MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO AGRESTE DE PERNAMBUCO

ALGORITMOS E ESTRUTURA DE DADOS II

TEMA: APLICAÇÃO DO ALGORITMO DE DIJKSTRA PARA CALCULAR O CUSTO MÍNIMO ENTRE AS CIDADE DO MUNDO FICTÍCIO DE GAME OF THRONES.

DISCENTES:

LUIZ FELLIPE DE ALMEIDA RODRIGUES BARBOSA;
MARIA EDUARDA DEODATO INTERAMINENSE.

GARANHUNS/2022

Sumário

Definição

O algoritmo de Dijkstra é um dos algoritmos que pode-se aplicar para encontrar o caminho de menor custo entre os vértices em um grafo. Sendo assim, gerando uma árvore de custo-mínimo. Esse algoritmo calcula o custo mínimo de um vértice, que é o vértice escolhido como pai na busca, para todos os outros vértices do grafo. Esse é um algoritmo simples e que tem uma boa performance. Entretanto, ele não dá a garantia de uma solução exata, caso exista presença de arcos que possuem valores negativos.

Dispositivos que possuem GPS com objetivo de encontrar o caminho mais curto e a localização atual e o destino é uma aplicação do algoritmo de Dijkstra, ele é um algoritmo que possui flexibilidade podendo atingir outras áreas para sua aplicação, além disso também é possível aplicar na indústria e em domínios que necessitam de modelagem de redes.

Esse algoritmo foi desenvolvido pelo holandês, cientista da computação e engenheiro de software Dr. Edsger W. Dijkstra em 1959 em um artigo *"A note on two problems in connexion with graphs"* (Uma nota sobre dois problemas em conexão com grafos, em tradução livre) onde ele detalhou sobre o algoritmo.

Esse algoritmo inicia escolhendo uma vértice, o qual será sua origem e a partir dessa raiz, ele analisa o menor caminho entre os outros vértices, enquanto isso ele armazena o caminho mais curto, quando ele encontrar o caminho de custo mínimo entre o nó raiz e o destino ele marca como visitado e adiciona ao caminho. Com isso, o processo segue até que todos os vértices tenham sido adicionados ao caminho. Desta forma, obtemos um caminho que liga o vértice de origem a outros vértices seguindo o menor custo para chegar a cada destino.

Contextualização

O problema: Daenerys, personagem do Mundo de Game of Thrones precisa se deslocar de uma cidade para outra em seu dragão para poder dominar o trono. Entretanto, existem vários caminhos que passam por diversas cidades. Porém, para que ela conquiste o trono é necessário uma trajetória que percorre o menor caminho entre as cidades chaves, para que ela conquiste e atinja seu objetivo.

E para a solução desse problema foi escolhido o algoritmo de Dijkstra, para calcular o custo mínimo entre essas cidades e assim conseguir o melhor caminho para Daenerys.

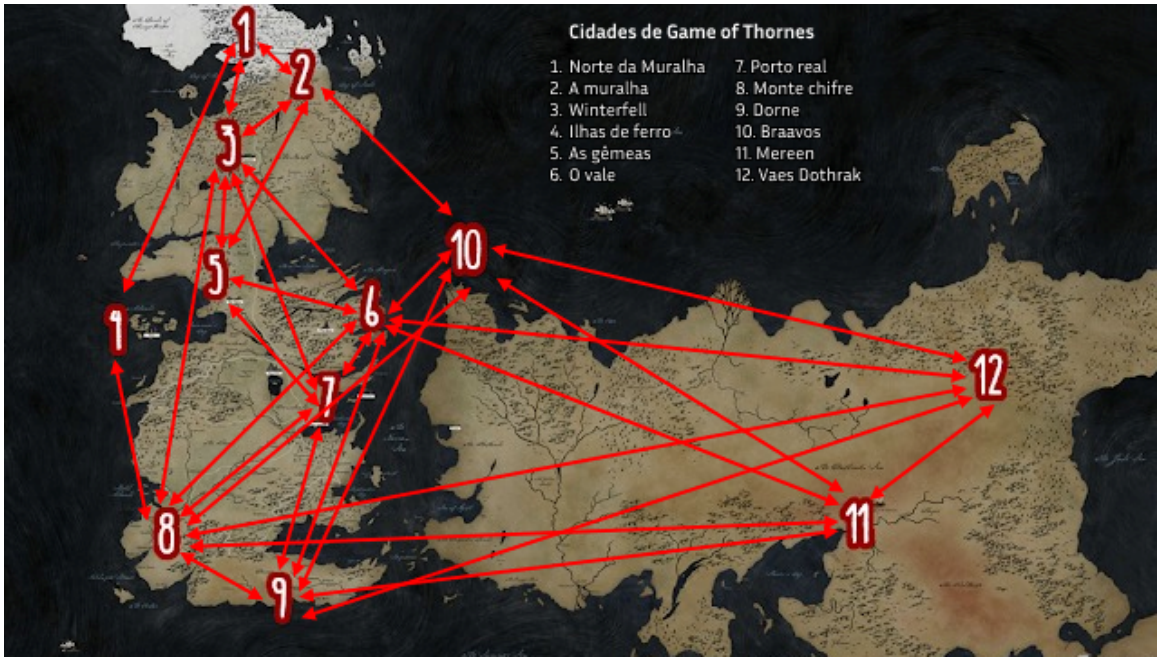
Características

O Algoritmo de Dijkstra só funciona de forma exata com grafos que têm pesos positivos. Isso porque, durante o processo, os pesos das arestas devem ser somados para encontrar o caminho de menor custo.

O algoritmo utilizado para determinar a rota de menor caminho é uma adaptação do algoritmo de Dijkstra. A mudança no método de Dijkstra foi feita dentro da interface gráfica, quando foram criados os vértices e as arestas, obrigando que o caminho de custo mínimo envolvesse necessariamente todos os vértices do grafo.

Na nossa problemática, o que soma-se são as distâncias entre as cidades, para poder-se calcular essa rota de menor custo. Os vértices representam as cidades e as arestas direcionadas são o caminho a ser percorrido, tanto indo, quanto voltando. No algoritmo proposto, considera-se que todos os vértices do grafo estão ligados entre si e que o custo de uma aresta independe do sentido do percurso.

Mapa das cidades e grafo gerado a partir dele:

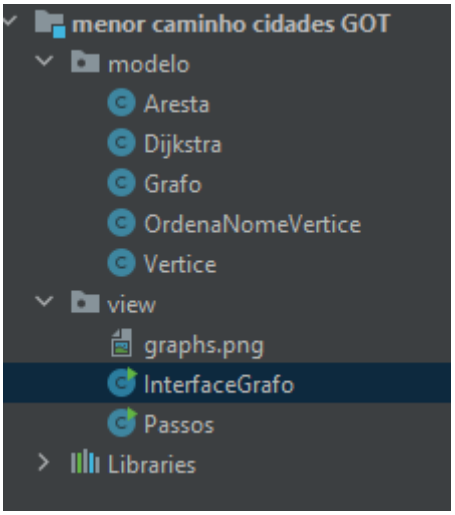


Metodologia

Foi usado o algoritmo de Dijkstra justamente para resolvermos a nossa problemática, calcular o caminho de custo mínimo entre as cidades de GOT.

Tinha-se as distâncias de uma cidade para a outra, então, fornece-se essa informação ao algoritmo, juntamente com os vértices e as arestas específicas, que foram definidas no código, assim o programa funciona.

Utilizamos interface gráfica, fornecida pela biblioteca ***"javax.swing.JFrame"***, a qual nos permitiu criar uma tela, onde nela, temos o mapa referente a essas cidades e pode-se fazer a comparação entre a distância de qualquer cidade para outra e o resultado será o melhor caminho possível, da origem para o destino.



Observando a imagem acima, tem-se uma melhor ideia de como o algoritmo foi pensado e construído. A parte modelo é composta pelos códigos que fomentam e são necessários para posteriormente criar-se a interface gráfica e seu grafo correspondente.

Já na parte View, tem-se a interface propriamente dita, a que vai gerar o visual, com as funções que tem e irá mostrar o que pode ser feito no programa. Quando compilado, apresentará o mapa e as cidades que pode-se calcular seu destino. O código de passos vem para complementar a interface gráfica e fazê-la funcionar mais completa, sendo através dele, possível observar, na interface, o que o programa está fazendo quando calculamos a rota, ou seja, o passo a passo.

Resultados

Funcionamento do algoritmo de Dijkstra propriamente dito:

Passo 1: Todos os vértices do grafo são considerados como não visitados

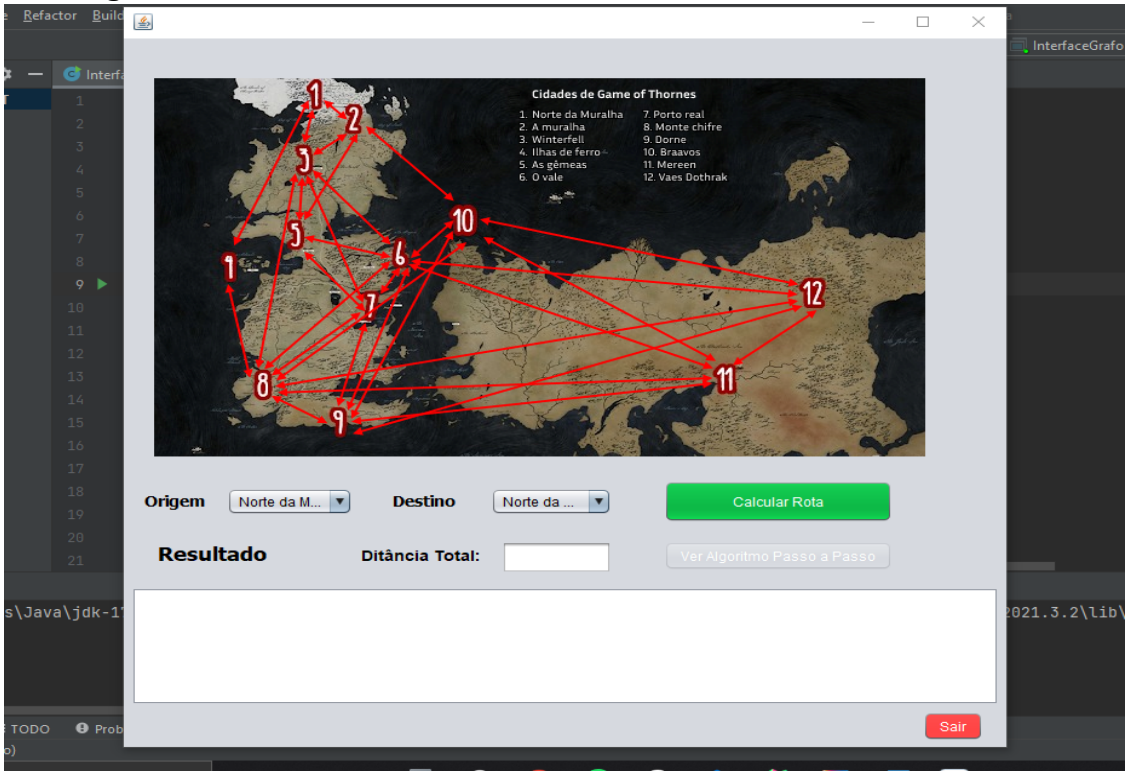
Passo 2: Uma distância é colocada para cada vértice. A origem do caminho tem distância 0 (você não gasta nada para chegar lá, é seu ponto de partida!), enquanto os outros vértices têm distância infinita (você ainda não chegou em nenhum deles). Além disso, para cada vértice vai ter uma informação anterior, para informar a partir de que outro ponto do grafo o caminho até ele foi construído;

Passo 3: O vértice de origem passa a ser seu vértice atual de análise, onde você vai realizar os seguintes passos: Atribuir distância a todos os vizinhos não visitados. Essa distância é o custo de chegar até o vizinho (peso da aresta) mais o custo para chegar até o vértice atual. Caso já exista uma distância diferente de infinito, ficará a menor entre a que já existia e a que acabou de ser calculada; Caso a distância atual seja a menor, alterar o vértice anterior do vizinho para o vértice atual;

Passo 4: O vértice atual é marcado como visitado;

Passo 5: Um novo vértice não visitado é escolhido para ser o vértice atual, e os procedimentos do passo 3 são repetidos até não existirem mais vértices não visitados.

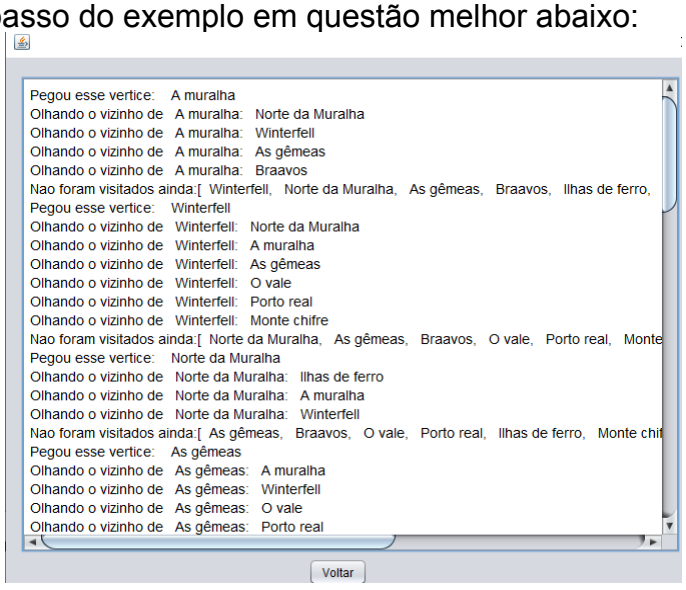
Interface gráfica em funcionamento:

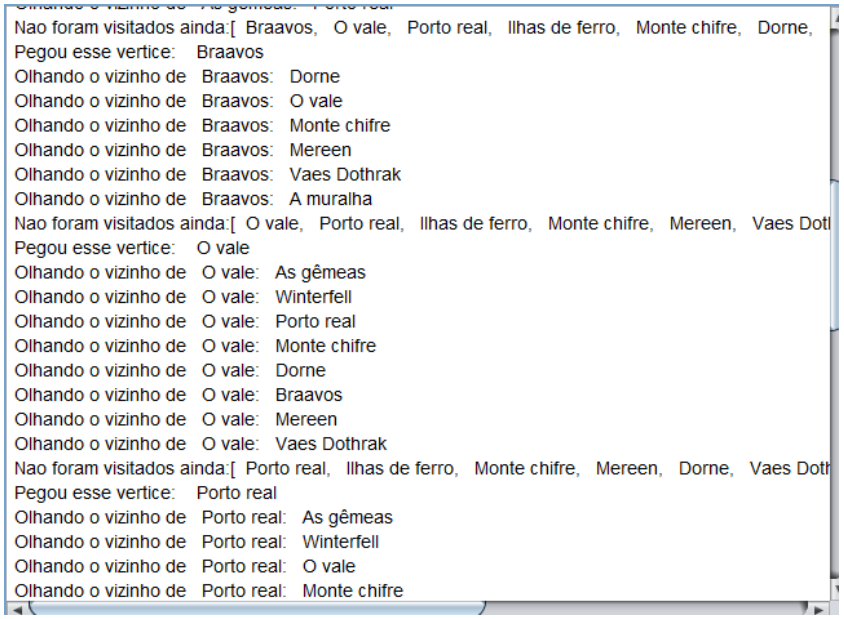


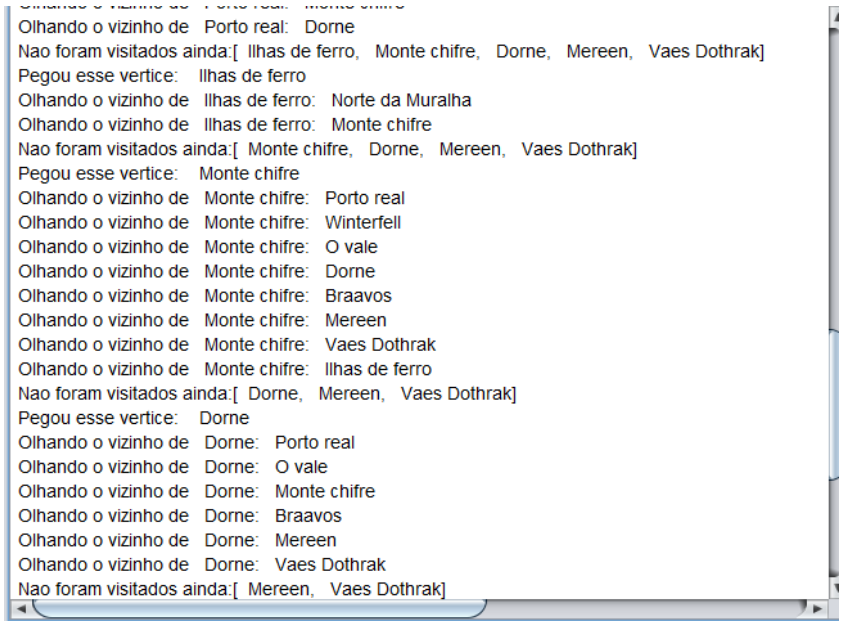
Vamos agora observar o algoritmo em funcionamento, para o teste vamos supor que Daenerys está em 2- A muralha e quer chegar em 9- Dorne, ao clicar calcular rota, pode-se ver o percurso de custo mínimo necessário a ser feito, observa-se melhor isso pela imagem abaixo:

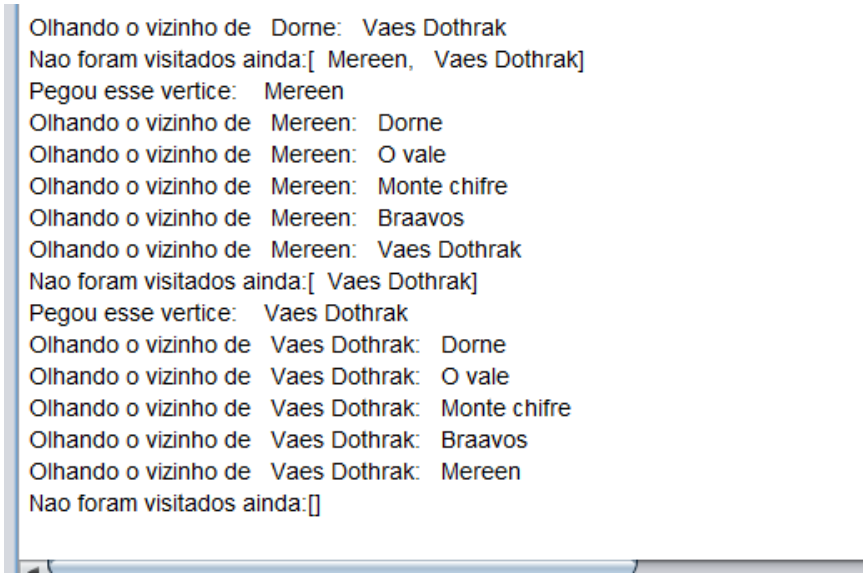
O percurso ficou, **origem: 2- A Muralha**, >> 3- As Gêmeas >> 7- Porto Real, **destino: 9- Dorne**. Totalizando uma jornada de 195 km, a qual seria a maneira mais otimizada de chegar ao seu destino.

Se clicarmos em “ver algoritmo **passo a passo**”, é possível ainda com mais detalhes ver como Dijkstra rodou e funcionou dentro do que foi solicitado. Vê-se esse passo a passo do exemplo em questão melhor abaixo:




Nao foram visitados ainda:[Braavos, O vale, Porto real, Ilhas de ferro, Monte chifre, Dorne,
Pegou esse vertice: Braavos
Olhando o vizinho de Braavos: Dorne
Olhando o vizinho de Braavos: O vale
Olhando o vizinho de Braavos: Monte chifre
Olhando o vizinho de Braavos: Mereen
Olhando o vizinho de Braavos: Vaes Dothrak
Olhando o vizinho de Braavos: A muralha
Nao foram visitados ainda:[O vale, Porto real, Ilhas de ferro, Monte chifre, Mereen, Vaes Dotl
Pegou esse vertice: O vale
Olhando o vizinho de O vale: As gêmeas
Olhando o vizinho de O vale: Winterfell
Olhando o vizinho de O vale: Porto real
Olhando o vizinho de O vale: Monte chifre
Olhando o vizinho de O vale: Dorne
Olhando o vizinho de O vale: Braavos
Olhando o vizinho de O vale: Mereen
Olhando o vizinho de O vale: Vaes Dothrak
Nao foram visitados ainda:[Porto real, Ilhas de ferro, Monte chifre, Mereen, Dorne, Vaes Dotl
Pegou esse vertice: Porto real
Olhando o vizinho de Porto real: As gêmeas
Olhando o vizinho de Porto real: Winterfell
Olhando o vizinho de Porto real: O vale
Olhando o vizinho de Porto real: Monte chifre


Olhando o vizinho de Porto real: Dorne
Nao foram visitados ainda:[Ilhas de ferro, Monte chifre, Dorne, Mereen, Vaes Dothrak]
Pegou esse vertice: Ilhas de ferro
Olhando o vizinho de Ilhas de ferro: Norte da Muralha
Olhando o vizinho de Ilhas de ferro: Monte chifre
Nao foram visitados ainda:[Monte chifre, Dorne, Mereen, Vaes Dothrak]
Pegou esse vertice: Monte chifre
Olhando o vizinho de Monte chifre: Porto real
Olhando o vizinho de Monte chifre: Winterfell
Olhando o vizinho de Monte chifre: O vale
Olhando o vizinho de Monte chifre: Dorne
Olhando o vizinho de Monte chifre: Braavos
Olhando o vizinho de Monte chifre: Mereen
Olhando o vizinho de Monte chifre: Vaes Dothrak
Olhando o vizinho de Monte chifre: Ilhas de ferro
Nao foram visitados ainda:[Dorne, Mereen, Vaes Dothrak]
Pegou esse vertice: Dorne
Olhando o vizinho de Dorne: Porto real
Olhando o vizinho de Dorne: O vale
Olhando o vizinho de Dorne: Monte chifre
Olhando o vizinho de Dorne: Braavos
Olhando o vizinho de Dorne: Mereen
Olhando o vizinho de Dorne: Vaes Dothrak
Nao foram visitados ainda:[Mereen, Vaes Dothrak]


Olhando o vizinho de Dorne: Vaes Dothrak
Nao foram visitados ainda:[Mereen, Vaes Dothrak]
Pegou esse vertice: Mereen
Olhando o vizinho de Mereen: Dorne
Olhando o vizinho de Mereen: O vale
Olhando o vizinho de Mereen: Monte chifre
Olhando o vizinho de Mereen: Braavos
Olhando o vizinho de Mereen: Vaes Dothrak
Nao foram visitados ainda:[Vaes Dothrak]
Pegou esse vertice: Vaes Dothrak
Olhando o vizinho de Vaes Dothrak: Dorne
Olhando o vizinho de Vaes Dothrak: O vale
Olhando o vizinho de Vaes Dothrak: Monte chifre
Olhando o vizinho de Vaes Dothrak: Braavos
Olhando o vizinho de Vaes Dothrak: Mereen
Nao foram visitados ainda:[]

Considerações Finais

Estudando sobre os grafos, seus diferentes tipos e definições, pode-se concluir o quão são úteis, para resolver diversos tipos de problemas. Sendo observado que para cada problema, existirá um algoritmo de melhor caso para a solução.

Também percebe-se que o algoritmo de Dijkstra é ineficiente para trabalhar com pesos negativos, pois fica traiçoeiro e muito mais difícil. Sendo mais viável e eficiente trabalhar com pesos positivos.

Na nossa problemática, usamos o que inferimos ser a solução mais adequada para a situação criada e isso foi comprovado através dos testes. Também, com a utilização da interface gráfica, foi trazido uma facilidade e eficácia para o programa e a aprendizagem não ficou de fora, muito se foi aprendido ao realizarmos a construção desses algoritmos para esse projeto.

Conclui-se que, o objetivo proposto para o presente trabalho foi atendido, ou seja, o método utilizado permitiu definir uma rota mínima entre as cidades de GOT para a Daenerys conquistar o seu trono, da maneira mais eficiente possível.