Relatório do Trabalho Final de Algoritmos

Relatório elaborado por: Carlos Silva Nº21583

[Introdução 3](#_Toc138194460)

[Ferramentas Utilizadas 4](#_Toc138194461)

[Objetivo 1- Implementação do Carteiro Chinês 5](#_Toc138194462)

[História 5](#_Toc138194463)

[Propósito 5](#_Toc138194464)

[Problemas 5](#_Toc138194465)

[Aplicação 5](#_Toc138194466)

[Objetivo 2- Prim e Kruskal 6](#_Toc138194467)

[História 6](#_Toc138194468)

[Propósito 6](#_Toc138194469)

[Problemas 6](#_Toc138194470)

[Aplicação 6](#_Toc138194471)

[Objetivo 3- Caixeiro Viajante(VMP e Clarke&Wright) 7](#_Toc138194472)

[História 7](#_Toc138194473)

[Propósito 7](#_Toc138194474)

[Problemas 7](#_Toc138194475)

[Aplicações 7](#_Toc138194476)

[Webgrafia 8](#_Toc138194477)

[Webgrafia dentro do código 8](#_Toc138194478)

# Introdução

Neste relatório será documentado o processo atrás da construção e implementação dos objetivos pedidos pela Professora Manuela Fernandes, estes sendo:

A implementação do teorema Carteiro Chinês em contexto de “Distribuição otimizada da Vacina COVID” em conjunto com demonstração gráfica e recrutamento de dados através da utilização do API Google Maps.

A implementação dos Algoritmos Prim & Kruskal com animações, sem extra condições ou requerimentos.

E a implementação do Teorema Caixeiro Viajante utilizando a metodologias de “Vizinho Mais Próximo” e “Poupança de Clarke e Wright”

# Ferramentas Utilizadas

As ferramentas utilizadas para concretização dos objetivos Anaconda Python DIstribution, Spyder IDE e o search engine Google. Adquiridos do seus respetivos websites: [Anaconda | The World’s Most Popular Data Science Platform](https://www.anaconda.com/) e [Home — Spyder IDE (spyder-ide.org)](https://www.spyder-ide.org/)

# Objetivo 1- Implementação do Carteiro Chinês

## História

Criado em 1962 por matemático chinês Meigu Gyan num estudo no tópico de grafos eulerianos. Denominado também de Problema de Inspeção de Rotas, o Carteiro Chinês supõe que, dado uma cidade com várias ruas de tamanhos diferentes e um posto de correios, é possível encontrar um caminho que possa percorrer por todas as ruas e voltar ao posto de correios com a menor possível distância percorrida. [[1]](#_Webgrafia)

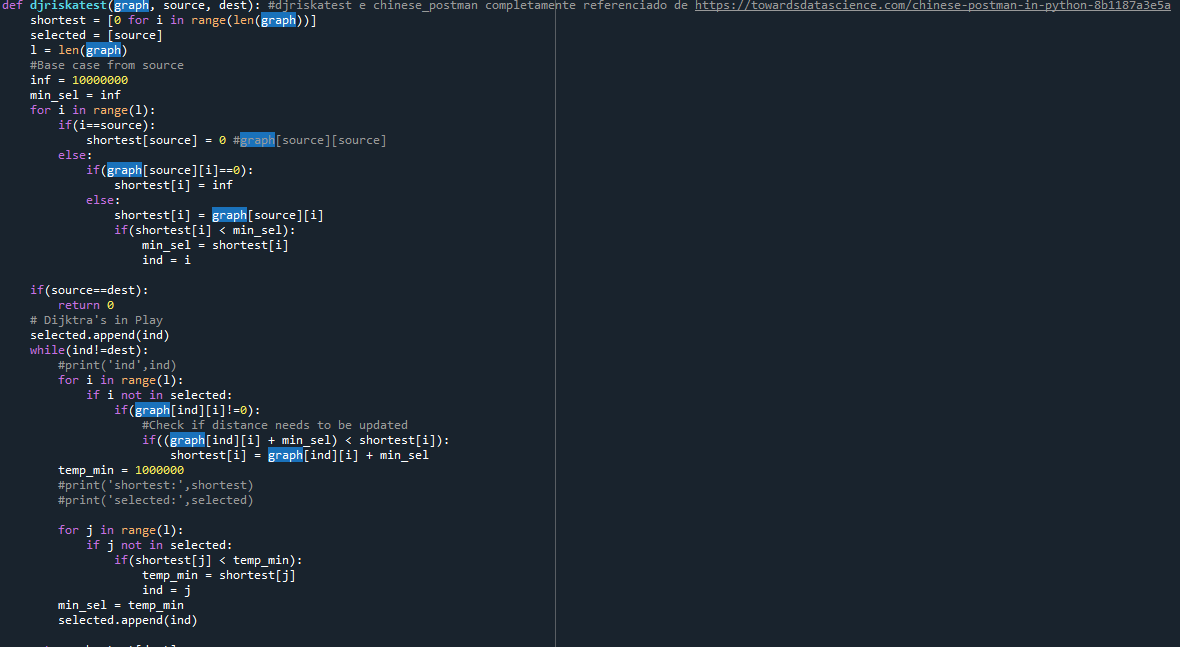
## Propósito

O propósito do Carteiro Chinês reside na descoberta de um percurso de maior poupança que percorre através o todo do grafo e volta ao vértice inicial. Exemplos da utilização do Carteiro Chinês é evidente no planeamento de manutenção de estradas.

## Problemas

Considerando o enunciado dado, foi-se encontrado múltiplos problemas para o desenvolvimento. Primeiramente, a instalação dos pacotes mpl\_toolkit.basemap após uma semana de tentativas concorrentes e obsessivas, que acabou por deteriorar a implementação de todos os objetivos. Segundo, a utilização do API do Google Maps, com pouca experiência e tempo, esta parte do objetivo não foi realizada.

## Aplicação

 Para a resolução parcial do Carteiro Chinês, com assistência de estudo e pesquisa, foi-se determinado que o algoritmo Dijsktra é o mais adequado para obter a distância mínima necessitada para percorrer a totalidade do Médio Tejo.

Algoritmo Dijsktra.

# Objetivo 2- Prim e Kruskal

## História

Algoritmo Prim for desenvolvido por Vojtěch Jarník em 1930, matemático da República Checa, e redescoberto por Robert C. Prim e Edsger W. Dijkstra em 1957(ou 1959). Nota-se que existe pouca quantidade de informação sobre a história atrás do desaparecimento e redescoberta do algoritmo em questão[[2]](#_Webgrafia). Em contraste, Algoritmo Kruskal foi desenvolvido por Joseph Kruskal em 1956 e possui um ponto de origem, este sendo a sua primeira demonstração no jornal *Proceedings of the American Mathematical Society* entre as páginas 48 a 50. [[3]](#_Webgrafia)

## Propósito

Ambos os algoritmos têm como propósito encontrar a árvore de mínimo custo, travessando pelas arestas e escolhendo conexões de leve peso. Mas onde este dois algoritmos se diferenciam é na inicialização dos seus procedimentos, Prim começando no primeiro vértice listado enquanto Kruskal começa no vértice que tem acesso às arestas de menor peso.

## Problemas

Na implementação do Prim e Kruskal sobre um grafo sem direções foi-se exibo mínimo desafio, no entretanto devido às ramificações de terríveis decisões feitas pelo elaborador deste trabalho, não se foi concretizado a implementação da animação da formulação das árvores.

## Aplicação

A picture containing text, screenshot, font

Description automatically generatedA picture containing text, screenshot, font

Description automatically generated Devido a absência de um enunciado, não se evidencia dificuldade, resultando numa simples implementação dos Algotimos Pirm e Kruskal. Com ajuda exterior, foi se elaborado os códigos quês segue, notando que se foi criado uma classe para a melhor execução do Kruskal.

Algoritmo Prim

Algoritmo Kruskal

# Objetivo 3- Caixeiro Viajante(VMP e Clarke&Wright)

## História

Com origens pouco claras, a mais comum sendo a sua origem manual, o problema do Caixeiro Viajante foi matematicamente formulado pelo matemático William R. Hamilton e Thomas Kirkman. Originalmente criado como um puzzle baseado em encontrar ciclos Hamilton, ao longo dos anos, o problema foi estudado com múltiplos métodos descobertos que apresentam respostas ao problema.[4]

## Propósito

O propósito do Problema do Caixeiro Viajante é a procura de um caminho de menor custo e menor tamanho, utilizando diferentes metodologia como as exigidas pela Professora. Com a metodologia do VMP, o Caixeiro começa num vértice randomizado e visita a cidade mais próxima, e a metodologia de Clarke&Wright utilizado para minimizar os custos.

## Problemas

Enquanto não se evidenciou grandes dificuldades na construção do VMP, a construção do Clarke&Wright verificou-se mais difícil para construir, mas utilizando o VMP como base, verificou-se um aliviamento de stresse.

## Aplicações

## 

A screen shot of a computer program

Description automatically generated with low confidenceA screen shot of a computer program

Description automatically generated with medium confidence Devido a uma absência, não houve complicações ou condições extrâneas que teriam de ser consideradas para a implementação

Vizinho Mais Próximo. Anteriormente abreviado como VMP

Clarke and Wright.

# Webgrafia

[[1]](#_História) <https://bcc.ime.usp.br/tccs/2020/gafeol/chinese-postman/tex/main.pdf>

[https://www.projectrhea.org/rhea/index.php/Walther\_MA279\_Spring2016\_topic1#:~:text=Real%2Dworld%20applications,-An%20early%20example&text=Conventional%20applications%20of%20the%20Chinese,used%20in%20network%20algorithm%20checking](https://www.projectrhea.org/rhea/index.php/Walther_MA279_Spring2016_topic1%23:~:text=Real%2Dworld%20applications,-An%20early%20example&text=Conventional%20applications%20of%20the%20Chinese,used%20in%20network%20algorithm%20checking)

[[2]](#_História_1) <https://en.wikipedia.org/wiki/Prim%27s_algorithm#:~:text=The%20algorithm%20was%20developed%20in,Dijkstra%20in%201959.>

[[2]](#_História_1) <https://en.wikipedia.org/wiki/Kruskal%27s_algorithm>

<https://www.ams.org/journals/proc/1956-007-01/S0002-9939-1956-0078686-7/>

[[4]](#_História_2)<https://en.wikipedia.org/wiki/Travelling_salesman_problem>

## Webgrafia dentro do código

<https://blog.devgenius.io/traveling-salesman-problem-nearest-neighbor-algorithm-solution-e78399d0ab0c>

<https://towardsdatascience.com/chinese-postman-in-python-8b1187a3e5a>

<https://www2.isye.gatech.edu/~mgoetsch/cali/VEHICLE/TSP/TSP007__.HTM>

[https://www.programiz.com/dsa/kruskal-algorithm](https://www.programiz.com/dsa/kruskal-algorithm%20%20%20%20)