Trabalho Prático I
Algoritmos I

Data de Entrega: **13 de Novembro de 2024**

1. Introdução

O imperador de Archadia, Gramis Gana Solidor, foi notificado pelo seu conselho de segurança sobre a ameaça de invasão das nações vizinhas, apesar do tratado de paz. Com o objetivo de proteger seu território e seu povo, ele propôs novas medidas de segurança em todos os estados na borda do império, para que haja uma resposta rápida em caso de alguma invasão.
Porém, há um problema. Os mapas disponíveis no arquivo imperial são muito antigos, feitos antes da Grande Unificação, e portanto não refletem a geopolítica atual.
Com isso, foram convocados os melhores cartógrafos de todo o império, para desenhar novos mapas e definir corretamente os estados. Ao examinar a situação, os cartógrafos decidiram primeiro criar esboços do que seriam as informações importantes mínimas para o conselho de segurança, de forma a atender o pedido do imperador. Assim, chegaram a conclusão de que um estado seria um recorte geográfico contendo:

- Uma capital
- Centros urbanos
- Estradas de mão única entre os centros urbanos

Durante o processo de definir os estados, os cartógrafos encontraram problemas que, ao tentar resolver manualmente, perceberam que levaria muito tempo e esforço. Eles decidiram portanto convocar você, um famoso matemático e cientista da computação, para auxiliar.

2. Parte I: Definindo a Capital

Ao desenhar um estado, a primeira questão é definir qual dos centros urbanos teria o privilégio de ser elevado de a capital, onde teria o centro do governo estadual e o

batalhão principal, que comandaria o exército naquele estado. Porém, encontrar qual dos centros urbanos seria uma boa capital se tornou uma tarefa nada simples.
De forma a atender o principal requisito de proteger todo o estado, foi definido que a capital seria o centro urbano que, a partir dele, o exército envia tropas que **conseguem alcançar todos centros urbanos daquele estado o mais rápido possível (ou seja, passando pelo menor número de estradas)** e se proteger de um possível ataque. Com isso, você tem a sua primeira tarefa:

Qual centro urbano seria o melhor candidato a capital?

Os cartógrafos te deram a garantia que, todo esboço de um estado terá algum centro urbano que atende esse requisito, e seu papel é determinar qual.

3. Parte II: Batalhões

Mesmo encontrando a capital, ainda temos mais um problema: Pode ser o caso que tropas enviadas da capital aos demais centro urbanos não consigam retornar, por ausência das estradas adequadas. De certa forma, o caminho para tal centro urbano só comporta passagem em uma única direção e não há estradas no sentido contrário.
Assim, a solução seria encontrar quais seriam os centros urbanos onde é necessário colocar um batalhão secundário, onde as tropas podem retornar para se reabastecer, se comunicar com a capital e se reorganizar.
É aí que entra sua segunda tarefa:

Quantos batalhões secundários seriam necessários para que todas as tropas possam se reagrupar?

Tenha em mente que, o conselho de segurança deixou claro que o mínimo de batalhões a serem instalados em um estado seria ideal, pois em uma guerra, cada soldado conta. É importante também que os batalhões sejam posicionados tal que nenhuma tropa fique sem retornar a um batalhão, ou seja, uma tropa em qualquer centro urbano tem que ser capaz de retornar ao batalhão mais próximo.

4. Parte III: Patrulhamento

Tendo já resolvido os problemas anteriores, agora surgiu um novo pedido do conselho de segurança: a possibilidade de patrulhamento preventivo. Uma ideia proposta em uma reunião do conselho de segurança com o conselho de guerra seria de estabelecer rotas de patrulhamento, caso as tensões entre nações se elevassem e uma invasão estivesse para acontecer. O patrulhamento seria feito da seguinte forma:

1. A patrulha iria começar do batalhão

2. Seriam patrulhadas todas as estradas possíveis na região protegida pelo batalhão

3. A patrulha voltaria ao batalhão de origem

Para melhor manter o estado protegido e as informações do exército estejam atualizadas para tomar as melhores decisões possíveis, você se deparou com o seguinte problema:

O patrulhamento é possível? Se sim, quantos e onde?

Caso não seja possível realizar nenhuma, o conselho de segurança pediu que fosse informado sobre, para repassar ao conselho de desenvolvimento urbano para elaborar planejamentos futuros.
Em caso de uma ou mais, você deve informar quais batalhões permitem o patrulhamento e apresentar uma rota válida, como referência.

5. Solução

5.1 Modelagem

A sua solução para o trabalho prático deverá modelar o problema utilizando grafos como a estrutura de dados fundamental, e os algoritmos para solucionar o problema devem operar com essa modelagem.

5.2 Entrada & Saída

Nessa seção teremos exemplos para demostrar o que é enviado como entrada e o que é esperado como resposta do seu programa.
A entrada é o esboço de um único estado feito pelos cartógrafos, contendo as estradas que vão de um centro urbano A para B.
A saída deve conter suas soluções para os problemas propostos, de forma a apresentar:

1. **Qual é a capital**
2. **Quantos batalhões adicionais são necessários e onde eles estão**
3. **Se é possível fazer patrulhas e a suas sugestões de rota**

O formato da entrada e da saída e demonstrado nos exemplos a seguir.

5.2.1 Exemplo 1

Segue abaixo um exemplo de entrada:

```
6 8
Rhedrise Vandrad
Vandrad Benith
Khudealine Thonet
Thonet Khudealine
```

```
Rhedrise Khudealine
Benith Vandrad
Vandrad Muafland
Muafland Vandrad
```

Na primeira linha temos dois números, o primeiro sendo $N_C = 6$ o número de centros urbanos e o segundo $N_E = 8$ o número de estradas. Nas N_E linhas a seguir, cada linha segue o formato $C_1\, C_2$, separado por espaço, indicando uma estrada que vai do centro urbano C_1 até C_2. A saída esperada seria:

```
Rhedrise
2
Khudealine
Vandrad
2
Khudealine Thonet
Vandrad Muafland Vandrad Benith
```

Na primeira linha, temos o nome do capital escolhida. Na segunda linha, temos o número $N_B = 2$ indicando quantos batalhões secundários são necessários, e nas próximas N_B linhas quais centros urbanos terão esses batalhões.
Em seguida, temos o número $P = 2$ indicando quantos patrulhamentos são possíveis. Nas próximas P linhas, temos uma sequência de centros urbanos $C_1 \ldots C_k$,separados por espaço, indicando uma rota válida, onde C_1 obrigatoriamente é o centro urbano com o batalhão de origem.

5.2.2 Exemplo 2

Um outro exemplo de entrada:

```
5 6
Lagrasse Rolmuth
Yemond Bamburgh
Bamburgh Urymond
Yemond Lagrasse
Rolmuth Yemond
Urymond Yemond
```

A saída esperada para essa entrada seria:

```
Yemond
0
```

```
1
Yemond Lagrasse Rolmuth Yemond Bamburgh Urymond
```

Segue o mesmo formato do exemplo anterior, mas note que não é necessário nenhum batalhão secundário, então $N_B = 0$. Porém, o patrulhamento é possível, indicado por $P = 1$. **Note que, para o patrulhamento, você deve considerar o batalhão principal, localizado na capital**. Na próxima linha, uma rota válida como descrito anteriormente.

5.2.3 Exemplo 3

Segue abaixo um outro exemplo:

```
6 5
Vliamgate Odorhull
Odorhull Istrunstin
Vliamgate Itard
Itard Broln
Broln Ylego
```

No qual a resposta esperada seria:

```
Vliamgate
5
Odorhull
Istrunstin
Itard
Broln
Ylego
0
```

Seguindo o mesmo formato anterior, porém note que, devido a posição dos batalhões e a direção das estradas, nenhuma patrulha é possível, pois ao sair de qualquer batalhão não é possível retornar e a saída termina aí.

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:

```
./tp1 < 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.
4. **Análise de Complexidade**: Uma análise assintótica de tempo e memória da sua solução, devidamente demonstrada e justificada.

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 `tp1`.
- 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 `tp1` 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:

$$\Delta p = \frac{2^{d-1}}{64}$$

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 7 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.