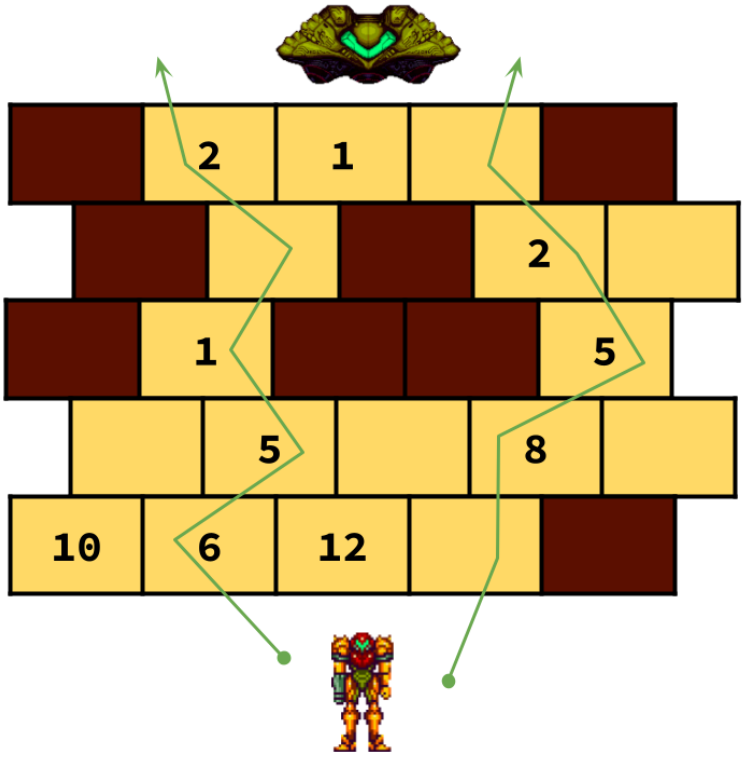
Universidade Federal de Viçosa – Campus UFV-Florestal
Ciência da Computação – Projeto e Análise de Algoritmos
Professor: Daniel Mendes Barbosa

Trabalho Prático 2

Este trabalho é **obrigatoriamente em grupo**. Os grupos já foram definidos nesta planilha e este trabalho deverá ser entregue no PVANet Moodle de acordo com as instruções presentes no final da especificação.

Em outro ponto distante do universo, a exploradora espacial Samus Aran está nas profundezas de uma caverna vulcânica na região de Norfair, do planeta Zebes. O sistema de IA de sua nave está com defeitos e por isso ela trouxe nessa missão uma equipe de estudantes de ciência da computação para auxiliar a interpretar as leituras da nave. Enquanto Samus realizava suas explorações, vocês foram alertados de uma erupção iminente, que encheria de lava todo o complexo de cavernas onde ela se encontra. Felizmente, os radares da nave fornecem para vocês um mapa dos locais por onde Samus pode passar, e onde ela vai encontrar mais perigos.

Assim, vocês precisam implementar, utilizando **programação dinâmica**, um algoritmo capaz de escolher um caminho que permita que Samus saia das cavernas em segurança no menor tempo possível. Algumas áreas da caverna são bloqueadas, e outras estão infestadas de monstros que Samus precisa derrotar se quiser escapar.



O mapa da caverna segue o padrão acima, sendo as áreas em amarelo as partes passáveis, e as áreas em marrom bloqueadas por rochas. As células com números indicam a presença de algum monstro. As setas ilustram caminhos possíveis.

Samus se encontra no fundo da caverna (linha mais abaixo no desenho), e deve sempre se mover para cima, seja para a esquerda ou para direita. Ela pode começar a subir a partir de qualquer uma das células inferiores, e ela pode sair da caverna por qualquer uma das células superiores. Ela não pode sair dos limites laterais da caverna. A cada movimento que Samus realiza, ela gasta t unidades de tempo. Cada vez que Samus encontra um monstro, representado por um número t' no mapa da caverna, ela gasta não apenas o tempo de se mover, mas também o tempo para enfrentar esse monstro (ou seja, $t + t'$). Por fim, a lava vai subindo na caverna nível a nível, ao longo de t_L unidades de tempo, portanto, para saber se Samus vai chegar sã e salva por algum dos caminhos, é preciso saber se o tempo total T gasto para subir a caverna é igual ou inferior ao tempo que a lava gasta para preenchê-la, ou seja, $T \leq t_L \times h$, sendo h a altura da caverna.

Seu programa deverá obrigatoriamente usar **programação dinâmica**.

Importante:
- você deverá definir as estruturas de dados necessárias ao algoritmo;
- na **documentação** você deverá explicar seu algoritmo com base nos conceitos de programação dinâmica, e como ele foi implementado;

Entrada

O espaço geográfico da caverna será a entrada para seu programa, que será fornecido a partir de um arquivo texto, previamente obtido a partir dos radares da nave. O arquivo terá um formato padronizado, sendo que na primeira linha do arquivo temos, separados por espaços: a altura h da caverna, a largura w de cada nível da caverna, o tempo t que Samus gasta para realizar cada movimento, e o tempo que a lava gasta para subir cada nível da caverna, t_L.

Nas linhas seguintes será informado o conteúdo de cada célula do mapa da caverna, sendo cada célula composta sempre de 3 caracteres, separados por espaços. Se for um espaço vazio, será denotado por zeros (`000`). Se for uma célula bloqueada por rochas, será denotada por *hashtags* (`###`). Se for um espaço com algum monstro, será denotado por um número entre `001` e `999`.

O exemplo abaixo representa a caverna da figura anterior. Como indicado pela primeira linha, a caverna tem 5 níveis de altura, e cada nível tem 5 células de largura. Lembre-se que esses números podem ser diferentes entre si. A primeira linha também indica que Samus gasta 4 unidades de tempo para realizar cada movimento, e a lava demora 7 unidades de tempo para subir cada nível.

`caverna1.txt`:

```
5 5 4 7
### 002 001 000 ###
  ### 000 ### 002 000
### 001 ### ### 005
  000 004 000 008 000
010 006 012 000 ###
```

Perceba que as linhas alternam seu alinhamento. Considere que a linha mais superior da caverna sempre vai começar alinhada à esquerda, a segunda à direita, e assim sucessivamente.

Saída

Os resultados do algoritmo devem ser, no mínimo, apresentados na saída padrão de acordo com as seguintes especificações. Cada linha representa um movimento de Samus, com as coordenadas (i, j) - linha e coluna - da caverna. A linha superior é a linha 0, e a linha mais inferior, de onde Samus começa, no fundo da caverna, é a linha $h - 1$. A célula mais à esquerda de cada nível é a célula 0, e a mais à direita é a célula $w - 1$. Vocês devem escrever na saída somente o passo a passo que Samus precisa fazer para chegar até a superfície pelo **melhor caminho possível.** Caso seja impossível escapar da caverna a tempo, deve ser escrito `"Samus falhou na missão"`, sem a necessidade de mostrar nenhuma outra informação.

Segue a solução para o exemplo de entrada:

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

Interface

O programa não precisa ter interface. É suficiente que ele receba por linha de comando o caminho do arquivo contendo o "mapa" da caverna, mostrando em seguida os resultados na saída padrão. Considere que os arquivos de entrada seguirão fielmente o formato que foi definido.

Tarefas extras

Tarefas que podem ser feitas além do básico (que podem servir inclusive como diferencial no ranqueamento das notas dos trabalhos):

1. Explicar e usar algum algoritmo para detectar células livres, porém inalcançáveis, antes da aplicação da programação dinâmica.
2. Plotar gráficos de desempenho de tempo do algoritmo para diferentes tamanhos de entrada, definidos pelo grupo.
3. Melhorar a interface do programa, mostrando de forma gráfica o melhor caminho.
4. Criar uma outra opção inventada pela dupla, com a devida especificação, implementação e resultados mostrados na documentação.
5. Criar um programa à parte para geração de arquivos de entrada, de preferência com alguns parâmetros para orientar a geração.

Faça exatamente o que está sendo pedido neste trabalho, ou seja, mesmo que o grupo tenha uma ideia mais interessante para o programa, deverá ser implementado exatamente o que está definido aqui no que diz respeito ao problema em si e ao paradigma programação dinâmica. No entanto, o grupo pode implementar algo além disso, desde que não atrapalhe a obtenção dos resultados necessários a esta especificação.

Formato e data de entrega

Os arquivos com o código-fonte (projeto inteiro do Codeblocks ou arquivos .c, .h e makefile), juntamente com um arquivo PDF (**testado, para ver se não está corrompido**) contendo a **documentação**. A documentação deverá conter:

- explicação do algoritmo projetado;
- implementação do algoritmo projetado (estruturas de dados criadas, etc);
- resultados de execução, mostrando entrada e saída;
- arquivos de entrada usados nos testes;
- explicação de como compilar o programa.

Mais direcionamentos sobre o formato da documentação podem ser vistos no documento "Diretrizes para relatórios de documentação".

Importante: Entregar no formato **ZIP**. As datas de entrega estarão configuradas no PVANet Moodle. É necessário que apenas um aluno do grupo faça a entrega, mas o PDF da documentação deve conter os nomes e números de matrícula de todos os alunos em sua capa ou cabeçalho.

Bom trabalho!