TP 02 - Trabalho Prático 02

Algoritmos I

Entrega: 18/06/2022

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 18/06/2022. A especificação do conteúdo do relatório e linguagens de programação aceitas serão detalhadas nas seções subsequentes.

2 Definição do problema

Infelizmente, catástrofes estão se tornando mais e mais comuns no estado de Minas Específicas. Nos últimos anos, em particular, o estado tem sido acometido por uma série de deslizamentos de terra que destruíram estradas, soterraram rios e levaram por terra inúmeras residências.

Você faz parte da Luz Vermelha, uma ONG (Organização Não-Governamental) de ajuda humanitária fundada pela líder religiosa Ricca. A Luz Vermelha tem trabalhado arduamente em operações de ajuda emergencial para resgatar e apoiar milhares de vítimas de desastres como os que assolam as Minas Específicas, resgatando pessoas em áreas de risco e levando alimento aos mais necessitados.

Contudo, a cada dia, as operações da Luz Vermelha ficam mais e mais difíceis. Devido às constantes catástrofes naturais, as estradas encontram-se mais e mais frágeis, de forma que o roteamento dos caminhões deve ser planejado com extremo cuidado. Para piorar a situação, a Luz Vermelha não possui muitos caminhões à sua disposição, de forma que apenas um caminhão pode ser enviado para cada cidade.

Você observa um grupo de voluntários, manualmente avaliando diversas rotas entre a capital Helo Borizonte e a cidade do petróleo, Eldorado. Eles buscam a rota que permite o transporte da **maior quantidade possível de suprimentos**. Após descobrirem a melhor rota, eles irão comunicar a outro grupo a quantidade de peso máxima que poderá ser transportada pelo caminhão na rota descoberta, para que o outro grupo possa começar a alocar os recursos necessários. Devido ao grande número de cidades, o processo de roteamento é demorado e saturado por erros de cálculo. Você sabe que a automatização do processo o tornaria muito mais ágil e confiável, porém não existem outros programadores no grupo.

Por esse motivo, você propõe-se a desenvolver um aplicativo que encontre o melhor caminho entre qualquer par de cidades, dado o peso máximo que pode ser transportado através de cada rodovia. Você chamará seu aplicativo de *MaxiMini App*, uma vez que ele busca a rota com o **maior limite** de peso possível. Isto é, seu aplicativo deverá:

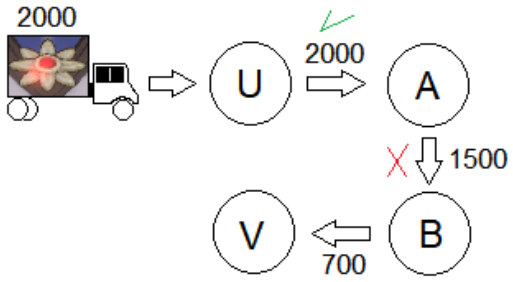

Figura 1: Com 2000 kg, o caminhão ultrapassa o limite de peso da via (A,B)

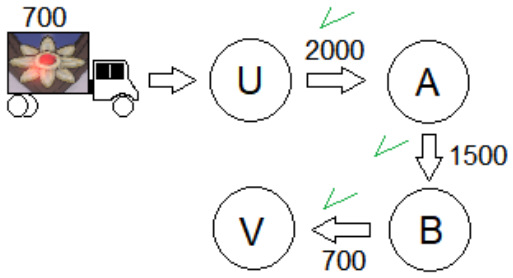

Figura 2: Com 700 kg, o caminhão pode atravessar toda a rota sem nenhum risco

- Receber uma descrição do sistema rodoviário do estado. Cada rodovia interliga duas cidades, u e v, e suporta apenas um certo limite de peso, w.
- Obter uma rota entre pares de cidades através das rodovias descritas, de forma a maximizar o peso que pode ser transportado por um caminhão ao longo da rota.
- Imprimir na tela o peso máximo que pode ser pelo caminhão da Luz Vermelha, correspondente à rota descoberta pelo seu algoritmo.

3 O que fazer?

O objetivo deste projeto é, dado um par de cidades u e v em uma malha rodoviária com n cidades e m rodovias interligando pares de cidades, encontrar o máximo de peso que pode ser transportado de u até v pelo caminhão da Luz Vermelha alocado para aquela expedição.

Considere, por exemplo, a rota descrita na Figura 1. Se um caminhão é carregado com 2000kg de suprimentos, ele poderá atravessar a rodovia interligando u e a, mas não poderá atravessar a rodovia interligando a e b, cujo limite de peso é de 1500kg. Dessa forma, a carga máxima que pode ser transportada através da rota é de 700kg (Note que uma carga mais pesada não atravessaria a rodovia interligando b e v), conforme ilustrado na Figura 2.

Isto é, um caminhão pode atravessar a rota com uma carga de peso p se, para todas as rodovias v na rota, temos $p \leq c_v$, onde c_v é o limite de peso que a rodovia v pode suportar. Desejamos, então, encontrar a rota que permite o caminhão atravessar com a maior carga possível.

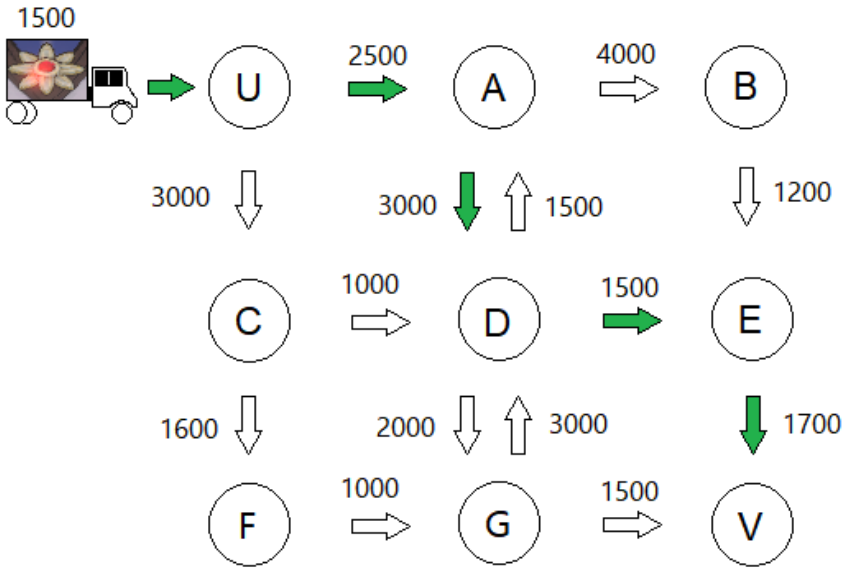

Figura 3: A melhor rota na malha rodoviária acima permite um carregamento de 1500 kg.

4 Exemplo do problema

A Figura 3 ilustra a melhor rota possível para o caminhão c na malha rodoviária descrita. No exemplo, o caminhão é carregado com 1500 kg de suprimentos. Note que qualquer carregamento mais pesado não poderá atravessar a rodovia (D,E). Note ainda que a rota U-A-D-G-V também permite o mesmo carregamento, sendo que o menor limite para o carregamento é encontrado na rodovia (G,V).

Tendo em vista que existem múltiplas rotas ótimas possíveis, o programa desenvolvido neste trabalho deve retornar apenas o carregamento permitido pela melhor rota, ao invés da própria rota (Obviamente, todas as rotas ótimas permitem o mesmo carregamento máximo).

5 Entrada

O problema terá como entrada um arquivo, que deverá ser lido pela entrada padrão (stdin), utilizando o comando "$./tp01.exe < caseTest01.txt$", por exemplo. Mais detalhes podem ser encontrados na seção 9. O arquivo de entrada contém uma descrição da malha rodoviária do estado, seguida de uma série de consultas a serem respondidas pelo seu programa. O arquivo está organizado da seguinte forma:

A primeira linha contém três valores inteiros, N, M e Q, tais que, $2 \leq N \leq 100$, $1 \leq M \leq min(N * (N-1), 1000)$ e $1 \leq Q \leq 1000$. Os valores indicam, respectivamente, o número de cidades, o número de rodovias interligando as cidades e o número de consultas a serem realizadas.

Cada uma das próximas M linhas descreve uma rodovia, contendo três valores inteiros, u, v e w, $1 \leq u, v \leq N$, $u \neq v$ e $1 \leq w \leq 100000$, indicando que a rodovia parte da cidade u e leva à cidade v, suportando um peso máximo de w. Suponha que as rodovias possuem mão única (Rodovias de mão dupla serão representadas em duas partes, (u, v) e (v, u), com pesos não necessariamente iguais).

Cada uma das próximas Q linhas descreve uma consulta. Uma consulta consiste em dois valores, u e v, indicando que desejamos saber qual é o maior carregamento que pode ser enviado de u até v. Você pode supor que sempre haverá um caminho (Por mais desgastado que o caminho esteja) de u até v.

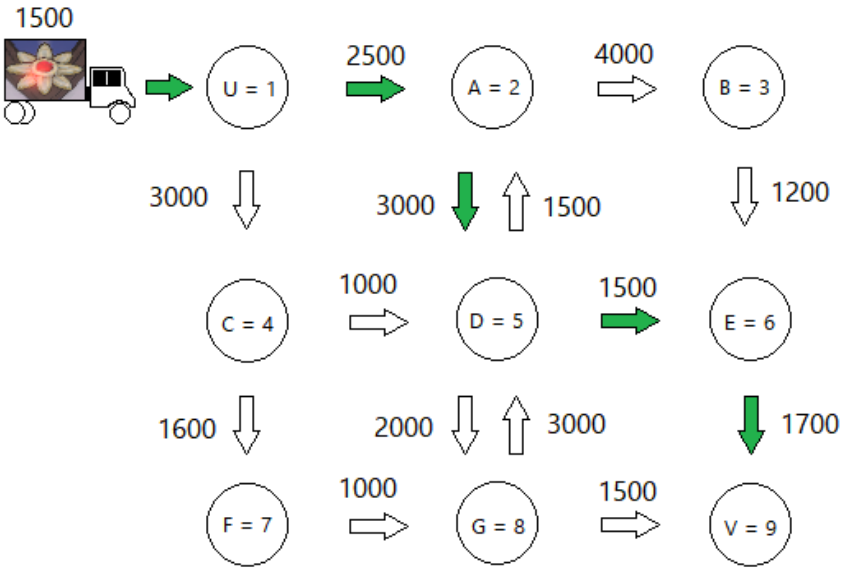

Figura 4: Equivalência entre os vértices na Figura 3 e os vértices descritos na entrada do primeiro exemplo.

6 Saída

A saída deve ser impressa na tela (*stdout*). A saída esperada consiste em Q linhas, uma para cada consulta. Para cada consulta, imprima uma linha contendo um único valor: O peso do maior carregamento que pode ser enviado da cidade u até a cidade v.

A seção seguinte ilustra com alguns exemplos a formatação utilizada na entrada e a formatação esperada para a saída.

7 Exemplo Prático de Entrada e Saída

Este primeiro exemplo ilustra a malha ferroviária descrita na Figura 3. Apenas uma consulta é realizada, entre os vértices 1 e 9 (u e v). A associação entre os índices dos vértices na entrada e seus equivalentes na Figura 3 é ilustrada na Figura 4. Conforme visto na seção 4, a saída esperada é 1500, o carregamento máximo que pode ser enviado do vértice 1 ao vértice 9.

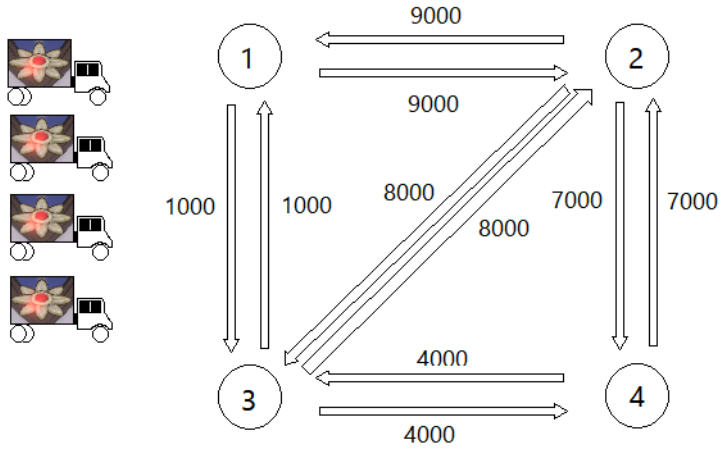

Figura 5: Grafo associado ao exemplo 2.

Arquivo de entrada

```
9 14 1
1 2 2500
2 3 4000
1 4 3000
2 5 3000
5 2 1500
3 6 1200
4 5 1000
5 6 1500
4 7 1600
5 8 2000
8 5 3000
6 9 1700
7 8 1000
8 9 1500
1 9
```

Saída esperada

```
1500
```

O próximo exemplo mostra uma série de consultas no grafo ilustrado pela Figura 5. Os caminhos ótimos são, respectivamente, 1-2-4, 2-1, 3-2-1 e 4-2-3.

Arquivo de entrada

```
4 10 4
1 2 9000
2 1 9000
1 3 1000
3 1 1000
2 3 8000
3 2 8000
2 4 7000
4 2 7000
3 4 4000
4 3 4000
1 4
2 1
3 1
4 3
```

Saída esperada

```
7000
9000
8000
7000
```

8 Especificação das entregas

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

- todos os arquivos de código-fonte implementados;
- sua documentação (arquivo pdf).
  - **ATENÇÃO:** O arquivo compactado NÃO deverá conter um arquivo *makefile*, haja visto que o comando "make"não está funcionando nas máquinas do laboratório.
  - **ATENÇÃO:** É de suma importância que a documentação contenha instruções expressas sobre a forma adequada de compilar o seu programa (Em particular, o comando ou sequência de comandos utilizado, exemplo: "g++ main.cpp").

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.

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., *$ ./tp01 < casoTeste01.txt*), e gerar o resultado também na saída padrão (não gerar saída em arquivo). Informações sobre entrada e saída padrão podem ser encontradas na página web a seguir: `https://cplusplus.com/doc/tutorial/basic_io/`.

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 e demais características do código.

ATENÇÃO: Falha em atender as especificações de entrada e saída descritas nessa seção resultará na rejeição do trabalho.

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.

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.

10.2 Fator parcial da documentação

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

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

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:

$$NF = 10 \times (0,6 \times fpi + 0,4 \times fpd)$$

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)