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

Trabalho Prático 1

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.

Fernando é um renomado jovem fazendeiro que possui uma plantação de batata. A fazenda de Fernando é retangular e possui diversos campos de 1x1 usados para o cultivo desse tipo de legume. Com objetivo de obter o maior lucro, Fernando contratou um engenheiro agrônomo que possui em sua equipe diversos programadores especialistas em otimizar rotas de colheitas. A equipe do engenheiro decidiu que a rota ótima para a fazenda de Fernando ser a mais produtiva é seguir uma sequência de fibonacci, ou seja, as batatas só serão colhidas se estiverem em uma rota que segue essa sequência. Então, o engenheiro numerou todos os campos da fazenda e pediu aos programadores encontrarem a melhor rota para que Fernando tenha o maior lucro possível. Observe um exemplo da fazenda mapeada:

1	2	3	8	1	9	1	2	1	6	1
5	2	3	5	1	1	1	1	2	1	2
1	8	13	1	1	3	2	1	6	4	7
1	1	2	1	1	2	3	5	1	1	2
5	3	8	5	3	2	1	1	8	5	3
1	1	13	1	1	5	6	7	9	10	0
1	8	13	1	2	3	5	8	-1	−2	3
6	2	3	4	4	7	9	13	-1	2	1
7	8	9	1	2	3	14	21	1	4	1
1	2	7	6	5	1	1	1	2	1	1

Sabemos que a sequência de fibonacci para um número n, recursivamente, é dado por $fib(n-1) + fib(n-2)$. Sabemos também que é possível determinar fibonacci para um número n pela fórmula fechada $fib(n) = \left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n$. No caso da rota

ótima para a colheita mais lucrativa, a sequência é composta por elementos da sequência de fibonacci como abaixo:

1

1 1

1 1 2

1 1 2 3

1 1 2 3 5

O primeiro campo de batata que será colhido deve ser numerado com 1, o segundo, terceiro, quarto e quinto campo também deve ter o número igual a 1. O sexto campo deve ser 2, formando assim, a sequência de fibonacci como acima: (1, 1, 1, 1, 1, 2, 1, 1, 2, 3, 1, 1, 2, 3, 5)

É garantido que exista uma ou nenhuma rota ótima para o problema de Fernando. Também sabe-se que as colheitadeiras não conseguem se movimentar pelas diagonais.

Você deve, portanto, projetar um algoritmo com **backtracking** para encontrar um caminho, neste contexto, implementá-lo em C e documentá-lo, conforme as especificações dadas.

Entrada

As dimensões da fazenda N, M tal que $1 \leq N \leq 100$, $1 \leq M \leq 100$. Logo após, será dado N linhas contendo M números que correspondem às numerações dos campos.

Segue um exemplo de entrada, com um mapa correspondente à figura inicial:

```
10 11
1 2 3 8 1 9 1 2 1 6 1
5 2 3 5 1 1 1 1 2 1 2
1 8 13 1 1 3 2 1 6 4 7
1 1 2 1 1 2 3 5 1 1 2
5 3 8 5 3 2 1 1 8 5 3
1 1 13 1 1 5 6 7 9 10 0
1 8 13 1 2 3 5 8 -1 -2 3
6 2 3 4 4 7 9 13 -1 2 1
7 8 9 1 2 3 14 21 1 4 1
1 2 7 6 5 1 1 1 2 1 1
```

Saída

O programa deverá calcular o caminho que obedece à sequência de fibonacci conforme descrito anteriormente e imprimir a rota ótima completa na tela. A saída esperada para o exemplo dado acima (marcado de cinza no mapa da fazenda) é:

```
1 5
2 5
2 6
2 7
1 7
1 8
2 8
3 8
3 7
3 6
3 5
4 5
4 6
4 7
4 8
4 9
4 10
4 11
5 11
5 10
5 9
5 8
5 7
5 6
5 5
5 4
5 3
6 3
6 4
7 4
7 5
7 6
7 7
7 8
8 8
```

```
9 8
10 8
```

Pode haver casos em que não existe um caminho ótimo e o programa deve relatar esse resultado:

```
IMPOSSÍVEL!
```

Obs.: o grupo é livre para definir outras formas mais interessantes de exibir essa saída, desde que essas informações mínimas estejam presentes.

Backtracking

Seu programa deverá obrigatoriamente usar **backtracking**. Uma função recursiva chamada `movimentar` (ou de nome similar) deverá ser criada.

Na documentação, o grupo deverá explicar seu algoritmo com base nos conceitos de backtracking, e como ele foi implementado, explicando as estruturas de dados necessárias. Fernando possui vários amigos fazendeiros que possuem interesse na sua solução, com diferentes áreas de cultivo, em outras palavras, as dimensões das fazendas dos amigos de Fernando diferem entre si. Logo, só se sabe o tamanho das estruturas depois de ler os arquivos de entrada, portanto deverá ser usada a **alocação dinâmica de memória**.

O grupo deve definir a possibilidade do programa ser executado em um "modo de análise", para verificar o funcionamento do algoritmo. Nesse modo, seu programa contabiliza quantas chamadas recursivas foram feitas, e qual foi o nível máximo de recursão alcançado. O modo de análise pode ser feito em tempo de compilação (por exemplo, através do uso de `#define`), ou em tempo de execução (alguma opção dentro da interface do programa que permita ligar e desligar esse modo). O importante é que o grupo escolha e documente como isso foi feito.

Quando o modo de análise estiver ligado, essas informações sobre a recursão devem constar na saída final do programa, independente se houve uma solução ou não.

Interface

O grupo é livre para definir os detalhes da interface do programa, desde que ela atenda aos seguintes requisitos:

- O programa deve aceitar arquivos de texto como entrada, segundo a formatação especificada.
- O programa deve ser capaz de aceitar novos arquivos de entrada depois de cada execução, encerrando apenas quando o usuário desejar.

Tarefas extras

1. A interface e como exibir o resultado fica a critério do grupo. Portanto, poderá ser oferecida mais de uma forma de se exibir os resultados, podendo o usuário escolher qual formato deseja.
2. Criar uma opção (ou até mesmo um outro programa) para a geração de arquivos de teste, considerando todos os dados envolvidos e o formato, como descrito acima. Os mapas não necessariamente possuem solução. Seu programa de geração de arquivos de teste deverá ter alguns parâmetros de configuração, como as dimensões da fazenda, entre outros, que vocês poderão colocar e especificar na documentação.

Faça <u>exatamente</u> o que está sendo pedido neste trabalho, ou seja, mesmo que você tenha uma ideia mais interessante para o programa, você deverá implementar <u>exatamente</u> o que está definido aqui no que diz respeito ao problema em si e ao paradigma backtracking. 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 Codeblocks ou arquivos .c, .h e makefile), 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 em modo normal e modo análise.
- nos resultados deverá constar a quantidade total de chamadas recursivas e o nível máximo de recursão obtido para cada teste através da execução no modo análise.

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

<u>Importante</u>: 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 do grupo que **<u>efetivamente</u>** participaram do trabalho em sua capa ou cabeçalho.

Bom trabalho!