



## **Relatório do Projeto**

### **Parte 1**

Nome do Integrante	RA
Guilherme Ferreira Martins Ramos	10418373
Felipe Viviani Schulze	10417996

## **Relatório**

### **➤ Nome do aplicativo**

LudoRadar

### **➤ ODS e por que se encaixa?**

O projeto de recomendação de jogos pode se encaixar na ODS de Saúde e Bem-Estar, pois auxilia o usuário a encontrar opções de entretenimento alinhadas aos seus interesses, evitando frustrações e promovendo momentos de lazer mais satisfatórios. O acesso facilitado a jogos adequados contribui para a redução do estresse, ansiedade e principalmente perda de tempo, todo mundo já perdeu tempo procurando um jogo que no fim não lhe interessava ou até teve que jogar um jogo para descobrir que não era o que esperava. Além disso, o estímulo a atividades de lazer personalizadas apoia o equilíbrio entre estudo, trabalho e descanso, fatores essenciais para o bem-estar integral. Essa iniciativa também pode favorecer a socialização em jogos cooperativos, ampliando laços sociais e prevenindo o isolamento, aspectos importantes da saúde psicológica.

### **➤ Códigos Fonte (parte relevante, estarão completos no diretório do Github)**



## comparacoes.py

```
1 import jogos as jogosPy
2 from grafoMatriz import GrafoRot as grafo
3
4 jogos = list(vars(jogosPy).items())
5 comparacoes = grafo(0)
6
7 for i in range(8, len(jogos)):
8     comparacoes.insererV(jogos[i][1][1], jogos[i][1][0])
9
10 for i in range(8, len(jogos)):
11     for j in range(8, i):
12         peso = len(jogos[i][1][2] & jogos[j][1][2]) # soma 1 pra cada gênero em comum
13         peso += len(jogos[i][1][3] & jogos[j][1][3]) # soma 1 pra cada tema em comum
14         peso += len(jogos[i][1][4] & jogos[j][1][4]) # soma 1 pra cada qtde de jogador em comum
15         comparacoes.insererA(i-8, j-8, peso)
16
17 comparacoes.makeFileFromGraph("grafo.txt")
```



## grafoMatriz.py

```
9 class GrafoRot:
10     TAM_MAX_DEFAULT = 100 # qtde de vértices máxima default
11     # construtor da classe grafo
12     def __init__(self, n = TAM_MAX_DEFAULT):
13         self.vertices = n # número de vértices
14         self.arestas = 0 # número de arestas
15         # matriz de adjacência
16         self.adj = [[INF for i in range(n)] for j in range(n)]
17         # lista de rótulos
18         self.rot = [("", 0) for i in range(n)]
19
20     # Insere um vértice no Grafo tal que
21     # p é o peso no vértice e r é o rótulo do vértice
22     def insereV(self, p, r):
23         for i in range(self.vertices):
24             # Adiciona o vértice em todas as linhas
25             self.adj[i].append(INF)
26         self.vertices += 1 # atualiza o número de vértices
27         # Adiciona uma linha a mais
28         self.adj.append([INF for i in range(self.vertices+1)])
29         self.rot.append([r, p])
30
31     # Remove um vértice v do grafo
32     """ Explicação do algoritmo utilizado
33
34     0 1 2
35     0 A B C
36     1 D E F
37     2 G H I
38
39     remover o 1:
40     vértices removidos: B, D, E, F, H
41     0 1 2
42     0 A C
43     1
44     2 G I
45     (Espaços vazios para representar que seus valores não importam mais)
46
```



# UNIVERSIDADE PRESBITERIANA MACKENZIE

## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



```
58 # v é o índice do vértice que será removido
59 def removeV(self, v):
60     for i in range(self.vertices):
61         if i != v:
62             self.rot[i - (i > v)] = self.rot[i]
63             for j in range(self.vertices):
64                 # Copia os elementos para preencher as posições vagas do vértice removido na matriz
65                 if j != v:
66                     self.adj[i - (i > v)][j - (j > v)] = self.adj[i][j]
67
68     # Atualiza o número de vértices, remove a última linha e as últimas posições de cada linha da matriz
69     self.vertices -= 1
70     self.adj.pop()
71     self.rot.pop()
72     for i in range(self.vertices):
73         self.adj[i].pop()
74
75     # Insere uma aresta no Grafo tal que
76     # v é adjacente a w, com peso p
77     def insereA(self, v, w, p):
78         if self.adj[v][w] == INF:
79             self.adj[v][w] = p
80             self.adj[w][v] = p
81             self.arestas += 1 # atualiza qtd arestas
82
83     # remove uma aresta v->w do Grafo
84     def removeA(self, v, w):
85         # testa se temos a aresta
86         if self.adj[v][w] != INF:
87             self.adj[v][w] = INF
88             self.adj[w][v] = INF
89             self.arestas -= 1 # atualiza qtd arestas
```



# UNIVERSIDADE PRESBITERIANA MACKENZIE

## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



```
94     def show(self):
95         print(f"\n n: {self.vertices:2d} ", end="")
96         print(f"m: {self.arestas:2d}\n")
97
98         for i in range(self.vertices):
99             for w in range(self.vertices):
100                 if self.adj[i][w] == INF:
101                     print(f"Adj[{i:2d},{w:2d}] = ∞ | ", end="")
102                 else:
103                     print(f"Adj[{i:2d},{w:2d}] = {self.adj[i][w]} | ", end="")
104             print("\n")
105         print("\nfim da impressao do grafo." )
106
107
108     # Apresenta o Grafo contendo
109     # número de vértices, arestas
110     # e a matriz de adjacência obtida
111     # Apresentando apenas os valores 0 ou 1
112     def showMin(self):
113         print(f"\n n: {self.vertices:2d} ", end="")
114         print(f"m: {self.arestas:2d}\n")
115
116         for i in range(self.vertices):
117             for w in range(self.vertices):
118                 if self.adj[i][w] == INF:
119                     print("∞ ", end="")
120                 else:
121                     print(str(self.adj[i][w]) + " ", end="")
122             print('\n')
123         print("\nfim da impressao do grafo." )
124
125     ##### Métodos implementados para aula de Teoria dos Grafos
126
127     # Retorna o grau do vértice v
128     def degree(self, v):
129         degree = 0
130         for i in range(self.vertices):
```



# UNIVERSIDADE PRESBITERIANA MACKENZIE

## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



```
135     def makeFileFromGraph(self, arq):
136         with open(arq, "w") as f:
137             f.write(str(3) + "\n") # insere o tipo do grafo
138             f.write(str(self.vertices) + "\n") # insere o núm de vértices
139             for i in range(self.vertices): # insere os n vértices
140                 f.write(str(i) + " " + self.rot[i][0] + " " + str(self.rot[i][1]) + "\n")
141
142             f.write(str(self.arestas) + "\n") # insere o núm de arestas
143             for i in range(self.vertices): # insere todas as arestas caso existam
144                 for j in range(self.vertices):
145                     if self.adj[i][j] != INF and i > j:
146                         f.write(str(i) + " " + str(j) + " " + str(self.adj[i][j]) + "\n")
147
148         # Lê um arquivo a e monta o grafo presente nele
149         def makeGraphFromFile(self, arq):
150             # abre o arquivo e o lê linha por linha
151             with open(arq, "r") as f:
152
153                 # primeiro lemos as duas primeiras linhas para pegar o tipo e o número de vértices do grafo
154                 f.readline() # tipo (sempre será 3 nessa aplicação)
155                 line = f.readline() # número de vértices
156                 self.__init__(0) # inicia a matriz vazia para inserir os vértices depois
157
158                 for i in range(int(line)): # para as i linhas seguintes, insere os vértices
159                     line = f.readline().split() # [i, "rótulo", "peso"]
160                     rot = ""
161                     for i in range(len(line)-2):
162                         rot += line[i+1] + " "
163                     rot = rot[:-1] # remove o espaço extra
164                     self.insereV(line[-1], rot)
165
166
167                 line = f.readline() # lê o número de arestas
168
169                 # Percorre o resto do arquivo, inserindo as arestas de cada par de vértices
170                 for i in range(int(line)):
171                     line = f.readline().split()
172                     v = int(line[0])
```



# UNIVERSIDADE PRESBITERIANA MACKENZIE

## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



```
177 # Verifica se o grafo é completo
178 def isComplete(self):
179     for i in range(self.vertices):
180         for j in range(self.vertices):
181             if self.adj[i][j] == INF:
182                 return 0 # Não é completo
183     return 1 # É completo
184
185 def Floyd(self):
186     D = self.adj
187     for i in range(self.vertices):
188         D[i][i] = 0
189
190     R = [[(j+1) for i in range(self.vertices)] for j in range(self.vertices)]
191     for i in range(self.vertices):
192         for j in range(self.vertices):
193             if self.adj[i][j] < INF:
194                 R[i][j] = (j+1)
195             else:
196                 R[i][j] = 0
197
198     for k in range(self.vertices):
199         for i in range(self.vertices):
200             for j in range(self.vertices):
201                 if i != j and D[i][k] + D[k][j] < D[i][j]:
202                     D[i][j] = D[i][k] + D[k][j]
203                     R[i][j] = R[i][k]
204
205     print("D: ")
206     for i in range(self.vertices):
207         for j in range(self.vertices):
208             print(D[i][j], end=" ")
209         print()
210
211     print("R: ")
212     for i in range(self.vertices):
213         for j in range(self.vertices):
214             print(R[i][j], end=" ")
```



```
204
205     print("D: ")
206     for i in range(self.vertices):
207         for j in range(self.vertices):
208             print(D[i][j], end=" ")
209         print()
210
211     print("R: ")
212     for i in range(self.vertices):
213         for j in range(self.vertices):
214             print(R[i][j], end=" ")
215         print()
216
217     # Verifica se o gravo é conexo ou desconexo
218     def eConexo(self):
219         # Apenas precisamos ver se 1 vértice está conectado a todos os outros
220         # Usamos o vértice 0 como inicial
221         visitados = [0]
222
223         for v in visitados:
224             for i in range(self.vertices):
225                 if self.adj[v][i] == 1 and i not in visitados:
226                     visitados.append(i)
227
228         if len(visitados) != self.vertices:
229             return 1
230         return 0
```





## jogos.py

```
1  minecraft = ["Minecraft", 83,  
2      [{"simulador", "aventura"},  
3      {"fantasia", "sobrevivencia", "sandbox", "infantil", "mundo aberto"},  
4      {"single", "multi", "coop"}]  
5  
6  silksong = ["Hollow Knight: Silksong", 90,  
7      [{"plataforma", "aventura", "indie"},  
8      {"acao", "fantasia"},  
9      {"single"}]  
10  
11 hollowKnight = ["Hollow Knight", 91,  
12     [{"plataforma", "aventura", "indie"},  
13     {"acao", "fantasia"},  
14     {"single"}]  
15  
16 babaIsYou = ["Baba Is You", 78,  
17     [{"puzzle", "estrategia", "indie"},  
18     {"fantasia"},  
19     {"single"}]  
20  
21 redDead2 = ["Red Dead Redemption 2", 89,  
22     [{"tiro", "rpg", "aventura"},  
23     {"acao", "drama", "mundo aberto", "ocidental"},  
24     {"single", "multi", "coop"}]  
25  
26 skyrim = ["The Elder Scrolls V: Skyrim", 86,  
27     [{"rpg", "aventura"},  
28     {"fantasia", "furtivo", "sandbox", "mundo aberto", "acao"},  
29     {"single"}]  
30  
31 gtaV = ["Grand Theft Auto V", 85,  
32     [{"tiro", "aventura", "corrida"},  
33     {"acao", "comedia", "sandbox", "mundo aberto"},  
34     {"single", "multi", "coop"}]  
35  
36 godOfWar = ["God of War", 91,  
37     [{"puzzle", "aventura", "bater"},  
38     {"acao", "fantasia", "historia"},  
39     {"single"}]
```

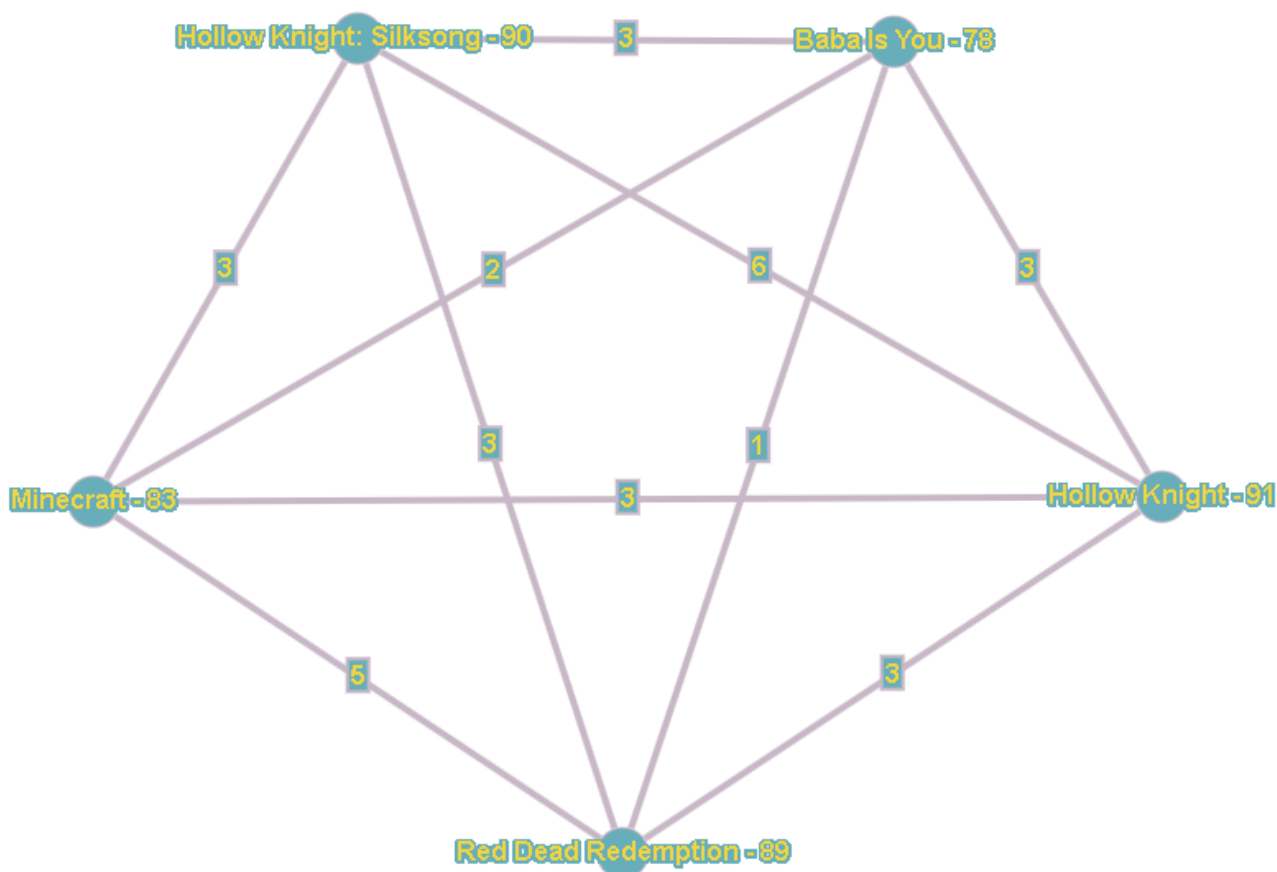


➤ Execuições do menu

**Grafo com 5 jogos (arquivo: grafo\_5.txt)**

**grafo\_5.txt (representado no graphonline):**

Vértices: "Nome do jogo - nota"





UNIVERSIDADE PRESBITERIANA MACKENZIE  
Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira  
Teoria dos Grafos



## Telas:

```
Menu
0: Ler dados do arquivo grafo.txt
1: Gravar dados no arquivo grafo.txt
2: Inserir vértice
3: Inserir aresta
4: Remover vértice
5: Remover aresta
6: Mostrar conteúdo do arquivo
7: Mostrar grafo
8: Apresentar conexidade do grafo
9: Encerrar o programa
Escolha uma opção: 7
Qual modo do grafo deseja (0 - Completo, 1 - Simples): 0

n: 5 m: 10

Adj[ 0, 0] = ∞ | Adj[ 0, 1] = 3 | Adj[ 0, 2] = 3 | Adj[ 0, 3] = 2 | Adj[ 0, 4] = 5 |
Adj[ 1, 0] = 3 | Adj[ 1, 1] = ∞ | Adj[ 1, 2] = 6 | Adj[ 1, 3] = 3 | Adj[ 1, 4] = 3 |
Adj[ 2, 0] = 3 | Adj[ 2, 1] = 6 | Adj[ 2, 2] = ∞ | Adj[ 2, 3] = 3 | Adj[ 2, 4] = 3 |
Adj[ 3, 0] = 2 | Adj[ 3, 1] = 3 | Adj[ 3, 2] = 3 | Adj[ 3, 3] = ∞ | Adj[ 3, 4] = 1 |
Adj[ 4, 0] = 5 | Adj[ 4, 1] = 3 | Adj[ 4, 2] = 3 | Adj[ 4, 3] = 1 | Adj[ 4, 4] = ∞ |

fim da impressao do grafo.
```



**UNIVERSIDADE PRESBITERIANA MACKENZIE**  
**Faculdade de Computação e Informática**

Prof. Dr. Ivan Carlos Alcântara de Oliveira  
Teoria dos Grafos



```
Escolha uma opção: 6
--- Conteúdo de grafo.txt ---
3
5
0 Minecraft 83
1 Hollow Knight: Silksong 90
2 Hollow Knight 91
3 Baba Is You 78
4 Red Dead Redemption 2 89
10
1 0 3
2 0 3
2 1 6
3 0 2
3 1 3
3 2 3
4 0 5
4 1 3
4 2 3
4 3 1
--- fim ---
```

```
Menu
0: Ler dados do arquivo grafo.txt
1: Gravar dados no arquivo grafo.txt
2: Inserir vértice
3: Inserir aresta
4: Remover vértice
5: Remover aresta
6: Mostrar conteúdo do arquivo
7: Mostrar grafo
8: Apresentar conexidade do grafo
9: Encerrar o programa
Escolha uma opção: 8
Conexidade do grafo: Desconexo
```



**UNIVERSIDADE PRESBITERIANA MACKENZIE**  
**Faculdade de Computação e Informática**

Prof. Dr. Ivan Carlos Alcântara de Oliveira  
Teoria dos Grafos



```
Menu
0: Ler dados do arquivo grafo.txt
1: Gravar dados no arquivo grafo.txt
2: Inserir vértice
3: Inserir aresta
4: Remover vértice
5: Remover aresta
6: Mostrar conteúdo do arquivo
7: Mostrar grafo
8: Apresentar conexidade do grafo
9: Encerrar o programa
Escolha uma opção: 7
Qual modo do grafo deseja (0 - Completo, 1 - Simples): 1

n: 5 m: 10

∞ 3 3 2 5
3 ∞ 6 3 3
3 6 ∞ 3 3
2 3 3 ∞ 1
5 3 3 1 ∞

fim da impressao do grafo.
```

Foram utilizados somente 5 jogos porque a representação com 60 jogos/vértices é extremamente poluída e ilegível já que conta com mais de 1700 arestas, para fins de expor o projeto, fizemos esse subgrafo.



# UNIVERSIDADE PRESBITERIANA MACKENZIE

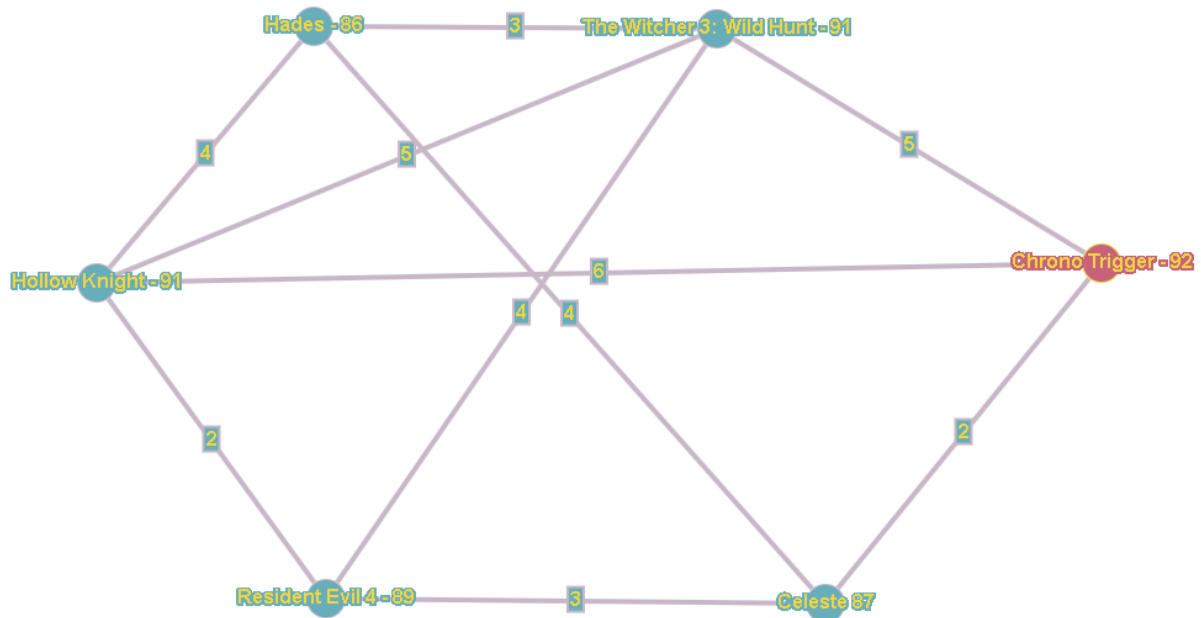
## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira  
Teoria dos Grafos



Segunda leva de execuções mostrando as outras funcionalidades do menu para um grafo de 6 jogos/vértices.

**grafo\_6.txt:**



```
Menu
0: Ler dados do arquivo grafo.txt
1: Gravar dados no arquivo grafo.txt
2: Inserir vértice
3: Inserir aresta
4: Remover vértice
5: Remover aresta
6: Mostrar conteúdo do arquivo
7: Mostrar grafo
8: Apresentar conexidade do grafo
9: Encerrar o programa
Escolha uma opção: 0
grafo.txt lido e grafo montado.
```

```
Menu
0: Ler dados do arquivo grafo.txt
1: Gravar dados no arquivo grafo.txt
2: Inserir vértice
3: Inserir aresta
4: Remover vértice
5: Remover aresta
6: Mostrar conteúdo do arquivo
7: Mostrar grafo
8: Apresentar conexidade do grafo
9: Encerrar o programa
Escolha uma opção: 1
grafo gravado em grafo.txt.
```



# UNIVERSIDADE PRESBITERIANA MACKENZIE

## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira  
Teoria dos Grafos



```
Menu
0: Ler dados do arquivo grafo.txt
1: Gravar dados no arquivo grafo.txt
2: Inserir vértice
3: Inserir aresta
4: Remover vértice
5: Remover aresta
6: Mostrar conteúdo do arquivo
7: Mostrar grafo
8: Apresentar conexidade do grafo
9: Encerrar o programa
Escolha uma opção: 2
Insira a nota do jogo (peso do vértice): 67
Insira o nome do jogo (rótulo do vértice): Exemplo
vértice inserido.
```

```
Menu
0: Ler dados do arquivo grafo.txt
1: Gravar dados no arquivo grafo.txt
2: Inserir vértice
3: Inserir aresta
4: Remover vértice
5: Remover aresta
6: Mostrar conteúdo do arquivo
7: Mostrar grafo
8: Apresentar conexidade do grafo
9: Encerrar o programa
Escolha uma opção: 4
Insira o índice do vértice a remover: 8
vértice removido.
```

```
Escolha uma opção: 6
--- Conteúdo de grafo.txt ---
3
6
0 Hollow Knight 91
1 Hades 86
2 The Witcher 3: Wild Hunt 91
3 Resident Evil 4 89
4 Chrono Trigger 92
5 Celeste 87
10
1 0 4
2 0 5
2 1 3
3 0 2
3 2 4
4 0 6
4 2 5
5 1 4
5 3 3
5 4 2
--- fim ---
```



# UNIVERSIDADE PRESBITERIANA MACKENZIE

## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira  
Teoria dos Grafos



57. Encerra o programa

Escolha uma opção: 7

Qual modo do grafo deseja (0 - Completo, 1 - Simples): 0

n: 6 m: 10

Adj[ 0, 0] = ∞ | Adj[ 0, 1] = 4 | Adj[ 0, 2] = 5 | Adj[ 0, 3] = 2 | Adj[ 0, 4] = 6 | Adj[ 0, 5] = ∞ |  
Adj[ 1, 0] = 4 | Adj[ 1, 1] = ∞ | Adj[ 1, 2] = 3 | Adj[ 1, 3] = ∞ | Adj[ 1, 4] = ∞ | Adj[ 1, 5] = 4 |  
Adj[ 2, 0] = 5 | Adj[ 2, 1] = 3 | Adj[ 2, 2] = ∞ | Adj[ 2, 3] = 4 | Adj[ 2, 4] = 5 | Adj[ 2, 5] = ∞ |  
Adj[ 3, 0] = 2 | Adj[ 3, 1] = ∞ | Adj[ 3, 2] = 4 | Adj[ 3, 3] = ∞ | Adj[ 3, 4] = ∞ | Adj[ 3, 5] = 3 |  
Adj[ 4, 0] = 6 | Adj[ 4, 1] = ∞ | Adj[ 4, 2] = 5 | Adj[ 4, 3] = ∞ | Adj[ 4, 4] = ∞ | Adj[ 4, 5] = 2 |  
Adj[ 5, 0] = ∞ | Adj[ 5, 1] = 4 | Adj[ 5, 2] = ∞ | Adj[ 5, 3] = 3 | Adj[ 5, 4] = 2 | Adj[ 5, 5] = ∞ |

fim da impressao do grafo.

Escolha uma opção: 7

Qual modo do grafo deseja (0 - Completo, 1 - Simples): 1

n: 6 m: 10

∞ 4 5 2 6 ∞

4 ∞ 3 ∞ ∞ 4

5 3 ∞ 4 5 ∞

2 ∞ 4 ∞ ∞ 3

6 ∞ 5 ∞ ∞ 2

∞ 4 ∞ 3 2 ∞

fim da impressao do grafo.





**UNIVERSIDADE PRESBITERIANA MACKENZIE**  
**Faculdade de Computação e Informática**

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



```
Menu
0: Ler dados do arquivo grafo.txt
1: Gravar dados no arquivo grafo.txt
2: Inserir vértice
3: Inserir aresta
4: Remover vértice
5: Remover aresta
6: Mostrar conteúdo do arquivo
7: Mostrar grafo
8: Apresentar conexidade do grafo
9: Encerrar o programa
Escolha uma opção: 8
Conexidade do grafo: Desconexo
```

```
Menu
0: Ler dados do arquivo grafo.txt
1: Gravar dados no arquivo grafo.txt
2: Inserir vértice
3: Inserir aresta
4: Remover vértice
5: Remover aresta
6: Mostrar conteúdo do arquivo
7: Mostrar grafo
8: Apresentar conexidade do grafo
9: Encerrar o programa
Escolha uma opção: 9
Fim
```

➤ **Apêndice:**

<https://github.com/EspadaDeArthur11/Atividade-Projeto-1>