UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL REI
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
ALGORITMOS E ESTRUTURAS DE DADOS III

1º semestre de 2023
Professor: Leonardo Chaves Dutra da Rocha

Trabalho Prático 1
Data de Entrega: 18 de Setembro 2023. Trabalho Dupla

Introdução

Você foi contratado pelo mago mais famoso do planeta a vencer um jogo, trata-se de Harry Potter! Harry recebe um grid mágico S com R linhas e C colunas contendo a localização de um artefato poderoso, O objetivo do jogo é levar Harry a este artefato sem deixar sua energia cair a 0. Cada célula do grid possui ou uma poção ou um monstro. Um monstro na célula (i,j) retira S[i][j] de energia de Harry, enquanto uma poção na célula (i,j) aumenta a energia de Harry em S[i][j]. Se Harry ficar com 0 de energia ou menos ele morre e Voldemort ficará indestrutível!

A jornada de Harry sempre começa na célula (1,1) e o artefato sempre está na célula (R,C). De uma célula (i,j) Harry pode se movimentar apenas para baixo (i+1, j) ou para direita (i, j+1) e não pode se mover para fora do grid mágico. Sua tarefa é definir qual seria a energia mínima de Harry na célula (1,1) para ele continuar com uma energia positiva ao longo de sua jornada até a célula (R, C).

Assim, seu objetivo nesse trabalho é implementar duas estratégias para solucionar esse problema.

Entrada e Saída

A primeira linha do arquivo de entrada contem um inteiro T indicando o número de casos de testes. Cada caso de teste consiste em R e C na primeira linha seguido pela descrição do grid em R linhas, cada uma contendo C inteiros. As linhas são numeradas de 1 a R de cima para baixo e as colunas são numeradas de 1 a C da esquerda para a direita. Células com S[i][j] < 0 contêm monstros, outras contêm poções mágicas.

Exemplo de entrada:

```
3
2 3
0 1 -3
1 -2 0
2 2
0 1
2 0
3 4
0 -2 -3 1
-1 4 0 -2
1 -2 -3 0
```

O arquivo de saida deve conter T linhas, uma para cada caso contendo a força mínima com a qual o mago deve começar a partir da célula (1, 1) para ter uma força positiva ao longo de sua jornada até a célula (R, C).

Exemplo de saída:

```
2
1
2
```

Seu executável deve chamar tp2 e será chamado da seguinte forma:

```
./tp2 <estrategia> entrada.txt Onde:
```

- <estrategia> é 1 ou 2 para cada uma das suas soluções.

Seu programa deve criar um arquivo **saida.txt** com a resposta, na tela, o programa deve imprimir apenas os tempos de usuário e os tempos de sistema para comparação. Para avaliação do tempo, utilize as funções *getrusage* e *gettimeofday*.

Documentação

Deve ser clara e objetiva, descrevendo as soluções adotadas e justificando bem as escolhas realizadas. Devem possuir também uma análise de complexidade detalhada das soluções. Em termos de análise de resultados, avalie o desempenho e funcionamento de seus algoritmos para diversas configurações e avalie também o tempo de execução dos mesmos (compare-os). Lembre-se, o importante é você apresentar maturidade técnica em suas discussões.

Observações:

- O código fonte do trabalho deve ser submetido para compilação e execução em ambiente Linux, tendo como padrão os computadores dos laboratórios do DCOMP.
- Deve ser escrito na linguagem C (trabalhos implementados em outras linguagens como C++/Java/Python e outras não serão aceitos);
- As estruturas de dados devem ser alocadas dinamicamente e o código deve ser modularizado utilizando os arquivos .c .h.
- O utilitário Make deve ser utilizado para compilar o programa;
- A saída deve ser impressa no arquivo pedido seguindo estritamente o formato da especicação caso contrário o resultado será considerado errado;
- O arquivo executável deve ser chamado de **tp2** e deve receber como parâmetro apenas o nome do arquivo de entrada de dados e a estratégia escolhida pelo usuario. Não serão aceitos outros nomes de executáveis além dos mencionados.
- Faça seu código de forma legível

Avaliação

Deverão ser entregues:

- listagem das rotinas;

- documentação contendo:;
  - descrição das soluções e estruturas de dados utilizadas;
  - análise da complexidade das rotinas;
  - análise dos resultados obtidos.
  - a documentação não pode exceder 12 páginas.

Distribuição dos pontos:

- execução (E)
  execução correta: 80%
- estilo de programação
  código bem estruturado: 10%
  código legível: 10%
- documentação (D)
  comentários explicativos: 40%
  análise de complexidade: 30%
  análise de resultados: 30%

A nota final é calculada como a média harmônica entre execução (E) e documentação (D):

$$\frac{D * E}{\frac{D+E}{2}}$$