E
13. C
14. C
15. E
16. D
17. B
18. C

LISTA DE QUESTÕES - ESTRUTURAS DE DADOS - ÁRVORE #42 - MULTIBANCAS

1. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As operações de busca em uma árvore binária não a alteram, enquanto operações de inserção e remoção de nós provocam mudanças sistemáticas na árvore.**

2. **(CETAP - 2010 - AL-RR - Analista de Sistemas - A) Uma árvore binária é aquela que tem como conteúdo somente valores binários.**

3. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) O tipo de dados árvore representa organizações hierárquicas entre dados.**

4. **(CETAP - 2010 - AL-RR - Analista de Sistemas - B) Uma árvore é composta por duas raízes, sendo uma principal e a outra secundária.**

5. **(CESPE - 2010 - DETRAN-ES - Analista de Sistemas) Denomina-se árvore binária a que possui apenas dois nós.**

6. **(FUNCAB - 2010 - SEJUS-RO - Analista de Sistemas - II) Árvores são estruturas de dados estáticas com sua raiz representada no nível um.**

7. **(CESPE - 2009 - ANAC - Especialista em Regulação - Economia) Considerando-se uma árvore binária completa até o nível 5, então a quantidade de folhas nesse nível será 24.**

8. **(CESPE - 2009 – ANAC - Analista de Sistemas) Uma árvore binária completa até o nível 10 tem 2.047 nós.**

9. **(FGV – 2015 – DPE/MT – Analista de Sistemas) No desenvolvimento de sistemas, a escolha de estruturas de dados em memória é especialmente relevante. Dentre outras classificações, é possível agrupar essas estruturas em lineares e não lineares, conforme a quantidade de sucessores e antecessores que os elementos da estrutura possam ter. Assinale a opção que apresenta, respectivamente, estruturas de dados lineares e não lineares.**

   a) Tabela de dispersão e fila.

   b) Estrutura de seleção e pilha.

c) Pilha e estrutura de seleção.

d) Pilha e árvore binária de busca.

e) Fila e pilha.

10. (CESPE - 2010 – TRE/MT – Analista de Sistemas – B) As listas, pilhas, filas e árvores são estruturas de dados que têm como principal característica a sequencialidade dos seus elementos.

11. (FCC - 2012 - MPE-AP - Analista Ministerial - Tecnologia da Informação – A) A árvore é uma estrutura linear que permite representar uma relação de hierarquia. Ela possui um nó raiz e subárvores não vazias.

12. (CESPE - 2010 – TRE/MT - Analista de Sistemas – E) O uso de recursividade é totalmente inadequado na implementação de operações para manipular elementos de uma estrutura de dados do tipo árvore.

13. (FCC - 2011 - TRT - 19ª Região (AL) - Técnico Judiciário - Tecnologia da Informação) Em uma árvore binária, todos os nós têm grau:

a) 2.

b) 0, 1 ou 2.

c) divisível por 2.

d) maior ou igual a 2.

e) 0 ou 1.

14. (CESPE - 2011 – STM – Analista de Sistemas) Enquanto uma lista encadeada somente pode ser percorrida de um único modo, uma árvore binária pode ser percorrida de muitas maneiras diferentes.

15. (FCC - 2016 - Prefeitura de Teresina - PI - Analista Tecnológico - Analista de Suporte Técnico) Considerando a estrutura de dados denominada árvore,

a) a sua altura é definida como a profundidade média de todos os seus vértices.

b) um vértice com um ou dois filhos é denominado folha.

c) cada nó tem no mínimo dois filhos em uma árvore binária.

d) as folhas de uma árvore binária completa podem ter profundidades distintas entre si.

e) a profundidade de um vértice em uma árvore é definida como o comprimento da raiz da árvore até esse vértice.

16. (CESPE – 2017 – TRE/BA - Analista de Sistemas) No estabelecimento de uma estrutura hierárquica, foi definida a seguinte árvore binária S:

S = (12(10(9(8))(11))(14(13)(15)))

Considerando o resultado da operação de exclusão do nó 12, assinale a opção que corresponde a nova estrutura da árvore S.

a) (10(9(8))(11(14(13)(15)))

b) (11(9(8)(10))(14(13)(15)))

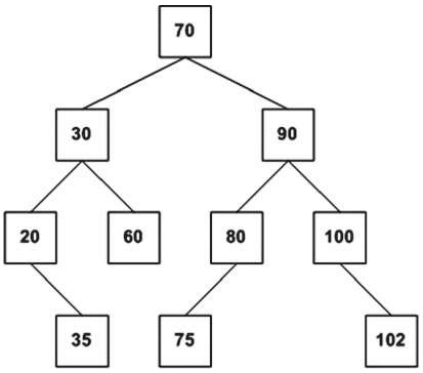
c) (11(10(9(8))(14(13)(15)))

d) (13(10(9)(11))(14(15)))

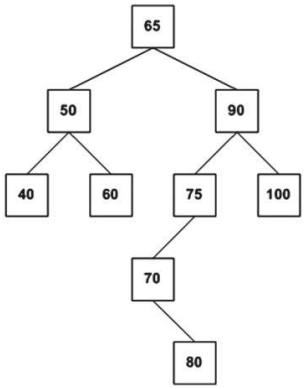
e) (13(11(9)(10))(14(15)))

17. (CESGRANRIO – 2012 – PETROBRÁS - Analista de Sistemas) Qual figura representa uma árvore AVL?

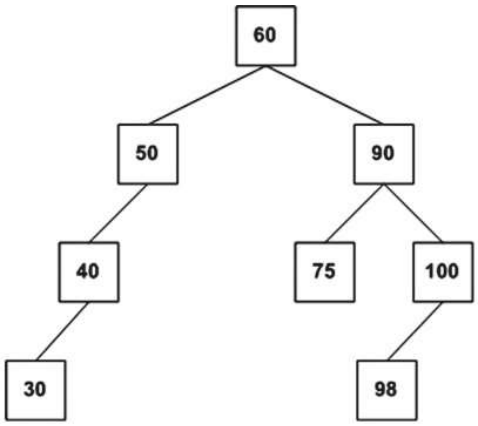
a)



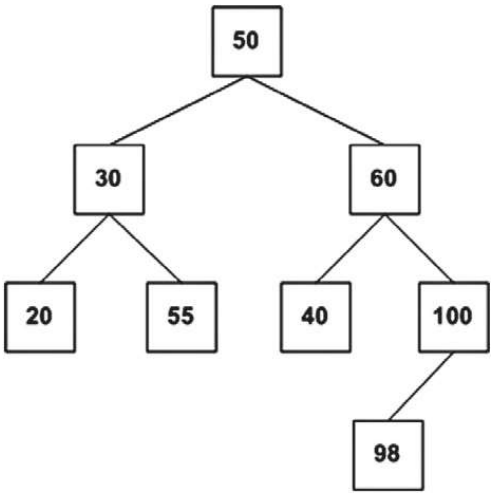
b)



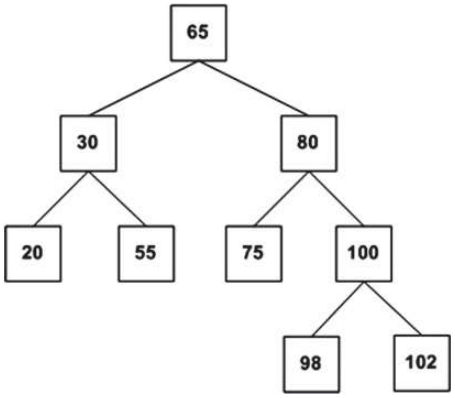
c)


60
50
90
40
75
100
30
98

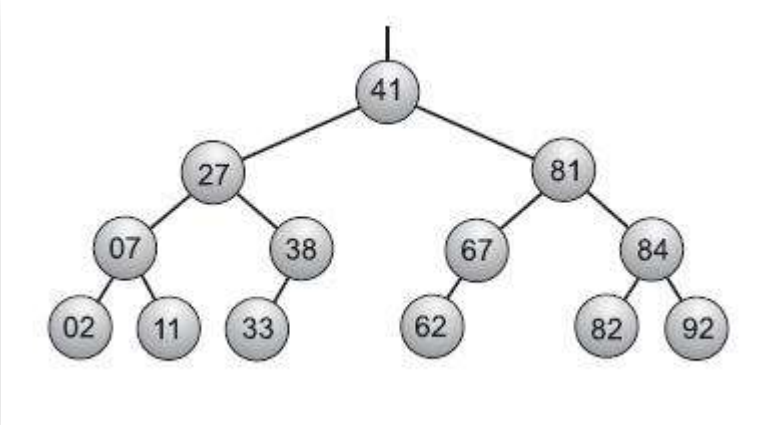
d)


50
30
60
20
55
40
100
98

e)


65
30
80
20
55
75
100
98
102

18. (CESGRANRIO – 2006 – DECEA - Analista de Sistemas) Suponha a seguinte árvore AVL.



A inserção do elemento 30 nessa árvore:

a) aumenta a profundidade da árvore após uma rotação.

b) provoca uma rotação à direita.

c) deixa os nós 02 e 07 no mesmo nível.

d) altera a raiz da árvore (nó 41).

e) torna o nó 33 pai do nó 27.

19. (CESPE – 2012 – TJ/RO - Analista de Sistemas) Assinale a opção em que é apresentado exemplo de estrutura de informação do tipo abstrata, balanceada, não linear e com relacionamento hierárquico.

a) lista duplamente encadeada

b) árvore binária

c) pilha

d) árvore AVL

e) deque

20. (FCC – 2008 – TRT/18 - Analista de Sistemas) Árvore AVL balanceada em altura significa que, para cada nó da árvore, a diferença entre as alturas das suas sub-árvores (direita e esquerda) sempre será:

a) menor ou igual a 2.

b) igual a 0 ou -1.

c) maior que 1.

d) igual a 1.

e) igual a -1, 0 ou 1.

21. (CESGRANRIO – 2010 – PETROBRÁS - Analista de Sistemas) Uma árvore AVL é uma estrutura de dados muito usada para armazenar dados em memória. Ela possui algumas propriedades que fazem com que sua altura tenha uma relação muito específica com o número de elementos nela armazenados. Para uma folha, cuja altura é igual a um, tem-se uma árvore AVL com 6 nós.

Qual é a altura máxima que esta árvore pode ter?

a) 6

b) 5

c) 4

d) 3

e) 2

22. (CESGRANRIO – 2011 – PETROBRÁS - Analista de Sistemas) Uma árvore AVL é uma árvore binária de busca autobalanceada que respeita algumas propriedades fundamentais. Como todas as árvores, ela tem uma propriedade chamada altura, que é igual ao valor da altura de sua raiz.

Sabendo que a altura de uma folha é igual a um e que a altura de um nó pai é igual ao máximo das alturas de seus filhos mais um, qual estrutura NÃO pode representar uma árvore AVL?

a) Uma árvore vazia

b) Uma árvore com dois nós

c) Uma árvore com três nós e altura igual a dois

d) Uma árvore com três nós e altura igual a três

e) Uma árvore com seis nós e altura igual a três

23. (CESGRANRIO – 2011 – PETROBRÁS - Analista de Sistemas) Após a inserção de um nó, é necessário verificar cada um dos nós ancestrais desse nó inserido, relativamente à consistência com as regras estruturais de uma árvore AVL.

PORQUE

O fator de balanceamento de cada nó, em uma árvore AVL, deve pertencer ao conjunto formado por {-2, -1, 0, +1, +2}.

Analisando-se as afirmações acima, conclui-se que:

a) as duas afirmações são verdadeiras, e a segunda justifica a primeira.

b) as duas afirmações são verdadeiras, e a segunda não justifica a primeira.

c) a primeira afirmação é verdadeira, e a segunda é falsa.

d) a primeira afirmação é falsa, e a segunda é verdadeira.

e) as duas afirmações são falsas.

24. (CESGRANRIO – 2010 – EPE - Analista de Sistemas) Um programador decidiu utilizar, em determinado sistema de análise estatística, uma árvore AVL como estrutura de dados. Considerando-se n a quantidade de elementos dessa árvore, o melhor algoritmo de pesquisa, com base em comparações, possui complexidade de tempo, no pior caso, igual a:

a) O(1)

b) O(log n).

c) Ω(n)

d) Ω(n log n)

e) $\Omega(n^2)$

25. (CESGRANRIO – 2012 – PETROBRÁS - Analista de Sistemas) Todos os N nomes de uma lista de assinantes de uma companhia telefônica foram inseridos, em ordem alfabética, em três estruturas de dados: uma árvore binária de busca, uma árvore AVL e uma árvore B.

As alturas resultantes das três árvores são, respectivamente,

a) O(Log(N)), O(Log(N)), O(1)

b) O(Log(N)), O(N), O(Log(N))

c) O(N), O(Log(N)), O(1)

d) O(N), O(Log(N)), O(Log(N))

e) O(N), O(N), O(Log(N))

26. (IBFC – 2014 – TRE/AM - Analista de Sistemas) Quanto ao Algoritmo e estrutura de dados no caso de árvore AVL (ou árvore balanceada pela altura), analise as afirmativas abaixo, dê valores Verdadeiro (V) ou Falso (F) e assinale a alternativa que apresenta a sequência correta de cima para baixo:

( ) Uma árvore AVL é dita balanceada quando, para cada nó da árvore, a diferença entre as alturas das suas sub-árvores (direita e esquerda) não é maior do que um.

( ) Caso a árvore não esteja balanceada é necessário seu balanceamento através da rotação simples ou rotação dupla.

Assinale a alternativa correta:

a) F-F

b) F-V

c) V-F

d) V-V

27. (CESGRANRIO – 2010 – PETROBRÁS - Analista de Sistemas) Uma sequência desordenada de números armazenada em um vetor é inserida em uma árvore AVL. Após a inserção nesta árvore, é feito um percurso em ordem simétrica (em ordem) e o valor de cada nó visitado é inserido em uma pilha. Depois de todos os nós serem visitados, todos os números são retirados da pilha e apresentados na tela. A lista de números apresentada na tela está:

a) ordenada ascendentemente de acordo com os números.

b) ordenada descendentemente de acordo com os números.

c) na mesma ordem do vetor original.

d) na ordem inversa do vetor original.

e) ordenada ascendentemente de acordo com sua altura na árvore.

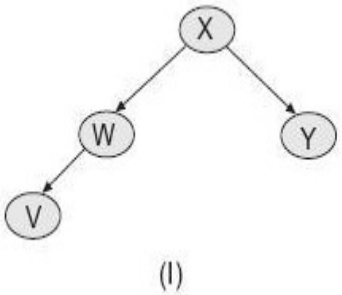
28. (FGV – 2009 – MEC - Analista de Sistemas) Acerca das estruturas de dados Árvores, analise as afirmativas a seguir.

I. A árvore AVL é uma árvore binária com uma condição de balanço, porém não completamente balanceada.

II. Árvores admitem tratamento computacional eficiente quando comparadas às estruturas mais genéricas como os grafos.

III. Em uma Árvore Binária de Busca, todas as chaves da subárvore esquerda são maiores que a chave da raiz.

29. (CESPE – 2014 – TJ/SE - Analista de Sistemas) Em uma árvore AVL (Adelson-Velsky e Landis), caso a diferença de altura entre as sub-árvores de um nó seja igual a 2 e a diferença de altura entre o nó filho do nó desbalanceado seja igual a -1, deve-se realizar uma rotação dupla com o filho para a direita e o pai para a esquerda a fim de que a árvore volte a ser balanceada.



(I)

(II)

30. (CESPE – 2010 – PETROBRÁS - Analista de Sistemas) As árvores usadas como estruturas de pesquisa têm características especiais que garantem sua utilidade e propriedades como facilidade de acesso aos elementos procurados em cada instante. A esse respeito, considere as afirmações abaixo.

I - A árvore representada na figura (I) acima não é uma árvore AVL, pois as folhas não estão no mesmo nível.

II - A sequência 20, 30, 35, 34, 32, 33 representa um percurso sintaticamente correto de busca do elemento 33 em uma árvore binária de busca.

III - A árvore representada na figura (II) acima é uma árvore binária, apesar da raiz não ter filhos.

É (São) correta(s) APENAS a(s) afirmativa(s):

a) I.

b) II.

c) III.

d) I e II.

e) II e III.

31. (CESPE – 2010 – PETROBRÁS - Analista de Sistemas) No sistema de dados do Departamento de Recursos Humanos de uma grande empresa multinacional, os registros de funcionários são armazenados em uma estrutura de dados do tipo árvore binária AVL, onde cada registro é identificado por uma chave numérica inteira. Partindo de uma árvore vazia, os registros cujas chaves são 23, 14, 27, 8, 18, 15, 30, 25 e 32 serão, nessa ordem, adicionados à árvore.

Dessa forma, o algoritmo de inserção na árvore AVL deverá realizar a primeira operação de rotação na árvore na ocasião da inserção do elemento:

a) 30

b) 25

c) 18

d) 15

e) 8

32. (CESPE – 2014 – TJ/SE - Analista de Sistemas) Existem dois vetores, chamados A e B, que estão ordenados e contêm N elementos cada, respeitando a propriedade A[N-1]<B[0], onde os índices de ambos os vetores vão de 0 a N-1. Retiram-se primeiro todos os elementos de A na ordem em que se apresentam e inserem-se esses elementos em uma árvore binária de busca, fazendo o mesmo depois com os elementos de B, que são inseridos na mesma árvore de busca que os de A. Depois, retiram-se os elementos da árvore em um percurso pós ordem, inserindo-os em uma pilha. Em seguida retiram-se os elementos da pilha, que são inseridos de volta nos vetores, começando pelo elemento 0 do vetor A e aumentando o índice em 1 a cada inserção, até preencher todas as N posições, inserindo, então, os N elementos restantes no vetor B da mesma maneira.

Ao final do processo, tem-se que os vetores:

a) estão ordenados e A[i] < B[i], para todo i=0,...., N-1.

b) estão ordenados e A[i] > B[i], para todo i=0,...., N-1.

c) estão ordenados e não existe mais uma propriedade que relacione A[i] e B[i].

d) não estão ordenados e A[i] < B[i], para todo i=0,...., N-1.

e) não estão ordenados e A[i] > B[i], para todo i=0,...., N-1.

GABARITO

1. C
2. E
3. C
4. E
5. E
6. E
7. E
8. C
9. D
10. E
11. E
12. E
13. B
14. C
15. E
16. X
17. E
18. B
19. D
20. E
21. D
22. D
23. C
24. B
25. D
26. D
27. B
28. B
29. C
30. E
31. D
32. A

LISTA DE QUESTÕES - GRAFOS - MULTIBANCAS

1. (CESPE - 2010 – TJ/ES – Analista de Suporte) Considerando-se a implementação de um grafo denso, direcionado e ponderado, se o número de vértices ao quadrado tem valor próximo ao número de arcos, o uso de uma matriz de adjacência simétrica apresenta vantagens em relação ao uso de uma lista de adjacência.

2. (FCC - 2013 – MPE/SE – Analista do Ministério Público) Considere:

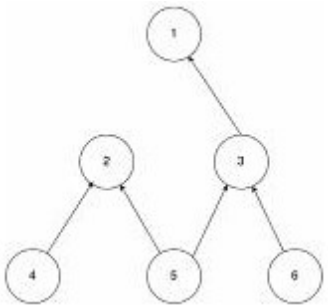
   I. Estrutura de dados que possui uma sequência de células, na qual cada célula contém um objeto de algum tipo e o endereço da célula seguinte.

   II. Podem ser orientados, regulares, completos e bipartidos e possuir ordem, adjacência e grau.

   III. Possuem o método de varredura esquerda-raiz-direita (e-r-d).

   Os itens de I a III descrevem, respectivamente,
   a) árvores binárias, listas ligadas e arrays.
   b) arrays, árvores binárias e listas ligadas.
   c) grafos, árvores binárias e arrays.
   d) listas ligadas, grafos e árvores binárias.
   e) grafos, listas ligadas e árvores binárias.

3. (CESPE - 2013 – TCE/ES – Analista de Sistemas) Considerando o grafo ilustrado acima, assinale a opção em que é apresentada a descrição em vértices (V) e arestas (A).



   a) V = {1, 2, 3, 4, 5, 6 }
   A = {(2, 4), (2, 3), (2, 5), (3, 6), (1, 5)}

   b) V = { 2, 4, 1, 3, 6, 5 }
   A = {(4, 2), (1, 3), (5, 2), (6, 3), (5, 3)}

   c) V = {1, 2, 3, 4, 5, 6 }
   A = {(4, 2), (3, 4), (5, 2), (6, 3), (5, 3)}

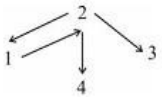
   d) V = {1, 2, 3, 4, 5, 6 }

A = {(4, 2), (3, 1), (5, 1), (6, 2), (5, 3)}

e) V = { 2, 4, 1, 3, 6, 5 }
A = {(4, 2), (3, 1), (5, 2), (6, 3), (5, 3)}

4. (CESPE - 2012 – TJ/RO – Analista de Suporte – ITEM B) Grafo corresponde a uma estrutura abstrata de dados que representa um relacionamento entre pares de objetos e que pode armazenar dados em suas arestas e vértices, ou em ambos.

5. (CESPE - 2012 – PEFOCE – Perito Criminal) Considere que um grafo G seja constituído por um conjunto (N) e por uma relação binária (A), tal que G = (N, A), em que os elementos de N são denominados nós (ou vértices) e os elementos de A são denominados arcos (ou arestas). Em face dessas informações e do grafo abaixo, é correto afirmar que esses conjuntos são N = {1,2,3,4} e A = {(1,2),(2,1),(2,4),(2,3)}.



6. (CESPE - 2012 – BASA – Analista de Sistemas) É misto o grafo com arestas não dirigidas que representam ruas de dois sentidos e com arestas dirigidas que correspondem a trechos de um único sentido, modelado para representar o mapa de uma cidade cujos vértices sejam os cruzamentos ou finais de ruas e cujas arestas sejam os trechos de ruas sem cruzamentos.

7. (CESPE - 2012 – BASA – Analista de Sistemas) Para modelar a rede que conecta todos os computadores em uma sala de escritório com a menor metragem possível de cabos, é adequado utilizar um grafo G cujos vértices representem os possíveis pares (u, v) de computadores e cujas arestas representem o comprimento dos cabos necessários para ligar os computadores u e v, determinando-se o caminho mínimo, que contenha todos os vértices de G, a partir de um dado vértice v.

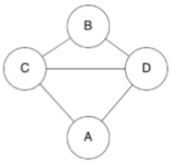
8. (CESPE - 2012 – BASA – Analista de Sistemas) Um grafo completo contém pelo menos um subgrafo ponderado.

9. (CESPE - 2012 – BASA – Analista de Sistemas) Um grafo não direcionado é dito conectado quando há pelo menos um caminho entre dois vértices quaisquer do grafo.

10. (CESPE - 2012 – TJ/AC – Analista de Sistemas) Define-se um grafo como fortemente conexo se todos os nós puderem ser atingidos a partir de qualquer outro nó.

11. (CESPE - 2013 – CRPM – Analista de Sistemas) Considere que o grafo não orientado representado na figura abaixo possua as seguintes características:

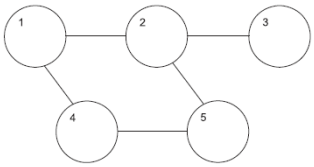
G1 = (V1, A1)
V1 = {A, B, C, D}
A1 = {(A, C), (A, D), (B, C), (B, D), (A,B)}.



Nesse caso, é correto afirmar que o grafo G1 possui quatro vértices, nomeados de A, B, C e D, e cinco arcos, que conectam pares de vértices, conforme especificado em A1.

12.(CESPE - 2012 – BASA – Analista de Sistemas) A implementação de um grafo do tipo ponderado e direcionado na forma de uma matriz de adjacência utiliza menor quantidade de memória que a implementação desse mesmo grafo na forma de uma lista encadeada.

13.(CESGRANRIO - 2014 – BASA – Analista de Sistemas) O grafo acima pode ser representado pela seguinte matriz:



a) $\begin{vmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{vmatrix}$

b) $\begin{vmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{vmatrix}$

c) $\begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{vmatrix}$

d) $\begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{vmatrix}$

e) $\begin{vmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{vmatrix}$

14.(CESPE - 2012 – TJ/SE – Analista de Sistemas) Um grafo é formado por um par de conjuntos de vértices e arestas, não podendo o conjunto de vértices ser particionado em subconjuntos.

15.(CESPE - 2012 – TRT/AM – Analista de Sistemas) Um grafo é uma estrutura de dados consistida em um conjunto de nós (ou vértices) e um conjunto de arcos (ou arestas). O grafo em que os arcos possuem um número ou peso associados a eles, é chamado de grafo:
a) predecessor.
b) adjacente.
c) incidente.
d) ponderado.
e) orientado.

GABARITO

1. E
2. D
3. E
4. C
5. C
6. C
7. E
8. E
9. C
10. C
11. E
12. E
13. A
14. E
15. D

LISTA DE QUESTÕES - HASHING - MULTIBANCAS

1. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) A pesquisa sequencial é aplicável em estruturas não ordenadas.**

2. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) As colisões ocorrem na utilização de tabela hash porque várias chaves podem resultar na mesma posição.**

3. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação - Administração de Dados) Ocorre o hashing quando não há o armazenamento de cada entrada de uma tabela em um específico endereço calculado a partir da aplicação de uma função chave da entrada.**

4. **(FCC - 2008 - METRÔ-SP - Analista Treinee - Análise de Sistemas) O objetivo de fazer uma busca rápida a partir de uma chave de pesquisa simples e obter o valor desejado é alcançado pela estrutura de dados especial denominada:**

   a) array.

   b) lista.

   c) vetor.

   d) árvore binária.

   e) tabela de hashing.

5. **(CESPE - 2012 - Banco da Amazônia - Técnico Científico - Administração de Dados) A busca que utiliza uma tabela hash realiza comparação das chaves para encontrar a posição do elemento que está sendo buscado.**

6. **(CESPE - 2010 - DETRAN-ES - Analista de Sistemas) No método de hashing, por meio de acesso sequencial, são utilizados tabelas e mapas para recuperar informações de endereço de arquivos de forma rápida e eficiente.**

7. **(FCC - 2015 - DPE-SP - Programador) Um Programador da Defensoria Pública do Estado de São Paulo foi solicitado a propor uma solução para o problema: Há uma quantidade grande de dados classificáveis por chave e estes dados devem ser divididos em subconjuntos com base em alguma característica das chaves. Um método eficiente deve ser capaz de localizar em qual subconjunto deve-se colocar cada chave e depois estes subconjuntos bem menores devem ser gerenciados por algum método simples de busca para que se localize uma chave**

rapidamente. O Programador propôs como solução, corretamente, a implementação de:

a) Deques.

b) Tabela e função hash.

c) Pilhas.

d) Fila duplamente encadeada.

e) Árvore Binária de Busca.

GABARITO

1. C
2. C
3. E
4. E
5. E
6. E
7. B

LISTA DE QUESTÕES - BITMAP - MULTIBANCAS

1. (FGV - 2015 – TJ-RO – Analista Judiciário - Análise de Sistemas)

ID	Nome	Curso
1210	A	Física
356	B	Química
23	C	Matemática
57	D	Física
45	E	Física
6	F	Matemática
210	G	Matemática

Se fosse construído um índice de banco de dados do tipo “bitmap" para essa tabela, tendo o campo Curso como chave, o conteúdo desse índice seria:

a)

6 3

23 3

45 1

57 1

210 3

356 2

1210 1

b)

Física 1001100

Química 0100000

Matemática 0010011

c)

6 001

23 001

45 100

57 100

210	001
356	010
1210	100

d)

0011001100
0100100000
0010010011

e)

00000000110	Matemática
00000010111	Matemática
00000101101	Física
00000111011	Física
00011010010	Matemática
00101100100	Química
10010111010	Química

2. (FGV - 2014 – DPE-RJ – Técnico Superior Especializado - Administração de Dados)

Candidato	
inscrição	**candidatoNome**
101	João
102	Maria
105	Gabriela
106	João

Prova	
provaNome	**data**
Português	12/01/2014
Matemática	12/01/2014
Prática	19/01/2014

Avaliação		
inscrição	**provaNome**	**nota**
101	Português	10
102	Português	8
105	Português	3
106	Português	5
101	Matemática	7
102	Matemática	4
105	Matemática	8
101	Prática	5
102	Prática	5
105	Prática	NULL
106	Prática	4

Índices do tipo Bitmap podem ser usados em tabelas de bancos de dados. O quadro abaixo representa o mapa de bits na indexação da tabela Avaliação quando a chave considerada é a concatenação dos atributos Inscrição e provaNome.

101Matemática 00001000000
101Português 10000000000
101Prática 00000001000
102Matemática 00000100000
102Português 01000000000
102Prática 00000000100
105Matemática 00000010000
105Português 00100000000
105Prática 00000000010
106Português 00010000000
106Prática 00000000001

Num mapa de bits para a mesma tabela, usando apenas o atributo Inscrição, o mapa de bits seria

a)
101 10001001000
102 01000100100
105 00100010010
106 00010000001

b)
101 1000
102 0100
105 0010
106 0001

c)
101 10000000000
102 01000100000
105 00100000000
106 00010000000

d)
101 00000001000
102 00000000100
105 00000000010
106 00000000001

e)
101 111
102 111
105 111
106 011

GABARITO

1. B
2. A

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.