PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática

Disciplina	Curso	Período
Técnicas Avançadas de Programação	Sistemas de Informação	5°

Professor
Kleber Jacques F. de Souza (klebersouza@pucminas.br)

Trabalho Prático - Son Goku

Instruções

- O trabalho deverá ser feito em grupo de no máximo 5 pessoas.
- A interpretação do problema faz parte do trabalho.

1 Especificação do Problema

Son Goku é um poderoso Saiyajin (raça de poderosos guerreiros). Ele está na busca pelas 7 esferas do dragão[1] com o objetivo de ressuscitar seu amigo Kuririn que foi morto por Freeza.

Goku já reuniu 6 das 7 esperas. Ele descobriu que a última esfera do dragão está em uma caverna escondida nos confins da selva. Até hoje nenhum guerreiro conseguiu recuperar a esfera, pois ela é bem guardada por terríveis monstros. Mas Goku não é um guerreiro qualquer e decidiu preparar-se para recuperar a última esfera que precisa para ressuscitar seu amigo.

Goku dispõe de uma certa quantidade de Ki (uma espécie de energia mágica) e de uma lista de M magias. Cada monstro tem um determinado número de pontos de vida. Cada vez que Goku lança uma magia contra um monstro, Goku gasta uma certa quantidade de Ki (o custo da magia) e inflinge um certo dano ao monstro. O dano infligido provoca a perda de pontos de vida do monstro (o número de pontos perdidos depende da magia). Um monstro está morto se tiver zero ou menos pontos de vida. Goku sempre luta contra um monstro a cada vez. Como é um guerreiro poderoso, ele pode usar a mesma magia várias vezes, desde que possua a quantidade necessária de Ki.

Em suas pesquisas, Goku conseguiu o mapa que leva a esfera do dragão. A caverna é representada como um conjunto de salões conectados por galerias. Os salões são identificados sequencialmente de 1 a N. Goku sempre inicia no salão 1 e a esfera do dragão está sempre no salão N. Existem K monstros identificados sequencialmente de 1 a K. Cada monstro vive em um salão, do qual não sai (note que é possível que mais de um monstro viva no mesmo salão). Durante a busca pela esfera, Goku pode sair ou recuperar a esfera de um salão somente se o salão estiver vazio (sem monstro). Em outras palavras, Goku deve sempre, antes de sair ou de recuperar a esfera de um salão, matar o(s) monstro(s) que lá viver(em).

Dadas as descrições das magias, dos monstros e da caverna, sua tarefa é determinar a quantidade mínima inicial de Ki necessária para que Goku consiga recuperar a esfera do dragão.

[1] São bolas laranjas e cristalinas que podem invocar o Deus Dragão Shenlong que tem a habilidade de conceder desejos para quem conseguir juntar as sete esferas.

2 Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém quatro inteiros M, N, G e K, indicando respectivamente o número de magias ($1 \leq M \leq 1.000$), de salões ($1 \leq N \leq 1.000$), de galerias ($0 \leq G \leq 1.000.000$) e de monstros ($0 \leq K \leq 1.000$).

Cada uma das M linhas seguintes descreve uma magia. A descrição de uma magia contém dois números inteiros, a quantidade de Ki consumida (entre 1 e 1.000) e o número de pontos de danos provocados (também entre 1 e 1.000).

Em seguida, há G linhas, cada uma descrevendo uma galeria. Uma galeria é descrita por dois números inteiros A e B ($A \neq B$), representando os salões que a galeria conecta. Goku pode utilizar a galeria nos dois sentidos, ou seja, para ir de A para B ou de B para A.

Finalmente, as últimas K linhas de um caso de teste descrevem os monstros. A descrição de um monstro contém dois números inteiros representando respectivamente o salão no qual ele vive (entre 1 e N inclusive) e o seu número inicial de pontos de vida (entre 1 e 1.000 inclusive).

O final da entrada é indicado por $M = N = G = K = 0$.

A entrada deve ser lida de um arquivo texto.

3 Saída

Para cada caso de teste da entrada seu programa deve produzir uma linha na saída contendo um número inteiro, a quantidade mínima inicial de Ki necessária. Caso não seja possível recuperar a esfera do dragão, você deve imprimir -1.

A saída deve ser impressa na tela do programa.

4 Exemplo

Entrada:

```
3 4 4 2
7 10
13 20
25 50
1 2
2 4
1 3
3 4
2 125
3 160
3 4 4 1
7 10
13 20
25 50
1 2
2 4
1 3
3 4
2 125
1 3 1 1
1000 1000
1 2
3 1000
0 0 0 0
```

Saída:

```
70
0
-1
```

5 Entrega

O trabalho deverá ser desenvolvido e entregue em duas parte:

- **Primeira parte**: Esboço da solução - Documentação descrevendo a solução do problema, os algoritmos a serem utilizados, com suas principais características. (O uso de diagramas poderá facilitar a descrição da solução.)
- **Segunda parte**: Desenvolvimento - Código e apresentação da solução proposta.

Cada grupo deverá implementar 5 soluções para o problema, uma usando as abordagem: Força Bruta, Retrocesso, Divisão e Conquista, Algoritmo Guloso e Programação Dinâmica. Deverá ser entregue um relatório técnico com a descrição da sua solução, o código fonte e os arquivos de testes utilizados.

O relatório deve conter uma breve descrição da solução utilizada, explicando a modelagem do problema e a solução em cada técnica de programação. Este arquivo deve ser um PDF.

5.1 Perguntas que devem ser respondidas no relatório

O relatório deve possuir respostas para as seguintes perguntas:

1. Como esse problema pode ser modelado para cada uma das técnicas de projeto de algoritmos?
2. Seu Algoritmo dá a solução ótima? Por quê?
3. Discuta a sub-estrutura ótima e a sobreposição dos problemas.
4. Se algum algoritmo clássico foi adaptado para resolver o problema, qual foi ele?
5. Como sua implementação se comporta em termos de tempo de execução e memória alocada? Apresente uma variação experimental da solução, com entradas de tamanho diferente tentando, pelo menos, discutir também o melhor e o pior caso. Deve ser usado no mínimo 5 arquivos de testes diferentes.

Bom Trabalho!