Licenciatura em Engenharia Informática



**Estruturas de Informação**

**2ª Parte**

Ano Letivo 2019/2020

Breno Pacheco – 1180005

João Ferreira – 1181436



Novembro 2020

Índice

[Diagrama de Classes 3](#_Toc23942913)

[Construção do grafo 4](#_Toc23942914)

[Colorir mapa 5](#_Toc23942915)

[Caminho mais curto entre duas capitais 5](#_Toc23942916)

[Caminho mais curto entre duas capitais passando por outras capitais 8](#_Toc23942917)

[Circuito de menor comprimento 10](#_Toc23942918)

[Conclusão 11](#_Toc23942919)

[Bibliografia 12](#_Toc23942920)

# Diagrama de Classes



Figura – Diagrama de classes do projeto criado

# Construção do grafo

Para representar os vários países e as suas fronteiras recorreu-se à representação por via de um grafo. Sendo os nós e os vértices os países e as fronteiras carregados a partir dos ficheiros fornecidos.

Para isso, começou-se por carregar os países, criar os respetivos nós e inseri-los na instância de grafo criada. Com este objetivo foi criado o método carregarPaises:

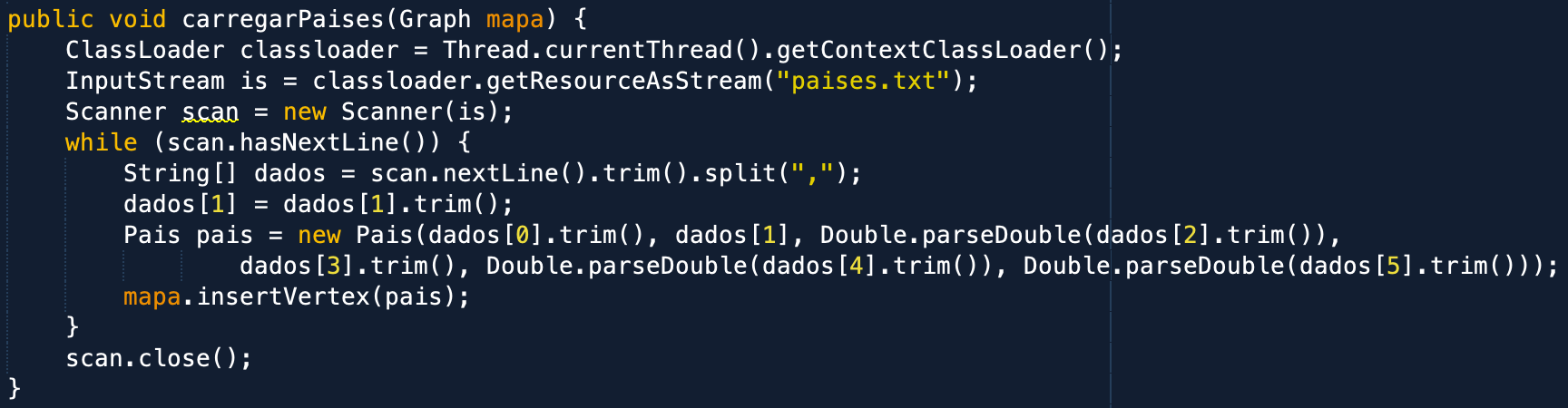


Figura – Implementação do método carregarPaíses

De seguida foram carregadas as fronteiras para que pudessem ser criados os vértices do grafo. Sendo que o peso de cada vértice corresponde à distância entre os países. O cálculo dessa distância foi feito recorrendo às coordenadas geográficas de cada país [1]. Para isso, foi criado o método carregarFronteiras:

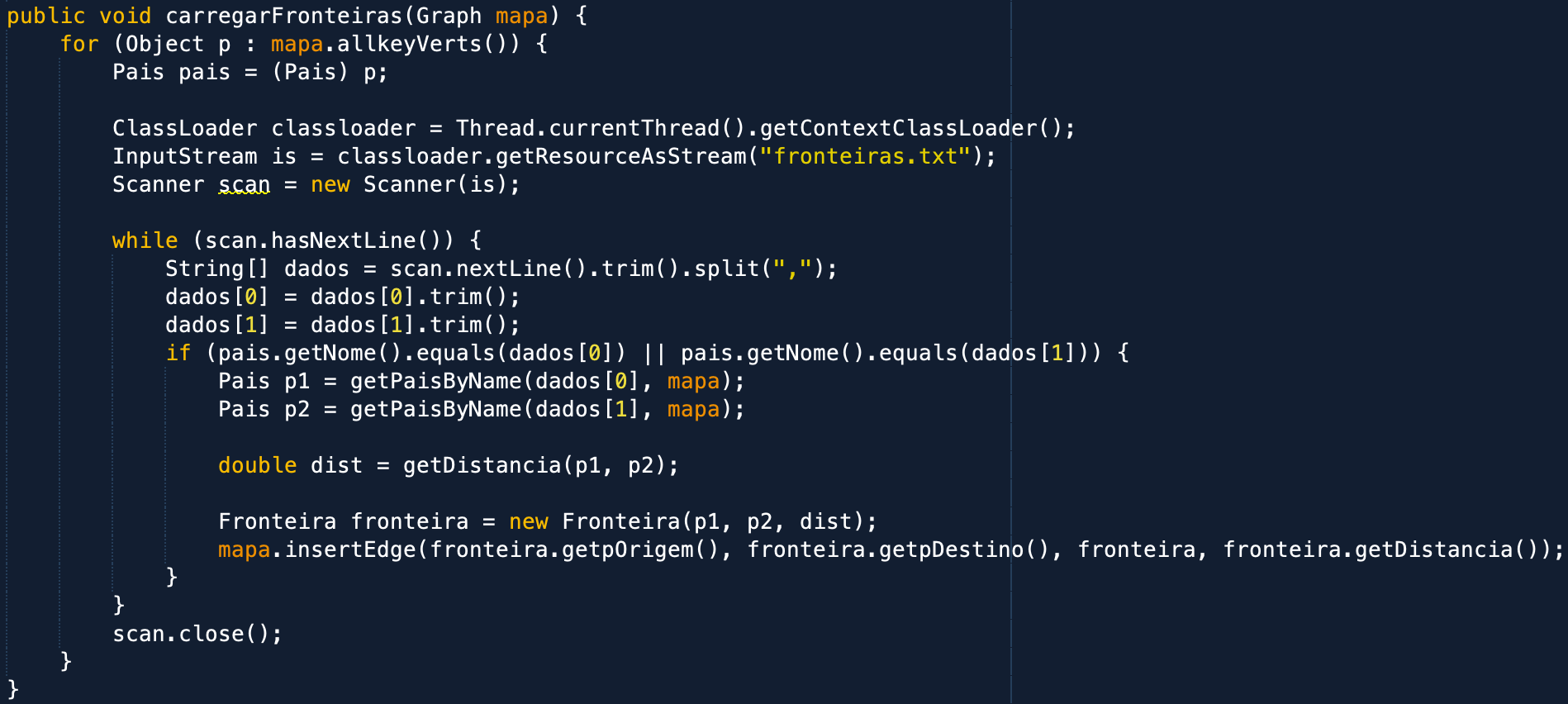


Figura – Implementação do método carregarFronteiras

# Colorir mapa

A problemática de colorir os vários vértices de um grafo é um problema NP-completo. Ou seja, não existem algoritmos com complexidade temporal que obtenham uma resposta exata para o problema. Como tal, recorrem-se a algoritmos com soluções aproximadas [2].

Foram então testados três algoritmos distintos, ambos considerados algoritmos *greedy*. O algoritmo *gree*dy genérico com ordenação por grau de saída, algoritmo de Welsh-Powell [3] e o algoritmo DSatur [4]. O que os distingue é a heurística utilizada. Enquanto que o primeiro segue a ordem dos vértices pelo grau de saída, o segundo segue tenta colorir todos os vértices com uma cor de cada vez e, por fim, o último segue a heurística do próximo nó não colorido com a maior saturação. Ou seja, escolhe sempre o nó seguinte cujos países vizinhos já coloridos possuem a maior variedade de cores. Em ambos os casos a complexidade temporal é O(V2) [3] [4].



Figura -Implementação do algoritmo de Welsh-Powell

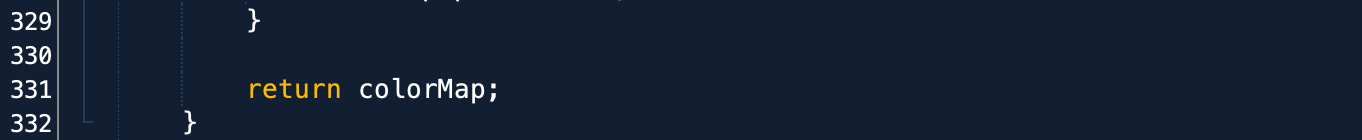


Figura – Implementação do algoritmo greedy com ordenação inicial dos vértices por grau de saída

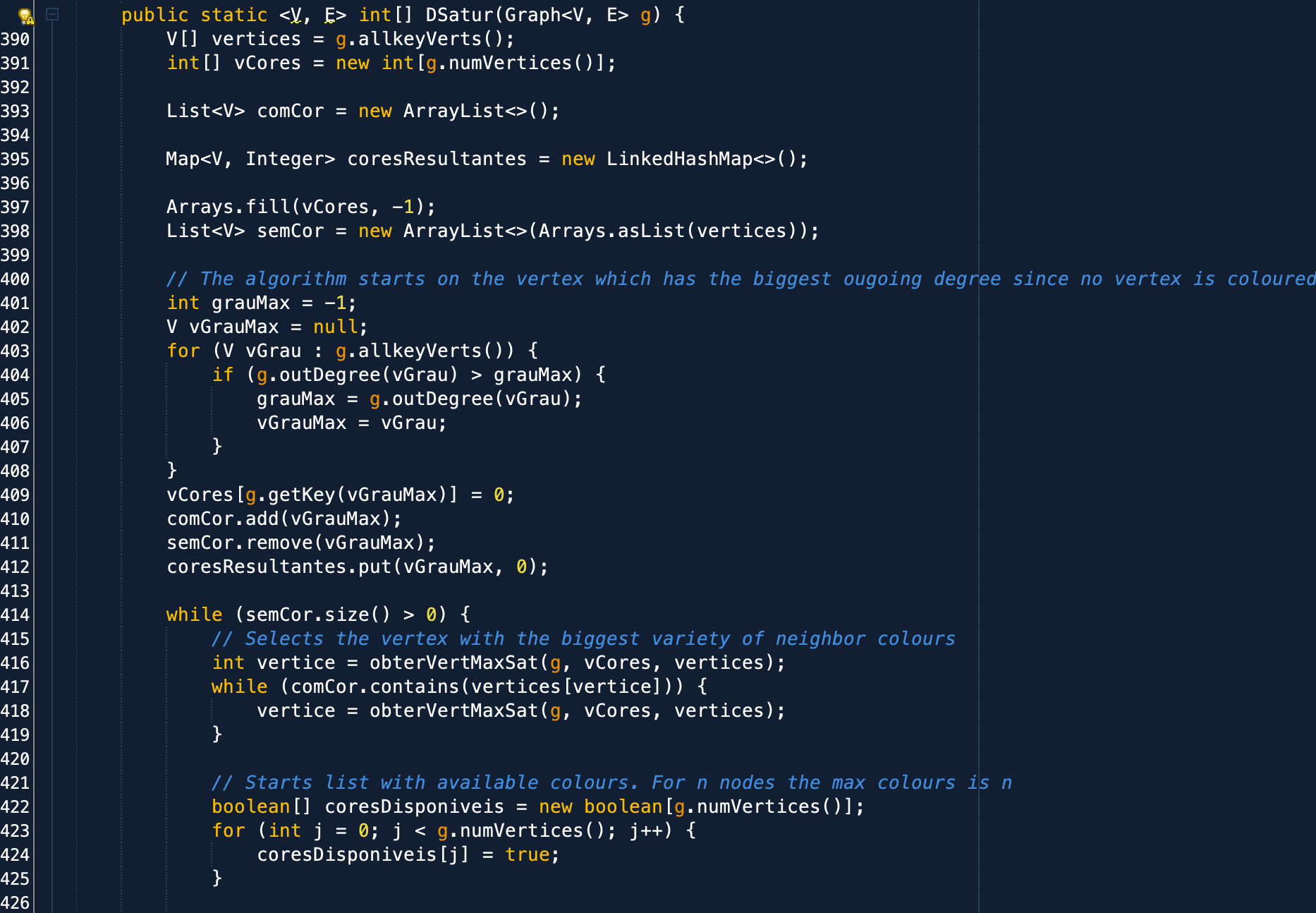


Figura - Implementação do algoritmo DSatur

Como referido anteriormente, estes algoritmos apenas garantem uma solução aproximada do ideal. Não garantindo uma solução exata. Para se garantir uma solução exata teriam de ser usados algoritmos com uma complexidade bastante superior como é o caso dos algoritmos *brute force* [5].

Apesar de todos os algoritmos terem conseguido colorir o mapa utilizando 4 cores. Considerou-se que os algoritmos que necessitavam de uma ordenação pelos graus de saída carecem de mais cuidados na implementação uma vez que não só é necessária a ordenação inicial dos nós como é necessário analisar com especial cuidado vértices que possuem graus de saída iguais.

# Caminho mais curto entre duas capitais

Para a determinação do caminho mais curto entre duas capitais foi aplicado o algoritmo de Dijkstra. Este algoritmo tem complexidade temporal O(V log V) e é considerado *greedy* [6] uma vez que opta pela opção ótima numa determinada iteração sem ter em atenção problemas que possam derivar dessa decisão. Fazendo sempre escolhas definitivas. Como alternativa poderia ter sido utilizado o algoritmo de Bellman-Ford que apesar de ter uma implementação mais simples apresenta uma complexidade superior ao de Dijkstra sendo esta O(VE) [7].

Para isto foram implementados três métodos:

1. shortestPath: Método que retorna o caminho mais curto entre dois países



Figura – Implementação do método shortestPath

1. shortestPaths: Método que retorna os caminhos mais curtos entre um determinado país e todos os outros



Figura - Implementação do método shortestPaths

1. shortestPathLength: Método que aplica o algoritmo de Dijkstra e determina os caminhos mais curtos entre um determinado país e todos os outros



Figura – Implementação do método shortestPathLength. Que contém a implementação do algoritmo de Dijkstra

# Caminho mais curto entre duas capitais passando por outras capitais

Para a determinação do caminho mais curto entre duas capitais passando obrigatoriamente por outras capitais intermédias foi necessário calcular todos os caminhos possíveis entre a capital de origem, as diferentes capitais intermédias e a capital de destino. Após isto determinou-se qual dos vários caminhos apresenta o comprimento mais curto e qual o seu comprimento. A complexidade temporal deste algoritmo é O(V!).

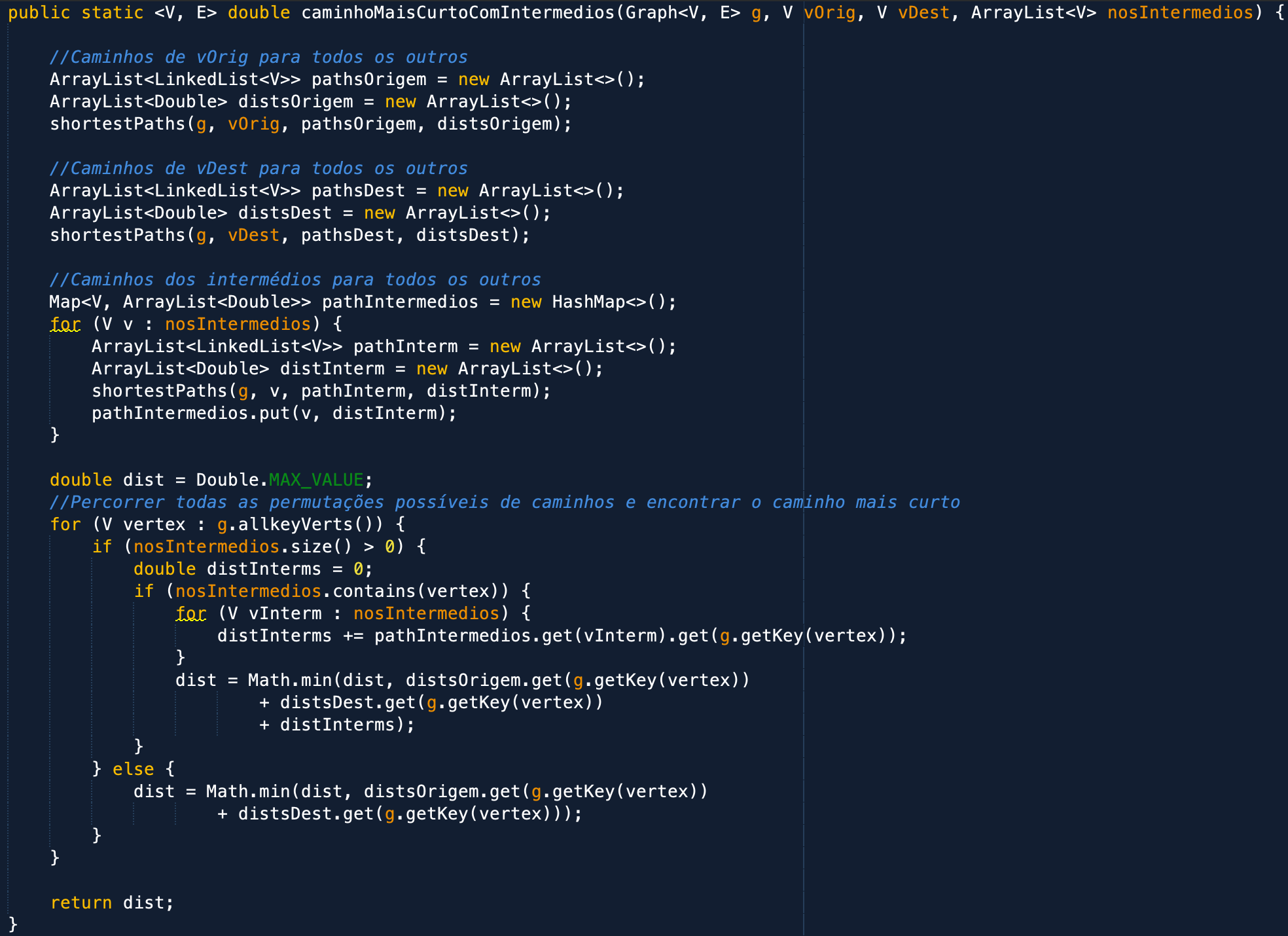


Figura 10 – Implementação do método caminhoMaisCurtoComIntermédios

# Circuito de menor comprimento

Neste exercício os países são visitados a partir de uma origem utilizando a heurística do vizinho mais próximo realizando um circuito fechado. Para tanto, constrói-se para cada país visitado uma fila ordenada pela distância entre a capital deste país e a capital dos países adjacentes.

Visita-se o primeiro país da fila de forma recursiva e, sendo possível chegar a partir deste nó no país destino (ou origem neste caso) propaga-se a informação de que o circuito foi fechado e devolve-se o vetor de países visitados. Não sendo possível completar o circuito utilizando o primeiro país da fila de adjacências, este é considerado um beco-sem-saída e adicionado à uma lista para que não seja novamente visitado, sendo o país mais próximo seguinte então visitado.

Cada país visitado é adicionado a uma lista para que não sejam novamente visitados dentro da pilha de chamadas recursvas em que se encontra. Isto evita que ocorram “loops” no algoritmo, como por exemplo a visita seguida de Paraguai para Argentina para Paraguai.

Foram criadas duas funções para conclusão deste algoritmo. A função nearestNeighborCycle é chamado na função main(), cria as estruturas de dados necessárias e chama a função recursiva cycleRecursive. implementa a heurística do vizinho mais próximo, cria a fila de prioridades para visitas e realiza as visitas de forma recursiva, não permitindo a visita de países “impossiveis” e já visitados. O tipo de retorna é booleano, pois permite propagar à pilha de chamadas da função se o vértice de destino foi ou não encontrado a partir de um “branch” das chamadas à função.

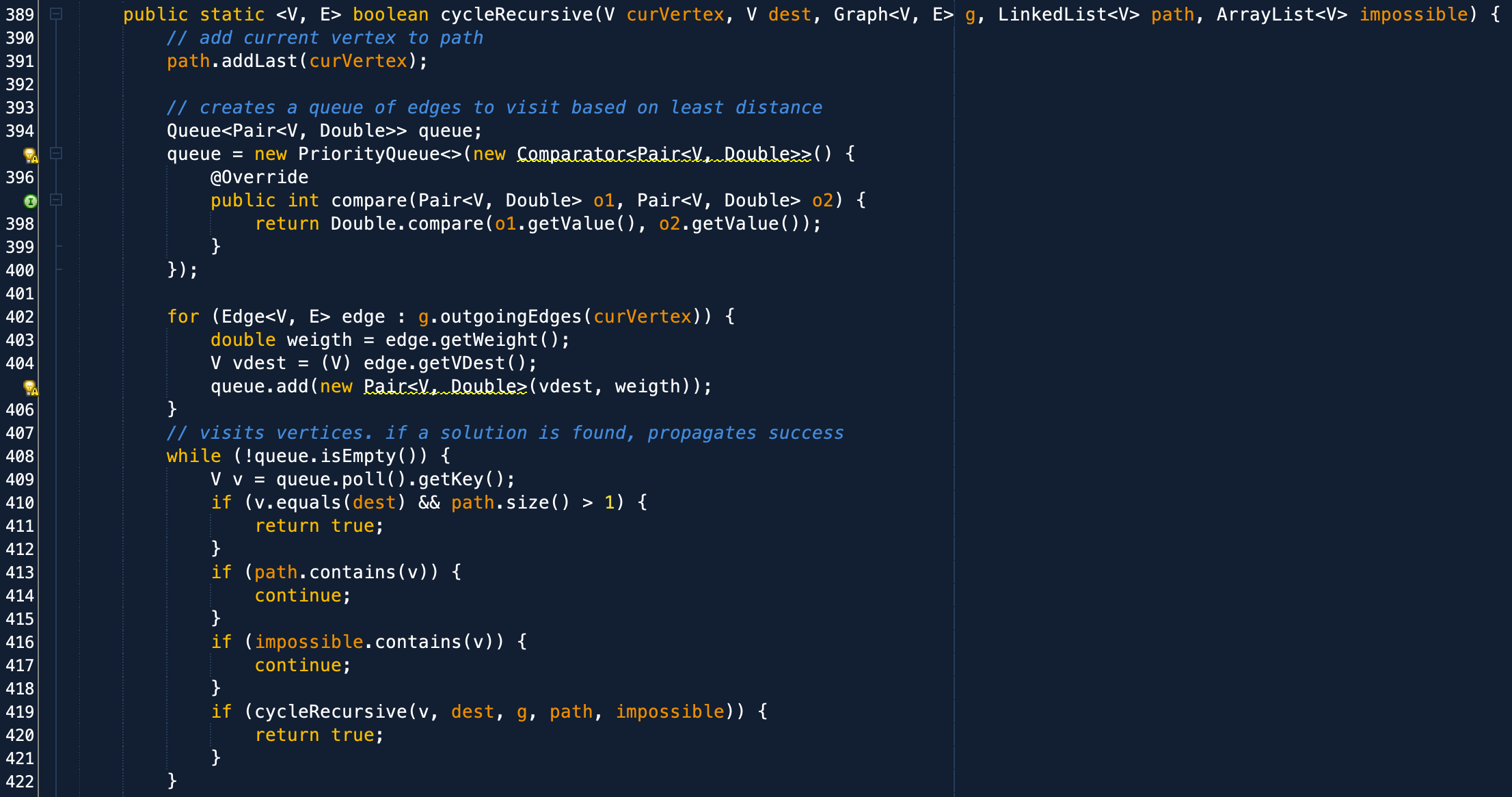


Figura - Algoritmo implementado para o circuito de menor comprimento

# Conclusão

Este trabalho permitiu consolidar conhecimentos relativamente à implementação de uma estrutura de grafo recorrendo às várias estruturas de dados associados ao Java. Foi também relevante para o treino de implementação e aplicação de diferentes algoritmos, já bastantes discutidos na bibliografia, em exemplos concretos e práticos. Assim como, ganhar uma maior perceção acerca da influência de algoritmos com uma elevada complexidade temporal.

**Bibliografia**

|  |  |
| --- | --- |
| [1] | M. T. Ltd, "Calculate distance, bearing and more between Latitude/Longitude points," 6 Novembro 2019. [Online]. Available: http://www.movable-type.co.uk/scripts/latlong.html. |
| [2] | R. M. R. Lewis, A Guide to Graph Colouring: Algorithms and Applications, Cardiff, UK: Springer, 2016. |
| [3] | R. E. Campello and N. Maculan, Algoritmos e Heurísticas: Desenvolvimento e Avaliação de Performance, Niterói: EDUFF, 1994. |
| [4] | D. Brélaz, “New Methods to Color the Vertices of a Graph,” *Commun. ACM,* vol. 22, pp. 251-256, April 1979. |
| [5] | O. IQ, “Overview of Graph Colouring Algorithms,” [Online]. Available: https://iq.opengenus.org/overview-of-graph-colouring-algorithms/. [Accessed 8 Novembro 2019]. |
| [6] | U. d. S. Paulo, “Algoritmos Gulosos,” 6 Novembro 2019. [Online]. Available: https://www.ime.usp.br/~pf/analise\_de\_algoritmos/aulas/guloso.html. |
| [7] | G. f. Geeks, “Bellman-Ford Algorithm,” 6 Novembro 2019. [Online]. Available: https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/. |