Busca do Menor Caminho entre Bairros de São Luís usando Grafos e Algoritmo de Dijkstra

Antônio José Lobato Nogueira e Felipe Murilo Ribeiro Ribeiro

ALGORITMOS E ESTRUTURAS DE DADOS II

IFMA – CAMPUS MONTE CASTELO

Motivação e Contexto

Desafio Urbano

São Luís enfrenta diariamente desafios de mobilidade entre bairros como Cohama, Centro e Aeroporto. A determinação eficiente de rotas mais curtas é essencial para navegação GPS, planejamento logístico e otimização de transporte urbano.

Aplicações Práticas

- Sistemas de navegação GPS
- Planejamento de entregas e logística
- Otimização de transporte urbano
- Aplicativos de mobilidade

Objetivos do Trabalho

Objetivo Principal

Implementar um TAD Grafo genérico baseado em listas de adjacência e o algoritmo de Dijkstra para resolver o problema do caminho mínimo de fonte única, aplicado à modelagem de rotas urbanas entre bairros de São Luís-MA.

Objetivos Específicos

- Modelar malha viária como grafo ponderado
- Desenvolver API genérica para grafos
- Implementar Dijkstra com PriorityQueue
- Validar com casos reais locais

Fundamentação Teórica: Grafos

Um grafo $G = (V, A)$ é composto por um conjunto finito de vértices V e um conjunto de arestas A que conectam pares de vértices, representando relacionamentos entre objetos.

Neste trabalho utilizamos **grafos ponderados não direcionados**, onde cada aresta $(u, v) \in A$ possui um peso $p(u, v) \geq 0$ correspondente à distância em quilômetros entre bairros.

Vértices

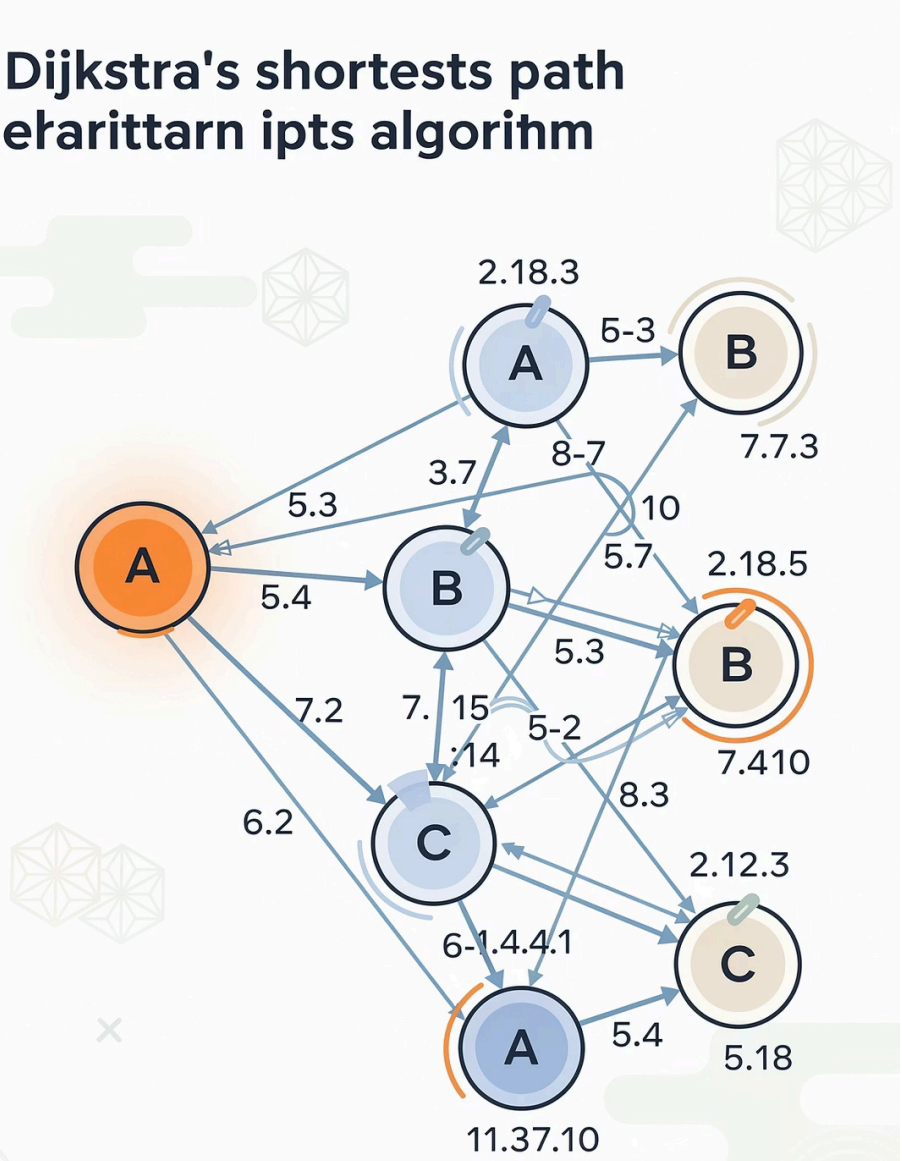
Bairros de São Luís

Arestas

Ligações viárias

Pesos

Distâncias em km



Algoritmo de Dijkstra

01

Problema do Caminho Mínimo

Dado um vértice de origem $s \in V$, busca o caminho mais curto até todos os outros vértices

02

Pré-condição

Funciona apenas com pesos não negativos nas arestas

03

Fila de Prioridade

Expande sempre o vértice com menor distância estimada

04

Complexidade

Tempo de execução: $O((V + A) \log V)$

Modelagem do Problema

Estrutura do Grafo

36

Vértices

Bairros representativos de São Luís

66

Arestas

Ligações viárias bidirecionais

Densidade aproximada: $\frac{|A|}{|V|^2} \approx 0.05$

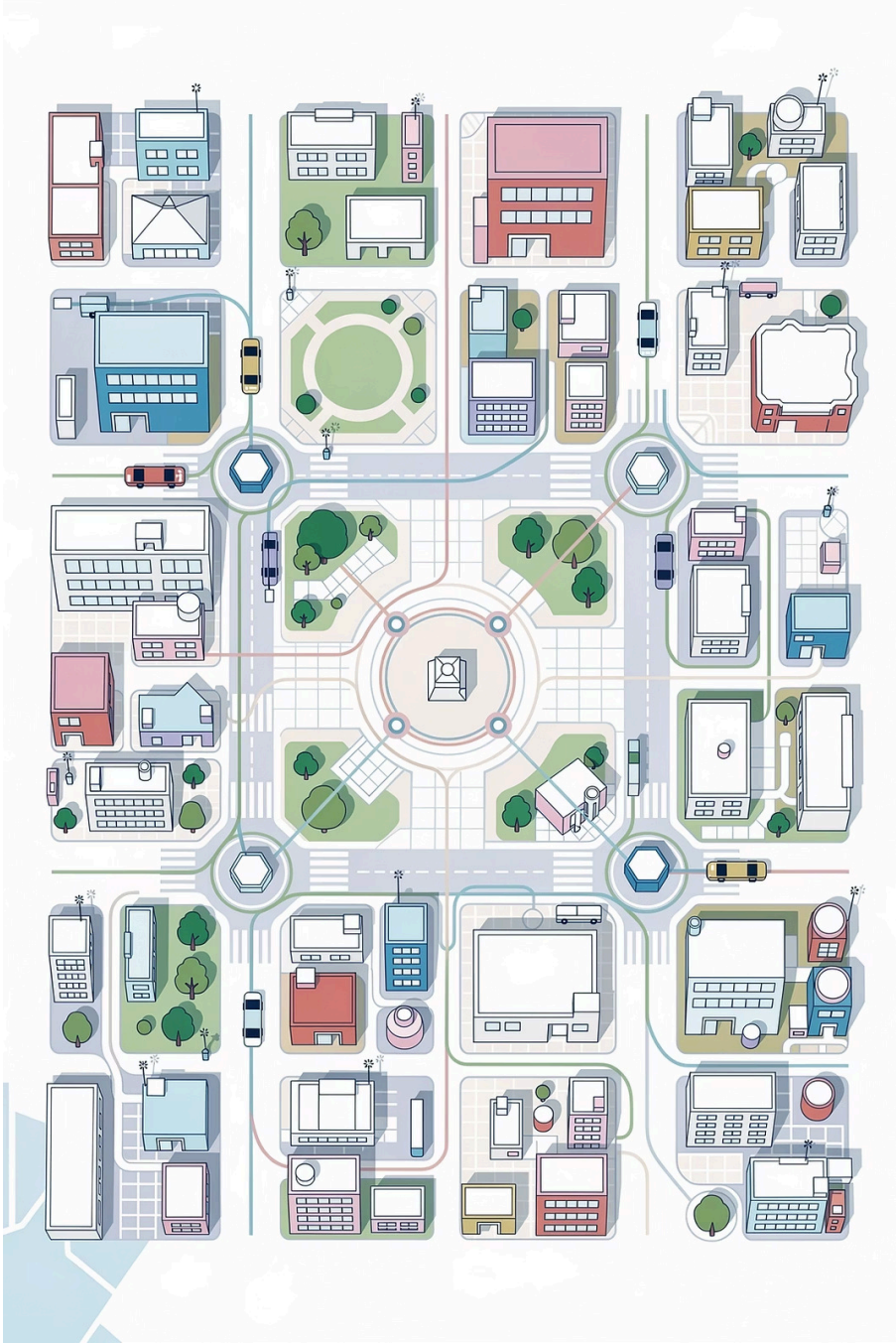
Característica típica de redes viárias urbanas (grafo esparso)

Fonte de Dados

Leitura do arquivo `entrada.txt` com formato:

```
bairro1 bairro2 distancia
cohama cohafuma 3.2
cohafuma calhau 4.5
...
```

Exemplos de bairros: Cohama, Centro, Aeroporto, Ipase, São Cristóvão, Divineia, Tiririca, Coroado, Turu



Implementação: Estrutura de Dados

1

Linguagem e Estrutura

Implementação em **Java** utilizando `Map<String, List<Aresta>>` para representar listas de adjacência

Acesso aos adjacentes em tempo constante $O(1)$

2

TAD Grafo Genérico

API independente do domínio com operações:

- `inserirAresta(u, v, peso)`
- `obterAdjacentes(vertice)`
- Inserção/remoção de vértices e arestas

3

Componentes do Sistema

Separação em três classes principais:

- **Grafo:** estrutura de dados
- **Dijkstra:** algoritmo de caminho mínimo
- **Main:** leitura/escrita de arquivos

Algoritmo de Dijkstra: Visão Geral

Inicialização

Distância da origem = 0

Demais vértices = ∞

Mapa de pais vazio

Relaxamento

Extrai vértice com menor distância da PriorityQueue

Atualiza distâncias dos vizinhos

Reconstrução

Usa mapa de predecessores

Gera caminho completo da origem ao destino

Importante: O sistema opera sem interação com usuário, lendo dados de `entrada.txt` e gerando resultados em `saida.txt`

Experimentos Realizados

Critérios de Teste

Foram selecionados quatro cenários estratégicos para validar a implementação:

- **Rota Curta**

  Ligação direta entre bairros adjacentes

- **Rota Média**

  Percurso com múltiplos vértices intermediários

- **Rota Longa**

  Trajeto extenso atravessando vários bairros

- **Teste Aleatório**

  Validação adicional com par não planejado

Objetivos da Validação

1. Verificar capacidade de encontrar caminhos mínimos corretos
2. Confirmar reconstrução adequada dos trajetos via mapa de predecessores
3. Analisar comportamento das distâncias em rotas de diferentes extensões
4. Garantir ausência de ciclos nos caminhos calculados

Resultados Obtidos

Caso	Origem	Destino	Distância	Caminho
Curta	Cohama	Cohafuma	3,2 km	cohama → cohafuma
Média	Cohama	Centro	13,7 km	cohama → cohafuma → calhau → renascença → centro
Longa	Ipase	Aeroporto	23,0 km	ipase → centro → monte-castelo → joão-paulo → coroado → tiririca l → aeroporto
Aleatório	São Cristóvão	Divineia	14,2 km	são-cristovão → tiririca l → coroado → turu → divineia

Todos os caminhos foram reconstruídos corretamente, sem ciclos, com distâncias crescentes proporcionais ao número de arestas percorridas.

Análise dos Resultados

Correção Algorítmica

O algoritmo de Dijkstra identificou corretamente os caminhos mínimos em todos os cenários testados, sem ocorrência de ciclos.

A reconstrução via mapa de predecessores confirmou a integridade das rotas calculadas.

Comportamento Esperado

Rotas curtas utilizam apenas uma ligação direta, enquanto rotas longas exploram múltiplos vértices intermediários.

Distâncias crescentes refletem o comportamento real de redes urbanas.

Eficiência Estrutural

O grafo esparso com densidade ≈ 0.05 beneficia-se do uso de listas de adjacência e PriorityQueue.

Complexidade $O((V + A) \log V)$ adequada para redes urbanas médias.

Conclusão

Objetivos Alcançados

TAD Grafo Genérico

Implementação bem-sucedida com listas de adjacência e API flexível

Algoritmo de Dijkstra

Cálculo correto de rotas mínimas com PriorityQueue eficiente

Validação Prática

Rotas realistas para São Luís com resultados consistentes

Aplicabilidade Prática

A modelagem demonstrou flexibilidade para cenários reais de:

- Navegação GPS urbana
- Logística de entregas
- Planejamento de transporte
- Otimização de rotas

A implementação genérica permite adaptação para outras cidades e domínios sem modificação da estrutura central.

Benefícios e Trabalhos Futuros

Benefícios do Uso de Grafos

- Modelagem natural de redes viárias urbanas
- Representação eficiente de relacionamentos espaciais
- Escalabilidade para redes de diferentes tamanhos
- Base sólida para algoritmos de otimização

Vantagens do Dijkstra

- Garantia de caminho ótimo para pesos não negativos
- Complexidade adequada para aplicações práticas
- Implementação direta com estruturas de dados clássicas
- Amplamente validado e utilizado na indústria

Possibilidades Futuras

- Incorporação de dados de trânsito em tempo real
- Extensão para grafos direcionados (mão única)
- Interface gráfica para visualização interativa
- Integração com APIs de mapas e geolocalização