

Trabalho Prático 02 - Algoritmos I

7 de maio de 2025

1 Introdução

Após diversas batalhas, um novo rei surgiu no domínio central. Para estabelecer seu comando, o rei gostaria de garantir o controle da sua capital, e precisa da sua ajuda, como conselheiro real. Devido às recentes batalhas e à necessidade de estabelecer a ordem interna, o rei gostaria de utilizar o **mínimo de tropas** para defender a capital contra seus inimigos.

Outros conselheiros já fizeram uma pesquisa e determinaram quantos soldados seriam necessários para defender cada posição no mapa contra possíveis invasores. Além disso, esses conselheiros já mapearam o reino para identificar suas montanhas, pelas quais um exército inimigo não conseguiria passar e, portanto, não seria necessária uma mobilização de soldados para sua defesa. Por fim, por causa da confiança em seus súditos, não existe possibilidade de rebeliões internas, e **todo território que fique além do mapa analisado pelos conselheiros deve ser considerado uma potencial fonte de invasores**.

Os inimigos do rei podem se deslocar **apenas horizontal e verticalmente** no mapa (uma limitação técnica conhecida entre os reinos vizinhos). Assim, sua função como o atual conselheiro do rei é decidir qual o **menor número de soldados** necessários para a proteção da capital. O território considerado como protegido é composto por todos os territórios que os inimigos não consigam chegar, incluindo os territórios onde os soldados estão.

2 Descrição do Problema

Serão dados os valores n e m representando as dimensões do mapa. Em seguida, as próximas n linhas terão m inteiros separados por espaço, representando a quantidade de soldados necessária para defender a posição. As posições indicadas com valor 0 representam as montanhas, por onde os inimigos não conseguem passar, mesmo sem tropas para defendê-las. Por fim, uma última linha apresenta dois números separados por espaço, representando as coordenadas x, y da capital.

Os casos de teste possuem as seguintes restrições: $3 \leq n, m \leq 300$ e $A_{ij} \leq 10^4$.

2.1 Parte

O rei precisa saber quantas tropas devem ser disponibilizadas para a proteção da capital. Assim, sua saída deve consistir de um único número inteiro: o menor número de soldados necessários para que a capital esteja protegida. Portanto, nenhum inimigo deve conseguir, **partindo de fora do reino**, chegar à capital sem passar por uma posição totalmente defendida.

3 Solução

3.1 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.

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
A11 A12 ... A1M
A21 A22 ... A2M
.
.
.
AN1 AN2 ... ANM
X Y
```

Saída:

```
S
```

3.3 Exemplos

3.3.1 Exemplo 01

Entrada:

```
5 7
9 1 1 1 9 9 9
9 0 9 1 1 1 9
9 0 9 50 9 1 9
9 1 1 1 9 1 9
9 9 9 1 9 9 9
3 4
```

Saída:

```
13
```


9	1		1	1	9	9	9
9		0	9		1	1	9
9		0	9		50		9
9	1		1		1	9	1
9	9	9	1	9	9	9	9

Figura 1: Solução do Exemplo 01

3.3.2 Exemplo 02

Entrada:

5 7
 1 1 1 1 1 1 0
 1 0 4 4 4 1 0
 1 0 4 20 4 1 0
 1 1 4 4 4 1 0
 1 1 1 1 1 1 0
 3 4

Saída:

9


1	1		1		1		1		0
1		0	4	4	4		1		0
1		0	4		20	4	1		0
1		1	4	4	4	1		0	
1	1		1		1		1		0

Figura 2: Solução do Exemplo 02

3.4 Exemplo 03

Entrada:

9 4
 1 1 1 1
 1 2 1 1
 1 0 1 1
 1 1 1 1
 1 1 1 1
 1 1 1 1
 0 0 1 1
 1 1 1 1
 1 1 1 1
 2 2

Saída:

2





1	1	1	1
1		2	1
1		0	1
1	1	1	1
1	1	1	1
1	1	1	1
	0		0
1	1	1	1
1	1	1	1

Figura 3: Solução do Exemplo 03

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, e eles não funcionarão na VPL. 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/>). A maneira mais fácil de conferir isso é rodando o código nos VPL's de entrega, caso tudo compile corretamente, é um forte indicativo de que o programa está compatível com o ambiente das máquinas do DCC.

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 análise da complexidade assintótica 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.

Ademais, também serão disponibilizados VPL's para a correção automática, para que você receba a pontuação pela correção do código é necessário enviar sua solução para esses VPL's

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`, 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 correção, 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.

8 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).

8.1 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.

8.2 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.

8.3 Nota Final do Trabalho Prático (NF)

A nota final do trabalho prático será obtida pela equação:

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

8.4 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.

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.