Trabalho Prático 02 - Algoritmos I

Data de Entrega: 16/11/2025

1 Introdução

O prefeito de Triangulândia, Roberto Ângulo, é conhecido por sua paixão por triângulos e simetrias. Ao caminhar pela cidade, percebeu que muitos muros estavam em ruínas e que algumas praças estavam desorganizadas. Inspirado por essa observação, decidiu reformar a cidade com base em formas geométricas perfeitas e, claro, triângulos seriam o símbolo dessa nova fase.

Para tornar sua visão realidade, o prefeito determinou duas tarefas principais que exigem não apenas senso estético, mas também soluções computacionais eficientes. Na primeira, ele precisa reconstruir um muro de madeira em formato de triângulo isósceles, aproveitando ao máximo os blocos já existentes — sem desperdício e sem adições. Na segunda, ele deseja delimitar uma área triangular entre três árvores escolhidas em um parque da cidade, de forma que o perímetro seja o menor possível, garantindo economia de recursos e preservação ambiental na instalação da nova cerca ecológica.

Você foi contratado pela prefeitura para descobrir como otimizar os projetos de Roberto Ângulo,

2 Descrição do Problema

O trabalho será dividido em duas partes, onde cada uma representa um dos problemas citados:

2.1 Parte 1

O antigo muro de Triangulândia tem N pilhas onde a i-ésima delas contém B_i blocos. Roberto acredita que transformar esse muro em um triângulo seria uma excelente representação do espírito da cidade. Como a prefeitura não possui mais blocos como esses, o prefeito deseja construir o maior triângulo isósceles possível apenas retirando os blocos que julgar necessário. Sua missão é informar qual a altura máxima do triângulo a ser construído.

A saída consiste em: uma string “Parte 1:” seguida de um número inteiro, representando a altura do maior triângulo isósceles possível sem adicionar nenhum bloco.

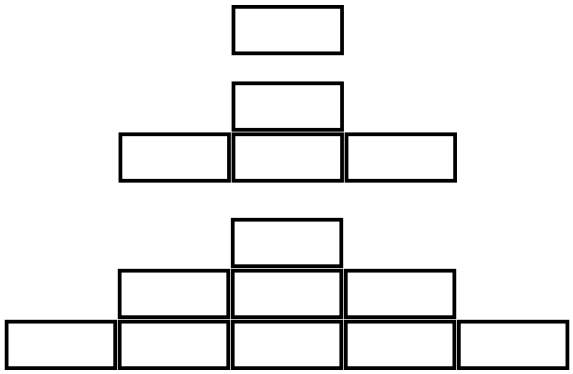

Figura 1: Possíveis resultados de altura 1, 2 e 3, respectivamente

2.2 Parte 2

A área verde da cidade tem Z árvores, cada uma representada pelas coordenadas X, Y que a localizam no parque, que é de livre circulação para a população. Contudo, a prefeitura julga necessário o estabelecimento de uma pequena área restrita para preservação ambiental. O prefeito deseja escolher três árvores para cercar, formando um triângulo. Para economizar na construção da cerca, o perímetro da área de preservação deve ser o menor possível.

A saída consiste em: uma string "Parte 2:", seguido do valor do perímetro e dos índices das três árvores escolhidas.

3 Solução

3.1 Modelagem

Este trabalho prático aborda a parte de algoritmos gulosos e de divisão e conquista da ementa desta disciplina. Por esse motivo, sua solução da Parte 1 deve utilizar um algoritmo guloso, e a Parte 2, divisão e conquista. Suas escolhas de implementação devem ser descritas na documentação do seu trabalho.

3.2 Entrada e Saída

Nesta 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
B1 B2 B3 ... BN
Z
X1 Y1
X2 Y2
...
XZ YZ
```

Saída:

```
Parte 1: H
Parte 2: P A1 A2 A3
```

Temos como **entrada**:

- 1ª linha: um inteiro positivo N, $1 \leq N \leq 50000$, indicando o número de pilhas.
- 2ª linha: N inteiros B_i, $1 \leq B_i \leq N$, para $1 \leq i \leq N$, indicando a quantidade de blocos em cada pilha.
- 3ª linha: um inteiro positivo Z, $3 \leq Z \leq 50000$, indicando o número de árvores disponíveis.
- Próximas Z linhas: cada linha contém dois números inteiros X_i e Y_i, $-10^6 \leq X_i, Y_i \leq 10^6$, indicando as coordenadas da árvore i. Podem existir pontos repetidos.

Como **saída**, temos:

1. **Parte 1: H**
   - A string "Parte 1:"e a altura máxima do triângulo encontrado.
2. **Parte 2: P A1 A2 A3**
   - A string "Parte 2:", seguida do perímetro do triângulo formado pelos três pontos, impresso com exatamente quatro casas decimais, e da tripla de índices das três árvores encontradas pelo algoritmo, em ordem crescente e separados por espaço. A tripla deve ser lexicograficamente mínima.

Uma tripla A é lexicograficamente menor que uma tripla B se no primeiro índice onde as duas se diferenciam, a tripla A tem um elemento menor que a de B. Ou seja, se duas triplas de pontos, por exemplo, "1 2 3" e "1 2 4" resultarem no mesmo menor perímetro, deve-se imprimir "1 2 3".

3.3 Exemplos

3.3.1 Exemplo 01

Entrada:

```
16
5 6 5 8 9 10 5 8 9 5 7 9 9 9 6 3
5
0 0
1 1
3 0
0 3
3 3
```

Saída:

```
Parte 1: 6
Parte 2: 6.6503 1 2 3
```

3.3.2 Exemplo 02

Entrada:

```
8
5 1 1 1 1 1 1 3
3
0 0
0 1
1 0
```

Saída:

```
Parte 1: 1
Parte 2: 3.4142 1 2 3
```

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 com sistema operacional **Linux**, 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.
- Seu código deve ser legível e devidamente comentado.

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:

```
./tp2.out < 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 `tp2.out` (desnecessário para resoluções em Python);
- Todos os arquivos de código fonte.
- No caso de Python, seu arquivo de código principal deve se chamar `tp2.py`

Atenção: caso seu trabalho não siga as instruções dadas nessa seção, **você será penalizado.**

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 `tp2.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.