Trabalho Prático 3
Algoritmos I

Data de Entrega: 03/02/2025

1 Introdução

A empresa *DelivExpress*, uma *startup* no ramo de e-commerce, está realizando estudos e pesquisas de mercado para determinar onde estabelecer pontos de distribuição centrais dos produtos das suas vendas online. Há vários possíveis investidores interessados, e uma demanda alta por esse tipo de serviço, onde os clientes têm como um requerimento fundamental a entrega do seu pedido na porta de casa. Como uma ideia inicial de planejamento com baixo custo e alta flexibilidade, a empresa iria estabelecer o ponto de distribuição central de uma região, e a partir dele, mandar caminhões com as entregas para todas as outras cidades daquela região, seguindo uma rota que passa por cada uma apenas uma vez e volta à cidade de origem.
Entretanto, surge um problema: o custo da operação depende fortemente do custo da rota, e determinar em qual cidade daquela região colocar o ponto de distribuição central e qual a melhor rota é um problema difícil. Considerando todas as rotas possíveis, determinar qual seria a de menor custo é algo que pode se tornar computacionalmente intratável conforme o número de cidades e estradas entre elas cresce. O problema então seria responder a pergunta:

Dado uma região com cidades e estradas, sendo que uma estrada entre duas cidades é caracterizada por uma distância $d \in \mathbb{N}$***, qual a menor rota que passa por todas as cidades sem repetir nenhuma e retorna a origem?***

Considere que existe sempre pelo menos uma rota na região, e as estradas podem ter qualquer tamanho inteiro maior do que zero. A região pode ter qualquer estrutura arbitrária.
A equipe de planejamento logístico que você faz parte, decidiu estabelecer três abordagens diferentes que podem ser usadas dependendo da situação da região

de forma flexível e escalável. Você foi escolhido para fazer uma primeira implementação das três abordagens que serão apresentadas, e fazer um estudo comparativo com as vantagens e desvantagens de cada um.

2 Estratégia 1: Força Bruta

A primeira e mais óbvia estratégia seria testar todos os possíveis caminhos e desses selecionar o ótimo. Apesar de simples e direta, essa estratégia não é escalável e se torna intratável com regiões muito grandes. Porém, para regiões menores, é uma possibilidade devido a garantia de otimalidade da solução e tempo de resposta razoável.
Sua tarefa seria identificar à partir de qual tamanho de região essa estratégia se torna inviável em comparação as outras estratégias a seguir.

3 Estratégia 2: Programação Dinâmica

A segunda estratégia é um pouco mais elaborada e menos óbvia, porém possui ganhos significativos em relação à força bruta. Em princípio, consiste em reduzir o tempo necessário para chegar a solução ideal utilizando mais memória, guardando soluções já computadas para evitar repetições desnecessárias, acelerando todo o processo.
Sua tarefa então seria determinar quanto é o ganho em tempo e também a utilização a mais de memória em relação a estratégia de força bruta.

4 Estratégia 3: Algoritmo Guloso

A terceira estratégia é diferente das anteriores no sentido de que o objetivo não é mais necessariamente chegar na solução ótima, mas em uma solução considerada "boa o suficiente" em comparação com a rota ideal. Partindo disso, é possível aproximar a solução com uma estratégia que constrói a resposta incrementalmente olhando sempre a melhor solução local, ou seja, decidir o próximo passo de forma "gulosa".
Nesse caso, sua tarefa é determinar não apenas o ganho de tempo e espaço em relação às outras duas, mas também comparar a qualidade da resposta com a solução ideal.

5 Solução

5.1 Modelagem

A sua solução para o trabalho prático deve consistir em resolver o problema proposto através de três paradigmas de programação diferentes, cada um atendendo os requisitos e restrições descritos a seguir.

5.1.1 Força Bruta

Seu algoritmo deve enumerar todas as soluções possíveis e determinar a melhor solução. Detalhes de implementação ficam ao seu critério, mas a ideia geral deve ser de uma busca exaustiva.

5.1.2 Programação Dinâmica

Seu algoritmo deve estabelecer e utilizar alguma estrutura auxiliar que faça com que a complexidade em tempo diminua e a de espaço aumente.
Sua solução deve ser necessariamente ótima e ter performance melhor que a estratégia de força bruta.
Soluções aproximadas para este paradigma não serão consideradas respostas corretas.

5.1.3 Algoritmo Guloso

Seu algoritmo deve satisfazer o princípio de uma busca local gulosa e construir a solução incrementalmente a partir disso. Qual a estratégia para determinar o ótimo local e construir a resposta aproximada fica a seu critério.
Sua implementação deve ser um algoritmo com complexidade de tempo e espaço no máximo polinomial.
Sua solução não precisa ser ótima em todos os casos, mas também não pode ser a pior solução. Você deve ser capaz de mostrar a qualidade da sua solução em comparação com o ótimo.

5.2 Entrada & Saída

Considere os exemplos de entrada a seguir e a saída esperada do seu programa. Os exemplos demonstram como a correção automática será feita, os estudos comparativos devem ser feitos independentemente.

5.2.1 Exemplo 1

Considere o exemplo de entrada a seguir:

```
b
4 6
Brierbornedon Budwleyland 10
Brierbornedon Hallow 15
Brierbornedon Melodis 20
Budwleyland Hallow 35
Budwleyland Melodis 25
Hallow Melodis 30
```

Na primeira linha, temos um único caractere que indica qual a estratégia a ser utilizada. Considere o seguinte:

- `b` se refere a **força bruta**
- `d` se refere a **programação dinâmica**
- `g` se refere a **estratégia gulosa**

Na segunda linha, temos dois números $V, E \in \mathbb{N}$ que são o número de cidades e o número de estradas, respectivamente.
Nas próximas E linhas, temos cada linha com três elementos, sendo duas cidades V_i e V_j e um número W, indicando que existe uma estrada conectando as cidades i e j de tamanho W. Em outras palavras, a distância entre i e j é W.
Nesse caso, a saída esperada seria:

```
80
Brierbornedon Budwleyland Melodis Hallow
```

Onde a primeira linha é um número $W_t > 0$ que indica o custo total da melhor rota, e a segunda linha é uma sequência de cidades $V_{x_1}, \ldots, V_{x_n}$ onde V_{x_1} é a cidade escolhida para ser o ponto de distribuição central, e a sequência indica a rota ótima encontrada.

5.2.2 Exemplo 2

Considere agora o exemplo a seguir:

```
g
4 6
Triompagne Nébulenne 12
Triompagne Mélodie 18
Triompagne Auranis 24
Nébulenne Mélodie 36
Nébulenne Auranis 28
Mélodie Auranis 32
```

Para o qual uma possível saída seria:

```
104
Triompagne Auranis Mélodie Nébulenne
```

Seguindo o mesmo padrão descrito anteriormente.
Note que a saída não é a solução ótima, o que é aceitável e esperado pois a estratégia utilizada foi o algoritmo guloso.
Estratégias gulosas diferentes irão ter resultados diferentes.

6 Implementação

6.1 Linguagem

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

6.1.1 C

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

6.1.2 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 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.

6.2 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 disponibilizadas pelo Centro de Recursos Computacionais (para mais informações, acesse o site: https://www.crc.dcc.ufmg.br/).

6.3 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 na suas escolhas de indentação, formatação, nomes de variáveis, funções, estruturas, classes e outros, etc.
- 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.
- 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.
- Evite funções muito grandes ou muito pequenas, que fazem várias coisas diferentes ao mesmo tempo, ou que tenham ou retornem muitos parâmetros diferentes.

6.4 Compilação

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.

6.5 Parâmetros

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

```
./tp3 < testCase01.txt
```

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

7 Documentação

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

1. **Introdução**: Uma breve explicação em suas palavras qual 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 mais alto nível e utilizar pseudocódigo, sem trechos diretos do código fonte. **Você deve explicar cada algoritmo não apenas demonstrando como cada um atende os critérios para o paradigma proposto, mas também de forma comparativa mostrando as diferenças entre cada um.**

4. **Análise de Complexidade**: Uma análise assintótica de tempo e memória da sua solução, devidamente demonstrada e justificada. **Você deve fazer um estudo comparativo com os casos de teste, com gráficos mostrando como cada solução se comporta conforme a entrada cresce. Para os paradigmas dinâmico e guloso, você deve mostrar o ganho em performance e o *tradeoff* tempo-espaço. No caso guloso, mostrar também a diferença na qualidade da solução.**

5. **Considerações Finais**: Descreva sua experiência em realizar este trabalho prático, quais partes foram mais fáceis, quais foram mais difíceis e porquê.

6. **Referências**: Coloque 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.

8 Entrega

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

- A documentação em um arquivo `.pdf`.
- Um arquivo `Makefile` que crie um executável com o nome `tp3`.
- Todos os arquivos de código fonte.

9 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 `tp3` para que seu código seja avaliado corretamente.
Serão avaliados casos de teste básicos como também casos mais complexos e específicos que testarão não somente a corretude mas também performance da sua solução. Para que seu código seja avaliado, você deverá garantir que seu programa dê a resposta correta pelo menos nos casos de teste mais básicos. Os mesmos 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 informados. É importante fazer o trabalho por conta própria e não simplesmente copie a solução inteira de outra pessoa. Caso você tenha suas próprias ideias inspiradas em outras, deixe 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 zerado.

10 Avaliação

A nota final (NF) do trabalho prático será composta por dois fatores: o Fator Parcial de Implementação (FPI) e o Fator Parcial de Documentação (FPD).

10.1 Fator Parcial de Implementação (FPI)

A implementação será avaliada com base nos seguintes aspectos:

Aspecto	**Sigla**	**Valores Possíveis**
Compilação no ambiente de correção	co	0 ou 1
Respostas corretas nos casos de teste	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.

A fórmula para cálculo do FPI é:

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

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

10.2 Fator Parcial da Documentação (FPD)

A documentação será avaliada com base nos seguintes aspectos:

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	ac	0 a 100%

Tabela 2: Aspectos de avaliação da documentação.

A fórmula para cálculo do FPD é:

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

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

10.3 Nota Final do Trabalho Prático (NF)

A nota final do trabalho prático será obtida pela equação:

$$NF = 10 \times (0,6 \times FPI + 0,4 \times FPD)$$

10.4 Atraso

Para trabalhos entregues com atraso, haverá uma penalização conforme a fórmula:

$$\Delta p = \frac{2^{d-1}}{16}$$

onde d representa a quantidade de dias de atraso. Assim, sua nota final será atualizada da seguinte forma:

$$NF := NF \times (1 - \Delta p)$$

Note que a penalização é exponencial e, **após 5 dias de atraso, a penalização irá zerar a sua nota**.

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