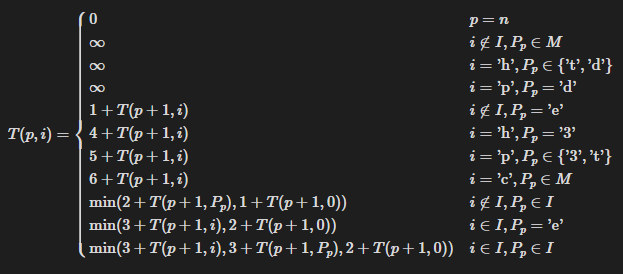
Resolução do Problema

## Solução recursiva

A função recursiva que resolve este problema é a seguinte:



Legenda:

I->lista de itens

M->lista de monstros

n->número total de carateres da string

P->string total

p->índice da casa atual

Pp->casa atual

i->item correspondente ao que o Harry e o Ron estão a carregar naquele momento

### Explicação:

#### Casos base:

Quando chegámos ao fim do caminho, o custo é 0;

Quando chegamos a uma casa com um monstro e não temos item, ou o item que temos não serve, retornar infinito para que essa possibilidade seja eliminada.

#### Casos gerais:

Se não tivermos com item e a casa for fácil sem objeto, o custo é 1 + o custo de percorrer o resto do caminho começando na casa seguinte sem item;

Se tivermos uma harpa e a casa atual tiver um cão de três cabeças, o custo é 4 + o custo de percorrer o resto do caminho começando na casa seguinte com a harpa;

Se tivermos uma poção e a casa atual tiver um cão de três cabeças ou um troll, o custo é 5 + o custo de percorrer o resto do caminho começando na casa seguinte com a poção;

Se tivermos um manto de invisibilidade e a casa atual tiver um cão de três cabeças, um troll ou um dragão, o custo é 6 + o custo de percorrer o resto do caminho começando na casa seguinte com o manto;

Se não tivermos item e a casa atual tiver um item, o custo é o mínimo entre 1 + o custo de percorrer o resto do caminho começando na casa seguinte sem item e 2 + o custo de percorrer o resto do caminho começando na casa seguinte com o item da casa atual;

Se tivermos um item e a casa atual não tiver item nem monstro, o custo é o mínimo entre 2 + o custo de percorrer o resto do caminho começando na casa seguinte sem item e 3 + o custo de percorrer o resto do caminho começando na casa seguinte com o item que tínhamos;

Se tivermos um item e a casa atual tiver um item, o custo é o mínimo entre 2 + o custo de percorrer o resto do caminho começando na casa seguinte sem item, 3 + o custo de percorrer o resto do caminho começando na casa seguinte com o item que tínhamos e 3 + o custo de percorrer o resto do caminho começando na casa seguinte com o item da casa atual.

## Solução iterativa

Inicialmente, implementámos uma solução recursiva do problema e depois, transformámos essa solução numa iterativa:

Num dado momento do caminho, existem 4 possíveis estados: O Harry e o Ron podem não ter nada, podem ter uma harpa, uma poção ou um manto de invisibilidade. Foi por isso que definimos quatro variáveis que gerem essas 4 possibilidades e os respetivos tempos. Em cada campo, essas variáveis são atualizadas.

A nossa solução iterativa é a seguinte:

Quando a casa atual não tem item nem monstro, noItemTime é igual ao mínimo de todas as variáveis + 1 + foundMonster\* e às outras variáveis somamos 3 unidades;

Quando a casa atual tem um item, noItemTime é igual ao mínimo de todas as variáveis + 1 + foundMonster\*, a variável correspondente ao item da casa atual é igual a noItemTime +1 e às outras variáveis somamos 3 unidades.

Quando a casa atual tem um monstro, noItemTime é igual a infinito, as variáveis correspondentes a itens fracos demais para o monstro da casa atual passam para infinito e para às restantes variáveis somamos o tempo correspondente a usar o item.

No fim, devolvemos o mínimo das quatro variáveis.

\*foundMonster é 1 se a casa anterior tiver um monstro e 0 se não tiver. Somamos esta variável, visto que se tivermos estado numa casa com monstro, entramos sempre nessa casa com um item. Como entramos nessa casa com um item, é necessário somar uma unidade de tempo.

# Complexidade Temporal

Complexidade temporal= L0+L1+L2…+LT=

=

ϴ(LT) = ϴ(n)

n=número total de carateres

A complexidade temporal da solução implementada é linear, visto que cada carater é analisado uma e só uma vez. É ϴ(n), visto que tanto no melhor, no pior e no caso esperado é ϴ(n), pois todos os carateres precisam de ser analisados.

Complexidade Espacial

Usamos no total seis variáveis. Quatro delas são do tipo inteiro e representam o tempo duma possibilidade (noItemTime, harpTime, PotionTime, cloakTime). Outra é do tipo inteiro e é 1 se a casa anterior tiver um monstro e 0 se não tiver (foundMonster).

Estas variáveis são usadas sempre. As variáveis correspondentes aos plots anteriores não são guardadas pelo que o valor da variável T (número de plots) não afeta a complexidade espacial.

Complexidade espacial = ϴ(noItemTime) + ϴ(harpTime) + ϴ(PotionTime) + ϴ(cloakTime) + ϴ(foundMonster) =

ϴ(1) + ϴ(1) + ϴ(1) + ϴ(1) + ϴ(1) =

ϴ(5) =

ϴ(1)

A complexidade espacial é constante, visto que não depende do input. É ϴ(1), porque tanto no melhor, no pior e no caso esperado a complexidade espacial é ϴ(1).

Conclusões

Um ponto forte é a complexidade temporal ser linear. Outro ponto forte é a espacial ser constante. Não há maneira de as reduzir. Um dos pontos fracos é o cálculo constante de mínimos. Estudámos uma alternativa que envolvia percorrer a string da direita para a esquerda, visto que sabendo qual o maior monstro, seria possível saber qual o instrumento necessário. No entanto, achámos a solução implementada mais simples.