Trabalho Prático 01 - Algoritmos I

Data de Entrega: 10/10/2025

1 Introdução

A prefeita de Somatório, Maria Quadras, foi cobrada pela população a investir em mais áreas abertas e verdes, como praças, parques, calçadões e ciclovias, por uma cidade mais verde e menos poluída, onde as pessoas possam caminhar mais e aproveitar espaços ao ar livre.

A prefeita, porém, foi informada pelo DINFESO (Departamento de Infraestrutura de Somatório) que não haviam terrenos disponíveis para tal projeto, e que a solução seria então liberar terrenos já construídos. Após uma análise da infraestrutura da cidade, o DINFESO percebeu que algumas ruas haviam sido construídas sem muito planejamento, e que seria viável destruir essas ruas para a construção das áreas verdes, sem que rotas importantes fossem prejudicadas.

O grande problema que o DINFESO percebeu é que, para manter o trânsito da cidade funcionando corretamente, existem ruas que, se destruídas, afetariam negativamente os trajetos mais rápidos da cidade, e apenas com os registros existentes não é possível identificá-las. Mas sabe-se que, se as rotas importantes forem analisadas com cuidado, é possível descobrir quais são as ruas que não deveriam ser destruídas, pois alongariam os trajetos mais curtos da cidade, e para isso o DINFESO contratou uma pessoa especialista em computação, você, para resolver esse problema.

2 Descrição do Problema

A cidade de Somatório possui N regiões numeradas de 1 a N, conectadas por M ruas. Todas as ruas são de duplo sentido e possuem uma distância associada. Para garantir uma intervenção efetiva, a prefeita, juntamente com o DINFESO, precisam responder três perguntas:

2.1 Parte 1

A prefeita Maria Quadras está preocupada em garantir que a praça central continue sendo de rápido acesso para o principal parque ecológico da cidade, de forma a manter o bom fluxo no ponto principal de Somatório. Para isso, ela precisa saber qual é a distância mínima da praça (região 1) para o parque ecológico (região N), considerando todas as ruas da cidade em funcionamento.

A saída consiste em: uma string "Parte 1:" seguida de um número inteiro, representando a distância mínima da região 1 para a região N. É garantido que existe pelo menos um caminho ligando os dois locais.

2.2 Parte 2

O DINFESO deseja compreender melhor a infraestrutura urbana. Sabe-se que, ao se locomover pela cidade, os moradores sempre preferem os trajetos mais curtos. Pensando nisso, a prefeita quer identificar todas as ruas que participam de pelo menos uma rota de menor distância da praça para o parque ecológico. Essas ruas são candidatas importantes, pois estão envolvidas diretamente nos caminhos que os cidadãos utilizam com mais frequência.

A saída consiste em: uma string "Parte 2:", seguida dos índices das ruas que participam de pelo menos um caminho mínimo entre a região 1 e a região N.

2.3 Parte 3

Como o projeto da prefeitura deseja tornar a cidade mais ecológica sem afetar sua urbanização, a prefeita deseja descobrir quais são as ruas críticas da cidade. Essas ruas não podem ser destruídas de forma alguma, uma vez que, caso removidas, fariam com que a menor distância entre as duas regiões aumentasse, ou até mesmo tornasse o acesso impossível.

A saída consiste em: uma string "Parte 3:", seguida dos índices das ruas críticas, em ordem crescente. Os índices são definidos pela ordem em que as ruas foram fornecidas na entrada, começando em 1. Se não houver nenhuma, imprima -1".

3 Solução

3.1 Modelagem

Este trabalho prático aborda a parte de grafos da ementa desta disciplina. Por esse motivo, sua modelagem deve, necessariamente, utilizar grafos para a resolução do problema. Suas escolhas de implementação devem ser descritas na documentação do seu trabalho.

3.2 Entrada e Saída

Nessa sessão está especificado o que será enviado como entrada e o que é esperado como saída da sua solução. Para cada problema, o programa deve imprimir a parte do problema que está resolvendo, seguido da resposta no formato especificado na seção anterior. Abaixo segue o formato generalizado:

Entrada:

```
N M
A1 B1 C1
A2 B2 C2
...
AM BM CM
```

Saída:

```
Parte 1: D
Parte 2: R1 R2 ... RN
Parte 3: R1 R2 ... RN
```

Neste exemplo, temos como **entrada**:

- 1ª linha: dois números inteiros positivos N e M, onde N indica o número de regiões e M indica o número de ruas.
- Próximas M linhas: cada linha contém três números inteiros positivos A_i, B_i e C_i. A_i e B_i representam as regiões que a rua i conecta, enquanto C_i indica o comprimento da rua i.

Como **saída**, temos:

1. **Parte 1: D**
   - A string "Parte 1:"e a distância encontrada na primeira tarefa.
2. **Parte 2: R1 R2 ... RN**
   - A string "Parte 2:"e os índices das ruas encontradas na segunda tarefa, separados por espaço.
3. **Parte 3: R1 R2 ... RN**
   - A string "Parte 3:"e os índices das ruas encontradas na terceira tarefa, separados por espaço.

3.3 Exemplos

3.3.1 Exemplo 01

Entrada:

```
3 3
1 2 5
1 3 10
2 3 6
```

Saída:

```
Parte 1: 10
Parte 2: 2
Parte 3: 2
```

3.3.2 Exemplo 02

Entrada:

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

Saída:

```
Parte 1: 11
Parte 2: 1 3 4 6 7
Parte 3: 7
```

3.3.3 Exemplo 03

Entrada:

```
5 6
1 2 1
3 1 5
2 5 3
4 1 2
3 5 1
5 4 2
```

Saída:

```
Parte 1: 4
Parte 2: 1 3 4 6
Parte 3: -1
```

4 Implementação

4.1 Linguagem

O trabalho prático deverá ser implementado em `C`, `C++` ou `Python` e utilizar apenas as bibliotecas padrão das respectivas linguagens. Não será permitido o uso de bibliotecas exclusivas de um sistema operacional ou compilador.

4.1.1 C

No caso de `C`, você deve usar apenas bibliotecas do padrão ANSI C, das versões `C99`, `C11` ou `C17`.

4.1.2 C++

No caso de `C++`, utilize bibliotecas do padrão `ISO/IEC C++`, das versões `C++11`, `C++14`, `C++17` ou `C++20`.

Poderão ser utilizadas bibliotecas que implementam algumas funcionalidades mais básicas, como:

- Algumas estruturas de dados simples: `<vector>`, `<set>`, `<list>`, etc.

- Entrada e saída: `<iostream>`, `<fstream>`, etc.
- Manipulação de strings: `<string>`, `<sstream>`, etc.
- Algumas utilidades simples: `<utility>`, `<compare>`, etc.

Não poderão ser utilizadas bibliotecas que realizam funcionalidades de mais alto nível, como:

- Algoritmos mais complexos: `<algorithm>`, `<functional>`, `<ranges>`, etc.
- Gerenciamento automático de memória: `<memory>`, `<memory_resource>`, etc.

Em caso de dúvidas, pergunte aos monitores.

4.1.3 Python

No caso de Python, serão aceitas versões feitas em Python 3.9+, com acesso a todas as bibliotecas padrão da linguagem. Pacotes adicionais não serão permitidos. Em caso de erro do código devido a importação de bibliotecas proibidas, ele NÃO SERÁ AVALIADO.

4.2 Ambiente

O aluno pode implementar em qualquer ambiente de programação que desejar, mas deve garantir que o programa seja compilável nas máquinas do DCC, que são acessíveis aos alunos e disponibilizadas pelo Centro de Recursos Computacionais (para mais informações, acesse o site: https://www.crc.dcc.ufmg.br/).

4.3 Código

Utilize boas práticas de programação que você aprendeu em outras disciplinas, para que seu código seja legível e possa ser interpretado corretamente pelo leitor. A qualidade do seu código será avaliada. Algumas dicas úteis:

- Seja consistente nas suas escolhas de indentação, formatação, nomes de variáveis, funções, estruturas, classes e outros.
- Escolha nomes descritivos e evite nomear variáveis como `aux`, `tmp` e similares.
- Comente o seu código de forma breve e objetiva para descrever funções, procedimentos e estruturas de dados, mas evite comentários muito longos e de várias linhas.
- Para c,c++, separe seu código em diferentes arquivos, como `.c` e `.h` para C ou `.cpp` e `.hpp` para C++, de forma que facilite navegar pelo seu código e compreender o fluxo de execução.
- Para o Python, não é necessária a utilização de arquivos de headers.
- Evite funções muito grandes ou muito pequenas, que fazem várias coisas diferentes ao mesmo tempo, ou que tenham ou retornem muitos parâmetros diferentes.

4.4 Compilação

4.4.1 C,C++

Ao compilar o programa, você deverá utilizar no mínimo as seguintes flags:

```
-Wall -Wextra -Wpedantic -Wformat-security -Wconversion -Werror
```

Se seu programa apresentar erros de compilação, seu código não será corrigido.

4.4.2 Python

Para Python, por ser uma linguagem interpretada, o código não precisa ser compilado. Ainda assim, ele não deve apresentar erros em sua execução

4.5 Parâmetros

O aluno deve ler o arquivo de entrada do programa pela entrada padrão através de linha de comando, como por exemplo:

```
./tp1 < testCase01.txt
```

E gerar o resultado na saída padrão, não por arquivo.

5 Documentação

O aluno deverá fornecer uma documentação do trabalho contendo as seguintes informações:

1. **Introdução:** Uma breve explicação, em suas palavras, sobre qual é o problema computacional a ser resolvido.
2. **Modelagem:** Como você modelou o problema, traduzindo a situação fictícia em uma estrutura de dados, e quais algoritmos foram utilizados.
3. **Solução:** Como os algoritmos que você implementou resolvem o problema proposto e qual a ideia geral de cada um deles. A explicação deve ser de alto nível e/ou utilizar pseudocódigo, sem trechos diretos do código fonte.
4. **Análise de Complexidade:** Para cada um dos três problemas deve haver uma analíse da complexidade assíntotica demonstrada e explicada de tempo e memória da solução escolhida.
5. **Considerações Finais:** Descreva sua experiência em realizar este trabalho prático, quais partes foram mais fáceis, quais foram mais difíceis e por quê.
6. **Referências:** Liste aqui as referências que você utilizou, considerando aquilo que foi relevante para a resolução deste trabalho prático.

A documentação deverá ser **sucinta e direta**, explicando com clareza o processo, contendo não mais do que 5 páginas.

6 Entrega

A entrega deverá ser feita através do Moodle, sendo um arquivo `.zip` ou `.tar.gz` no formato `MATRICULA_NOME`, contendo os seguintes itens:

- A documentação em um arquivo `.pdf`;
- Um arquivo `Makefile` que crie um executável com o nome `tp1` (desnecessário para resoluções em Python);
- Todos os arquivos de código fonte.

7 Correção

Seu trabalho prático será corrigido de forma automática, e portanto, você deverá garantir que, ao rodar o comando `make` na pasta contendo os arquivos extraídos, seja gerado um binário executável com o nome `tp1.out` na raiz, para que seu código seja avaliado corretamente.

Serão avaliados casos de teste básicos, bem como casos mais complexos e específicos, que testarão não somente a corretude, mas também a performance da sua solução. Para que seu código seja avaliado, você deverá garantir que seu programa dê a resposta correta e ótima, conforme pedido na descrição dos problemas. Esses casos serão disponibilizados no Moodle para que você possa testar seu programa.

Você deve garantir que seu programa não apresente erros de execução ou vazamentos de memória. Caso seu programa apresente erros de execução em algum caso de teste, sua nota será zerada para o caso específico. Vazamentos de memória serão penalizados.

Em caso de suspeita de plágio, seu trabalho será zerado e os professores serão informados. É importante fazer o trabalho por conta própria e não simplesmente copiar a solução inteira de outra pessoa. Caso você tenha suas próprias ideias inspiradas em outras, deixe isso claro na seção de referências.

A entrega do código-fonte e da documentação é **obrigatória**. Na ausência de algum desses, seu trabalho não será corrigido, e portanto, será zerado.

Tenha em mente que seu código **será testado para casos muito grandes**, portanto seu código deve apresentar um **tempo de execução razoável** nas máquinas do DCC, e para isso é necessário que seu código tenha **complexidade menor que quadrática**. Esses casos não serão usados no VPL, mas alguns exemplos estão disponíveis no Moodle.

8 Avaliação

A nota final (NF) do trabalho prático será composta por três fatores: corretude (CR) e clareza (CL) do código, e documentação (DC). A corretude dirá respeito ao código do aluno passar nos casos testes de correção, a clareza dirá respeito à facilidade de compreensão do código escrito, tal como nomes de variáveis e comentários, e a documentação dirá respeito à qualidade do texto e atenção ao formato proposto na seção 5 acima.

8.1 Nota Final do Trabalho Prático (NF)

A nota final do trabalho prático será obtida pela equação:

$$NF = 0{,}4 \times CR + 0{,}2 \times CL + 0{,}4 \times DC$$

8.2 Atraso

Para trabalhos entregues com **mais do que 2 dias (corridos) de atraso**, para além do prazo de entrega, **não serão corrigidos, e portanto serão zerados**. As submissões feitas com atrasos de um e dois dias serão avaliadas em até 90% e 60% da nota máxima, respectivamente.

9 Considerações Finais

Leia atentamente a especificação e comece o trabalho prático o quanto antes. A interpretação do problema faz parte da modelagem. Em caso de dúvidas, procure os monitores, eles estão a disposição para solucionar dúvidas e ajuda-los.

Busque primeiro usar o fórum de dúvidas no Moodle, pois a dúvida de um geralmente é a dúvida de muitos.