



**MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO AGRESTE DE PERNAMBUCO**

ALGORITMOS E ESTRUTURA DE DADOS II

TEMA: OTIMIZAÇÃO DE CUSTO E DE EFICIÊNCIA EM VIAGENS PELO AGRESTE DE PERNAMBUCO ATRAVÉS DA APLICAÇÃO DO ALGORITMO DE DIJKSTRA

DISCENTES

GUSTAVO FERREIRA WANDERLEY
JEFFERSON ALAN MONTEIRO MARTINS
RAFAELA FOERSTER DE MENEZES

LISTA DE TABELAS

Tabela 1: Exemplo de iterações do Algoritmo de Dijkstra.....	9
--	---

LISTA DE FIGURAS

Figura 1: Fluxograma de funcionamento do algoritmo.....	5
Figura 2: Mapa geográfico do Pernambuco.....	6
Figura 3: divisão territorial dos municípios do agreste de Pernambuco.....	7
Figura 4: Exemplo simples de um grafo.....	8
Figura 5: imagem do arquivo teste.txt.....	10
Figura 6: logo da plataforma gitHub.....	11
Figura 7: logo da ferramenta Visual Studio Code.....	11
Figura 8: Organização dos arquivos no VS Code.....	12
Figura 8: logo da Linguagem Java.....	13
Figura 9: Saída do algoritmo.....	14

SUMÁRIO

INTRODUÇÃO	5
Contextualização	5
DESENVOLVIMENTO	6
Definição	6
Características	7
Metodologia	9
Tecnologias utilizadas	10
RESULTADOS	13

INTRODUÇÃO

Contextualização

Um indivíduo está planejando uma viagem de pesquisa pelo agreste de Pernambuco, uma região rica em cultura, história e marcada por paisagens deslumbrantes e cidades históricas. Este viajante está interessado em explorar algumas das principais cidades da região como parte de seu projeto de estudo.

No entanto, há um desafio: o viajante possui um prazo pequeno para concluir sua jornada e precisa maximizar seu tempo para visitar todas essas cidades que ele deseja de forma eficiente. Além disso, ele está ciente das possíveis dificuldades das estradas do agreste de Pernambuco, com trechos sinuosos e não pavimentados, o que torna essencial escolher o caminho mais curto e eficaz entre as cidades.

Portanto, o problema que o viajante enfrenta é encontrar a rota mais curta que o leve a todas as cidades desejadas, garantindo que ele otimize o tempo de viagem para cada local. Para resolver esse problema logístico, torna-se necessário utilizar o algoritmo de Dijkstra, uma ferramenta poderosa em teoria dos grafos para determinar a rota ideal entre as cidades que ele planeja visitar. A representação das cidades e as estradas que as conectam são as vértices e as arestas de um grafo, respectivamente, atribuindo os comprimentos das estradas como pesos às arestas.

Ao aplicar o algoritmo de Dijkstra a esse grafo, o viajante será capaz de encontrar a rota mais curta entre as cidades desejadas, levando em consideração tanto a distância quanto o tempo médio de deslocamento em cada trecho. Isso permitirá que ele planeje sua viagem de forma eficiente, economizando tempo e recursos, enquanto visita as fascinantes cidades do agreste de Pernambuco para sua pesquisa.



Figura 2: Mapa geográfico do Pernambuco

DESENVOLVIMENTO

Definição

O Algoritmo de Dijkstra é um dos algoritmos mais importantes na teoria dos grafos e é amplamente utilizado para encontrar o caminho mais curto de um vértice de origem para todos os outros vértices em um grafo ponderado, onde cada aresta tem um peso não negativo. A ideia básica por trás do algoritmo é manter uma lista de distâncias mais curtas conhecidas a partir do vértice de origem para todos os outros vértices. Inicialmente, todas as distâncias, exceto a do vértice de origem, são consideradas infinitas. O algoritmo então explora cada vértice do grafo, atualizando as distâncias mais curtas à medida que encontra caminhos mais curtos.

Visão geral do algoritmo:

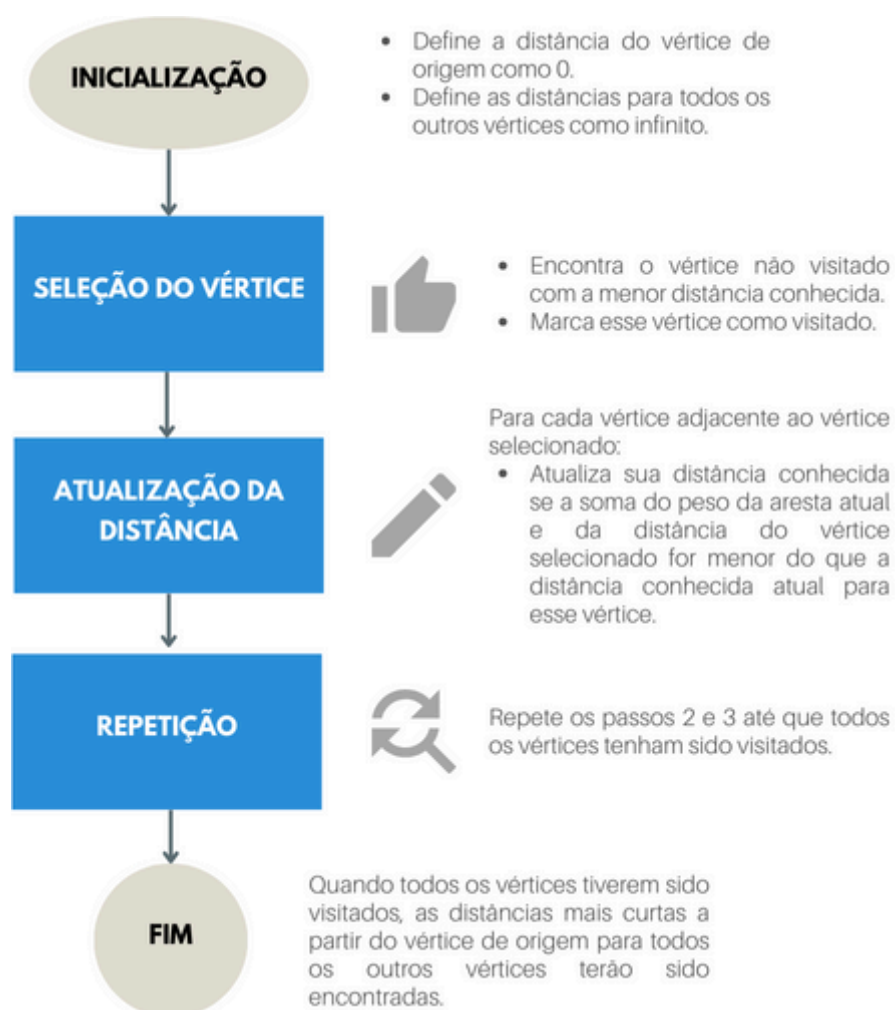


Figura 1: Fluxograma de funcionamento do algoritmo

Em termos de complexidade, o algoritmo de Dijkstra tem uma complexidade de tempo de $O(V^2)$ para implementações simples, onde V é o número de vértices no grafo. No

entanto, utilizando estruturas de dados como filas de prioridade, é possível reduzir a complexidade para $O((V + E)\log V)$, onde E é o número de arestas no grafo.

Características

O algoritmo de Dijkstra é comumente utilizado para encontrar o caminho mais curto entre dois vértices em um grafo ponderado, onde as arestas representam as conexões entre os vértices e possuem um peso associado que indica o custo de se mover de um vértice para outro. No contexto de uma busca de menor caminho entre as cidades do Agreste pernambucano, o grafo pode ser representado da seguinte forma:



Figura 3: divisão territorial dos municípios do agreste de Pernambuco

Suponha que temos um conjunto de cidades representadas por vértices, e as estradas que conectam essas cidades são representadas por arestas ponderadas com os custos associados (nesse caso, a distância).

Por exemplo, considere as cidades Garanhuns, Cachoeirinha, Belo Jardim, Caruaru e Surubim, onde as conexões entre elas são:

- Garanhuns para Cachoeirinha com peso 58
- Garanhuns para Belo Jardim com peso 81
- Cachoeirinha para Caruaru com peso 41
- Belo Jardim para Caruaru com peso 53
- Caruaru para Surubim com peso 64

Neste caso, o grafo correspondente seria:

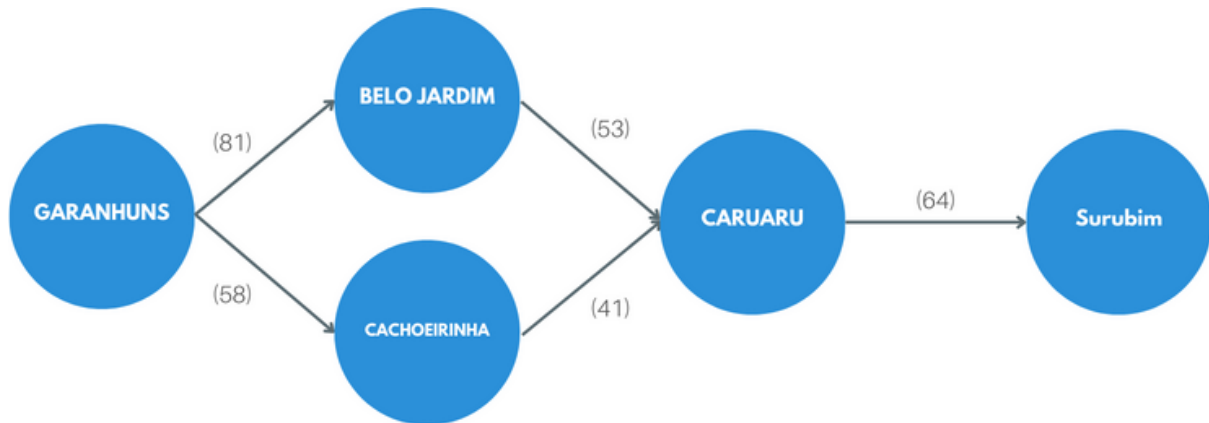


Figura 4: Exemplo simples de um grafo

Neste grafo, os números entre parênteses indicam o peso de cada aresta.

Aplicando o Algoritmo de Dijkstra para encontrar o menor caminho entre Garanhuns e Surubim, temos o seguinte passo a passo:

Passo 1: Inicialização

- Começa-se atribuindo a cidade que será o ponto de partida e a que será o ponto de chegada. A distância, de início, será infinita para as arestas do grafo.
- Defini-se um conjunto de vértices não visitados, inicialmente contendo todos os vértices.

Passo 2: Exploração

- Seleciona-se o vértice não visitado com a menor distância.
- Para cada vizinho desse vértice, calcula-se a distância atualizada, que é a distância do vértice selecionado mais o peso da aresta para esse vizinho, e o tempo gasto, que será contabilizado em minutos e será somado até chegar no ponto de chegada.
- Se a nova distância calculada for menor do que a distância atualmente conhecida para esse vizinho, atualiza-se a distância.
- Marca-se o vértice selecionado como visitado e o remove do conjunto de vértices não visitados.

Passo 3: Repetição

- Repete-se o passo 2 até que todos os vértices sejam visitados ou até que o vértice de destino seja alcançado.

Aqui está a tabela de distâncias atualizada durante as iterações do algoritmo:

Tabela 1: Exemplo de iterações do Algoritmo de Dijkstra

VÉRTICE	DISTÂNCIA DE GARANHUNS
GARANHUNS	0
CACHOEIRINHA	58
BELO JARDIM	81
CARUARU	99
SURUBIM	163

Com essa tabela, podemos ver que o menor caminho de Garanhuns para cada cidade é:

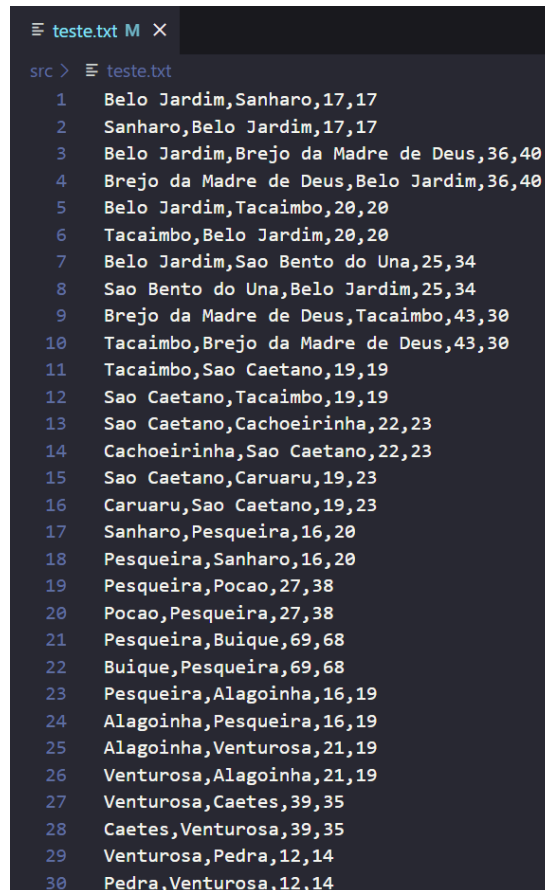
Garanhuns para Garanhuns: 0 (partindo de Garanhuns)
Garanhuns para Cachoeirinha: 58 (Garanhuns -> Cachoeirinha)
Garanhuns para Belo Jardim: 81 (Garanhuns -> Belo Jardim)
Garanhuns para Caruaru: 99 (Garanhuns -> Cachoeirinha -> Caruaru)
Garanhuns para Surubim: 163 (Garanhuns -> Cachoeirinha -> Caruaru -> Surubim)

Assim, usando o algoritmo de Dijkstra, encontramos os menores caminhos de Garanhuns para todas as outras cidades no grafo.

Metodologia

Para o problema abordado, optou-se por utilizar o Algoritmo de Dijkstra para o cálculo do custo mínimo entre as cidades de Pernambuco, para que, assim, o passageiro pudesse escolher a carona mais barata.

Nesse contexto, uma vez que as distâncias (que são os pesos do grafo) já foram estabelecidas junto aos vértices – as cidades em si – e as arestas ponderadas – os caminhos que podem ser percorridos de uma cidade para outra –, a solução pôde ser implementada. Tais dados foram depositados em um arquivo *.txt* com quatro campos: o primeiro, referente ao ponto de partida, o segundo, referente ao ponto de chegada, o terceiro, referente à distância da aresta (caminho do ponto de partida ao ponto de chegada), e o quarto, referente ao tempo gasto em minutos para percorrer esse caminho (o peso da aresta).

A screenshot of a text editor window titled 'teste.txt'. The editor shows a list of 30 lines of text, each representing a location and its coordinates. The lines are numbered 1 through 30 on the left. The text in the editor is as follows:

```
1 Belo Jardim,Sanharo,17,17
2 Sanharo,Belo Jardim,17,17
3 Belo Jardim,Brejo da Madre de Deus,36,40
4 Brejo da Madre de Deus,Belo Jardim,36,40
5 Belo Jardim,Tacaimbo,20,20
6 Tacaimbo,Belo Jardim,20,20
7 Belo Jardim,Sao Bento do Una,25,34
8 Sao Bento do Una,Belo Jardim,25,34
9 Brejo da Madre de Deus,Tacaimbo,43,30
10 Tacaimbo,Brejo da Madre de Deus,43,30
11 Tacaimbo,Sao Caetano,19,19
12 Sao Caetano,Tacaimbo,19,19
13 Sao Caetano,Cachoeirinha,22,23
14 Cachoeirinha,Sao Caetano,22,23
15 Sao Caetano,Caruaru,19,23
16 Caruaru,Sao Caetano,19,23
17 Sanharo,Pesqueira,16,20
18 Pesqueira,Sanharo,16,20
19 Pesqueira,Pocao,27,38
20 Pocao,Pesqueira,27,38
21 Pesqueira,Buique,69,68
22 Buique,Pesqueira,69,68
23 Pesqueira,Alagoinha,16,19
24 Alagoinha,Pesqueira,16,19
25 Alagoinha,Venturosa,21,19
26 Venturosa,Alagoinha,21,19
27 Venturosa,Caetes,39,35
28 Caetes,Venturosa,39,35
29 Venturosa,Pedra,12,14
30 Pedra,Venturosa,12,14
```

Figura 5: imagem do arquivo teste.txt

Tecnologias utilizadas

Para a realização da implementação do grafo para a resolução do problema, utilizou-se as seguintes tecnologias:

GitHub: O GitHub é uma plataforma de desenvolvimento colaborativo que oferece controle de versão robusto com o Git, facilitando o gerenciamento do código do algoritmo de Dijkstra no caso em questão. Com recursos como solicitações de pull e problemas, ele permite uma colaboração eficaz entre os desenvolvedores, enquanto sua ampla adoção e integração com outras ferramentas simplificam o processo de desenvolvimento e revisão de código.

Tal plataforma possibilitou que os códigos pudessem ficar mais organizados e serem manipulados mais facilmente.

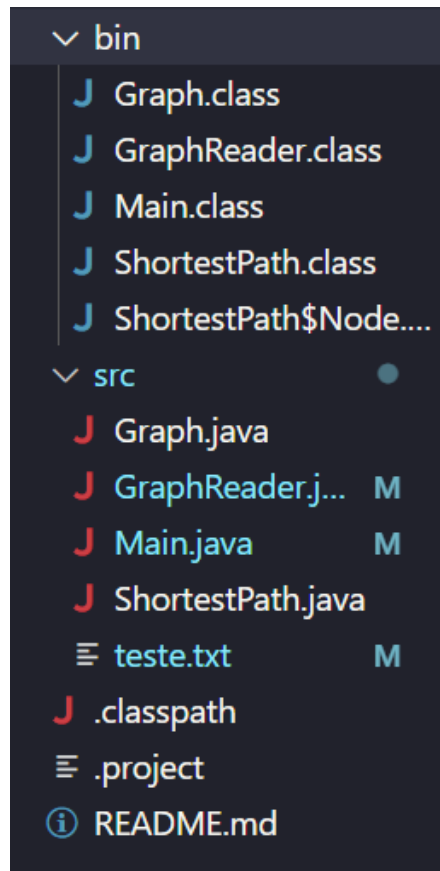


Figura 8: Organização dos arquivos no VS Code

Java: Java é uma linguagem de programação amplamente utilizada, conhecida por sua portabilidade, facilidade de uso e robustez. Ela é especialmente adequada para a implementação do algoritmo de Dijkstra devido a várias razões.

Primeiramente, Java oferece uma ampla gama de bibliotecas padrão que facilitam a manipulação de estruturas de dados complexas, como grafos, que são fundamentais para a execução do algoritmo de Dijkstra. Além disso, a sintaxe clara e orientada a objetos de Java torna mais fácil traduzir os conceitos matemáticos do algoritmo de Dijkstra para código executável.

Outro aspecto importante é a segurança proporcionada pelo Java. Como uma linguagem fortemente tipada e com um sistema de gerenciamento de memória automático, Java ajuda a evitar erros comuns de programação, como vazamentos de memória e acesso indevido a variáveis.

Além disso, Java é altamente interoperável e pode ser executado em uma ampla variedade de plataformas, desde computadores desktop até dispositivos móveis e servidores web. Isso significa que o código Java desenvolvido para implementar o algoritmo de Dijkstra pode ser facilmente integrado a outros sistemas e aplicativos.



Figura 8: logo da Linguagem Java

Após a escolha das tecnologias utilizadas para a realização do trabalho, seguimos para a criação das classes necessárias para a criação do grafo e do funcionamento do algoritmo de Dijkstra

RESULTADOS

Para aplicar o algoritmo de Dijkstra ao grafo que representa as cidades do Agreste de Pernambuco, começamos de um vértice inicial (neste caso, Belo Jardim) e calculamos as distâncias mínimas para todos os outros vértices do grafo. O algoritmo explora os vértices vizinhos e atualiza as distâncias mínimas conforme necessário. Este processo continua até que todos os vértices sejam visitados ou o vértice de destino seja alcançado.

O algoritmo de Dijkstra garante que, quando terminar, teremos encontrado o menor caminho e o menor tempo de viagem de Belo Jardim para qualquer outro vértice no grafo. Se houver múltiplos caminhos entre duas cidades, o algoritmo escolherá o caminho mais curto com base nos pesos das arestas.

Dessa forma, ao executar o programa com o algoritmo de Dijkstra, obteremos o seguinte resultado:

```
Diretório atual: C:\Users\jeffe\OneDrive\Área de Trabalho\AEDII_Project
Aguas Belas -> Paratama (Distância: 67 km, Tempo: 66 minutos)
Paratama -> Garanhuns (Distância: 23 km, Tempo: 26 minutos)
O tempo total da rota mais curta entre Aguas Belas e Garanhuns é: 1 horas e 32 minutos
Distância total percorrida: 90 km
```

Figura 9: Saída do algoritmo

Em conclusão, a aplicação do algoritmo de Dijkstra proporcionou uma valiosa análise das rotas entre as cidades do Agreste de Pernambuco, fornecendo informações precisas sobre as distâncias mínimas e os tempos de viagem associados. Esses resultados são essenciais para planejamento logístico, otimização de rotas e tomada de decisões estratégicas em diversas áreas, como transporte, turismo e desenvolvimento regional. Ao garantir a identificação do caminho mais eficiente entre os pontos de interesse, o algoritmo de Dijkstra demonstra sua utilidade como uma ferramenta poderosa na resolução de

problemas de otimização de trajetos em redes complexas, contribuindo para uma mobilidade mais eficaz e econômica na região do Agreste pernambucano.