Trabalho Prático 01 - Algoritmos I

13 de abril de 2025

1 Introdução

Em um animado festival de música, Ana e Bernardo decidiram explorar as delícias gastronômicas oferecidas pelas diversas barraquinhas espalhadas pelo local. Enquanto Ana foi experimentar os doces caseiros, Bernardo preferiu os sabores salgados. No entanto, o clima mudou abruptamente, e uma chuva intensa começou a cair, transformando o festival em uma corrida contra o tempo para encontrar abrigo.

Felizmente, o festival estava bem preparado. Além das barraquinhas, diversos guarda-sóis foram montados, cada um formando um abrigo circular perfeito para proteger os visitantes da chuva. Ana e Bernardo, agora separados, precisam se encontrar sem se molhar. Eles têm seus celulares e podem compartilhar suas localizações em tempo real.

Cada abrigo da chuva é representado por um círculo de raio R centrado em (x, y). As pessoas conseguem se mover livremente dentro de qualquer abrigo, inclusive passar de um abrigo para outro se eles cobrirem uma mesma região (inclusive se forem tangentes um ao outro). Entretanto, sair da cobertura dos abrigos resultaria em se molhar, o que não é desejado por nenhuma das pessoas.

2 Descrição do Problema

Serão dadas coordenadas (A_x, A_y), as coordenadas iniciais de Ana, e (B_x, B_y), as coordenadas iniciais de Bernardo. Também será dado um número N, o número de abrigos, seguido por N linhas de três números, R, X e Y, o raio R, e as coordenadas x e y do abrigo. Podemos considerar que inicialmente Ana e Bernardo estão sob um único abrigo cada, podendo ser o mesmo abrigo. Além disso, considere que todos os dados do problema são inteiros e nenhum abrigo está totalmente dentro de outro.

2.1 Parte 1

Como as regiões de passagem estão muito cheias, Ana e Bernardo desejam atravessar o menor número possível de abrigos. Ajude-os, determinando o número mínimo de abrigos para se encontrarem.

A saída consiste em: uma string "Parte 1: "para indicar o problema resolvido, seguida de um único número inteiro S, determinando o número de abrigos mínimos que eles precisam atravessar para se encontrarem, e -1 caso não seja possível o encontro.

2.2 Parte 2

A organização do festival está preocupada com a facilidade de movimentação das pessoas durante a chuva, garantindo que todos possam visitar as diversas barraquinhas gastronômicas durante o evento. Após uma pesquisa, percebeu que as pessoas sempre utilizam o menor caminho para se movimentar, quando já definidas barracas de início e destino. Pensando em novos abrigos em edições futuras, a organização gostaria de saber qual é o maior número de abrigos que uma pessoa qualquer precisaria atravessar no festival após o início da chuva.

A saída consiste em uma string "Parte 2: "para indicar o problema resolvido, seguida de um único número inteiro T, que indica o máximo de abrigos que uma pessoa em um lugar qualquer do festival precisa atravessar para chegar a qualquer outro lugar já acessível no início da chuva, considerando que ela faz um caminho ótimo.

2.3 Parte 3

A organização também tem uma outra preocupação: todas as barracas que estavam acessíveis após o início da chuva continuem acessíveis ao longo da tempestade. Assim, deseja identificar quais abrigos são críticos, isto é, aqueles cuja remoção (seja por um problema de infraestrutura) poderia interromper a conexão entre regiões do festival. Dessa forma, a organização gostaria de reforçar esses abrigos para evitar maiores complicações e garantir a continuidade do evento.

A saída consiste em uma string "Parte 3: "para indicar o problema resolvido, seguida do número de abrigos críticos M, seguido por seus índices i (os índices são definidos pela ordem de entrada, a começar por 1).

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:

```
Ax Ay
Bx By
N
R1 x1 y1
R2 x2 y2
...
Rn xn yn
```

Saída:

```
Parte 1: S
Parte 2: T
Parte 3: M i1 i2 ... im
```

3.3 Exemplos

3.3.1 Exemplo 01

Entrada:

```
9 -8
-6 -9
5
3 -4 -4
3 -8 -8
5 7 -3
3 0 0
4 10 -9
```

Saída:

```
Parte 1: 4
Parte 2: 4
Parte 3: 3 1 3 4
```

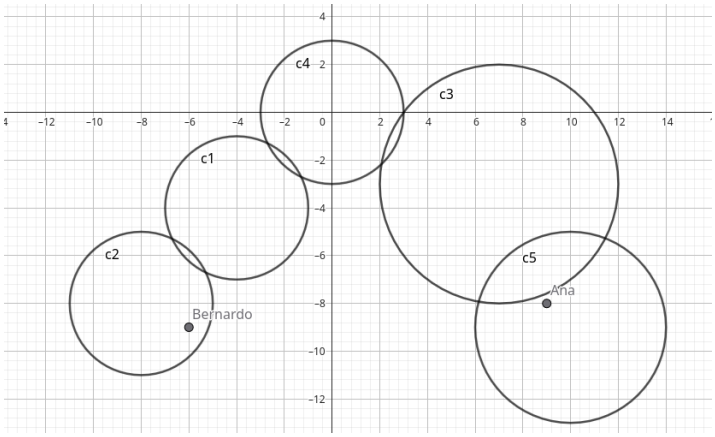

Figura 1: Representação do Exemplo 01

Neste exemplo, temos como **entrada**:

- 1ª linha indicando as coordenadas de Ana: $(9, -8)$
- 2ª linha indicando as coordenadas de Bernardo: $(-6, -9)$
- 3ª linha indicando o número de abrigos: 5
- Próximas 5 linhas especificando o *raio* e as *coordenadas* de cada abrigo

Como **saída**, temos:

1. **Parte 1:** o número S de abrigos mínimos que Ana e Bernardo terão que atravessar para se encontrarem — **4**
2. **Parte 2:** o número T de abrigos que uma pessoa teria que atravessar para ir de um lugar qualquer para outro acessível — **4**
3. **Parte 3:** o número M de abrigos críticos — **3**, e seus respectivos índices: **1, 3 e 4**

3.3.2 Exemplo 02

Entrada:

```
9 -8
-6 -9
10
3 -4 -4
3 -8 -8
5 7 -3
3 0 0
4 10 -9
2 -4 -10
1 -2 -12
2 -2 -14
2 0 -16
4 5 -13
```

Saída:

```
Parte 1: 4
Parte 2: 5
Parte 3: 0
```

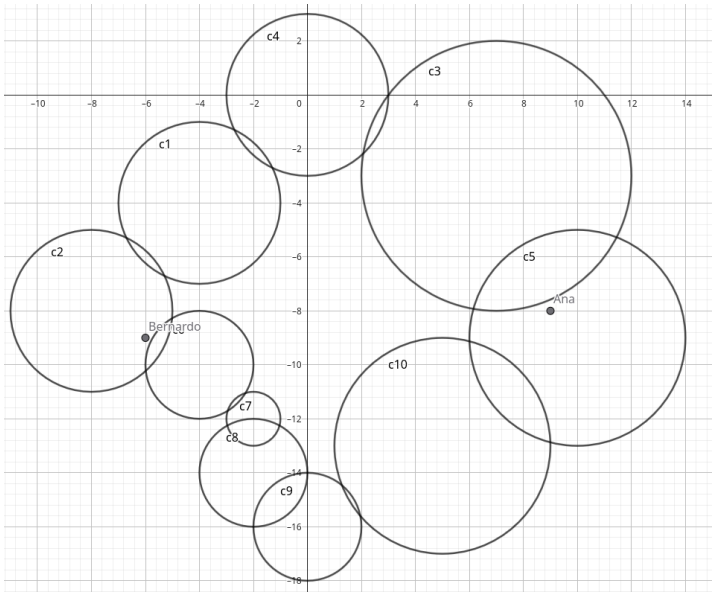

Figura 2: Representação do Exemplo 02

3.3.3 Exemplo 03

Entrada:

```
0 0
75 0
9
10 0 0
10 15 0
10 30 0
10 15 15
10 40 30
10 30 15
10 50 40
10 45 0
10 75 0
```

Saída:

```
Parte 1: -1
Parte 2: 5
Parte 3: 4 2 3 5 6
```

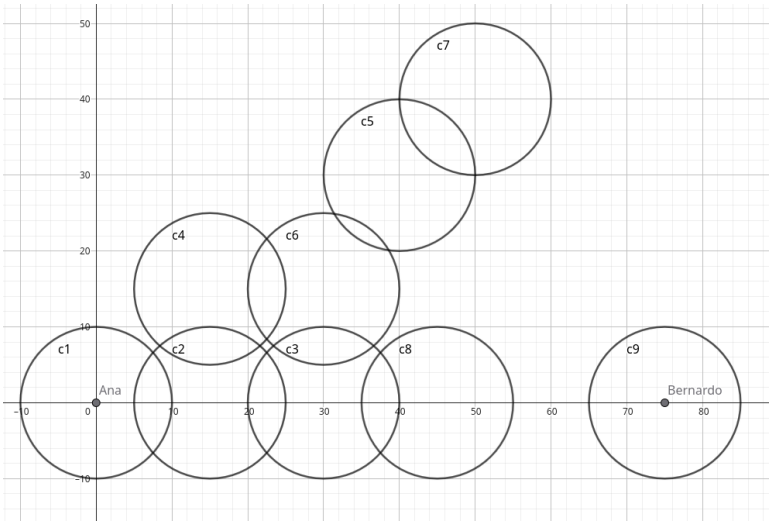

Figura 3: Representação do Exemplo 03

Embora pessoas que estavam no abrigo 9 não consigam acessar os demais abrigos após o início da chuva, o maior número de abrigos que uma pessoa atravessaria no festival seria 5: A pessoa está no abrigo 1 e gostaria de chegar no abrigo 7, para isto ela poderia fazer o caminho $1 \to 2 \to 4 \to 6 \to 5 \to 7$.

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 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.

Ademais, também serão disponibilizados VPL's para a correção automática, para que você receba a pontuação pela corretude 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 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.

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.