

## Movimentos

### Transcrição

[00:00] Já temos um código que movimenta nossos fantasmas. Ela localiza os fantasmas varrendo nossa matriz de array, de strings, e quando encontra o fantasma, ela movimenta. Só que sempre para a direita, é muito chato. É só a gente sempre andar para a esquerda. Não é legal.

[00:35] Temos que fazer o fantasma pelo menos andar para todos os lados. Tem quatro lados então que ele pode ir. Para baixo, para cima, para direita e para esquerda. Quero saber quais as posições válidas a partir da posição atual. Vamos pegar todas as posições a partir desse mapa, na posição linha coluna. E depois me devolve. Vamos criar a função.

[01:26] Vou começar com uma array vazia e no final vou devolver essa array. Vou verificar as quatro posições possíveis. Primeiro baixo, que é posição zero, posição um. Se eu quero olhar para baixo, tenho que somar uma linha. O de cima é o menos um então. Direita é a posição zero e posição um somado um. A esquerda é o inverso.

[02:25] Se a posição é válida, coloca baixo. Se a posição da direita é válida, coloca direita. Em todas a mesma coisa. Eu recebo um mapa e uma posição inicial, crio uma array de posições válidas. Se for válida, adiciona na array. No final, retorno uma array com todas as posições válidas a partir da atual.

[03:30] A posição que eu vou querer ir é uma dessas. Eu vou primeiro pegar a primeira de todas, que é zero. Pelo menos o movimento do fantasma vai ser para todos os lados. Eu simplesmente mando andar, porque sei que a posição é válida.

[04:12] E se não tiver nenhuma posição válida? E se o fantasma está travado no mapa porque o mapa foi malfeito? Ou ainda se ele estiver bloqueado por outro fantasma. Qualquer situação em que ele não pode se locomover. Se isso acontecer, posições vai ser vazio. Se posições é vazio, não faz nada.

[04:48] Em Ruby podemos escrever `return if posições.empty?`. É uma maneira de confirmar algo de guarda, de segurança. Estou garantindo que só vou executar o resto do código se isso não acontecer. É uma maneira clássica de escrever essa validação. Se está vazio, para por aqui, faz o `return`. Costumamos escrever muito em Ruby esse tipo de cláusula. Se você acha que não é natural para você ler da direita para a esquerda, não tem problema, escreva `if posições return`.

[05:40] Salvamos nosso código. Rodando isso, o fantasma anda. Mas um deles se teletransportou. Não só foi para baixo como foi para a direita. Está funcionando, mas não muito.