TP 03 - Trabalho Prático 03

Algoritmos I

Entrega: 10/02/2022

1 Objetivos do trabalho

O objetivo deste trabalho é modelar o problema computacional descrito a seguir utilizando uma estrutura de dados que permita resolvê-lo de forma eficiente com os algoritmos estudados nesta disciplina.

Serão fornecidos alguns casos de teste bem como a resposta esperada para que o aluno possa verificar a corretude de seu algoritmo. Não obstante, recomenda-se que o aluno crie casos de teste adicionais a fim de validar sua própria implementação.

O código-fonte da solução e uma documentação sucinta (relatório contendo não mais do que 5 páginas) deverão ser submetidos via *moodle* até a data limite de 10/02/2022. A especificação do conteúdo do relatório e linguagens de programação aceitas serão detalhadas nas seções subsequentes.

2 Definição do problema

O novo ano já iniciou e com isso os trabalhos planejados devem começar a serem realizados. Para a indústria de colheita frutífera, o planejamento sobre quando iniciar os trabalhos e de qual forma estes serão executados é de extrema importância para que a performance da colheita seja otimizada.

Em Fevereiro é iniciado o período da colheita de maçãs e, com isso, um grande produtor do estado de Santa Catarina decidiu implementar uma nova estratégia para otimizar o trabalho das colheitadeiras. É sabido que cada macieira produz uma quantidade Q ($0 <= Q <= 10^9$) diferente da fruta e, por limitação de tempo de colheita e disponibilidade de equipamento, nem todas as árvores terão seus frutos colhidos. A máquina colheitadeira possui limitação de movimentos e a realização de manobras pode demandar um tempo elevado, diminuindo assim sua performance na colheita. Portanto, garantir que o percurso da colheitadeira seja otimizado e que a quantidade de frutas colhidas a cada viagem da colheitadeira seja a maior possível pode resultar em uma maximização do volume final colhido pelo produtor.

Seguindo um estudo realizado anteriormente, o produtor realiza a plantação das macieiras de forma alinhada, criando assim F ($1 <= F <= 10^6$) fileiras com a mesma quantidade W ($1 <= W <= 50$) de árvores. Para superar a limitação de movimentação da colheitadeira, foi definido que esta deverá iniciar a colheita sempre pela primeira fileira (a primeira fileira é a mais acima do campo de plantação, ou seja, a primeira linha lida no arquivo de entrada) e mover-se para a fileira seguinte selecionando entre três opções: a macieira com mesmo alinhamento da atual, a macieira diagonalmente a direita da atual ou a macieira que se encontra diagonalmente a esquerda da atual. A colheitadeira nunca se movimenta para trás ou para os lados, ou seja, nunca para uma macieira da fileira anterior ou para uma macieira na mesma fila da atual.

O seu trabalho é definir qual o maior número de maçãs que pode ser colhido com uma viagem da colheitadeira obedecendo as restrições de movimentação e definir o mapa do caminho que deve ser seguido para obter esta quantidade de fruta.

3 O que fazer?

O objetivo deste trabalho é definir a quantidade máxima de maçãs que uma colheitadeira poderá colher a cada viagem dada uma disponibilidade de frutas em cada árvore e as limitações de movimentação pré-definidas por um especialista da área. Também deve ser informado uma a sequência de F inteiros, onde cada um indica qual a posição da macieira a ser colhida em cada fileira F_i.

Este algoritmo deverá:

- Ler os números F e W correspondentes, respectivamente, ao número de fileiras de macieiras e a quantidade de macieiras em cada fileira;
- Ler a disponibilidade Q_i de frutas em cada macieira i;
- Definir o início da colheita por qualquer árvore na primeira fileira de macieiras;
- Respeitar as restrições de movimentação da colheitadeira, sempre movimentando-se para uma macieira na próxima fileira e que seja alinhada a macieira atual ou diagonalmente seguinte;
- Encontrar a quantidade máxima da fruta que pode ser colhida pela colheitadeira com uma viagem, obedecendo as regras de movimentação.
- Imprimir na tela uma sequência de inteiros informando qual é o caminho para se obter a colheita máxima.

4 Arquivos de entrada

O problema terá como entrada um arquivo contendo as informações disponibilizadas. A primeira linha conterá o número F de fileiras de macieiras e a quantidade de macieiras W em cada fileira, separados por espaço. As próximas F linhas irão conter W números separados por espaço, indicando a quantidade $Q(0 <= Q <= 10^9)$ disponível da fruta em cada macieira.

5 Saída

A saída deve ser impressa na tela (*stdout*) e seguir o seguinte padrão: Na primeira linha deverá ser impresso um número R indicando a quantidade máxima possível que pode ser colhida em uma única rota e, na linha seguinte, uma sequência de F inteiros, onde cada número na posição i indica qual é o índice da macieira na fileira F_i, detalhando a rota a ser seguida pela colheitadeira. Os índices de cada fileira e de cada macieira são definidos pela ordem de leitura do arquivo de entrada, iniciando por 0 (zero).

Fique atento para imprimir exatamente como foi descrito pois a correção do algoritmo será feita de forma automática.

6 Exemplo do problema

A seguir é apresentado um exemplo:

Dados de Entrada

```
6 5
3 1 7 4 2
2 1 3 1 1
1 2 2 1 8
2 2 1 5 3
2 1 4 4 4
5 2 7 5 1
```

A imagem abaixo ilustra a leitura deste exemplo após realizada a leitura das quantidades.

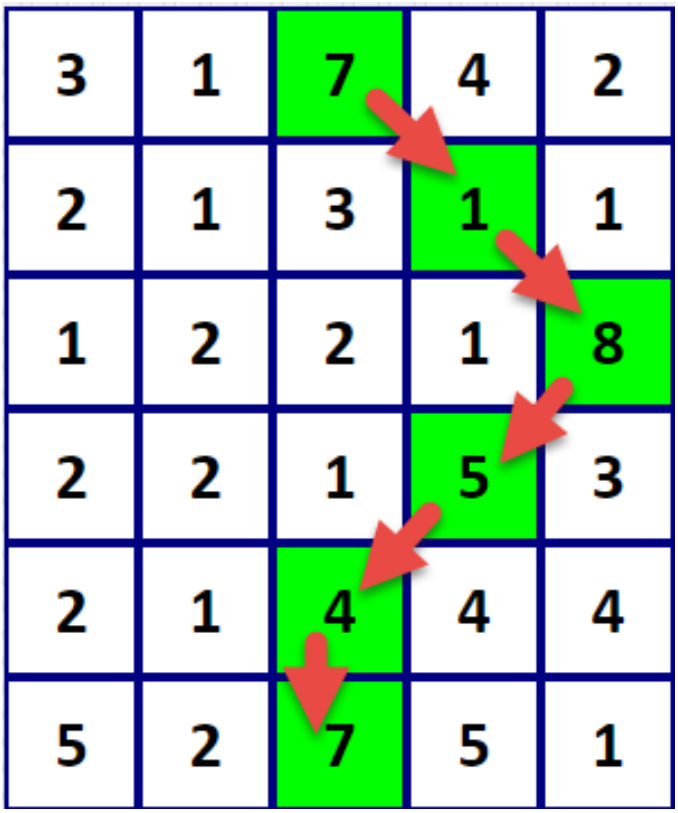

Figura 1: Exemplo de caminho com a máxima quantidade

Pode existir vários caminhos que representam o caminho máximo, como este segundo exemplo

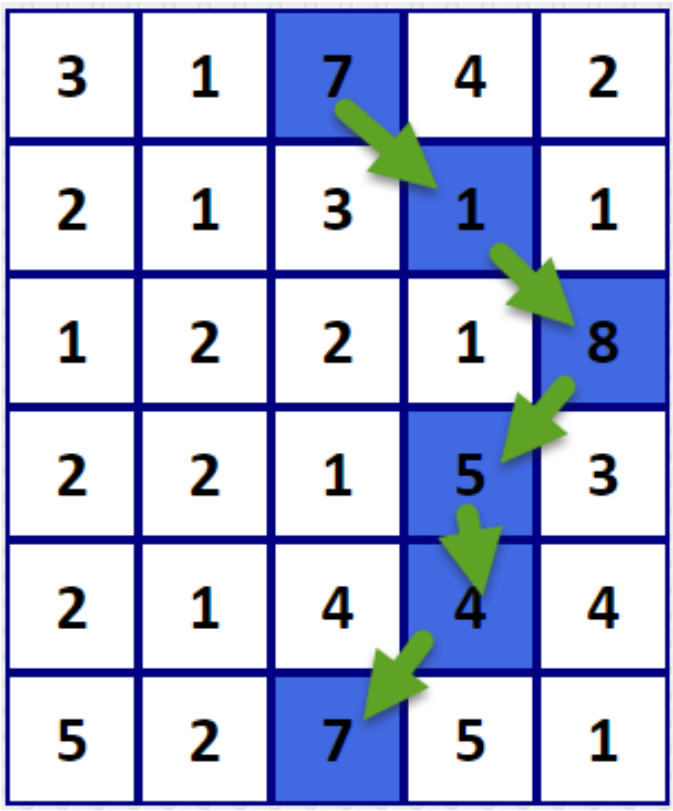

Figura 2: Exemplo de outro caminho com a máxima quantidade

No primeiro caminho, a soma total seria $7 + 1 + 8 + 5 + 4 + 7 = 32$ com índices:
$7 \rightarrow$ índice 2
$1 \rightarrow$ índice 3
$8 \rightarrow$ índice 4
$5 \rightarrow$ índice 3
$4 \rightarrow$ índice 2
$7 \rightarrow$ índice 2
No segundo caminho, a soma total seria $7 + 1 + 8 + 5 + 4 + 7 = 32$ $7 \rightarrow$ índice 2
$1 \rightarrow$ índice 3
$8 \rightarrow$ índice 4
$5 \rightarrow$ índice 3
$4 \rightarrow$ índice 3
$7 \rightarrow$ índice 2
Portanto, um possível resultado final é:

Dados de Saída

```
32
2 3 4 3 2 2
```

7 Exemplo Prático de Entrada e Saída

Arquivo de entrada

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

Exemplo prático da saída

```
69
6 7 7 8 8 7 6 7 6 7
```

---

8 Especificação das entregas

Você deve submeter um arquivo compacto (zip) no formato **MATRICULA_NOME** via Moodle contendo:

- todos os arquivos de código-fonte implementados;
- um arquivo *makefile*[1] **que crie um executável com nome tp03**;
  - **ATENÇÃO:** O makefile é para garantir que o código será compilado da forma como vocês implementaram, evitando erros na compilação. É **essencial** que ao digitar "make" na linha de comando dentro da pasta onde reside o arquivo makefile, o mesmo compile o programa e gere um executável chamado **tp03**.
- sua documentação (arquivo pdf).

Sua documentação deverá ser sucinta e conter não mais do que 5 páginas com o seguinte conteúdo obrigatório:

- Modelagem computacional do problema;
- estruturas de dados e algoritmos utilizados para resolver o problema (pseudo-código da solução impelementada), bem como justificativa para tal escolha. Não transcreva trechos da código-fonte;
- análise de complexidade de tempo assintótica da solução proposta, devidamente justificada.

[1] https://pt.wikibooks.org/wiki/Programar_em_C/Makefiles

9 Implementação

9.1 Linguagem, Ambiente e Parâmetros

O seu programa deverá ser implementado na linguagem **C** ou **C++** e deverá fazer uso apenas de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação e/ou que façam uso de bibliotecas que não a padrão não serão aceitos.

O aluno pode implementar seu programa em qualquer ambiente (Windows, Linux, MacOS, etc...), no entanto, deve garantir que seu código compile e rode nas máquinas do DCC (tigre.dcc.ufmg.br ou jaguar.dcc.ufmg.br), pois será neste ambiente que o TP será corrigido. Note que essas máquinas são acessíveis a todos os alunos do DCC com seu login e senha, podendo inclusive ser realizado acesso remoto via ssh. O aluno pode buscar informações no site do CRC (Centro de Recursos Computacionais) do DCC (https://www.crc.dcc.ufmg.br/).

O arquivo da entrada deve ser passado ao seu programa como entrada padrão, através da linha de comando (e.g., *$ ./tp03 < casoTeste01.txt*) e gerar o resultado também na saída padrão (não gerar saída em arquivo).

ATENÇÃO: Não é necessário que o aluno implemente em ambiente Linux. Recomenda-se que o aluno teste seu código nas máquinas previamente especificadas, as quais serão utilizadas para correção do TP, a fim de conferir a funcionalidade, makefile e demais características do código.

ATENÇÃO: Não é permitido usar flags de otimização, para aumentar a memória ou similar, a compilação deve ser simplesmente um *gcc/g++ (input_files) -o tp03*

ATENÇÃO: Não é permitido soluções que utilizem força bruta para obter o resultado. Você deve modelar e implementar um algoritmo utilizando as técnicas aprendidas em sala de aula

9.2 Testes automatizados

A sua implementação passará por um processo de correção automatizado, utilizando além dos casos de testes já disponibilizados, outros exclusivos criados para o processo de correção. O formato da saída de seu programa deve seguir a especificação apresentada nas seções anteriores. Saídas diferentes serão consideradas erro para o programa. O aluno deve certificar-se que seu programa execute corretamente para qualquer entrada válida do problema.

ATENÇÃO: O tempo máximo esperado para execução do programa, dado o tamanho máximo do problema definido em seções anteriores, é de 5 segundos.

ATENÇÃO: O executável produzido pelo comando make deve ser chamado "tp03", e não "tp03.out, tp03.exe, tp03.o"e deve estar localizado no diretório atual (.).

ATENÇÃO: O arquivo zip a ser enviado deve seguir exatamente a seguinte estrutura:

```
XXXXXXXXX_NOME.zip ---------> O nome de arquivo tem que ser ZIP e ter a extensao ".zip"
|------ Makefile
|------ File1
|------ File2
```

```
|------- File3
|------- Folder1
|------- Folder2
```

Certifique-se de que os comandos abaixo sejam aceitos corretamente por seu entregável

```
unzip XXXX_NAME.zip
make
./tp03 < file_input_name
```

*Não é aceito sem < (./tp03 file_input_name)

ATENÇÃO: O tempo máximo esperado para execução do programa, dado o tamanho máximo do problema definido em seções anteriores, é de 5 segundos.

9.3 Qualidade do código

Preze pela qualidade do código-fonte, mantendo-o organizado e comentado de modo a facilitar seu entendimento para correção. Caso alguma questão não esteja clara na documentação e no código fonte, a nota do trabalho pode ser penalizada.

10 Critérios para pontuação

A nota final do TP (NF) será composta por dois fatores: fator parcial de implementação (fpi) e fator parcial da documentação (npd). Os critérios adotados para pontuação dos fatores é explicado a seguir.

10.1 Fator parcial de implementação

Serão avaliados quatro aspectos da implementação da solução do problema, conforme a Tabela 1.

Aspecto	**Sigla**	**Valores possíveis**
Compilação no ambiente de correção	co	0 ou 1
Respostas corretas nos casos de teste de correção	ec	0 a 100%
Tempo de execução abaixo do limite	te	0 ou 1
Qualidade do código	qc	0 a 100 %

Tabela 1: Aspectos de avaliação da implementação da solução do problema

O fator parcial de implementação será calculado pela seguinte fórmula:

$$fpi = co \times (ec - 0,15 \times (1 - qc) - 0,15 \times (1 - te))$$

Caso o valor calculado do fator seja menor que zero, ele será considerado igual a zero.

Aspecto	Sigla	Valores possíveis
Apresentação (formato, clareza, objetividade)	ap	0 a 100%
Modelagem computacional	mc	0 a 100%
Descrição da solução	ds	0 a 100%
Análise de complexidade de tempo assintótica	at	0 a 100 %

Tabela 2: Aspectos de avaliação da documentação

10.2 Fator parcial da documentação

Serão avaliados quatro aspectos da documentação entregue pelo aluno, conforme a Tabela 2.

O fator parcial de documentação será calculado pela seguinte fórmula:

$$fpd = 0,4 \times mc + 0,4 \times ds + 0,2 \times at - 0,25 \times (1 - ap)$$

Caso o valor calculado do fator seja menor que zero, ele será considerado igual a zero.

10.3 Nota final do TP

A nota final do trabalho prático será obtida pela equação a seguir:

$$NF = 15 \times (0,6 \times fpi + 0,4 \times fpd)$$

É importante ressaltar que é obrigatória a entrega do código fonte da solução e documentação. Na ausência de um desses elementos, a nota do trabalho prático será considerada igual a zero, pois não haverá possibilidade de avaliar adequadamente o trabalho realizado.

Assim como em todos os trabalhos dessa disciplina é estritamente proibida a cópia parcial ou integral de código-fontes, seja da Internet ou de colegas. Se for identifcado o plágio, o aluno terá a nota zerada e o professor será informado para que as medidas cabíveis sejam tomadas.

ATENÇÃO: Os alunos que submeterem os TPs com atraso, terão a nota final penalizada em termos percentuais de acordo com a seguinte regra: $2^{d-1}/0,16$ (onde d é a quantidade de dias úteis de atraso na entrega do TP)