



## **Relatório do Projeto**

Parte 1

Nome do Integrante	RA
Lucas Kenzo Kawamoto	10396359

## **Relatório**

### **Título projeto:**

Criação de grafo das distâncias entre as refinarias e postos do estado de SP

**Objetivo ods:** Este projeto faz parte do Objetivo 7: Energia Limpa e Acessível: Analisar a distribuição geográfica das refinarias e postos de combustível pode ajudar a entender como o acesso a fontes de energia, como gasolina, diesel e etanol, está distribuído na cidade e se é acessível para todos os residentes.

### **Grafos criados:**



# UNIVERSIDADE PRESBITERIANA MACKENZIE

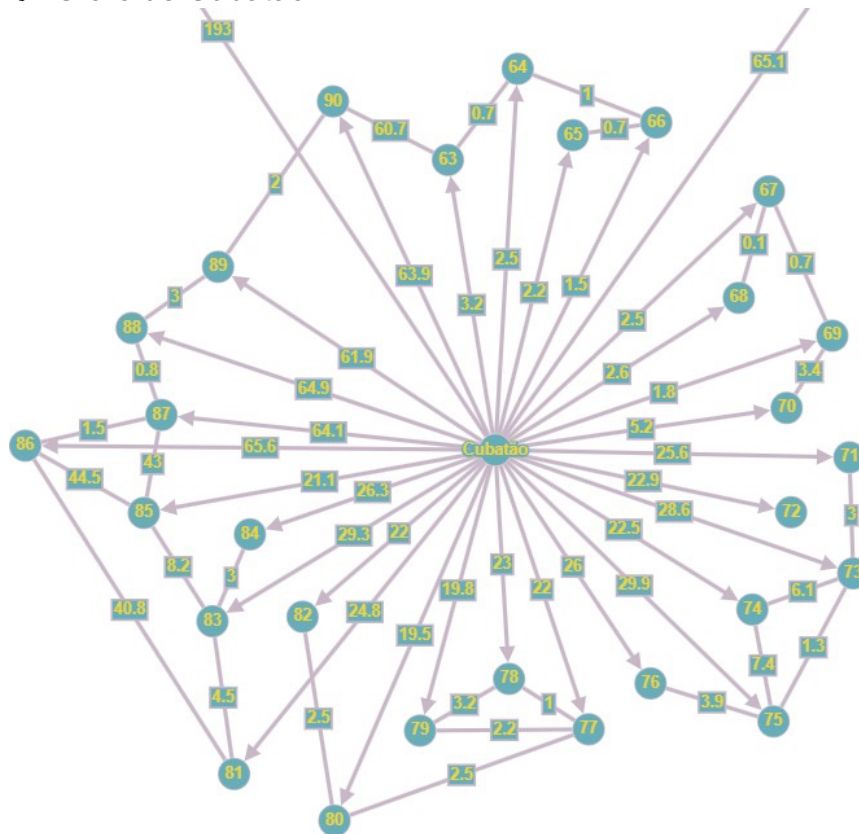
## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



◆ Grafo de Cubatão:



◆ Grafo de Paulínia:

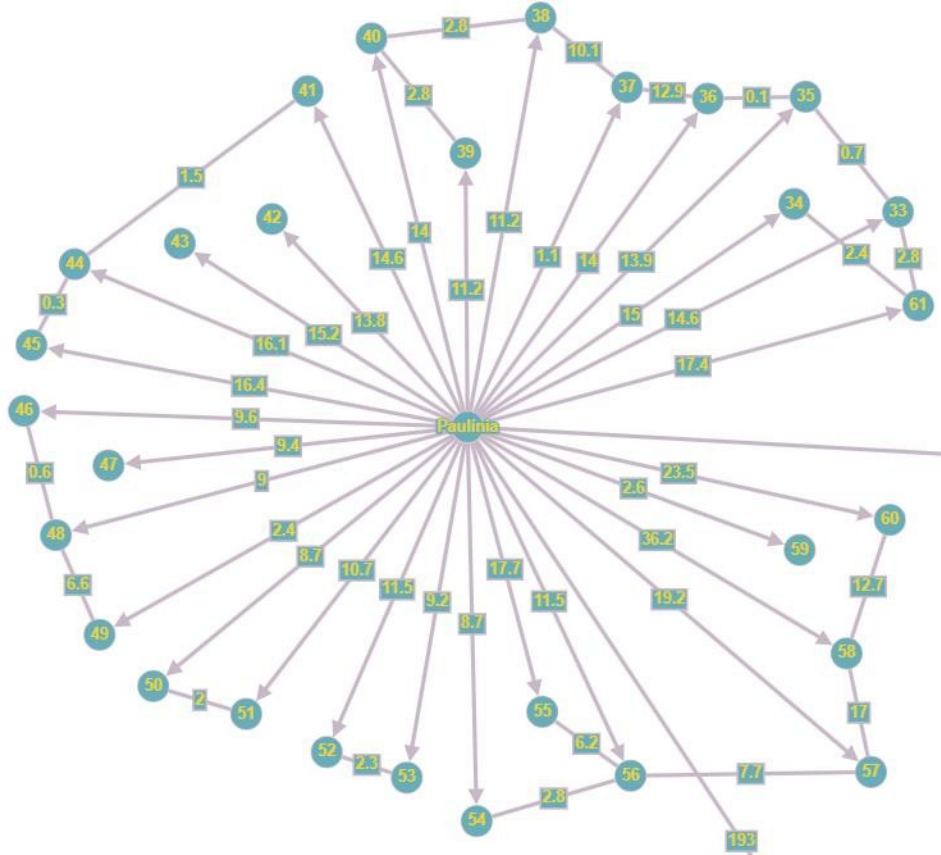


# UNIVERSIDADE PRESBITERIANA MACKENZIE

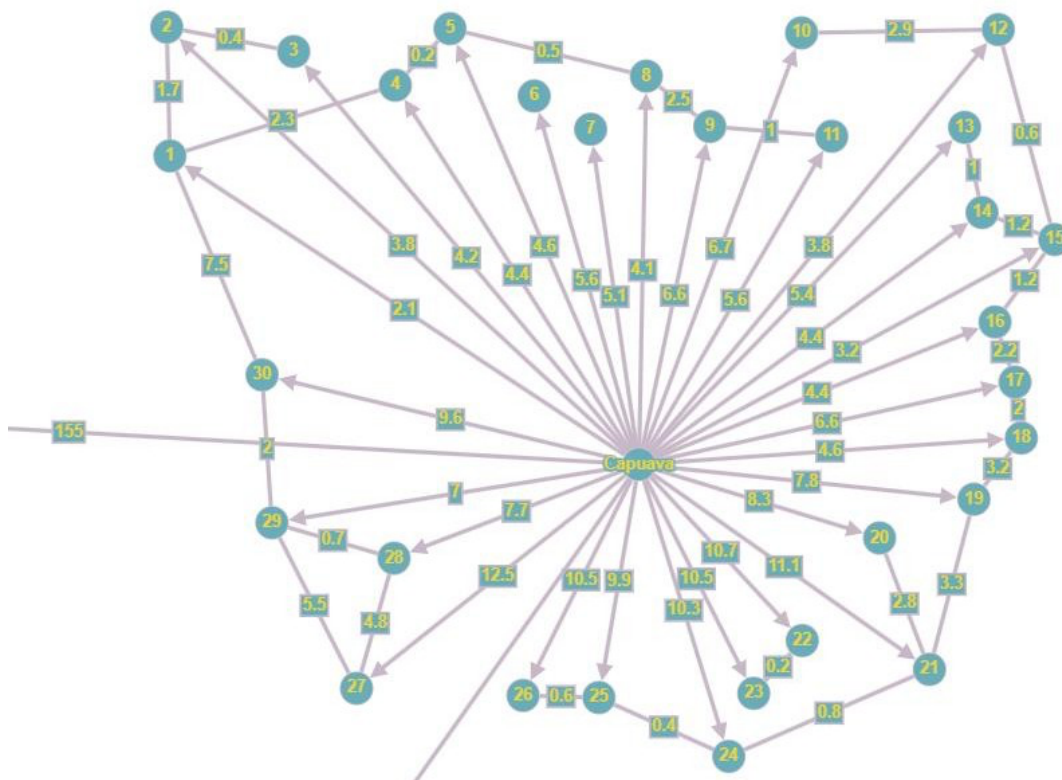
## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



◆ Grafo de Capuava:





UNIVERSIDADE PRESBITERIANA MACKENZIE

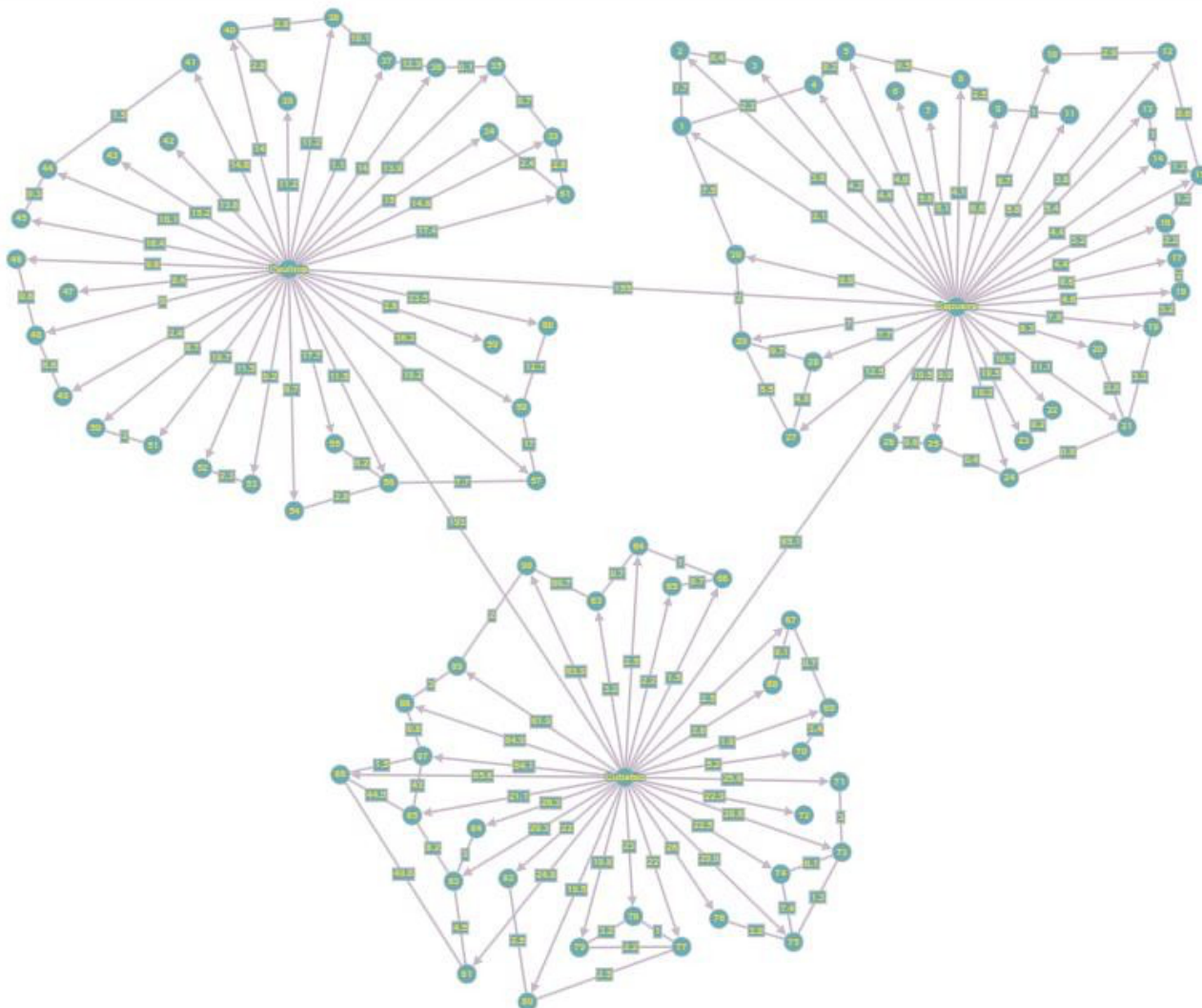
Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



◆ Grafo completo:



◆ Obs: Para mais detalhes, acessar o link do grafo que se encontra no final do relatório.



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

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



**Conteúdo do arquivo grafo.txt:**

Cubatao 63 3.2km	73 74 6.1km	Capuava 20 8.3km
Cubatao 64 2.5km	73 75 1.3km	Capuava 21 11.1km
Cubatao 65 2.2km	74 75 7.4km	Capuava 22 10.7km
Cubatao 66 1.5km	75 76 3.9km	Capuava 23 10.5km
Cubatao 67 2.5km	77 78 1km	Capuava 24 10.3km
Cubatao 68 2.6km	77 79 2.2km	Capuava 25 9.9km
Cubatao 69 1.8km	78 79 3.2km	Capuava 26 10.5km
Cubatao 70 5.2km	77 80 2.5km	Capuava 27 12.5km
Cubatao 71 25.6km	80 82 2.5km	Capuava 28 7.7km
Cubatao 72 22.9km	81 83 4.5km	Capuava 29 7km
Cubatao 73 28.6km	81 86 40.8km	Capuava 30 9.6km
Cubatao 74 22.5km	83 84 3km	Capuava Paulinia
Cubatao 75 29.9km	83 85 8.2km	155km
Cubatao 76 26km	85 86 44.5km	Capuava Cubatao
Cubatao 77 22km	85 87 43km	65.1km
Cubatao 78 23km	86 87 1.5km	1 2 1.7km
Cubatao 79 19.8km	87 88 0.8km	1 4 2.3km
Cubatao 80 19.5km	88 89 3km	2 3 0.4km
Cubatao 81 24.8km	89 90 2km	4 5 0.2km
Cubatao 82 22km	90 63 60.7km	5 8 0.5km
Cubatao 83 29.3km	Capuava 1 2.1km	8 9 2.5km
Cubatao 84 26.3km	Capuava 2 3.8km	9 11 1km
Cubatao 85 21.1km	Capuava 3 4.2km	10 12 2.9km
Cubatao 86 65.6km	Capuava 4 4.4km	12 15 0.6km
Cubatao 87 64.1km	Capuava 5 4.6km	13 14 1km
Cubatao 88 64.9km	Capuava 6 5.6km	14 15 1.2km
Cubatao 89 61.9km	Capuava 7 5.1km	15 16 1.2km
Cubatao 90 63.9km	Capuava 8 4.1km	16 17 2.2km
Cubatao Paulinia	Capuava 9 6.6km	17 18 2km
193km	Capuava 10 6.7km	18 19 3.2km
Cubatao Capuava	Capuava 11 5.6km	19 21 3.3km
65.1km	Capuava 12 3.8km	20 21 2.8km
63 64 0.7km	Capuava 13 5.4km	22 23 0.2km
64 66 1km	Capuava 14 4.4km	21 24 0.8km
65 66 0.7km	Capuava 15 3.2km	24 25 0.4km
67 68 0.1km	Capuava 16 4.4km	25 26 0.6km
67 69 0.7km	Capuava 17 6.6km	27 28 4.8km
69 70 3.4km	Capuava 18 4.6km	28 29 0.7km
71 73 3km	Capuava 19 7.8km	29 30 2km



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

Prof. Dr. Ivan Carlos Alcântara de Oliveira



Teoria dos Grafos

30 1 7.5km	44 45 0.3km
Paulinia 33 14.6km	46 48 0.6km
Paulinia 34 15km	48 49 6.6km
Paulinia 35 13.9km	50 51 2km
Paulinia 36 14km	52 53 2.3km
Paulinia 37 1.1km	54 56 2.8km
Paulinia 38 11.2km	55 56 6.2km
Paulinia 39 11.2km	56 57 7.7km
Paulinia 40 14km	57 58 17km
Paulinia 41 14.6km	58 60 12.7km
Paulinia 42 13.8km	61 34 2.4km
Paulinia 43 15.2km	61 33 2.8km
Paulinia 44 16.1km	
Paulinia 45 16.4km	
Paulinia 46 9.6km	
Paulinia 47 9.4km	
Paulinia 48 9km	
Paulinia 49 2.4km	
Paulinia 50 8.7km	
Paulinia 51 10.7km	
Paulinia 52 11.5km	
Paulinia 53 9.2km	
Paulinia 54 8.7km	
Paulinia 55 17.7km	
Paulinia 56 11.5km	
Paulinia 57 19.2km	
Paulinia 58 36.2km	
Paulinia 59 2.6km	
Paulinia 60 23.5km	
Paulinia 61 17.4km	
Paulinia Capuava 155km	
Paulinia Cubatao 193km	
33 35 0.7km	
35 36 0.1km	
36 37 12.9km	
37 38 10.1km	
38 40 2.8km	
39 40 2.8km	
41 44 1.5km	





## Programa Python Completo:

```
def ler_grafo2(nome_arquivo):  
    grafo = {}  
    vertices = set()  
    with open(nome_arquivo, 'r') as arquivo:  
        linhas = arquivo.readlines()  
        for linha_num, linha in enumerate(linhas, start=1):  
            valores = linha.split()  
            if len(valores) == 3:  
                origem, destino, distancia = valores  
                if origem not in grafo:  
                    grafo[origem] = {}  
                distancia = float(distancia[:-2]) if len(distancia) >= 2 else 0.0  
                grafo[origem][destino] = distancia  
                vertices.add(origem)  
                vertices.add(destino)  
            else:  
                print(  
                    f"Erro na linha {linha_num}: A linha '{linha.strip()}' não possui o formato esperado (origem destino distancia).")  
    # Adiciona as arestas ausentes  
    for v in vertices:  
        if v not in grafo:  
            grafo[v] = {}  
    return grafo
```

```
def gerar_matriz_adjacencia(grafo):  
    vertices = sorted(grafo.keys())  
    tamanho = len(vertices)  
    matriz = [[0] * tamanho for _ in range(tamanho)]  
    indice_vertices = {v: i for i, v in enumerate(vertices)}  
    for origem in grafo:  
        for destino, distancia in grafo[origem].items():  
            matriz[indice_vertices[origem]][indice_vertices[destino]] = distancia  
    return matriz  
  
def mostrar_matriz(matriz):  
    for linha in matriz:  
        print(" ".join(map(str, linha)))  
  
def dfs(grafo, vertice, visitados):  
    visitados.add(vertice)  
    for vizinho in grafo[vertice]:  
        if vizinho not in visitados:  
            dfs(grafo, vizinho, visitados)
```



# UNIVERSIDADE PRESBITERIANA MACKENZIE

## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



```
def verificar_conectividade(grafo):  
    visitados = set()  
    num_componentes = 0  
    for vertice in grafo:  
        if vertice not in visitados:  
            dfs(grafo, vertice, visitados)  
            num_componentes += 1  
  
    if num_componentes == 1:  
        return "C0", "O grafo não é conexo."  
    elif num_componentes == 2:  
        return "C1", "O grafo possui 2 componentes conexos."  
    elif num_componentes == 3:  
        return "C2", "O grafo possui 3 componentes conexos."  
    else:  
        return "C3", "O grafo possui mais de 3 componentes conexos."
```

```
def criar_grafo_reduzido(grafo):  
    grafo_reduzido = {}  
    mapeamento = {}  
    novo_vertice = 1  
  
    for origem in grafo:  
        if origem not in mapeamento:  
            mapeamento[origem] = novo_vertice  
            novo_vertice += 1  
  
        for destino in grafo[origem]:  
            if destino not in mapeamento:  
                mapeamento[destino] = novo_vertice  
                novo_vertice += 1  
  
            vertice_origem = mapeamento[origem]  
            vertice_destino = mapeamento[destino]  
  
            if vertice_origem not in grafo_reduzido:  
                grafo_reduzido[vertice_origem] = {}  
  
            if vertice_destino not in grafo_reduzido[vertice_origem]:  
                grafo_reduzido[vertice_origem][vertice_destino] = grafo[origem][destino]  
            else:  
                grafo_reduzido[vertice_origem][vertice_destino] += grafo[origem][destino]  
  
    # Adicionar vértices que não estão conectados a nenhuma aresta  
    for vertice in mapeamento.values():  
        if vertice not in grafo_reduzido:  
            grafo_reduzido[vertice] = {}  
  
    return grafo_reduzido
```





# UNIVERSIDADE PRESBITERIANA MACKENZIE

## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



```
def mostrar_grafo2(grafo):
    for vertice, vizinhos in grafo.items():
        print(f"{vertice} -> {list(vizinhos.keys())}")

def mostrar_grafo3(grafo):
    for vertice, vizinhos in grafo.items():
        vizinhos_formatados = [str(v) for v in vizinhos]
        print(f"{vertice} -> {' '.join(vizinhos_formatados)}")

def mostrar_grafo4(grafo):
    for origem in grafo:
        for destino, peso in grafo[origem].items():
            print(f"{origem} -> {destino} : {peso}")

def ler_grafo(arquivo):
    with open(arquivo, 'r') as f:
        conteudo = f.read().splitlines()
        grafo = {}
        for linha in conteudo:
            vertice, *arestas = linha.split()
            grafo[vertice] = arestas
        return grafo

def gravar_grafo(arquivo, grafo):
    with open(arquivo, 'w') as f:
        for vertice, arestas in grafo.items():
            f.write(f"{vertice} {' '.join(arestas)}\n")

def inserir_vertice(grafo, vertice):
    if vertice not in grafo:
        grafo[vertice] = []
```

```
def inserir_aresta(grafo, inicio, fim):
    if inicio in grafo:
        grafo[inicio].append(fim)

def remover_vertice(grafo, vertice):
    if vertice in grafo:
        del grafo[vertice]

def remover_aresta(grafo, inicio, fim):
    if inicio in grafo:
        grafo[inicio].remove(fim)

def mostrar_grafo(grafo):
    with open('grafo.txt', 'r') as f:
        for linha in f:
            print(linha.strip()) # strip() remove espaços em branco e quebras de linha extras

def main():
    grafo = {}
    while True:
        print("\nMenu de opções:")
        print("(a) Ler dados do arquivo grafo.txt")
        print("(b) Gravar dados no arquivo grafo.txt")
        print("(c) Inserir vértice")
        print("(d) Inserir aresta")
        print("(e) Remover vértice")
        print("(f) Remover aresta")
        print("(g) Mostrar conteúdo do arquivo")
        print("(h) Mostrar grafo")
        print("(i) Apresentar a conexidade do grafo")
        print("(j) Encerrar a aplicação")
        opcao = input("Escolha uma opção: ")
```



# UNIVERSIDADE PRESBITERIANA MACKENZIE

## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



```
if opcao == 'a':
    grafo = ler_grafo('grafo.txt')
    print("Grafo lido do arquivo grafo.txt.")
elif opcao == 'b':
    gravar_grafo('grafo.txt', grafo)
    print("Grafo gravado no arquivo grafo.txt.")
elif opcao == 'c':
    vertice = input("Digite o vértice a ser inserido: ")
    inserir_vertice(grafo, vertice)
    print(f"Vértice {vertice} inserido com sucesso no grafo.")
elif opcao == 'd':
    inicio = input("Digite o vértice de início da aresta: ")
    fim = input("Digite o vértice de fim da aresta: ")
    inserir_aresta(grafo, inicio, fim)
    print(f"Aresta de {inicio} para {fim} inserida com sucesso no grafo.")
elif opcao == 'e':
    vertice = input("Digite o vértice a ser removido: ")
    remover_vertice(grafo, vertice)
    print(f"Vértice {vertice} removido com sucesso do grafo.")
elif opcao == 'f':
    inicio = input("Digite o vértice de início da aresta a ser removida: ")
    fim = input("Digite o vértice de fim da aresta a ser removida: ")
    remover_aresta(grafo, inicio, fim)
    print(f"Aresta de {inicio} para {fim} removida com sucesso do grafo.")
elif opcao == 'g':
    mostrar_grafo(grafo)
elif opcao == 'h':
    mostrar_grafo(grafo)
elif opcao == 'i':
    # Aqui você pode implementar a lógica para verificar a conexidade do grafo e apresentar o grafo reduzido
```

```
nome_arquivo = "grafo.txt"
grafo = ler_grafo2(nome_arquivo)
matriz_adjacencia = gerar_matriz_adjacencia(grafo)
mostrar_matriz(matriz_adjacencia)
categoria, comentario = verificar_conectividade(grafo)
print(f"Categoria: {categoria}")
print(f"Comentário: {comentario}")
grafo_reduzido = criar_grafo_reduzido(grafo)
print("Grafo Original:")
mostrar_grafo4(grafo)
print("\nGrafo Reduzido:")
print("\nNo código que estamos utilizando, o tipo de redução de grafo realizado é a contração de vértices. ")
print("\nNa contração de vértices, os vértices que estão conectados por uma")
print("\naresta no grafo original são combinados em um único vértice no grafo reduzido. ")
print("\nIsso reduz o número de vértices no grafo e simplifica sua estrutura, ")
print("\nmas mantém a conectividade entre os vértices.")
# Na implementação que estamos fazendo,
# a função contrair_grafo realiza a contração de vértices para
# criar o grafo reduzido a partir do grafo original.
# Ela percorre todas as arestas do grafo original e combina
# os vértices conectados em um novo grafo reduzido,
# mantendo as informações sobre os pesos das arestas.

mostrar_grafo4(grafo_reduzido)

pass
elif opcao == 'j':
    print("Aplicação encerrada.")
    break
else:
    print("Opção inválida!")
```

```
if __name__ == "__main__":
    main()
```



### Compilações do programa

Menu de opções:

- a) Ler dados do arquivo grafo.txt
- b) Gravar dados no arquivo grafo.txt
- c) Inserir vértice
- d) Inserir aresta
- e) Remover vértice
- f) Remover aresta
- g) Mostrar conteúdo do arquivo
- h) Mostrar grafo
- i) Apresentar a conexidade do grafo
- j) Encerrar a aplicação

Escolha uma opção: **a**

Grafo lido do arquivo grafo.txt.

Menu de opções:

- a) Ler dados do arquivo grafo.txt
- b) Gravar dados no arquivo grafo.txt
- c) Inserir vértice
- d) Inserir aresta
- e) Remover vértice
- f) Remover aresta
- g) Mostrar conteúdo do arquivo
- h) Mostrar grafo
- i) Apresentar a conexidade do grafo
- j) Encerrar a aplicação

Escolha uma opção: **i**





```
Categoria: C0
Comentário: 0 grafo não é conexo.
Grafo Original:
Cubatao -> 63 : 3.2
Cubatao -> 64 : 2.5
Cubatao -> 65 : 2.2
Cubatao -> 66 : 1.5
Cubatao -> 67 : 2.5
Cubatao -> 68 : 2.6
Cubatao -> 69 : 1.8
Cubatao -> 70 : 5.2
Cubatao -> 71 : 25.6
Cubatao -> 72 : 22.9
Cubatao -> 73 : 28.6
Cubatao -> 74 : 22.5
Cubatao -> 75 : 29.9
Cubatao -> 76 : 26.0
Cubatao -> 77 : 22.0
Cubatao -> 78 : 23.0
Cubatao -> 79 : 19.8
Cubatao -> 80 : 19.5
Cubatao -> 81 : 24.8
Cubatao -> 82 : 22.0
Cubatao -> 83 : 29.3
Cubatao -> 84 : 26.3
Cubatao -> 85 : 21.1
Cubatao -> 86 : 65.6
Cubatao -> 87 : 64.1
Cubatao -> 88 : 64.9
Cubatao -> 89 : 61.9
Cubatao -> 90 : 63.9
Cubatao -> Paulinia : 193.0
Cubatao -> Capuava : 65.1
63 -> 64 : 0.7
64 -> 66 : 1.0
65 -> 66 : 0.7
67 -> 68 : 0.1
67 -> 69 : 0.7
69 -> 70 : 3.4
71 -> 73 : 3.0
73 -> 74 : 6.1
73 -> 75 : 1.3
74 -> 75 : 7.4
75 -> 76 : 3.9
77 -> 78 : 1.0
77 -> 79 : 2.2
77 -> 80 : 2.5
78 -> 79 : 3.2
80 -> 82 : 2.5
81 -> 83 : 4.5
81 -> 86 : 40.8
83 -> 84 : 3.0
83 -> 85 : 8.2
85 -> 86 : 44.5
85 -> 87 : 43.0
86 -> 87 : 1.5
87 -> 88 : 0.8
88 -> 89 : 3.0
89 -> 90 : 2.0
90 -> 63 : 60.7
Capuava -> 1 : 2.1
Capuava -> 2 : 3.8
Capuava -> 3 : 4.2
Capuava -> 4 : 4.4
Capuava -> 5 : 4.6
Capuava -> 6 : 5.6
Capuava -> 7 : 5.1
Capuava -> 8 : 4.1
Capuava -> 9 : 6.6
Capuava -> 10 : 6.7
Capuava -> 11 : 5.6
Capuava -> 12 : 3.8
Capuava -> 13 : 5.4
Capuava -> 14 : 4.4
Capuava -> 15 : 3.2
Capuava -> 16 : 4.4
Capuava -> 17 : 6.6
Capuava -> 18 : 4.6
```



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



```
Capuava -> 20 : 8.3
Capuava -> 21 : 11.1
Capuava -> 22 : 10.7
Capuava -> 23 : 10.5
Capuava -> 24 : 10.3
Capuava -> 25 : 9.9
Capuava -> 26 : 10.5
Capuava -> 27 : 12.5
Capuava -> 28 : 7.7
Capuava -> 29 : 7.0
Capuava -> 30 : 9.6
Capuava -> Paulinia : 155.0
Capuava -> Cubatao : 65.1
1 -> 2 : 1.7
1 -> 4 : 2.3
2 -> 3 : 0.4
4 -> 5 : 0.2
5 -> 8 : 0.5
8 -> 9 : 2.5
9 -> 11 : 1.0
10 -> 12 : 2.9
12 -> 15 : 0.6
13 -> 14 : 1.0
14 -> 15 : 1.2
15 -> 16 : 1.2
16 -> 17 : 2.2
17 -> 18 : 2.0
18 -> 19 : 3.2
19 -> 21 : 3.3
20 -> 21 : 2.8
22 -> 23 : 0.2
21 -> 24 : 0.8
24 -> 25 : 0.4
25 -> 26 : 0.6
27 -> 28 : 4.8
28 -> 29 : 0.7
29 -> 30 : 2.0
30 -> 1 : 7.5
Paulinia -> 33 : 14.6
```





# UNIVERSIDADE PRESBITERIANA MACKENZIE

## Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Teoria dos Grafos



Grafo Reduzido:

No código que estamos utilizando, o tipo de redução de grafo realizado é a contração de vértices.

Na contração de vértices, os vértices que estão conectados por uma

aresta no grafo original são combinados em um único vértice no grafo reduzido.

Isso reduz o número de vértices no grafo e simplifica sua estrutura,

mas mantém a conectividade entre os vértices.

```
1 -> 2 : 3.2
1 -> 3 : 2.5
1 -> 4 : 2.2
1 -> 5 : 1.5
1 -> 6 : 2.5
1 -> 7 : 2.6
1 -> 8 : 1.8
1 -> 9 : 5.2
1 -> 10 : 25.6
1 -> 11 : 22.9
1 -> 12 : 28.6
1 -> 13 : 22.5
1 -> 14 : 29.9
1 -> 15 : 26.0
1 -> 16 : 22.0
1 -> 17 : 23.0
1 -> 18 : 19.8
1 -> 19 : 19.5
1 -> 20 : 24.8
1 -> 21 : 22.0
1 -> 22 : 29.3
1 -> 23 : 26.3
1 -> 24 : 21.1
1 -> 25 : 65.6
1 -> 26 : 64.1
1 -> 27 : 64.9
1 -> 28 : 61.9
1 -> 29 : 63.9
1 -> 30 : 193.0
```



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

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



```
Menu de opções:  
a) Ler dados do arquivo grafo.txt  
b) Gravar dados no arquivo grafo.txt  
c) Inserir vértice  
d) Inserir aresta  
e) Remover vértice  
f) Remover aresta  
g) Mostrar conteúdo do arquivo  
h) Mostrar grafo  
i) Apresentar a conexidade do grafo  
j) Encerrar a aplicação  
Escolha uma opção: ]  
Aplicação encerrada.
```

◆ Obs: Por ter uma compilação muito grande das opções, colocamos umas partes da compilação para o relatório não ficar muito grande e poluído.

Link do repositório no GitHub: <https://github.com/LUCASBR8/GRAFOS>

Link do grafo: <http://graphonline.ru/pt/?graph=PjVctqLhWkrUwThI>