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

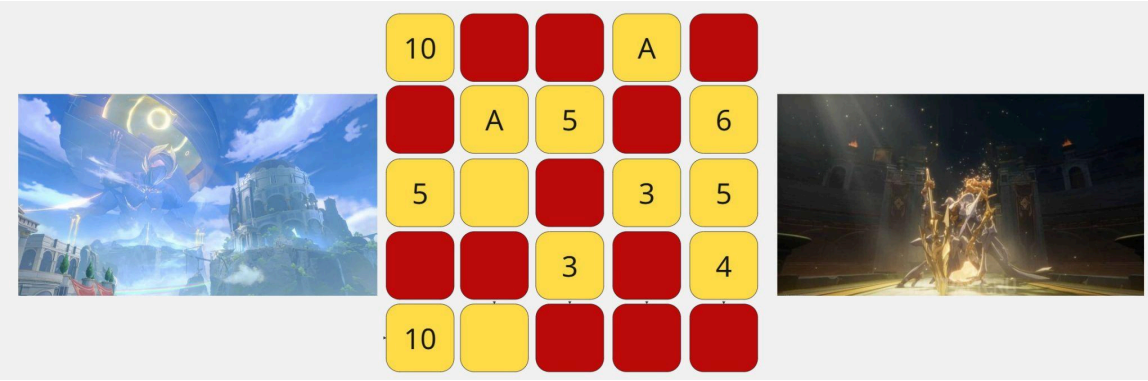
Trabalho Prático 2

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

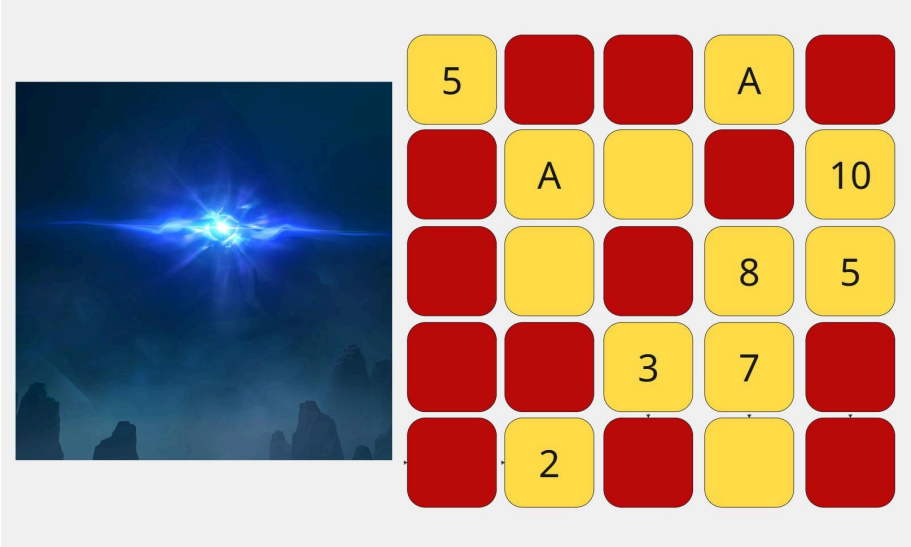
Após a última aventura rumo ao planeta das festividades, a tripulação do expresso interestelar está agora em terra no planeta Amphoreus e precisa novamente da ajuda dos estudantes de ciências da computação. Dessa vez, a tripulação está tentando ajudar a salvar o planeta de uma calamidade iminente e para isso precisarão se juntar aos locais e derrotar um poderoso inimigo, Nikador, o titã do conflito. Sua jornada é guiada por Kephale, o titã sustenta mundos o qual proporcionou um mapa para que possam chegar até Nikador, contudo existem âncoras temporais ao longo do caminho que abrem espaço para viagens temporais para pontos do passado. Kephale detém apenas os mapas do presente, mas felizmente Oronyx o titã do tempo concordou em fornecer os mapas do passado para o qual as âncoras temporais levam. O caminho está repleto de perigos e é necessário chegar com forças sobrando para enfrentar o titã do conflito.

Assim, vocês precisam implementar, utilizando **programação dinâmica**, um algoritmo capaz de escolher um caminho que permita que a tripulação chegue para a batalha com o menor desgaste possível. Algumas áreas do caminho são intransponíveis, e outras estão infestadas de monstros que precisam ser derrotados se quiserem chegar à Nikador.

MAPA CONCEDIDO POR KEPHALE



MAPA CONCEDIDO POR ORONYX



O mapa do trajeto segue o padrão acima, sendo áreas em amarelo as partes passáveis, e as áreas em vermelho intransponíveis. As células com números indicam a presença de algum inimigo. Vale lembrar que a disposição dos mapas em tamanho e largura é a mesma.

A aventura começa na pólis sagrada protegida por Kephale chamada Okhema (coluna mais à esquerda), e deve sempre se mover para direita, seja para baixo, para cima ou para frente. O percurso pode começar a partir de qualquer uma das células mais próximas da pólis sagrada, e ela pode sair da para enfrentar Nikador por qualquer uma das células mais próximas a ele.O mapa representa estritamente a passagem possível, não sendo possível passar dos limites definidos. A cada movimento realizado,a tripulação ou tem que lutar ou tem espaço para um breve descanso. Cada vez que encontram um inimigo, representado por um número **“X”** no mapa, as forças **“F”** da tripulação são gastas em um valor igual a **“X”**. Cada região sem inimigos permite um descanso que recupera as forças da tripulação em um valor **“D”**. Além disso, existem âncoras temporais **“A”** nos mapas que transportam para o outro tempo (se está no presente vai para o passado e vice versa) e quando se chega em uma delas tem-se uma relação na qual as âncoras estão sempre nas mesmas posições em ambos os mapas o que facilita a localização quando ocorre as trocas, quando se usa uma âncora não há perda ou ganho de forças. Por fim, Zagreus, o titã da trapaça, conseguiu informações sobre a força **“N”** de Nikador, dessa forma a tripulação pode saber como se preparar para a batalha no final do caminho, contudo o objetivo principal é saber se conseguirão chegar até Nikador com forças para enfrentá-lo ou se o caminho será uma provação maior do que esperavam.

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

Os mapas do caminho no presente e no passado serão a entrada para seu programa, que será fornecido a partir de um arquivo texto, previamente recebido de Kephale e Orinyx. 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 valor inicial de força **"F"** da tripulação,o valor **"D"** de força recuperada no descanso e o valor **"N"** da força de Nikador

Nas linhas seguintes será informado o conteúdo de cada célula do mapa do presente seguido de uma linha com "///" e em seguida o mapa do passado, sendo cada célula dos mapas composta sempre de 3 caracteres, separados por espaços. Se for um espaço vazio, será denotado por zeros (`000`). Se for uma célula de espaço intransponível, será denotada por *hashtags* (`***`). Se for um espaço com algum monstro, será denotado por um número entre `001` e `999`. Se for uma Âncora temporal será `AAA`

O exemplo abaixo representa a entrada para os mapas anteriores. Como indicado pela primeira linha, a região 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 a força inicial da tripulação é igual a 80, sendo que recuperam 2 a cada descanso e a força de Nikador é igual a 50.

```
5 5 80 2 50
010 *** *** AAA ***
*** AAA 005 *** 006
005 000 *** 003 005
*** *** 003 *** 004
010 000 *** *** ***
///
005 *** *** AAA ***
*** AAA 000 *** 010
*** 000 *** 008 005
*** *** 003 007 ***
*** 002 *** 000 ***
```

Saída

Os resultados do algoritmo devem ser, no mínimo, apresentados na saída padrão de acordo com as seguintes especificações. Cada linha representa um movimento da tripulação, com as coordenadas (i, j) - linha e coluna - do trajeto. A linha superior é a

linha 0, e a linha mais inferior 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 a tripulação precisa fazer para chegar até seu destino pelo **melhor caminho possível, ou seja com a maior quantidade de força possível.** Caso em algum ponto do trajeto as forças acabem e não houver caminho possível com algum nível de força sobrando significa que não é possível chegar ao destino,e, deve ser escrito `"A calamidade de Nikador é inevitável"`, sem a necessidade de mostrar nenhuma outra informação. Caso o trajeto seja possível é necessário mostrar ao final o destino que os aguarda, se F >= N deve-se exibir ao final `"A ruína de Nikador é iminente"`, caso F<N deve-se exibir ao final `"Será necessário mais planejamento para parar a calamidade"`

Segue a solução para o exemplo de entrada:

```
2 0
1 1
1 2
0 3
1 4
A ruína de Nikador é iminente
```

Para este caso exemplo como ao longo do caminho a força da tripulação na melhor rota diminui em 9 a força final é 71 logo maior que o valor de N=50 por isso a mensagem final, caso o valor de N fosse maior do que 71 a mensagem exibida seria “Será necessário mais planejamento para parar a calamidade” e caso o valor de F chegasse a zero durante o processo seria exibido apenas a mensagem “A calamidade de Nikador é inevitável” sem a necessidade de mostrar o caminho percorrido.

Interface

O programa não precisa ter interface. É suficiente que ele receba por linha de comando o caminho do arquivo contendo os “mapas” do trajeto, 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. Plotar gráficos de desempenho de tempo do algoritmo para diferentes tamanhos de entrada, definidos pelo grupo.
2. Melhorar a interface do programa, mostrando de forma gráfica o melhor caminho.

3. Criar uma outra opção , com a devida especificação, implementação e resultados mostrados na documentação. (ex: trabalhar com mapas de tamanhos variados, âncoras temporais em lugares distintos nos mapas)

4. 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 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!