07

Para saber mais - algorithm

A biblioteca `algorithm` tem algoritmos muito úteis já programados, como o comando `sort` para ordenar um vetor. Conhecer as principais funções dessa biblioteca ajuda muito a resolver problemas mais rapidamente, o que é essencial durante a competição.

Nesta seção, vamos conhecer mais duas funções dessa biblioteca: o `lower_bound` e o `upper_bound`, que realizam a busca binária em um vetor ordenado.

Utilizaremos o `lower_bound` para encontrar em um vetor ordenado o primeiro elemento maior ou igual a um determinado valor, enquanto que o `upper_bound` é utilizado para encontrarmos o primeiro elemento do vetor ordenado estritamente maior que um determinado valor.

Ambas funções recebem três parâmetros, o endereço do início do vetor, o endereço da posição seguinte ao último elemento do vetor e o valor que se quer buscar. As duas retornam **iteradores**, que são estruturas muito similares aos ponteiros.

Exemplo:

Se tivermos um vetor de 6 elementos, que chamaremos:

```
int s[] = {3, 5, 6, 10, 17};
```

O endereço do começo do vetor pode ser escrito por `&s[0]`, ou ainda, simplesmente por `s`, já o endereço da posição após o final do vetor será o `&s[6]`, ou mesmo, `s+6`. As duas representações de endereço são igualmente válidas, aqui usaremos a representação mais compacta.

Chamar a função `lower_bound(s, s+6, 6)` retorna um iterador indicando a primeira posição que contém um elemento maior ou igual a 6. Para acessarmos o valor deste iterador, usamos o operador `*`, assim como fazíamos com os ponteiros.

Sendo assim, `*lower_bound(s, s+6, 6)` retornará o valor do primeiro elemento da sequência maior ou igual a 6, que é o próprio 6.

Do mesmo modo, `*upper_bound(s, s+6, 6)` retornará o valor do primeiro elemento estritamente maior que 6, ou seja, o 10.

Além de conseguirmos saber o valor dos iteradores retornados usando `*`, como acabamos de ver, fazendo uma conta de subtração com iteradores também conseguimos descobrir a posição que o iterador retornado ocupa no vetor.

Em C++ quando subtraímos dois iteradores, recebemos um valor indicando a distância entre os iteradores subtraídos. Vamos nos aproveitar disso para descobrir a posição de um iterador retornado pelas funções `lower_bound` e `upper_bound`.

No exemplo, como discutimos, `*lower_bound(s, s+6, 6)` vale 6.

Para descobrir a posição desse elemento no vetor subtraímos o iterador retornado do `lower_bound` com o iterador, ou endereço, do começo do vetor: `(lower_bound(s, s+6, 6) - s)`.

Neste exemplo, a subtração valerá 2, que é justamente a posição do valor 6 no vetor dado.

Do mesmo modo, `(upper_bound(s, s+6, 6) - s)` retorna o valor 3, que é a posição do elemento 10 no vetor dado.

Tome cuidado!

Temos que tomar cuidado com duas coisas ao usar os comandos descritos:

1. O vetor passado como parâmetro deve estar ordenado, do contrário a função não funcionará como esperado;
2. Imagine que queremos procurar no vetor `s` do exemplo o valor 20. Como não existe nenhum valor na sequência maior ou igual a 20, usar `*lower_bound(s, s+6, 20)` retornará um valor aleatório, que chamamos de lixo de memória. Apesar de não podermos usar o operador `*`, ainda podemos usar a subtração de iteradores: `(lower_bound(s, s+6, 20) - s)` retorna 6, a posição logo após o final da sequência;

Para saber mais sobre essas funções e muitas outras, consulte o site referência de C++: lower_bound (http://www.cplusplus.com/reference/algorithm/lower_bound/), upper_bound (http://www.cplusplus.com/reference/algorithm/upper_bound/).