TP1 - Aritmofobia

Algoritmos I
Data de entrega: 05/05/2023

1 Objetivo do trabalho

O objetivo deste trabalho é modelar o problema computacional descrito a seguir utilizando estruturas de dados e algoritmos, em particular aqueles vistos na disciplina, que permitam resolvê-lo de forma eficiente.

Serão fornecidos alguns casos de teste bem como a resposta esperada dos casos fornecidos para que o aluno possa testar a corretude de seu algoritmo. Não obstante, recomenda-se que o aluno crie casos de teste adicionais a fim de validar sua própria implementação. A sua solução deve obrigatoriamente ser desenvolvida utilizando algoritmos de grafos.

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 05/05/2023. A especificação do conteúdo do relatório e linguagens de programação aceitas serão detalhadas nas seções subsequentes.

2 Definição do problema

Steven Jodds sofre de um transtorno peculiar: ele tem fobia a números ímpares. Felizmente, esse problema não o impediu de se tornar um empresário de sucesso e viajar pelo país inteiro para realizar reuniões com clientes. Infelizmente, Steve também tem medo de viajar de avião e precisa fazer todas as viagens de carro.

Steve deseja criar um algoritmo para planejar melhor suas viagens. No entanto, devido à sua aversão a números ímpares, ele nunca conseguiu aprender a programar. Por isso, ele pediu a ajuda de um exímio programador – você – para ajudá-lo nessa tarefa. Steve quer criar um algoritmo que determine o menor caminho entre duas cidades, dado um grafo que representa a rede de cidades e estradas que interconectam a região. No entanto, seu algoritmo deve levar em consideração as peculiaridades de Steve:

1. Steve só viaja entre duas cidades adjacentes se a estrada que conecta as duas cidades tiver comprimento par;
2. O caminho traçado pelo algoritmo deve passar por um número par de estradas.

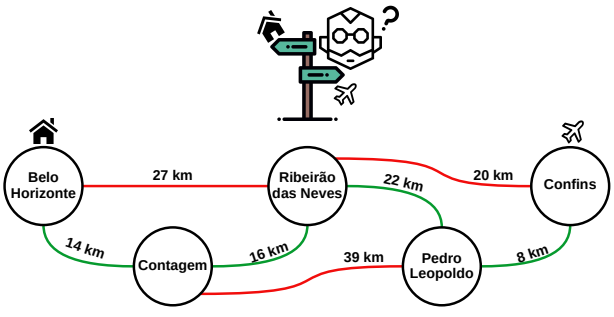

Figura 1: Caminho mínimo de tamanho par entre Belo Horizonte e Confins para o grafo apresentado. Nesse caso, o caminho mínimo de tamanho par é 60, passando por 4 estradas. Note que existem caminhos mais curtos no grafo, mas eles não respeitam às restrições do problema.

3 Exemplo do problema

3.1 Modelagem do problema

Este trabalho prático aborda a parte de grafos da ementa desta disciplina. Para a resolução do problema a sua modelagem **precisa** usar grafos e deve ser descrita sucintamente no relatório apresentado.

3.2 Formato da entrada esperada

O seu programa deverá processar um caso de teste em cada execução. A primeira linha de um cenário de teste é composta de dois número inteiros N e A, representando respectivamente o número de cidades ($2 \leq N \leq 10000$) e o número de estradas conectando pares de cidades ($1 \leq A \leq 100000$).

Cada uma das próximas A linhas descreve uma estrada entre duas cidades, representadas pelo identificador de uma cidade de origem, um identificador de uma cidade de destino e o comprimento da estrada que liga diretamente as respectivas cidades. Os dois identificadores e o comprimento da estrada são representados por inteiros positivos, a saber X_1, X_2 e D, os quais devem satisfazer as seguintes condições: $1 \leq X_1, X_2 \leq N$, $X_1 \neq X_2$ e $0 < D \leq 10000$. Pode-se assumir que: (1) todas as estradas são de mão dupla; (2) a cidade de origem sempre recebe o identificador 1; e (3) a cidade de destino sempre recebe o identificador N.

3.3 Formato da saída esperada

Para cada caso de teste seu programa deve imprimir uma linha com o tamanho do caminho mínimo satisfazendo as especificações do problema. Quando não existir tal caminho, o programa deve imprimir -1.

3.4 Casos de teste

Entrada	Saída
5 7 1 2 14 1 3 27 2 3 16 2 4 39 3 4 22 3 5 20 4 5 8	60

Entrada	Saída
5 6 1 2 3 2 3 5 3 5 2 5 1 8 2 4 1 4 5 4	-1

4 Implementação

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

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 ou login.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/).

Para facilitar o desenvolvimento vamos fornecer uma estrutura base de arquivos com Makefile já configurado. A pasta TP01-Template-CPP.zip, disponível para download na tarefa do Moodle, contém 4 arquivos: main.cpp, graph.cpp, graph.hpp e Makefile. O ponto de entrada do seu programa está no arquivo main.cpp. Para compilar seu programa basta executar o comando "make" no mesmo diretório que o Makefile está. Ao final deste comando, se a compilação for bem sucedida, será criado um arquivo executável chamado "tp01". Esse arquivo pode ser executado pela linha de comando usando "./tp01".

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).

Para avaliar automaticamente sua solução em todos os casos de teste disponibilizados, basta executar o comando "make eval", que irá testar sua solução com todos os casos de teste. Ao final, será informado quais casos falharam.

5 O que deve ser entregue

Deverá ser submetido um arquivo .zip contendo apenas uma pasta chamada tp1, esta pasta deverá conter: (i) Documentação em formato PDF e (ii) Implementação.

5.1 Documentação

A documentação deve ser sucinta e não ultrapassar 5 páginas. Você deve descrever cada solução do problema de maneira clara e precisa, detalhando e justificando os algoritmos e estruturas de dados utilizados. Para tal, artifícios como pseudo-códigos, exemplos ou diagramas podem ser úteis. Note que documentar uma solução não é o mesmo que documentar seu código. Não é necessário incluir trechos de código em sua documentação nem mostrar detalhes de sua implementação, exceto quando estes influenciem o seu algoritmo principal, o que se torna interessante. É essencial que a documentação contenha ao menos:

1. **Identificação:** Nome e Matrícula.
2. **Introdução:** Breve resumo do problema com suas palavras.
3. **Modelagem:** Modelo de grafos adotado bem como detalhamento e justificativa dos algoritmos e estruturas de dados escolhidos.

5.2 Implementação

O código fonte submetido deve conter todos os arquivos fonte e o Makefile usado para compilar o projeto. Lembre que seu código deve ser **legível**, então **evite variáveis com nomes não descritivos** (int a, aa, aaa;) e lembre-se de **comentar seu código.** Já estamos fornecendo uma implementação base com os arquivos necessários, então indicamos que você só o altere se for necessário.

5.3 Atrasos

Trabalhos poderão ser entregues após o prazo estabelecido, porém sujeitos a uma penalização regida pela seguinte fórmula:

$$\Delta_p = \frac{2^{(d-1)}}{0.32}\% \tag{1}$$

Nesta fórmula d representa dias de atraso. Por exemplo, se a nota dada pelo corretor for 70 e você entregou o TP com 4 dias corridos de atraso, sua penalização será de $\Delta_p = 25\%$ e, portanto, a sua nota final será: $N_f = 70(1 - \Delta_p) = 52.2$. Note que a penalização é exponencial e 6 dias de atraso resultam em uma penalização de 100%.

6 Considerações finais

Assim como em todos os trabalhos desta disciplina, é estritamente proibida a cópia parcial ou integral de códigos, seja da internet ou de colegas. Utilizaremos o algoritmo MOSS para detecção de plágio em trabalhos, seja honesto. Você não aprende nada copiando código de terceiros nem pedindo a outra pessoa que faça o trabalho por você. Se a cópia for detectada, sua nota será zerada e os professores serão informados para que as devidas providências sejam tomadas.

7 Atualizações

- É necessário que você realize a análise de complexidade do seu código no relatório;
- Soluções muito lentas não serão consideradas, uma vez que terão complexidades muito acima da adequada para este problema.