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

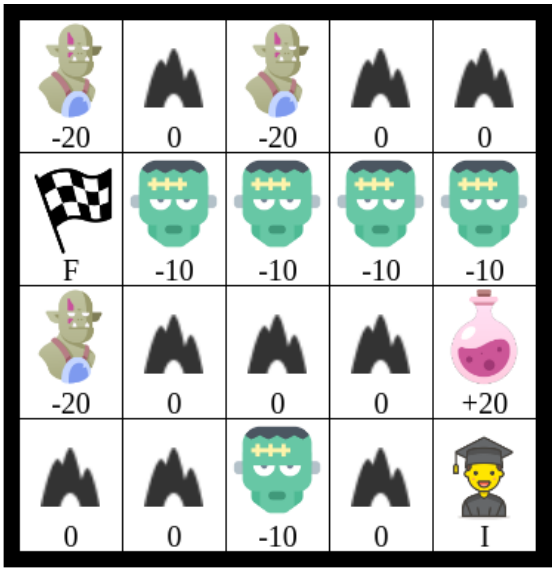
Trabalho Prático 2

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.

Como vocês já sabem, no trabalho 1 os estudantes escapavam de labirintos. Porém, após sair do labirinto, os estudantes encontraram cavernas parecidas com as do famoso jogo *Dungeons and Dragons*. Essas cavernas tinham monstros que atacavam os humanos desavisados e, após receber uma quantidade de ataques que zeraram seus Pontos de Vida, sucubiam e voltavam à prisão.

Portanto, para continuar a ser possível a escapada dos aprisionados, você precisa implementar utilizando **programação dinâmica** um algoritmo capaz de escolher um caminho que permita a saída do labirinto e ao mesmo tempo continuar com o máximo de Pontos de Vida para continuar a aventura de volta à Terra. E, para piorar sua vida, agora os alienígenas colocaram um portão em uma posição específica, ou seja, o objetivo não é chegar em qualquer posição no topo como antes. Uma boa notícia é que não há mais paredes, você pode passar por qualquer lugar e pode começar sua trajetória de qualquer ponto nas bordas da caverna.

Como anteriormente, a caverna pode ser fotografada utilizando um drone, mas agora temos algumas células com monstros e outras células com poções que permitem recuperar Pontos de Vida.



Você só pode se movimentar para cima ou para esquerda. A saída é representada por uma bandeira quadriculada em preto e branco e a célula marcada com F (de fim), os monstros e as poções são itens em cada célula. A posição inicial é representada pelo estudante com a posição marcada por I. Além disso, cada célula está marcada com um valor, se for negativo o estudante perderá Pontos de Vida ao passar por ali. Analogamente, se for positivo o estudante irá receber Pontos de Vida. Um detalhe importante: se você ficar sem Pontos de Vida (pontos <= 0) você **não conseguirá sair do labirinto**. A delimitação das cavernas são retângulos pretos, o estudante não pode passar por ali.

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;

Formato de entrada de dados:

O espaço geográfico da caverna será a entrada para seu programa, que será fornecido a partir de um arquivo texto, previamente obtido a partir da foto tirada pelo drone. O arquivo terá um formato padronizado, sendo que na primeira linha do arquivo temos o número de linhas, um espaço, um número de colunas, um espaço e o número de pontos de vida iniciais que o estudante possui no início.
Nas linhas seguintes será informado a diferença nos pontos de vida infligida ao aluno caso ele passe por ali. A separação entre as células é feita por espaço com um número simbolizando a alteração nos Pontos de Vida e a saída é representada por F.

O exemplo abaixo representa a caverna da figura anterior.

caverna1.txt

```
4 5 40
-20 0 -20 0 0
F -10 -10 -10 -10
-20 0 0 0 +20
0 0 -10 0 I
```

Este exemplo possui 4 linhas com 5 colunas cada e considera que o estudante está com 40 pontos de vida. E como no trabalho anterior, esses valores podem ser valores quaisquer, e o espaço geográfico do labirinto pode ser retangular (número de linhas diferente do número de colunas).

Formato de saída de dados:

A saída dos dados deve ser gravada em um arquivo: “resultado.txt”. Cada linha representa um movimento do estudante. Sendo a primeira coluna de cada linha a coordenada x (linha) da caverna e a segunda coluna a coordenada y (coluna). As coordenadas devem começar do zero, por exemplo, a primeira posição na extrema esquerda e cima é 0 0. Você somente deve escrever no arquivo o passo a passo que o aluno precisa fazer para chegar de I até F pelo **melhor caminho possível.** Caso o problema tenha solução, a primeira linha do arquivo deve ser a posição inicial do aluno. Caso seja impossível escapar do labirinto com vida, o aluno deve imprimir a string “impossível” no arquivo “resultado.txt” e nada mais deve ser escrito.

Solução exemplo para a caverna1:

resultado.txt

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

Aparência geral do programa:

O programa não deverá ter interface, ele deverá receber por linha de comando o caminho do arquivo contendo o “mapa” da caverna e gravar os passos em “resultado.txt”.

Considere que os arquivos de entrada seguirão fielmente o formato que foi definido.

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

1. Utilização de heurísticas para acelerar a busca. Sua solução será comparada a de outros alunos em tempo de processamento e memória. Então você é livre para utilizar estruturas de dados eficientes e heurísticas para acelerar a busca pelo melhor caminho. Lembrando que só serão aceitas soluções para o caminho que maximize os pontos de vida. Caso sua solução seja mais rápida mas não seja a melhor, ela será desconsiderada. Por isso é importante que seu programa tenha instruções para compilação.

2. Explicar como seria possível permitir todas as movimentações (norte, sul, leste e oeste) usando programação dinâmica.

3. Criar uma outra opção inventada pelo grupo, com a devida especificação, implementação e resultados mostrados na documentação.

4. Criar um programa à parte para geração de arquivos de entrada, de preferência com alguns parâmetros para orientar a geração.

5. Plotar um gráfico de complexidade x tempo. Basicamente, vocês devem criar a curva de complexidade do algoritmo criado, testando o mesmo para diferentes tamanhos de caverna. Considerem o eixo x como o tamanho da entrada e o eixo y como o tempo total gasto para cada entrada. Vocês podem usar qualquer software para a plotagem, mas devem colocá-lo nas referências do relatório.

Faça exatamente o que está sendo pedido neste trabalho, ou seja, mesmo que você tenha uma ideia mais interessante para o programa, você deverá implementar exatamente o que está definido aqui no que diz respeito ao problema em si e ao paradigma programação dinâmica. No entanto, você 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 IDE 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", também disponível no PVANet Moodle.

Importante: no PVANet Moodle você fará a entrega de dois arquivos: um para a documentação e um para o zip com o projeto todo. 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 em sua capa ou cabeçalho os nomes e números de matrícula de todos os alunos do grupo que efetivamente participaram do trabalho. **Assim como nos trabalhos anteriores, vocês devem utilizar a linguagem de programação C.**

Bom trabalho!