Magic Rescue  
Relatório

Uma imagem com ar livre, mola

Descrição gerada automaticamente

# Uma imagem com texto, carta Descrição gerada automaticamenteResolução do Problema

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamenteOnde E, M e I, representam as funções auxiliares Empty, Monster e Item, respetivamente.

## Empty

A função E(b, i) coincide com o método resolveEmpty(), da classe MagicRescue onde irá preencher uma coluna da tabela, com a resolução matemática, consoante o item.

## Monster

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

## Uma imagem com texto Descrição gerada automaticamente

A função auxiliar monster é subdividida em três outras funções auxiliares - Fill3, FillT e FillD - que preenche uma coluna, de acordo com a respetiva notação matemática associada ao mostro que estiver em b, consoante o item.

## Item

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

A função auxiliar Item é subdividida em três outras funções auxiliares - FillH, FillP e FillC - que preenche uma coluna, de acordo com a respetiva notação matemática associada ao mostro que estiver em b, consoante o item.

# Complexidade Temporal

Na nossa solução, recorremos ao preenchimento de um vetor com 4 posições, correspondentes ao objeto de entrada, na posição do caminho que se está a resolver.

Nesta medida, a complexidade temporal está apenas dependente do tamanho do caminho, *n, e* do número de objetos equipáveis (considerando o caso de não ter qualquer objeto equipado como mais um objeto equipável), *w,* visto que as operações de escrita e leitura de elementos do vetor e comparação entre os mesmos têm complexidades constantes.

Isto é:

Podemos então substituir *w* pelo número de objetos equipáveis referidos no enunciado. Ora:

Por fim, sendo constante, podemos omitir o 4, resultando numa complexidade temporal linear. Ou seja:

# Complexidade Espacial

Na nossa implementação, fazemos uso de 2 vetores cujos comprimentos correspondem ao comprimento do caminho a resolver, *n*, e ao número de objetos equipáveis, *w*. Assim, temos como complexidade espacial:

Mais uma vez, podemos substituir *w* pelo valor fornecido pelo enunciado, obtendo:

Por fim, podemos omitir a complexidade constante resultando numa complexidade espacial total de:

# Conclusões

Inicialmente otimizamos uma solução onde era preenchida uma tabela de tamanho C x L onde *C* era o tamanho do percurso a percorrer e *L* era os tipos de itens que podemos carregar, incluindo o vazio. Este preenchimento era feito através de programação dinâmica. Posteriormente, de modo a melhorar a complexidade espacial, passamos a utilizar um *array*, também preenchido através de programação dinâmica, de tamanho L onde *L* são os tipos de itens que podemos carregar, incluindo o vazio. Deste modo conseguimos reduzir o espaço utilizado na nossa solução em *C*.

Depois desta alteração não encontramos possíveis melhoramentos