04

Zeros

Transcrição

[00:00] Fizemos a função remove chamar ela mesma. Quando ela chama ela mesma, ela anda um passo para a direita e remove. Só que como ela fica chamando ela mesma, uma hora esse cara explode. Queremos fazer isso só com quatro casas. Quando vira zero, quero parar.

[01:07] Quando a quantidade for zero, return. Rodando o código, funciona. Eu poderia escrever o código com return if quantidade for igual a zero. Ou poderia escrever return unless quantidade maior do que zero. É bem comum o pessoal escrever com unless. Eu pessoalmente não gosto mesmo nesse caso, porque é uma negação, e o maior é a negação do igual. Não quero usar dupla negação.

[02:15] O remove recebe o mapa, a posição a partir da qual ele vai começar a remover e quantas casas para a direita você quer que ele remova. Se você pediu para ele remover zero, ele fica quieto. Senão, anda para a direita, remove e chama ele mesmo, para remover o que você tinha pedido menos um. Uma hora o número chega no zero e para.

[02:48] É o mesmo for que tínhamos antes, só que feito através de você se invocando. Antes tínhamos um laço que era interativo. Agora, voltamos para a própria função. É a recursão. Ele para quando chega na base.

[03:35] Recapitulando, o remove é chamado com o mapa, a posição nova, e quatro casas para a direita. Ele anda para a direita, remove, chama ele mesmo, três, anda para a direita, remove, chama ele mesmo com dois, anda para a direita, remove, chama ele mesmo com um, anda para a direita, remove, chama ele mesmo com zero. E para. Repare que a variável posição é acumulador. Ela muda todas as vezes, que nem antes. E quantidade é o limite.

[04:12] Continuo tendo um acumulador, só que antes era uma variável extraída. Agora é uma variável que passo como argumento. Essa é a diferença. Antes fizemos um laço através de interação. Agora fizemos um laço através de uma recursão. Invocamos nós mesmos e executamos o mesmo código diversas vezes. O importante é invocar você mesmo.