TP 01 - Trabalho Prático 01

Algoritmos I

Entrega: 30/06/2021

1 Objetivos do trabalho

O objetivo deste trabalho é modelar o problema computacional descrito a seguir utilizando uma estrutura de dados que permita resolvê-lo de forma eficiente com os algoritmos estudados nesta disciplina.

Serão fornecidos alguns casos de teste bem como a resposta esperada para que o aluno possa verificar a corretude de seu algoritmo. Não obstante, recomenda-se que o aluno crie casos de teste adicionais a fim de validar sua própria implementação.

O código-fonte da solução e uma documentação sucinta (relatório contendo não mais do que 5 páginas) deverão ser submetidos via *moodle* até a data limite de 30/06/2021. A especificação do conteúdo do relatório e linguagens de programação aceitas serão detalhadas nas seções subsequentes.

2 Definição do problema

Finalmente, após meses de espera, a população brasileira esta sendo vacinada após a aprovação nos testes das vacinas de imunização contra o vírus Sars-CoV-2, causador da doença covid-19.

A Secretaria de Saúde de uma capital brasileira realizou uma grande compra de vacinas de um fabricante e deseja distribuí-las de forma eficiente e justa para seus habitantes. Estas vacinas serão distribuídas em postos de vacinação espalhados por vários pontos da cidade, seguindo algumas diretrizes principais, após cadastro da população. Cada posto de vacinação possui um limite de pessoas que podem ser atendidas por dia. Isso se deve às normas sanitárias que devem ser seguidas e à capacidade de atendimento desses postos. As pessoas terão a prioridade de imunização dada pela sua idade e devem ser alocadas aos postos mais próximos, de acordo com a sua localidade. Ou seja, uma pessoa de 75 anos, cujo posto mais próximo fique a 3 km dela terá prioridade de alocação em relação a pessoas mais jovens, mesmo que mais próximas daquele posto.

Você foi contratado pelo prefeito da cidade para desenvolver um programa para auxiliar na alocação de pessoas para postos de vacinação para um único dia, seguindo as diretrizes supracitadas. Este programa deverá:

- Ler as localidades geográficas e as idades das pessoas cadastradas para vacinação;
- Ler as localidades geográficas e as capacidades dos postos de vacinação;
- Alocar as pessoas aos pontos de vacinação da cidade. Os postos devem alocar as pessoas com maior idade e mais próximas;
- Condição de estabilidade: se uma pessoa não for alocada num posto mais próximo a ela, não deve haver naquele posto vagas não alocadas ou alocações de pessoas mais jovens que ela. As situações de instabilidade serão detalhadas na próxima seção.

- Regra de desempate de idades: no caso de pessoas com mesma idade, terá prioridade aquela com o menor id (posição sequencial no arquivo de entrada).
- Regra de desempate de postos: se dois postos desejam alocar uma pessoa equidistante aos dois, ela deverá ser alocada ao posto com menor id (posição sequencial no arquivo de entrada).

3 O que fazer?

O objetivo deste trabalho será distribuir as pessoas entre as vagas de forma a satisfazer alguns critérios. Considere um processo de distribuição no qual n pessoas devem ser designadas para m postos de vacinação, onde Q_i é o número máximo de pessoas que podem ser vacinadas no posto i em um mesmo dia. Ao efetuar o cadastro para vacinação, cada pessoa p informa sua idade a_p e a localização geográfica (x_p, y_p) da sua residência, e para elas deve ser priorizada a alocação no posto mais próximo. De outro lado, cada posto de vacinação i possui um número de vagas e uma localização (x_i, y_i), e deverá propor as alocações priorizando as pessoas com maior idade. Dessa forma, uma pessoa p deverá ser designada para o posto de vacinação com a menor distância da sua residência (a) se existirem vagas disponíveis ou (b) se o posto, mesmo sem vagas disponíveis, tiver uma alocação de alguma pessoa q mais jovem que p, a qual deve ser ajustada. A distância considerada é a distância euclidiana, que pode ser calculada pela seguinte fórmula: $d(i,p) = \sqrt{(x_p - x_i)^2 + (y_p - y_i)^2}$.

A condição essencial que deve ser satisfeita para uma alocação é que ela não apresente **"instabilidades"**. Em outras palavras, queremos que a alocação seja a mais justa possível para a população, respeitando essas restrições do problema (idade e o número de vagas por posto). Uma alocação será instável e inválida quando alguma das seguintes situações ocorrer:

- Uma pessoa p_1 está alocada para um posto β, mas há um posto α mais próximo que possui vagas disponíveis ou está alocando uma pessoa p_2 de menor idade.
- Há uma pessoa sem alocação mas ainda existem vagas em um dos postos.

Além disso a alocação deve ser **simultaneamente a melhor possível para todas as pessoas**, respeitando as restrições descritas anteriormente. Ou seja, a alocação será **ótima do ponto de vista das pessoas**. Sendo assim, a pessoa que tiver a maior idade sempre conseguirá ser atendida no posto mais perto. Todos os postos ordenam as pessoas por idade (pessoas mais velhas devem ser atendidos primeiramente), logo, em essência, as preferências de cada posto são as mesmas. Cada pessoa possui um identificador p, $0 \leq p \leq (n-1)$ e cada posto possui um identificador c, $0 \leq c \leq (m-1)$. **Será adotada como premissa que, quando outros aspectos forem iguais, a pessoa ou posto com menor identificador terá prioridade**.

A imagem 1 ilustra três instabilidades e um desrespeito às regras de desempate: no quadrante A, C0 deveria ter alocado P1 (62 anos) e não P0 (28 anos), mesmo estando este mais próximo de C0; no quadrante B, C1 e C2 deveriam ter alocado P3 e P2, respectivamente, e não o inverso; no quadrante C, P4 não foi alocado, e deveria ser atribuído ao posto C3, que ainda tem capacidade para duas pessoas e está mais próximo que o posto C4, e ,por fim, no quadrante D a regra de desempate não foi observada: embora tenham a mesma idade e a mesma distância em relação ao posto C5, P5 deveria ser alocado ao invés de P6, pois tem menor id.

4 Exemplo do problema

A seguir é apresentado um exemplo de alocação estável para um caso de teste. Observe na Figura 2 uma estrutura logística na qual há 2 pontos de vacinação (C0, C1), ambos com capacidade de atendimento de

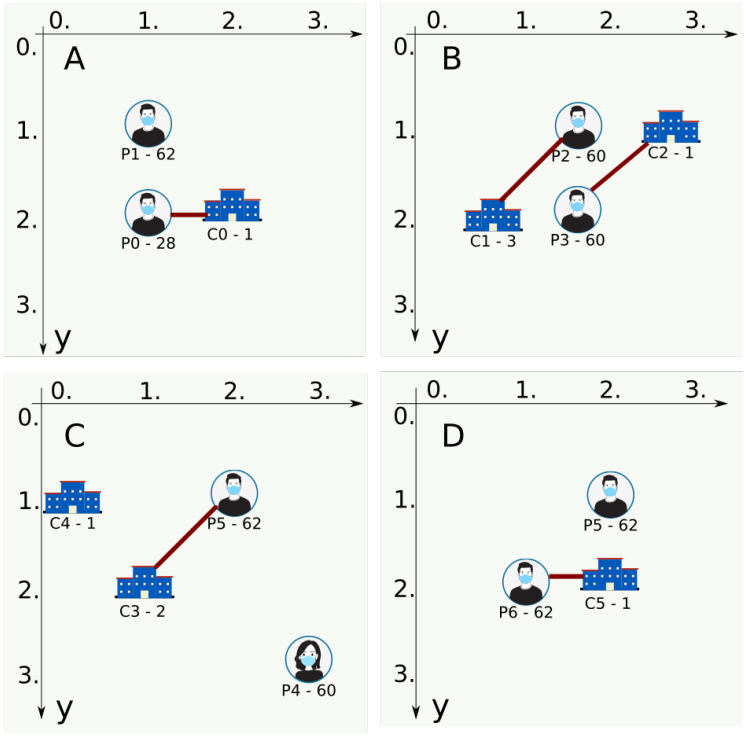

Figura 1: Exemplo de situações instáveis ou fora da regra de desempate

3 pessoas no dia. Há 7 pessoas inscritas para vacinação no dia (P0, P1, P2, P3, P4, P5, P6). Considere ainda a capacidade de cada um dos posto de vacinação e a idade das pessoas listadas na Figura 2.

Após calculada a distância euclidiana de cada pessoa para os postos teremos a seguinte lista de preferencia de postos para cada pessoa (desempatando pelo identificador dos postos caso apresentem a mesma distância):

Lista de proximidade por pessoa

```
P0 - {C0, C1}                          lista de proximidade para primeira pessoa
P1 - {C0, C1}
P2 - {C0, C1}
P3 - {C1, C0}
P4 - {C1, C0}
P5 - {C0, C1}
P6 - {C1, C0}
```

Já para os postos teremos a seguinte lista de prioridade de atendimento em relação a idade das pessoas (novamente desempatando pelo identificador das pessoas):

Lista de prioridade de pessoa nos postos

```
C0 - {P4, P2, P1, P5, P3, P0, P6}      lista de prioridade para primeiro posto
C1 - {P4, P2, P1, P5, P3, P0, P6}
```

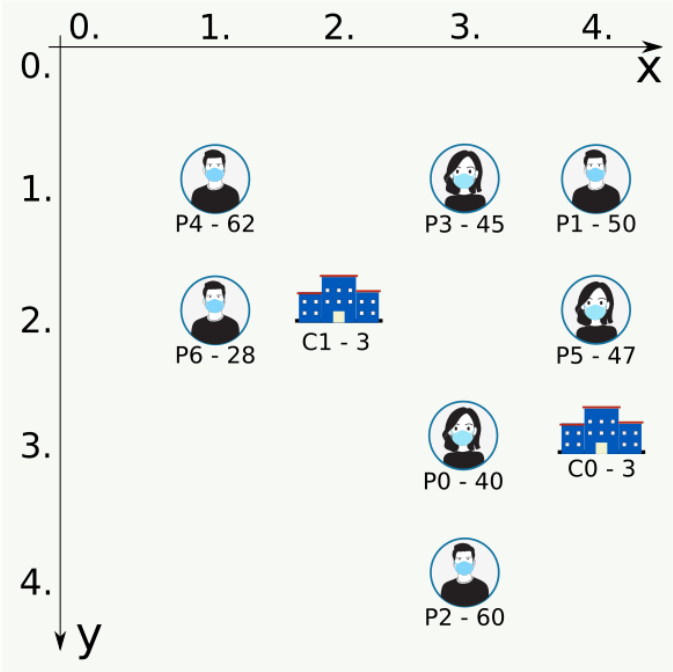

Figura 2: Exemplo de situação com dois postos de vacinação e sete pessoas inscritas no dia

Com base nestas informações teremos uma alocação estável de pessoa para os postos da seguinte forma:

Lista de atendimento de pessoa nos postos

```
C0 - {P2, P1, P5}
C1 - {P4, P3, P0}
```

Nesta alocação a pessoa P6 não foi contemplada para o dia em questão. Note que a alocação é a melhor possível do ponto de vista das pessoas visto que a pessoa mais velha sempre é alocada para o posto mais perto de sua residência. Note também que todos os critérios definidos anteriormente são mantidos nesta alocação.

No segundo exemplo, ilustrado na Figura 3, a solução dada envolve a observância de uma regra de desempate pelo id das pessoas. A alocação para o exemplo da Figura 3 deverá ser:

Lista de atendimento de pessoa nos postos

```
C0 - {P0}
C1 - {P1, P2}
```

No terceiro exemplo, ilustrado na Figura 4, a solução dada também envolve a observância de uma regra de desempate, mas agora pelo id dos postos. A alocação para o exemplo da Figura 4 deverá ser:

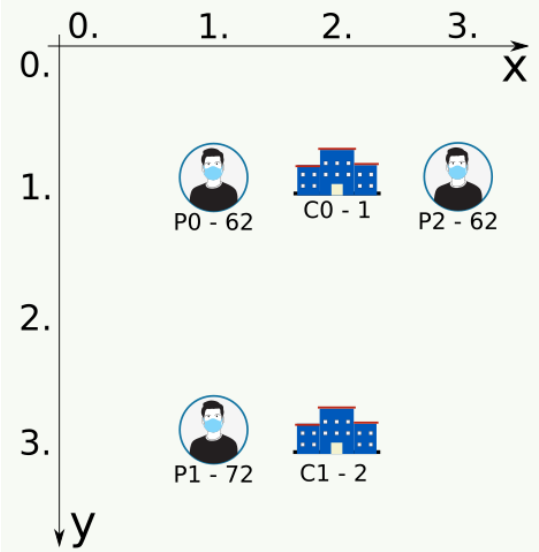

Figura 3: Exemplo de alocação com dois postos de vacinação e três pessoas inscritas no dia

Lista de atendimento de pessoa nos postos

```
C0 - {P0}
C1 - {P1, P2}
C2 - {P3}
```

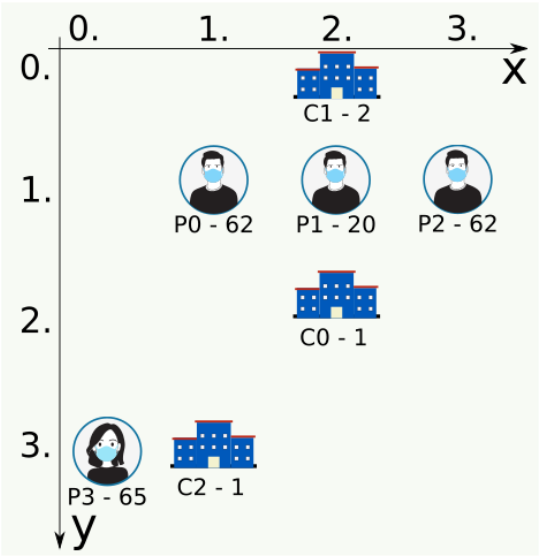

Figura 4: Exemplo de alocação com três postos de vacinação e quatro pessoas inscritas no dia

No quarto exemplo, ilustrado na Figura 5, temos uma demanda de alocação de quatro pessoas para três postos de vacinação. A alocação para o exemplo da Figura 5 deverá ser:

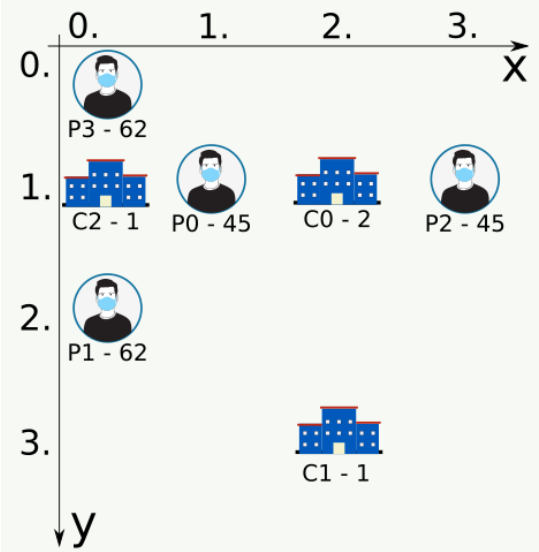

Figura 5: Exemplo de alocação com três postos de vacinação e quatro pessoas inscritas no dia

Lista de atendimento de pessoa nos postos

```
C0 - {P0}
C1 - {P1, P2}
C2 - {P3}
```

5 Arquivos de entrada

O problema terá como entrada um arquivos contendo a relação dos postos de vacinação e das pessoas inscritas. Os alunos terão que ler os arquivos e armazená-los em uma estrutura de dados para utilizar na resolução do problema. O arquivo está organizado da seguinte forma:

A primeira linha contém um número m inteiro, $0 < m \leq 10^4$, relacionado ao número total de postos de vacinação. Em sequência, cada uma das m linhas a seguir deverá contém uma tupla de inteiros com a capacidade do posto U_i, $0 < U \leq 10^4$, e sua localização x e y, $0 \leq x_i, y_i \leq 10^5$, todos separados por espaço.

A próxima linha conterá um número n inteiro, $0 < n \leq 10^5$, relacionado ao número total de pessoas. Em sequência, cada uma das n linhas a seguir deverá conter uma tupla de inteiros com, respectivamente, a idade I_i da pessoa i, $0 < I_i \leq m$, e sua localização x_i e y_i, $0 \leq x_i, y_i \leq 10^5$, todos separados por espaço também.

Note que há três cenários possíveis: há mais vagas que aplicações, há um número igual de vagas e aplicações ou há menos vagas que aplicações. A seguir está listado um exemplo de entrada e sua respectiva saída.

6 Saída

A saída deve ser impressa na tela (*stdout*) e seguir o seguinte padrão: Deverá ser impresso m saídas com a primeira linha com o identificador do posto e a segunda linha contendo uma lista de *ids* das pessoas que

foram alocadas para aquele posto.

A impressão desses grupos deve ser ordenado pelo identificador do inscrito. Lembrando que a identificação ocorre pela posição da pessoa ou posto no arquivo (0 para o primeiro posto/pessoa). Fique atento para imprimir exatamente como foi descrito pois a correção do algoritmo será feita de forma automática.

7 Exemplo Prático de Entrada e Saída

Arquivo contendo as universidades

```
6                                       // <Número de postos>
3 4 3                                   // <Capacidade> <localização> (Posto 0)
3 2 2
7 7 3
5 4 8
2 10 6
2 7 5
23                                      // <Número de pessoas>
40 3 3                                  // <Idade> <localização> (Pessoa 0)
50 4 1
60 3 4
45 3 1
62 1 1
47 4 2
28 1 2
30 6 2
40 8 3
55 8 2
43 8 1
60 11 1
70 10 3
37 12 5
31 12 7
56 8 8
51 9 5
48 7 4
33 5 5
28 6 7
50 3 6
70 1 5
54 2 8
```

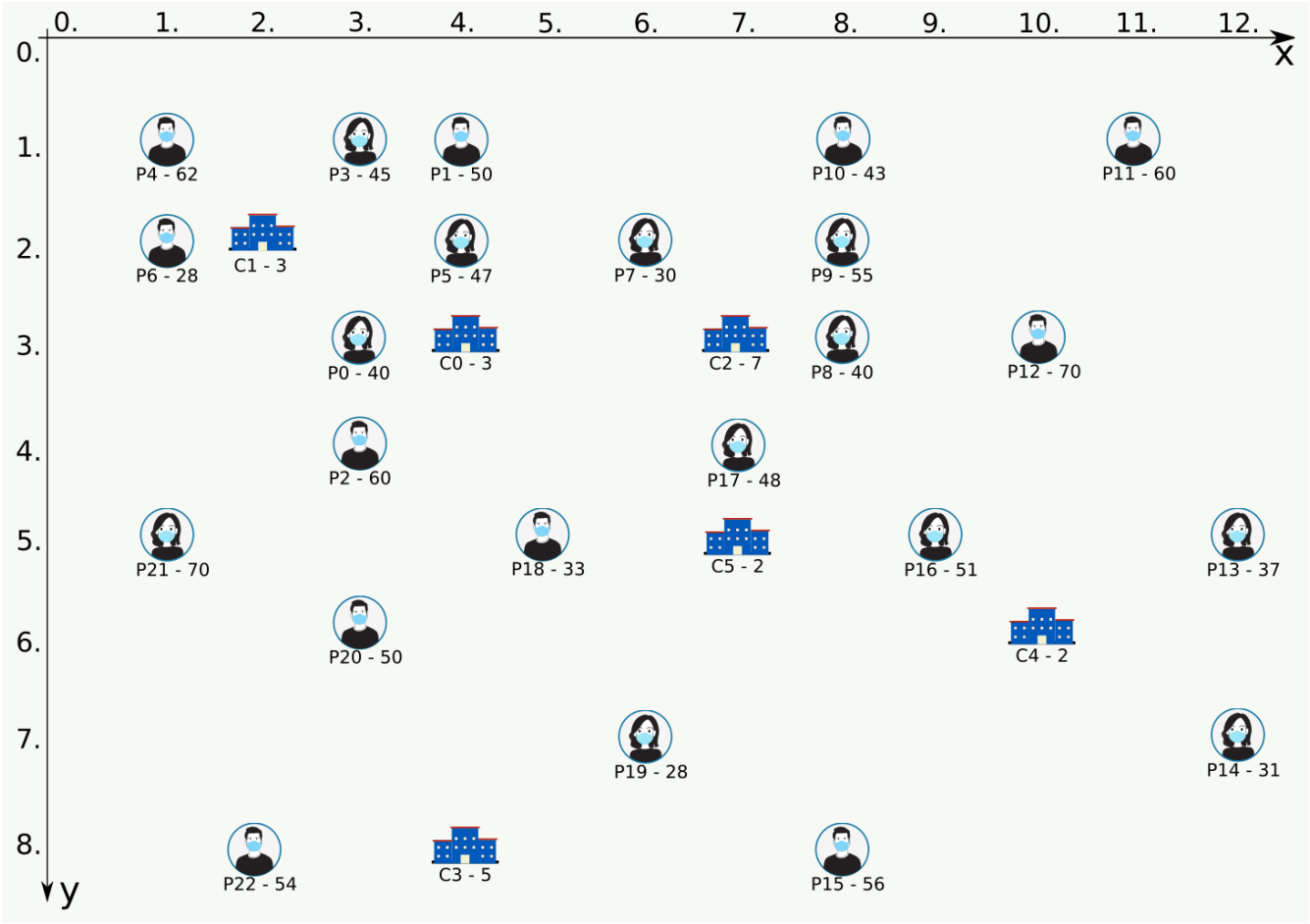

Figura 6: Exemplo de alocação com seis postos de vacinação e vinte e três pessoas inscritas no dia

Exemplo prático da saída

```
0                                   //Posto de vacinação 0
5 1 2                               // Lista das pessoas atendidas no posto 0
1
3 4 21
2
8 0 10 17 9 11 12
3
6 7 14 20 22
4
16 15
5
18 13
```

A representação desse exemplo é dada na Figura 6.

8 Especificação das entregas

Você deve submeter um arquivo compacto (zip ou tar.gz) no formato **MATRICULA_NOME** via Moodle contendo:

- todos os arquivos de código-fonte implementados;
- um arquivo *makefile*[1] **que crie um executável com nome tp01**;
  - **ATENÇÃO:** O makefile é para garantir que o código será compilado da forma como vocês implementaram, evitando erros na compilação. É **essencial** que ao digitar "make" na linha de comando dentro da pasta onde reside o arquivo makefile, o mesmo compile o programa e gere um executável chamado **tp01**.
- sua documentação (arquivo pdf).

Sua documentação deverá ser sucinta e conter não mais do que 5 páginas com o seguinte conteúdo obrigatório:

- Modelagem computacional do problema;
- estruturas de dados e algoritmos utilizados para resolver o problema (pseudo-código da solução impelementada), bem como justificativa para tal escolha. Não transcreva trechos da código-fonte;
- análise de complexidade de tempo assintótica da solução proposta, devidamente justificada.

9 Implementação

9.1 Linguagem, Ambiente e Parâmetros

O seu programa deverá ser implementado na linguagem **C** ou **C++** e deverá fazer uso apenas de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação e/ou que façam uso de bibliotecas que não a padrão não serão aceitos.

O aluno pode implementar seu programa em qualquer ambiente (Windows, Linux, MacOS, etc...), no entanto, deve garantir que seu código compile e rode nas máquinas do DCC (tigre.dcc.ufmg.br ou jaguar.dcc.ufmg.br), pois será neste ambiente que o TP será corrigido. Note que essas máquinas são acessíveis a todos os alunos do DCC com seu login e senha, podendo inclusive ser realizado acesso remoto via ssh. O aluno pode buscar informações no site do CRC (Centro de Recursos Computacionais) do DCC (https://www.crc.dcc.ufmg.br/).

O arquivo da entrada deve ser passado ao seu programa como entrada padrão, através da linha de comando (e.g., *$ ./tp01 < casoTeste01.txt*) e gerar o resultado também na saída padrão (não gerar saída em arquivo).

ATENÇÃO: Não é necessário que o aluno implemente em ambiente Linux. Recomenda-se que o aluno teste seu código nas máquinas previamente especificadas, as quais serão utilizadas para correção do TP, a fim de conferir a funcionalidade, makefile e demais características do código.

[1] `https://pt.wikibooks.org/wiki/Programar_em_C/Makefiles`

9.2 Testes automatizados

A sua implementação passará por um processo de correção automatizado, utilizando além dos casos de testes já disponibilizados, outros exclusivos criados para o processo de correção. O formato da saída de seu programa deve seguir a especificação apresentada nas seções anteriores. Saídas diferentes serão consideradas erro para o programa. O aluno deve certificar-se que seu programa execute corretamente para qualquer entrada válida do problema.

ATENÇÃO: O tempo máximo esperado para execução do programa, dado o tamanho máximo do problema definido em seções anteriores, é de 5 segundos.

9.3 Qualidade do código

Preze pela qualidade do código-fonte, mantendo-o organizado e comentado de modo a facilitar seu entendimento para correção. Caso alguma questão não esteja clara na documentação e no código fonte, a nota do trabalho pode ser penalizada.

10 Critérios para pontuação

A nota final do TP (NF) será composta por dois fatores: fator parcial de implementação (fpi) e fator parcial da documentação (npd). Os critérios adotados para pontuação dos fatores é explicado a seguir.

10.1 Fator parcial de implementação

Serão avaliados quatro aspectos da implementação da solução do problema, conforme a Tabela 1.

Aspecto	**Sigla**	**Valores possíveis**
Compilação no ambiente de correção	co	0 ou 1
Respostas corretas nos casos de teste de correção	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 da solução do problema

O fator parcial de implementação será calculado pela seguinte fórmula:

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

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

10.2 Fator parcial da documentação

Serão avaliados quatro aspectos da documentação entregue pelo aluno, conforme a Tabela 2.

O fator parcial de documentação será calculado pela seguinte fórmula:

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

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

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 de tempo assintótica	at	0 a 100 %

Tabela 2: Aspectos de avaliação da documentação

10.3 Nota final do TP

A nota final do trabalho prático será obtida pela equação a seguir:

$$NF = 10 \times (0,6 \times fpi + 0,4 \times fpd)$$

É importante ressaltar que é obrigatória a entrega do código fonte da solução e documentação. Na ausência de um desses elementos, a nota do trabalho prático será considerada igual a zero, pois não haverá possibilidade de avaliar adequadamente o trabalho realizado.

Assim como em todos os trabalhos dessa disciplina é estritamente proibida a cópia parcial ou integral de código-fontes, seja da Internet ou de colegas. Se for identifcado o plágio, o aluno terá a nota zerada e o professor será informado para que as medidas cabíveis sejam tomadas.

ATENÇÃO: Os alunos que submeterem os TPs com atraso, terão a nota final penalizada em termos percentuais de acordo com a seguinte regra: $2^{d-1}/0,16$ (onde d é a quantidade de dias úteis de atraso na entrega do TP)