TP 01 - Trabalho Prático 01

Algoritmos I

Entrega: 17/05/2022

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 17/05/2022. A especificação do conteúdo do relatório e linguagens de programação aceitas serão detalhadas nas seções subsequentes.

2 Definição do problema

A Lagoa dos Pimpolhos é um ponto turístico importante da cidade de Helo Borizonte. Todos os dias, várias pessoas visitam o local para explorar pontos turísticos, exercitar-se ou simplesmente admirar a paisagem.

Entre as atividades possíveis na Lagoa dos Pimpolhos estão os passeios de bicicleta. Visitantes interessados podem alugar bicicletas em certos pontos de vendas e apreciar um belo passeio ao longo da extensão de 18 quilômetros da lagoa.

Contudo, essa estrutura possui alguns problemas logísticos. O aluguel das bicicletas é cobrado por hora e, naturalmente, espera-se que o visitante retorne a bicicleta ao seu local de origem ao final do passeio. Naturalmente, isso pode ser um problema para visitantes que não estejam em forma, ou simplesmente desejem parar em algum ponto do percurso. A situação piora ainda mais em dias de muito movimento, quando certos trechos são interditados por carros e pedestres.

Para contornar esse problema e incentivar o ciclismo, a prefeitura de Helo Borizonte decidiu implementar o programa Mais Bicicletas. O programa fornece uma frota com dezenas de bicicletas espalhadas por diversos pontos estratégicos da Lagoa, de forma que o visitante interessado pode simplesmente escolher uma bicicleta em sua proximidade, passear pela distância desejada e deixar a bicicleta em um ponto de controle próximo ao final do passeio. Pagamentos serão efetuados em terminais eletrônicos nos pontos de controle, e as bicicletas contam com diversos dispositivos de segurança para impedir furtos e vandalismo.

O aplicativo *+Bikes* é parte integral do projeto. Através desse aplicativo, visitantes podem acompanhar em tempo real o estoque de cada ponto de controle, podendo assim escolher a bicicleta nas proximidades que mais se adeque ao seu porte físico. Note que vários visitantes podem estar interessados na mesma bicicleta, de forma que é necessário implementar um sistema de alocação que atribua cada bicicleta a um visitante, de acordo com suas preferências e sem esquecer a distância do visitante ao ponto de controle onde a bicicleta está localizada.

Como integrante júnior do time de desenvolvimento alocado ao programa, você ficou responsável por projetar um protótipo inicial do sistema de alocação. Nomeadamente, em um cenário em que hajam N bicicletas disponíveis no momento e N visitantes aguardando, seu programa deve ser capaz de alocar de forma *justa* uma bicicleta para cada visitante. Em outras palavras, seu programa deverá:

- Receber um mapa da lagoa, cuja formatação será descrita em detalhe na seção 5;
- Calcular as distâncias entre visitantes e bicicletas, de acordo com o mapa fornecido;
- Receber uma lista de preferências de cada visitante indicando, para cada bicicleta disponível, o grau de preferência do visitante por aquela bicicleta. Neste primeiro estágio, você pode assumir que a lista de preferências fornecida será sempre completa, ou seja, uma matriz NxN contemplando todos os pares *(visitante, bicicleta)*.
- Alocar visitantes a bicicletas, de forma que cada bicicleta esteja associada a exatamente um visitante, e vice-versa;
- Respeitar a seguinte *condição de estabilidade*: Se uma pessoa p_1 foi alocada a uma bicicleta b_1, **ambos** os seguintes critérios deve ser satisfeitos:

  - Se há uma bicicleta b_2 que p_1 considere preferível em relação a b_1, então há uma pessoa p_2 mais próxima de b_2 para a qual b_2 foi alocada;

  - Se há uma pessoa p_2 que está mais próxima de b_1 do que p_1, então p_2 foi alocada para alguma bicicleta b_2 que ela considera preferível a b_1.

- Um visitante não se importa em caminhar um pouco mais para obter uma bicicleta melhor. Contudo, o visitante sempre desejará obter a melhor bicicleta disponível, de forma que, entre a distância e a ordem de preferência, deve-se sempre priorizar a ordem de preferência;
- Regra de desempate por preferência: Se um visitante possui igual preferência por duas bicicletas, então é priorizada aquela com o menor ID (Posição sequencial no arquivo de entrada).
- Regra de desempate por distância: Se dois visitantes estão equidistantes da mesma bicicleta, então possui prioridade o visitante com o menor ID (Posição sequencial no arquivo de entrada).

Note que casos adicionais, como situações em que existem mais visitantes do que bicicletas, serão tratados pelo seu time em um momento posterior, de forma que você não precisa se preocupar com eles no momento. Da mesma forma, não é necessário se preocupar com outros aspectos do sistema, como os destinatários dos visitantes ou métodos para lidar com bicicletas que estão em movimento, mas prestes a chegar a seu destinatário.

3 O que fazer?

O objetivo deste trabalho é alocar visitantes a bicicletas, satisfazendo alguns critérios. Considere um processo de alocação no qual n visitantes devem ser alocados a n bicicletas, distribuídas entre diversos pontos de controle. Ao acessar o aplicativo, visitantes fornecem uma lista de preferências que, nesse primeiro momento, contempla todas as bicicletas disponíveis. Preferências são dadas na forma de um número inteiro, de forma que, se uma bicicleta b_1 apresenta $pref(b_1) > pref(b_2)$ em relação a outra bicicleta b_2, dizemos que o visitante prefere b_1 a b_2. Um visitante não se importa em caminhar um pouco mais para obter uma bicicleta melhor (segundo suas preferências).

Por outro lado, a prefeitura deseja priorizar, quando possível, os moradores locais, de forma que moradores de um bairro sejam alocados, preferencialmente, a bicicletas de seu próprio bairro. Essa estratégia de alocação foi proposta pelo Departamento de Estatística, com o obetivo de evitar certo problemas de logística. Nesta primeira implementação, isso significa que também devemos levar em conta a distância dos visitantes até as bicicletas.

A condição essencial para uma alocação é que ela seja *justa*. Dizemos que uma alocação é justa se, dada qualquer alocação de uma pessoa p_1 a uma bicicleta b_1, os seguintes critérios são satisfeitos:

- Se há uma bicicleta b_2 que p_1 considere preferível em relação a b_1, então há uma pessoa p_2 mais próxima de b_2 para a qual b_2 foi alocada;
- Se há uma pessoa p_2 que está mais próxima de b_1 do que p_1, então p_2 foi alocada para alguma bicicleta b_2 que ela considera preferível a b_1.

Em outras palavras, queremos respeitar as preferências dos usuários ao máximo possível, porém sem fazê-los andar em excesso para buscar sua bicicleta. Note que, utilizando essa regra de implementação, alguns visitantes poderão ser injustiçados em raras situações. Essas situações serão corrigidas em iterações futuras do projeto e, portanto, você não deve se preocupar com elas no momento.

Para possibilitar os cálculos da distância de visitantes a bicicletas, o aplicativo dispõe de um mapa (desenvolvido por outro membro da equipe) que descreve a lagoa como uma grade NxM contendo pessoas, bicicletas e obstáculos que não podem ser atravessados. Suponha que sempre haja um caminho de qualquer pessoa a qualquer bicicleta. Uma vez que estas localizações estão em constante mudança, está também sob a sua responsabilidade calcular as distâncias entre os visitantes e cada bicicleta.

Para simplificar os cálculos, suponha que uma pessoa pode mover-se apenas horizontalmente ou verticalmente. Isto é, um visitante que está na célula $mapa[i][j]$ pode se deslocar para as células $mapa[i][j+1]$, $mapa[i][j-1]$, $mapa[i+1][j]$ e $mapa[i-1][j]$, todas as quais estão a uma unidade de distância (Digamos metros) da célula original. Um visitante pode se deslocar por células vazias ou casas que contenham bicicletas e outros visitantes, mas não pode se deslocar por células que contém obstáculos. É importante também cuidar para que o visitante não saia do mapa (Por exemplo, a célula $mapa[-1][0]$ está fora do mapa, e não sabemos o que ela pode conter).

A distância percorrida no mapa de um ponto A a um ponto B é então dada pelo total de movimentos horizontais e verticais necessários para um visitante se deslocar de A até B, isto é, o caminho mais curto de A até B seguindo as diretrizes especificadas no parágrafo anterior. Por exemplo, na Figura 1, o visitante p_1 está a uma distância de 5 metros da bicicleta b_1, devendo se deslocar 2 posições para a direita e então 3 posições para baixo para alcançá-la.

Um exemplo de alocação injusta é demonstrado pela Figura 2, onde o visitante p_1 é alocado à bicicleta b_1, sendo que a bicicleta b_2 está mais próxima e é considerada preferível por p_1.

4 Exemplo do problema

A Figura 3 ilustra uma instância de alocação justa para um caso de teste. Existem cinco bicicletas (B1,B2,B3,B4,B5), que devem ser alocadas para cinco visitantes (P1,P2,P3,P4,P5). A ordem de preferências é descrita em separado, na Tabela 1.

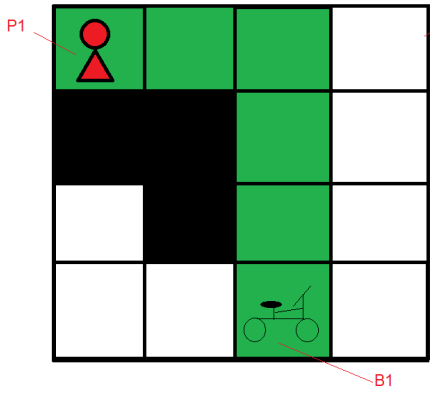

Figura 1: A bicicleta b_1 está a 5 unidades de distância (Digamos metros) de p_1. Note que os quadrados escuros indicam obstáculos, pelos quais os visitantes não podem passar.

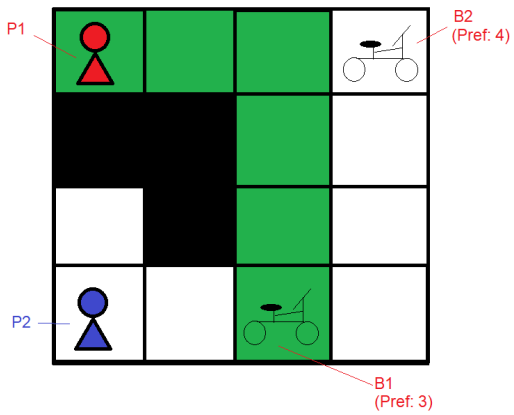

Figura 2: Exemplo de alocação injusta, uma vez que p_1 prefere a bicicleta b_2 e b_2, por sua vez, não possui nenhum visitante mais próximo que p_1.

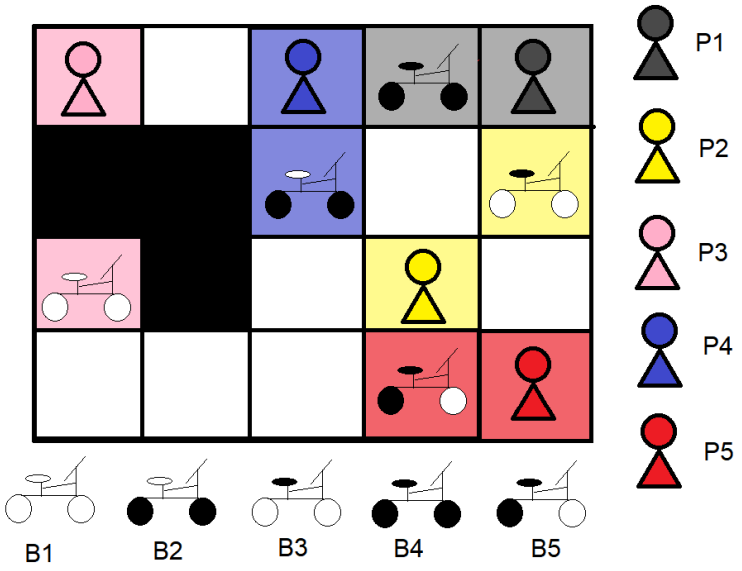

Figura 3: Exemplo de alocação justa, dadas as preferências especificadas na Tabela 1. Simbolizamos a alocação pintando com a mesma cor as regiões do visitante e da bicicleta pareados.

Ordem de preferências por pessoa (Figura 3)

```
P1 - {B4,B3,B5,B2,B1}          Ordem de preferências para primeira pessoa
P2 - {B3,B1,B2,B5,B4}
P3 - {B1,B2,B3,B4,B5}
P4 - {B4,B2,B1,B3,B5}
P5 - {B3,B2,B5,B4,B1}
```

Ordem de proximidade por bicicleta (Figura 3)

```
B1 - {P2,P5,P4,P1,P3}          Ordem de proximidade para a primeira bicicleta
B2 - {P4,P2,P1,P3,P5}
B3 - {P1,P2,P5,P4,P3}
B4 - {P1,P4,P2,P3,P5}
B5 - {P2,P5,P1,P4,P3}
```

Podemos observar, pela Figura 3, que a alocação não quebra o critério de estabilidade para nenhum par (bicicleta,visitante). Isto é, não existe um par (b, v) tal que a bicicleta b esteja mais próxima de v do que de seu par e v prefira b ao seu par.

Vale lembrar dos critérios de desempate. Por exemplo, embora P1 e P4 estejam equidistantes da bicicleta B4, a visitante P1 possui prioridade por ter menor ID e, portanto, o par (P4,B4) não é considerado "injusto".

5 Arquivos de entrada

O problema terá como entrada um arquivo contendo o mapa descrevendo a região, além de uma matriz de preferências onde cada linha descreve as preferências de um visitante. O arquivo está organizado da seguinte forma:

A primeira linha contém um valor inteiro indicando o número de visitantes e bicicletas, $1 \leq V \leq 10$. A segunda linha, por sua vez, contém dois valores inteiros separados por um espaço em branco, $1 \leq N, M \leq 1000$, indicando as dimensões do mapa (N linhas, M colunas).

As próximas N linhas descrevem o mapa da região. Cada linha contém uma string de texto com exatamente M caracteres, podendo conter os seguintes caracteres:

- '*', indicando um trecho atravessável do mapa. Visitantes podem transitar por essa região sem problemas.
- '-', indicando um obstáculo no mapa. Visitantes não podem atravessar essa região, devendo contorná-la de alguma forma.
- Letras minúsculas de 'a' até 'j', indicando um visitante, onde o visitante 'a' possui ID = 1, o visitante 'b' possui ID = 2 e assim por diante. Suponha que um visitante possa atravessar espaços ocupados por outros visitantes.
- Números, de '0' até '9', indicando uma bicicleta. A bicicleta '0' possui maior prioridade (ID = 1), enquanto a bicicleta '9' possui menor prioridade (ID = 10). Suponha que visitantes possam atravessar espaços ocupados por uma bicicleta.

Por fim, as próximas V linhas descrevem as preferências dos V visitantes. A i-ésima linha descreve as preferências do i-ésimo visitante (Lembrando que o visitante 'a' é o primeiro visitante, 'b' o segundo e assim por diante). Cada linha i, $1 \leq i \leq V$, contém V valores p_{ij}, indicando que o visitante i atribuiu um grau de preferência de p_{ij} à bicicleta j, $1 \leq j \leq V$.

6 Saída

A saída deve ser impressa na tela (*stdout*) e seguir o seguinte padrão: Serão impressas V linhas, onde cada linha contém um par (l_i,n_i): Uma letra l_i, de 'a' até 'j', seguida por um único espaço em branco e um número n_i, de '0' até '9', indicando que o visitante l_i foi alocado para a bicicleta n_i.

Os pares deverão ser impressos por ordem de identificação do visitante. Isto é, primeiro será impresso o par ('a',n_1), em seguida o par ('b',n_2) e assim por diante. Finalmente, note que a alocação descrita pelos V pares impressos deve descrever uma alocação justa, conforme especificado nas seções anteriores.

A seção seguinte ilustra com alguns exemplos a formatação utilizada na entrada e a formatação esperada para a saída.

7 Exemplo Prático de Entrada e Saída

Este primeiro exemplo ilustra uma alocação justa para a situação descrita na Figura 2.

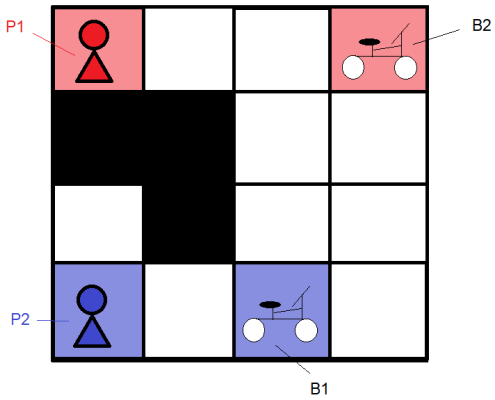

Figura 4: Alocação adequada para o exemplo descrito na figura 2

Arquivo de entrada

```
2                                   // <Número de visitantes e bicicletas>
4 4                                 // <Número de linhas> <Número de colunas>
a**1                                // <Mapa da orla>
--**                                // <Mapa da orla>
*-**                                // <Mapa da orla>
b*0*                                // <Mapa da orla>
3 4                                 // <Preferencias do primeiro visitante>
4 5                                 // <Preferencias do segundo visitante>
```

Saída esperada

```
a 1                                 // <Visitante 'a' pareado com a bicicleta 1>
b 0                                 // <Visitante 'b' pareado com a bicicleta 0>
```

A Figura 4 ilustra a alocação justa esperada. O par (a,1) é justo, pois 'a' prefere a bicicleta 1 à bicicleta 0. O par (b,0), por sua vez, é justo porque, embora 'b' prefira a bicicleta 1, 'b' está mais distante de 1 do que seu par, 'a'.

O próximo exemplo mostra como ficariam a entrada e a saída correspondentes ao pareamento descrito na Figura 3 (Sem o acréscimo de comentários dessa vez).

Arquivo de entrada

```
5
4 5
c*d3a
--1*2
0-*b*
***4e
1 2 4 5 3
4 3 5 1 2
5 4 3 2 1
3 4 2 5 1
1 4 5 2 3
```

Saída esperada

```
a 3
b 2
c 0
d 1
e 4
```

Finalmente, o próximo exemplo ilustra algumas situações interessantes, como visitantes atribuindo o mesmo grau de preferência a bicicletas diferentes (Lembrando, nesse caso priorizamos o menor ID).

Arquivo de entrada

```
8
7 7
*b*a*h*
*----*2
*----1c
fg---*5
*0---*d
4----*e
**637**
9 4 2 5 3 3 4 2
10 9 10 9 9 9 9 1
8 5 4 9 3 2 1 7
11 3 3 4 9 8 3 5
10 7 7 7 8 9 4 9
1 3 4 5 3 4 5 1
3 3 2 4 8 9 1 2
1 1 1 1 1 1 1 1
```

Saída esperada

```
a 6
b 0
c 1
d 5
e 7
f 3
g 4
h 2
```

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

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

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.

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.

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

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.

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)