Relatório do Projeto: Sistema de Localização Cajo Traffic System com Algoritmos de Dijkstra, Bellman-Ford e Floyd

Integrantes:
Caio Teixeira Torres; RA: 10417478

João Victor de Paula Silva; RA: 10418556

1. Introdução

O presente relatório documenta o desenvolvimento do **Cajo Traffic System**, um sistema de manipulação de grafos voltado para aplicações de localização urbana. O sistema permite carregar, salvar e manipular grafos representados por matrizes de adjacência, simulando a estrutura de uma malha urbana.

Como aprimoramento, foi incorporado os **algoritmos de Dijkstra, Bellman-Ford e Floyd** para o cálculo do **caminho mínimo entre vértices**, permitindo simular rotas otimizadas em redes urbanas, como deslocamentos por vias públicas.

A aplicação está dividida em três módulos principais:

- GrafoMatriz.py: implementação da estrutura de dados do grafo.
- GrafoTeste.py: operações e funções de entrada/saída para manipulação do grafo.
- main.py: interface de menu interativo com o usuário.

2. Objetivos

- Implementar uma estrutura de grafo orientado utilizando matriz de adjacência.
- Permitir operações como inserção e remoção de vértices e arestas.
- Permitir o carregamento e salvamento de dados em arquivos .txt.
- Determinar propriedades do grafo como grau, simetria e conexidade.
- **Implementar o algoritmo de Dijkstra** para cálculo do menor caminho entre dois pontos.

3. Estrutura dos Arquivos

3.1 GrafoMatriz.py

Contém a classe Grafo, que representa um grafo direcionado com as seguintes funcionalidades:

- Inserção e remoção de vértices e arestas.
- Impressão da matriz de adjacência.
- Cálculo de grau de entrada, grau de saída, e grau total.
- Verificação de simetria.
- Identificação de fontes e sorvedouros.
- Verificação de conexidade (a partir do vértice 0).
- **Algoritmo de Dijkstra** para menor caminho entre dois vértices.
- **Algoritmo de Bellman-Ford** para calcular o menor caminho.

- **Algoritmo de Floyd** para calcular menor caminho.

Funções adicionais:

- carregar_grafo(arquivo): carrega um grafo de um arquivo .txt.
- salvar_grafo(arquivo, grafo, nomes_vertices): salva o grafo em arquivo .txt.

3.2 GrafoTeste.py

Implementa funcionalidades para interação com o grafo:

- Leitura e gravação de dados.
- Inserção/remoção de vértices e arestas.
- Impressão do conteúdo do arquivo.
- Impressão da matriz do grafo.
- Apresentação da conexidade.
- **Execução do Dijkstra** com seleção de vértices de origem e destino.

3.3 main.py

Arquivo principal da aplicação com um menu interativo que permite ao usuário:

- Ler e gravar dados em grafo.txt.
- Inserir/remover vértices e arestas.
- Visualizar o grafo e seu estado no arquivo.
- Verificar conexidade e apresentar forma reduzida.
- **Executar os algoritmos de Dijkstra**, Bellman-Ford e Floyd para rotas mais curtas.
- Encerrar a aplicação.

4. Funcionamento Geral do Sistema

O sistema opera por meio de um menu textual com opções como:

- **a)** Ler dados de grafo.txt
- **b)** Gravar dados em grafo.txt
- **c)** Inserir vértice
- **d)** Inserir aresta
- **e)** Remover vértice
- **f)** Remover aresta
- **g)** Mostrar conteúdo do arquivo
- **h)** Mostrar grafo
- **i)** Verificar conexidade
- **j)** Calcular menor caminho com Dijkstra
- **k)** Calcular menor caminho com Bellman-Ford
- **k)** Calcular menor caminho com Floyd
- **m)** Encerrar aplicação

As funcionalidades interagem com o grafo em memória e com o arquivo grafo.txt, permitindo edições persistentes e análises dinâmicas.

🧪 5. Testes Recomendados

- Testar inserção e remoção de vértices e arestas.
- Verificar leitura e gravação com diferentes grafos.

- Avaliar conexidade em grafos com componentes isolados.
- Testar simetria em grafos direcionados e não direcionados.
- **Testar os 3 algoritmos citados com diferentes pares de vértices.**

🌎 6. Relação com os Objetivos de Desenvolvimento Sustentável (ODS)

Este projeto se relaciona com os seguintes ODS:

🎯 ODS 11: Cidades e Comunidades Sustentáveis

O algoritmo de Dijkstra possibilita a **simulação de rotas mais eficientes**, contribuindo para a redução do trânsito urbano, otimização de deslocamentos e emissão de poluentes — fatores fundamentais para tornar as cidades mais sustentáveis e inteligentes.

🎯 ODS 9: Indústria, Inovação e Infraestrutura

A manipulação de grafos e rotas com algoritmos como Dijkstra contribui para o **planejamento inteligente de infraestrutura urbana**, otimizando o uso de vias e reduzindo gargalos logísticos em sistemas de transporte.

🏁 7. Conclusão

O **Cajo Traffic System** apresenta uma solução educacional e prática para manipulação de grafos voltados a sistemas urbanos. Com a adição do **algoritmo de Dijkstra**, torna-se possível simular caminhos mínimos em mapas urbanos, aumentando o valor da aplicação para fins como roteamento, logística e mobilidade urbana sustentável.

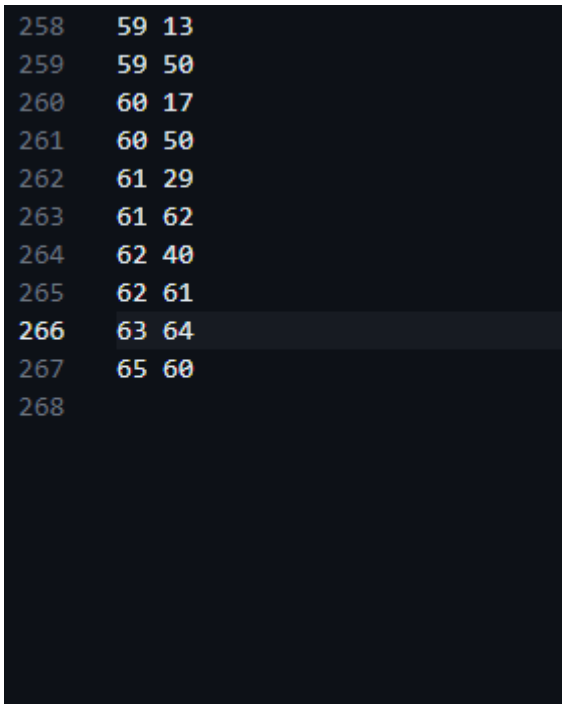
8. Testes

1. Ler dados do arquivo e salvar dados:

```
Bem vindo ao Cajo Traffic System
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) Apresentar o caminho com Dijkstra
k) Apresentar o caminho com Bellman-Ford
l) Apresentar o caminho com Floyd
m) Encerrar a aplicação
Escolha uma opção: a
Dados carregados com sucesso!

Bem vindo ao Cajo Traffic System
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) Apresentar o caminho com Dijkstra
k) Apresentar o caminho com Bellman-Ford
l) Apresentar o caminho com Floyd
m) Encerrar a aplicação
Escolha uma opção: b
Grafo salvo com sucesso!
Dados gravados com sucesso!

Bem vindo ao Cajo Traffic System
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) Apresentar o caminho com Dijkstra
k) Apresentar o caminho com Bellman-Ford
l) Apresentar o caminho com Floyd
m) Encerrar a aplicação
Escolha uma opção:
```

2. Inserir vértice e aresta:

```
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) Apresentar o caminho com Dijkstra
k) Apresentar o caminho com Bellman-Ford
l) Apresentar o caminho com Floyd
m) Encerrar a aplicação
Escolha uma opção: c
Digite o rótulo do vértice: Registro
Vértice inserido com sucesso!

Bem vindo ao Cajo Traffic System
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) Apresentar o caminho com Dijkstra
k) Apresentar o caminho com Bellman-Ford
l) Apresentar o caminho com Floyd
m) Encerrar a aplicação
Escolha uma opção: d
Digite o vértice de origem: 65
Digite o vértice de destino: 60
Aresta inserida com sucesso!

Bem vindo ao Cajo Traffic System
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) Apresentar o caminho com Dijkstra
k) Apresentar o caminho com Bellman-Ford
l) Apresentar o caminho com Floyd
m) Encerrar a aplicação
Escolha uma opção: b
Grafo salvo com sucesso!
```



O caminho vértice novo foi adicionado ao fim do grafo.txt

3. Remover vértice e aresta:

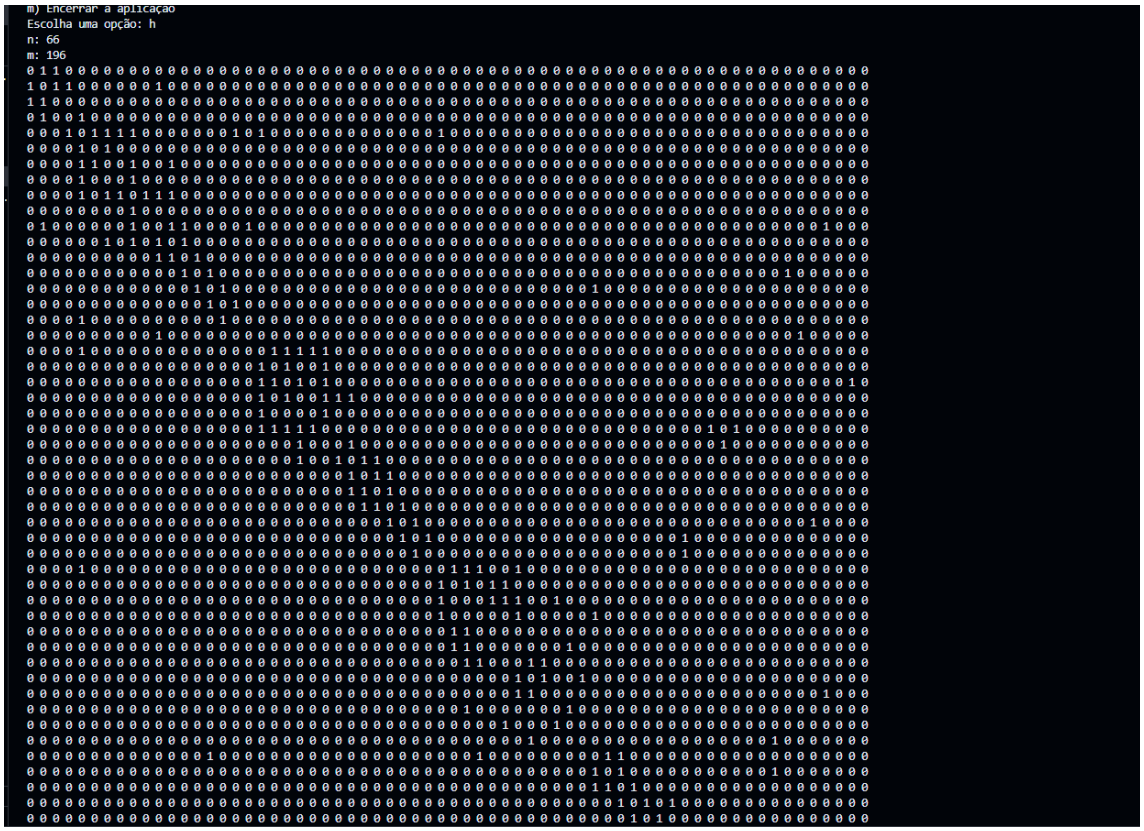
```
b) Gravar dados no arquivo grafo.txt
c) Inserir vértice
d) Inserir aresta
e) Remover vértice
f) Remover aresta
g) Mostrar conteúdo do arquivo
h) Mostrar grafo
i) Apresentar a conexidade do grafo
j) Apresentar o caminho com Dijkstra
k) Apresentar o caminho com Bellman-Ford
l) Apresentar o caminho com Floyd
m) Encerrar a aplicação
Escolha uma opção: e
Digite o vértice a ser removido: 65
Vértice removido com sucesso!

Bem vindo ao Cajo Traffic System
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) Apresentar o caminho com Dijkstra
k) Apresentar o caminho com Bellman-Ford
l) Apresentar o caminho com Floyd
m) Encerrar a aplicação
Escolha uma opção: f
Digite o vértice de origem: 65
Digite o vértice de destino: 60
Aresta removida com sucesso!

Bem vindo ao Cajo Traffic System
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) Apresentar o caminho com Dijkstra
k) Apresentar o caminho com Bellman-Ford
l) Apresentar o caminho com Floyd
m) Encerrar a aplicação
Escolha uma opção: b
Grafo salvo com sucesso!
Dados gravados com sucesso!
```

```
261   61 29
262   61 62
263   62 40
264   62 61
265   63 64
266
```

A aresta foi corretamente removida do grafo.txt.

4. Mostrar Conteúdo e Grafo:

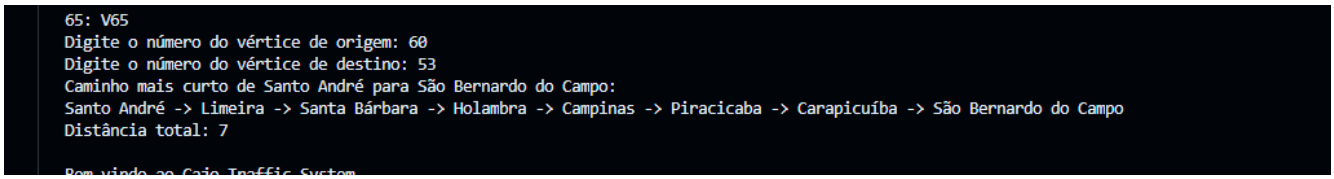
```
Bem vindo ao Cajo Traffic System
Menu de opções:
a) Ler dados do arquivo grafo.txt
b) Gravar dados no arquivo grafo.txt
c) Inserir vértice
d) Inserir aresta
e) Remover vértice
f) Remover aresta
g) Mostrar conteúdo do arquivo
h) Mostrar grafo
i) Apresentar a conexidade do grafo
j) Apresentar o caminho com Dijkstra
k) Apresentar o caminho com Bellman-Ford
l) Apresentar o caminho com Floyd
m) Encerrar a aplicação
Escolha uma opção: g
Conteúdo do arquivo grafo.txt:
Grafo
66
0 "Valinhos"
1 "Vinhedo"
2 "Hortolândia"
3 "Indaiatuba"
4 "Campinas"
5 "Paulínia"
6 "Itupeva"
7 "Salto"
8 "Holambra"
9 "Americana"
10 "Santa Bárbara"
11 "Rio Claro"
12 "Taquaritinga"
13 "Catanduva"
14 "São José do Rio Preto"
15 "Araraquara"
16 "São Carlos"
17 "Limeira"
18 "Piracicaba"
19 "Analândia"
20 "Morro Agudo"
21 "Águas de São Pedro"
22 "Jandira"
23 "Carapicuíba"
24 "Itapevi"
25 "Alphaville"
26 "Barueri"
27 "Osasco"
28 "Santana de Parnaíba"
29 "Taboão da Serra"
30 "Itapecerica da Serra"
31 "Embu das Artes"
32 "Cotia"
33 "Ilha Comprida"
34 "Iguape"
35 "Peruíbe"
```



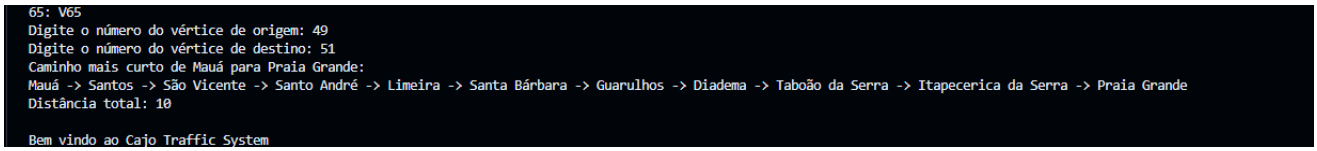
5. Apresentar Algoritmos de Dijkstra, Bellman-Ford e Floyd:

Importante citar que os 3 algoritmos listam as cidades e os números dos vértices.

Algoritmo de Dijkstra:



Algoritmo de Bellman-Ford:



Algoritmo de Floyd: