UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA - SEMESTRE SELETIVO 2025/1
ALGORITMOS E ESTRUTURAS DE DADOS 2 - DOCENTE ANDRÉ LUIZ MOURA

SISTEMA DE NAVEGAÇÃO PRIMITIVO

Autores: Ana Luísa Pereira dos Santos, Isadora Yasmim da Silva, Lucas Costa de Alvarenga e Verônica Ribeiro Oliveira Palmeira.

Goiânia - GO

2025

SUMÁRIO

1. INTRODUÇÃO

Este documento descreve a especificação e implementação do *Sistema de Navegação Primitivo*, desenvolvido como projeto final da disciplina Algoritmos e Estruturas de Dados 2 (AED2) do curso de Bacharelado em Engenharia de Software da Universidade Federal de Goiás (UFG). O sistema permite encontrar o menor caminho entre dois pontos em um grafo geográfico, cuja representação é feita através de arquivos no formato *.poly.*

1.1. Propósito do Documento

O propósito deste documento é detalhar o projeto do Sistema de Navegação Primitivo, incluindo seus requisitos, arquitetura, tecnologias, estruturas de dados, fluxo de execução, instruções de uso e resultados de testes. Ele serve como um guia técnico para quaisquer interessados em compreender o funcionamento interno do sistema.

1.2. Escopo do Projeto

O escopo do projeto abrange o desenvolvimento de um sistema capaz de importar mapas, calcular e visualizar o menor caminho entre dois pontos selecionados pelo usuário, bem como exibir estatísticas relevantes sobre o cálculo. O foco principal é a eficiência do algoritmo de busca de caminho em grafos representados geograficamente.

2. REQUISITOS DO SISTEMA

Esta seção detalha os requisitos funcionais e não funcionais que o Sistema de Navegação Primitivo deve atender.

2.1. Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades que o sistema deve fornecer aos usuários.

- RF01 – Permitir importar mapas reais (de arquivos .txt ou xlml etc., contendo coordenadas de vértices e arestas ref. a ruas, avenidas, rodovias etc.) e converter para grafos.
- RF02 – Disponibilizar opção de enumerar os vértices e rotular as arestas com pesos (distâncias).
- RF03 – Permitir ao usuário fazer/desfazer seleção de um vértice de origem e um de destino, com marcação, em cores distintas, para origem e destino.
- RF04 – Calcular e exibir, em cor diferenciada, rota indicativa do menor caminho entre dois vértices.
- RF05 – Permitir a criação e edição de grafos, adicionando/removendo vértices e arestas, com auxílio de clique do mouse.
- RF06 – Suporte a diferentes tipos de grafos (ponderado, não direcionado/direcionado). Ou seja, tratar vias com mão única e mão dupla.

- RF07 – Exibir estatísticas sobre a execução do algoritmo (tempo de processamento, número de nós explorados, custo total).
- RF08 – Permitir copiar para a área de transferência a imagem do grafo em qualquer momento.

2.2. Requisitos Não Funcionais

Os requisitos não funcionais descrevem atributos de qualidade do sistema.

- RNF01 – Os vértices e as arestas do grafo devem ter cores distintas.
- RNF02 – Diferenciar aresta representativa de via de mão única da aresta de via de mão dupla.
- RNF03 – A execução do programa deve ser otimizada para grandes grafos (milhares de vértices e arestas).
- RNF04 – O tempo de resposta para cálculos deve ser inferior a 2 segundos para grafos médios (~500 nós).
- RNF05 – Uso eficiente de memória para evitar sobrecarga em grafos extensos.
- RNF06 – Interface intuitiva e de fácil manipulação para usuários não técnicos.
- RNF07 – Suporte aos sistemas operacionais Windows ou Linux.
- RNF08 – Código modular e bem documentado para facilitar a manutenção.

3. ARQUITETURA DO SISTEMA

O sistema de Navegação Primitivo é concebido com uma arquitetura modular, dividida em dois componentes principais que interagem entre si.

3.1. Componentes Principais

- **Back-end (Linguagem C):** Responsável por todas as operações de processamento intensivo do grafo, incluindo a leitura do grafo a partir do arquivo *.poly* e a execução do algoritmo de Dijkstra para encontrar o menor caminho.
- **Front-end (Python + Pygame):** Responsável pela interface gráfica do usuário (GUI), permitindo a interação, visualização do grafo, seleção de pontos, e exibição dos resultados e estatísticas.

A comunicação entre o Back-end e o Front-end é realizada através da troca de informações via arquivos de texto dedicados: *entrada.txt* e *saida.txt*. O front-end escreve requisições em *entrada.txt*, e o back-end lê, processa e escreve os resultados em *saida.txt*, que são então lidos e exibidos pelo front-end.

A estrutura de pastas do projeto segue uma organização lógica para separar o código-fonte dos dados e arquivos de configuração.

3.2. Tecnologias Utilizadas

- **Back-end:** Linguagem de programação C.
- **Front-end:** Linguagem de programação Python 3.x com a biblioteca Pygame para desenvolvimento da interface gráfica.

4. INTERFACE GRÁFICA

A interface gráfica do usuário do Sistema de Navegação Primitivo é desenvolvida utilizando a biblioteca Pygame. Ela foi projetada para ser intuitiva, facilitando a interação do usuário com o grafo e as funcionalidades do sistema.

4.1. Elementos da Interface

A interface apresenta os seguintes elementos interativos principais:

- **Botões de Ação:** Para traçar caminhos, mudar o tamanho.
- **Área de Visualização do Grafo:** A área central onde o grafo é renderizado visualmente.

4.2. Visualização do Grafo

O grafo é renderizado de forma clara, utilizando distinção de cores para aprimorar a compreensão visual:

- **Vértices:** Representados por pontos com cores específicas.
- **Arestas:** Representadas por linhas, com cores que indicam sua natureza (mão única, dupla).
- **Origem e Destino:** Os vértices selecionados como origem e destino são destacados com cores diferentes para fácil identificação.
- **Caminho Mínimo:** O caminho resultante do algoritmo de Dijkstra é exibido com uma cor distinta para realçar a rota encontrada.

4.3. Imagem da Interface

5. ESTRUTURAS DE DADOS UTILIZADAS

O sistema faz uso de estruturas de dados eficientes para o armazenamento e manipulação do grafo e para o processamento do algoritmo de menor caminho.

5.1. Back-end

No back-end (implementado em C), as seguintes estruturas de dados são empregadas:

- **Lista de Adjacência:** Utilizada para representar o grafo, otimizando o armazenamento e a travessia de arestas, especialmente para grafos esparsos. Esta escolha permite uma representação eficiente das conexões entre os vértices.
- **Heap Mínima (Min-Heap):** Empregada como fila de prioridade na implementação do algoritmo de Dijkstra, garantindo a seleção eficiente do próximo vértice a ser processado com o menor custo acumulado.
- **Vetores:** Utilizados para armazenar informações auxiliares necessárias ao algoritmo de Dijkstra, como:
    - Distâncias: Armazena a menor distância conhecida do vértice de origem para cada outro vértice.
    - Predecessores: Rastreia o predecessor de cada vértice no caminho mínimo, permitindo a reconstrução do caminho ao final do algoritmo.
    - Visitados: Indica se um vértice já foi processado pelo algoritmo.

5.2. Front-end

No front-end (implementado em Python), uma estrutura de dados adicional é utilizada para facilitar a exibição:

- **Matriz de Adjacência:** Utilizada primariamente para a exibição dos pesos das arestas na interface gráfica, facilitando o acesso rápido a essa informação para fins de renderização.

6. FLUXO DE EXECUÇÃO

O fluxo de execução do Sistema de Navegação Primitivo segue uma sequência clara de interações entre o usuário, o frontend e o backend:

1. **Carregamento do Arquivo:** O usuário inicia a interface gráfica do sistema e carrega um arquivo de mapa no formato *.poly.*
2. **Seleção de Vértices:** O usuário seleciona visualmente dois vértices no grafo renderizado, definindo um ponto de origem e um ponto de destino para o cálculo do caminho.
3. **Comunicação Frontend-Backend:** O front-end, ao identificar a seleção, escreve os índices dos vértices de origem e destino no arquivo *entrada.txt.*
4. **Execução do Back-end:** O back-end, por sua vez, monitora o arquivo *entrada.txt*. Ao detectar a entrada, ele executa o algoritmo de Dijkstra, calculando o menor caminho entre os pontos especificados. Os resultados (o caminho, a distância total e outras estatísticas) são então gravados no arquivo *saida.txt.*
5. **Exibição dos Resultados:** O front-end lê o conteúdo de *saida.txt* e exibe o caminho encontrado na interface gráfica, destacando a rota no grafo e apresentando as estatísticas relevantes ao usuário.

7. INSTRUÇÕES DE EXECUÇÃO

Para executar o Sistema de Navegação Primitivo, deve-se seguir os passos abaixo:

7.1. Pré-requisitos

- **Python:** É necessário ter o Python 3.x instalado no sistema. Recomenda-se a versão 3.8 ou superior.

- **Dependências Python:** Instale as seguintes bibliotecas Python utilizando o gerenciador de pacotes *pip*:

    - ***pygame:*** Para a interface gráfica.
    - ***pywin32:*** (Apenas para Windows) Necessário para algumas funcionalidades de sistema em ambientes Windows.

Comandos de instalação:

```
pip install pygame pillow

# Para Windows:
pip install pywin32
```

pip install pygame pillow | # Para Windows: pip install pywin32

- **Compilador C:** É necessário ter um compilador C (e.g., GCC) instalado para compilar o código do backend.

7.2. Compilação do Backend

Navegue até o diretório do projeto onde o arquivo *backend.c* está localizado e compile-o usando o comando apropriado para o seu sistema operacional:

Windows:

```
gcc backend.c -o backend.exe -lm
```

gcc backend.c -o backend.exe -lm

Linux:

```
gcc backend.c -o backend -lm
```

gcc backend.c -o backend -lm

7.3. Execução do Frontend

Após a compilação do backend, execute o frontend. Certifique-se de estar no diretório raiz do projeto onde o arquivo *navegacao_primitiva_pygame.py* está localizado.

```
python navegacao_primitiva_pygame.py
```

python navegacao_primitiva_pygame.py

A interface gráfica do sistema será iniciada, permitindo a interação do usuário.

8. TESTES REALIZADOS

O sistema foi submetido a uma série de testes para verificar sua funcionalidade e robustez.

8.1. Cenários de Teste

- **Importação de Arquivos *poly:*** O sistema foi testado com diversos arquivos *.poly*, incluindo o *SetorGoiania2.poly*, que contém 54 vértices.
- **Cálculo de Menor Caminho:**
    - **Caminhos Existentes:** Verificou-se o correto traçado do menor caminho entre diferentes pares de origem e destino, garantindo que os resultados no arquivo de saída *(saida.txt)* e a visualização na interface fossem consistentes.
    - **Rotas Inexistentes:** O sistema foi testado com cenários onde não havia um caminho possível entre a origem e o destino selecionados. Nesses casos, o sistema exibe uma mensagem apropriada ao usuário, indicando a impossibilidade de encontrar a rota.

8.2. Resultados

Os testes confirmaram que o sistema atende aos requisitos propostos, sendo capaz de importar e processar grafos complexos, calcular o menor caminho de forma precisa e exibir os resultados de maneira compreensível. A capacidade de lidar com rotas inexistentes demonstra a robustez do tratamento de exceções.

9. CONCLUSÃO

O projeto do Sistema de Navegação Primitivo atendeu com sucesso aos requisitos propostos, resultando em uma solução visual, funcional e otimizada para o cálculo de menor caminho em grafos geográficos.

Durante o desenvolvimento, a equipe adquiriu conhecimentos aprofundados sobre a implementação de estruturas de dados avançadas (como Lista de Adjacência e Heap Mínima), a integração entre linguagens de programação distintas (C para backend e Python para frontend) e o desenvolvimento de interfaces gráficas interativas utilizando a biblioteca Pygame. Os desafios de otimização para grafos grandes e a comunicação entre processos

foram superados com sucesso, consolidando um valioso aprendizado em engenharia de software.

10. EQUIPE

- Ana Luísa Pereira Dos Santos
- Isadora Yasmim Da Silva
- Lucas Costa De Alvarenga
- Veronica Ribeiro Oliveira Palmeira