Algoritmia e Estrutura de Dados

Travel Salesman Problem

Discente(s):
Pedro Roldan
Leandro Moreira

Docente:
Doutor Faroq AlTam

28 de Junho de 2017

Conteúdo

1 Introdução

O travel salesman problem (TSP) é um problema bastante comum largamente encontrado em diversas aplicações tais como: empresas de transporte (e.g. UPS), escalas de tripulação de companhias aéreas, etc.
Em principio, um vendedor necessita de efetuar uma viagem por diversas cidades, onde inicia a viagem numa determinada cidade (Casa), visita todas as cidades para vender os seus produtos, e retorna a casa.

1.1 Requisitos Minimos

O TSP pode ser representado por uma lista de nós, sendo o objetivo descobrir uma serie de caminhos (Edges) entre cada um dos nós.

Sendo que:

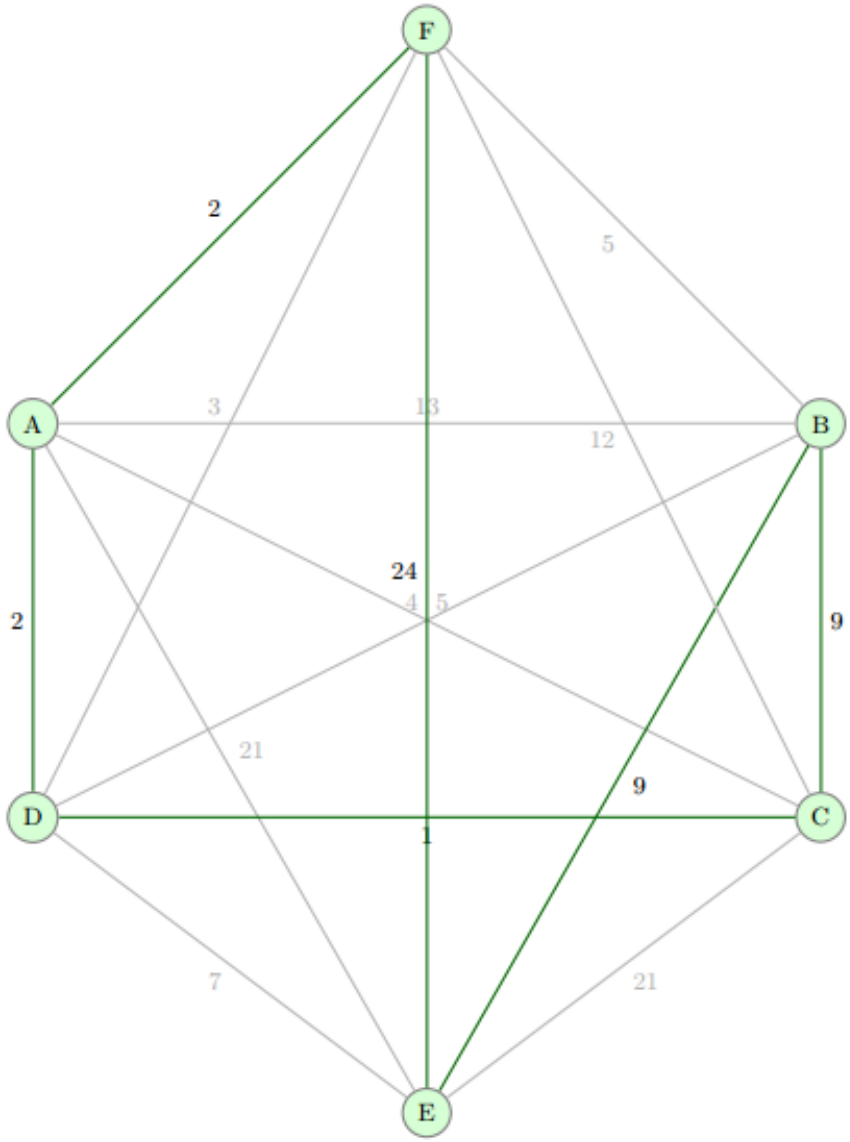
- Cada nó (Cidade) pode ser visitado apenas 1 vez.
- Os caminhos formam uma Tour.
- O custo da Tour deve ser o mínimo possível.

A tour TSP é um gráfico direcionado, onde cada nó representa uma cidade, e cada edge representa um caminho entre 2 cidades.
Cada edge têm um peso, que é no seu caso mais simples a distancia euclidiana entre os seus nós.
Este peso pode ser composto por diversos fatores, no entanto neste projeto apenas se considera a distancia entre nós.

2 2-Opt

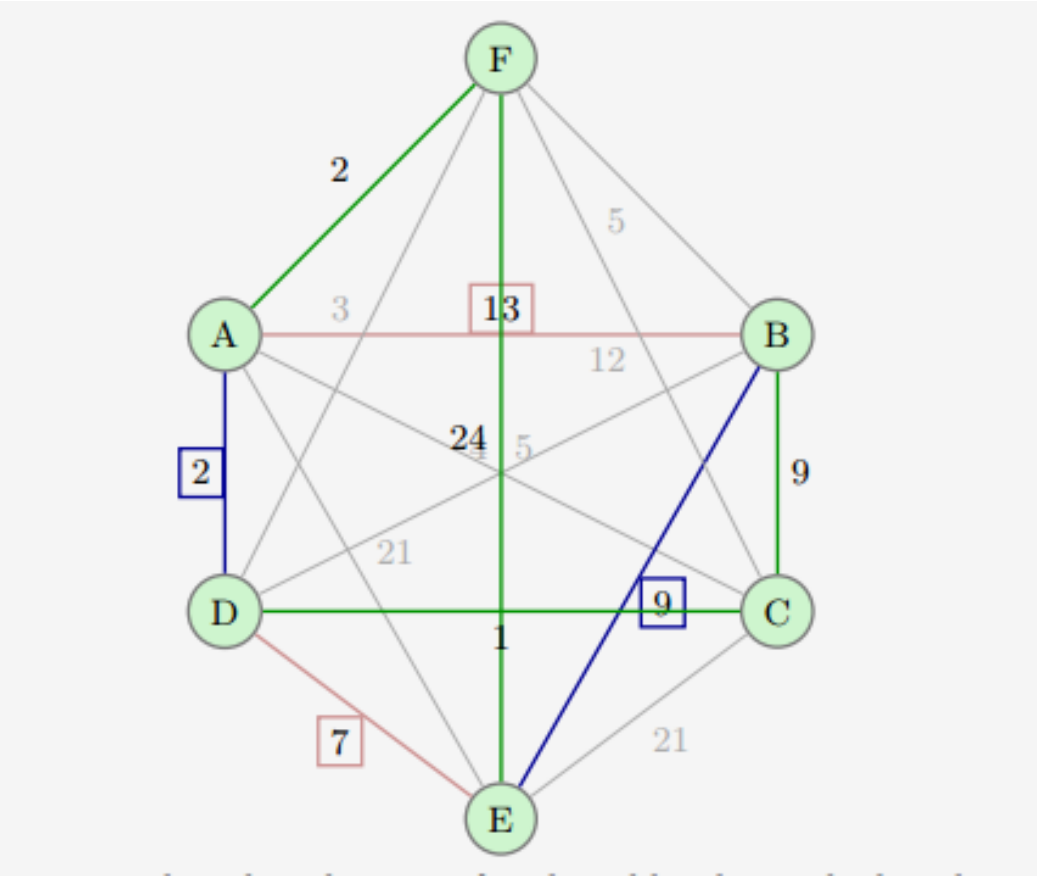
O algoritmo 2-Opt foi proposto por Croes em 1958, embora o movimento básico tenha sido sugerido por Flood em 1956.
O algoritmo 2-Opt basicamente remove 2 nós (Edges) da tour, e conecta os dois caminhos criados. Isto é normalmente referido como um movimento 2-Opt.
Quando falamos da complexidade deste algoritmo K-Opt, não podemos omitir que um movimento pode levar até $\mathcal{O}(n)$ a ser efetuado. A implementação simples do 2-Opt corre em $\mathcal{O}(n^2)$, isto envolve selecionar um edge (c1,c2), procurar outro edge (c3,c4), completar o movimento apenas se a dist(c1,c2) + dist(c3,c4) > dist(c2,c3) + dist(c1,c4.)

2.1 Técnica 2-Opt



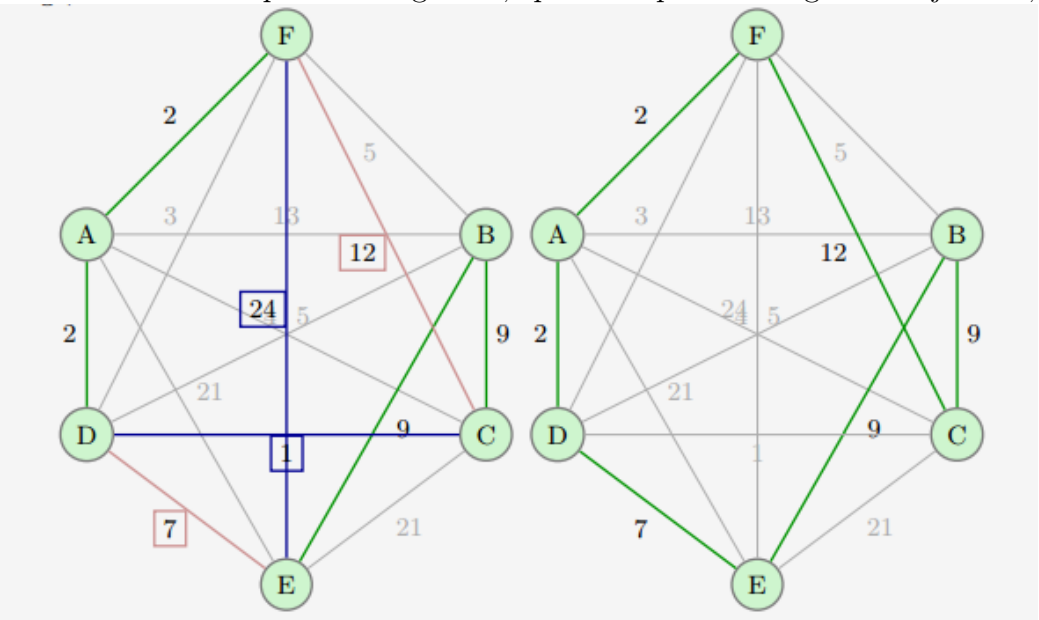
Aqui vamos utilizar uma lista de 6 nós, em que a distancia total da tour é de 47.

Em primeiro lugar selecionamos um edge, neste caso **AD**, vamos de seguida selecionar outro edge que não seja adjacente ao egde em uso,**AD**.
Neste exemplo apenas dispomos de um edge possível **BE**.
Podemos assim trocar **AD** e **BE** por **AB** e **BE**.



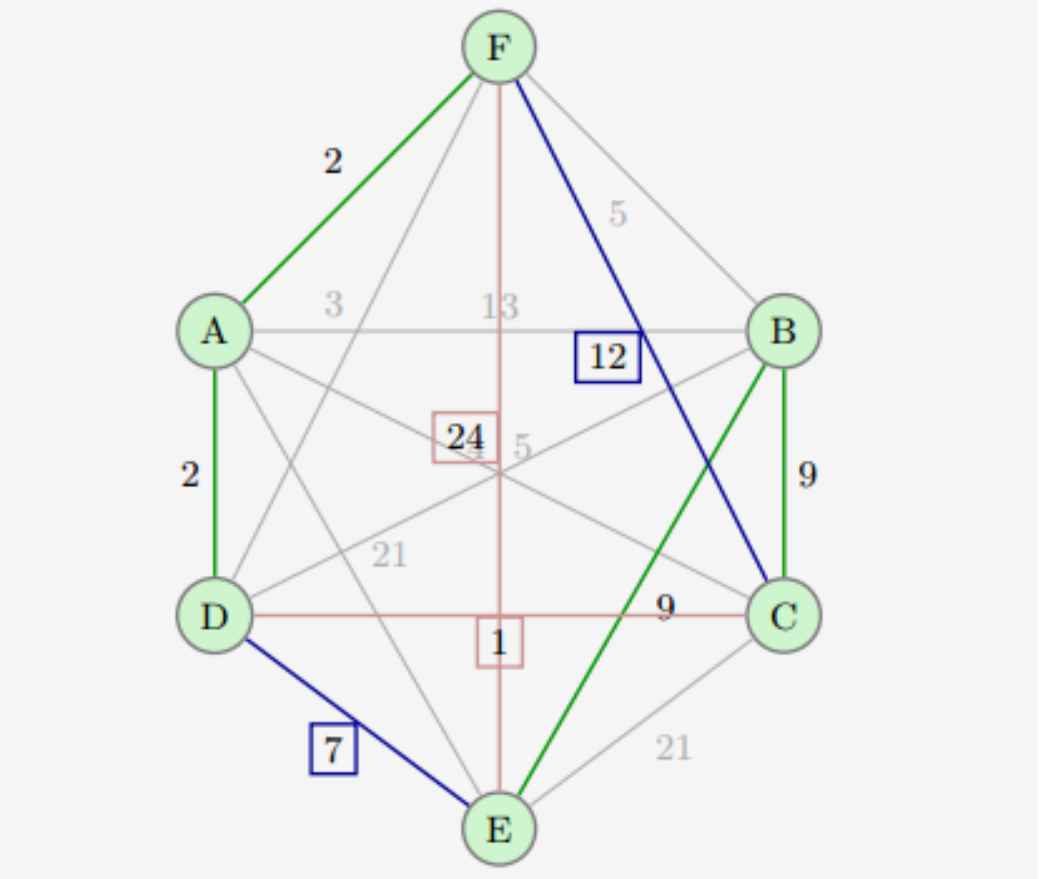
No entanto, a soma das distancias dos edges possíveis de trocar é **maior** que a soma das distancias dos edges originais. Assim sendo não trocamos os edges.

Considerando o próximo edge **DC**, apenas dispõe de 1 edge não-adjacente,



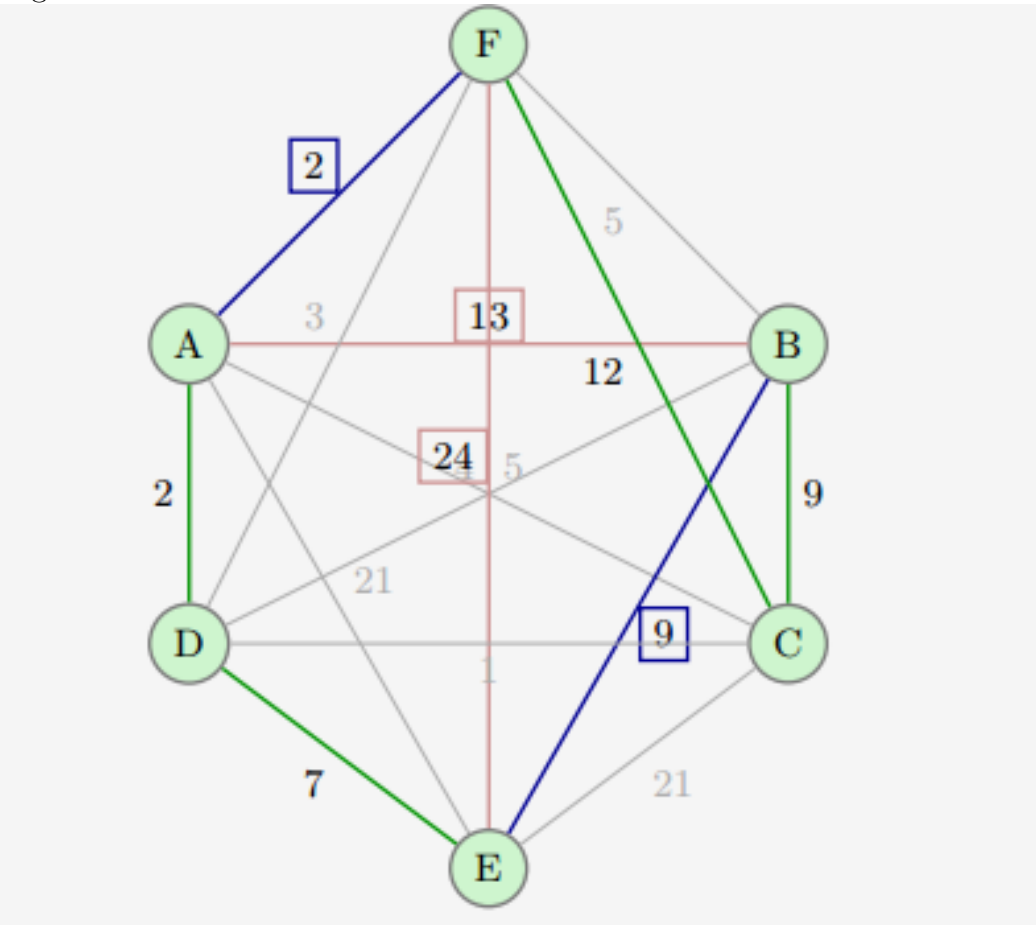
Aqui, como a soma das distancias dos edges originais é **maior** que a soma das distancias dos edges possíveis de trocar, efetuamos a troca de edges.

Considerando o edge **DE**, este edge dispõe do edge não-adjacente **CF**.



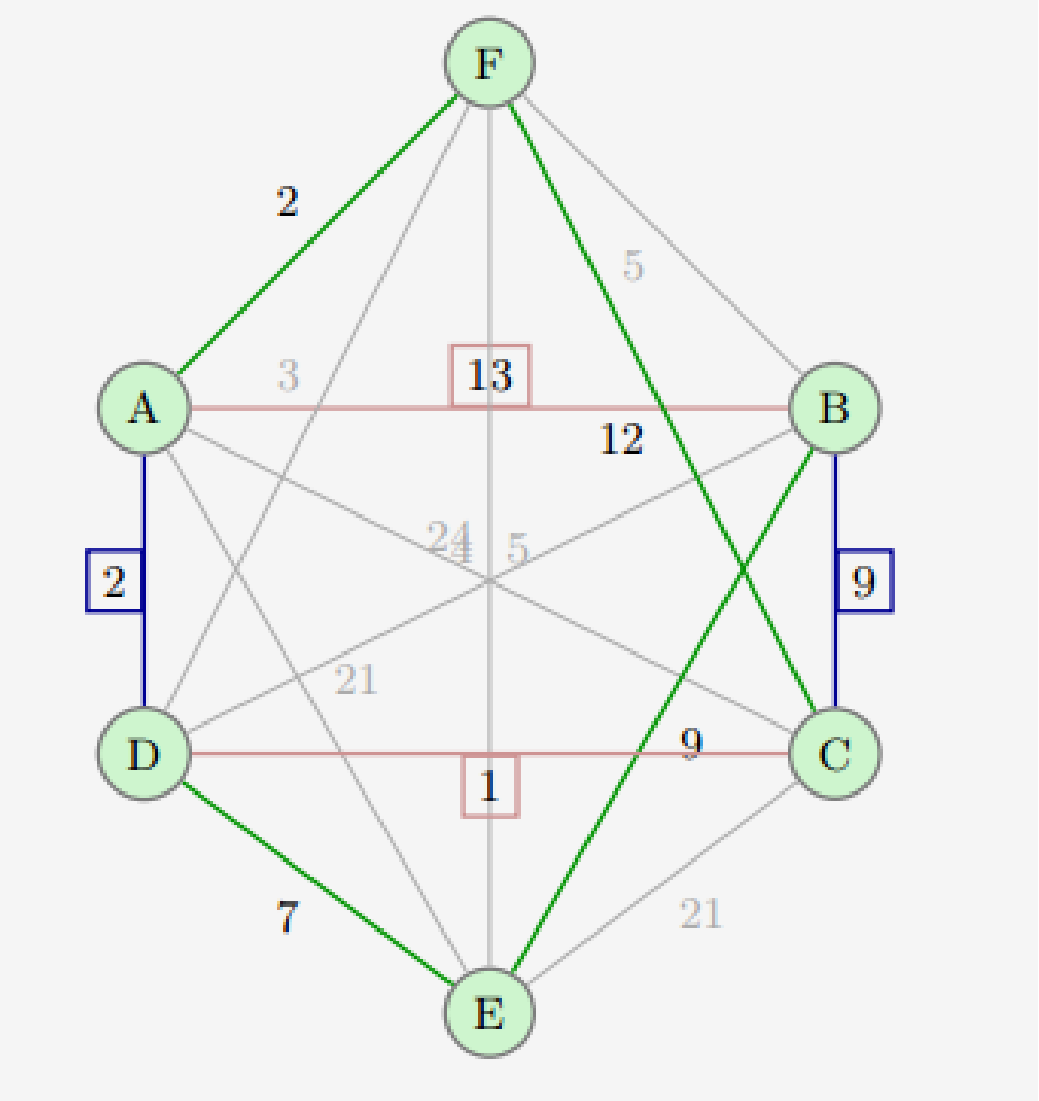
Como a soma das distancias dos edges possíveis de trocar são maiores que a soma das distancias dos edges originais, não se efetua trocas.

No proximo passo o edge selecionado é **EB**.O edge não-adjacente a este edge é **FA**.



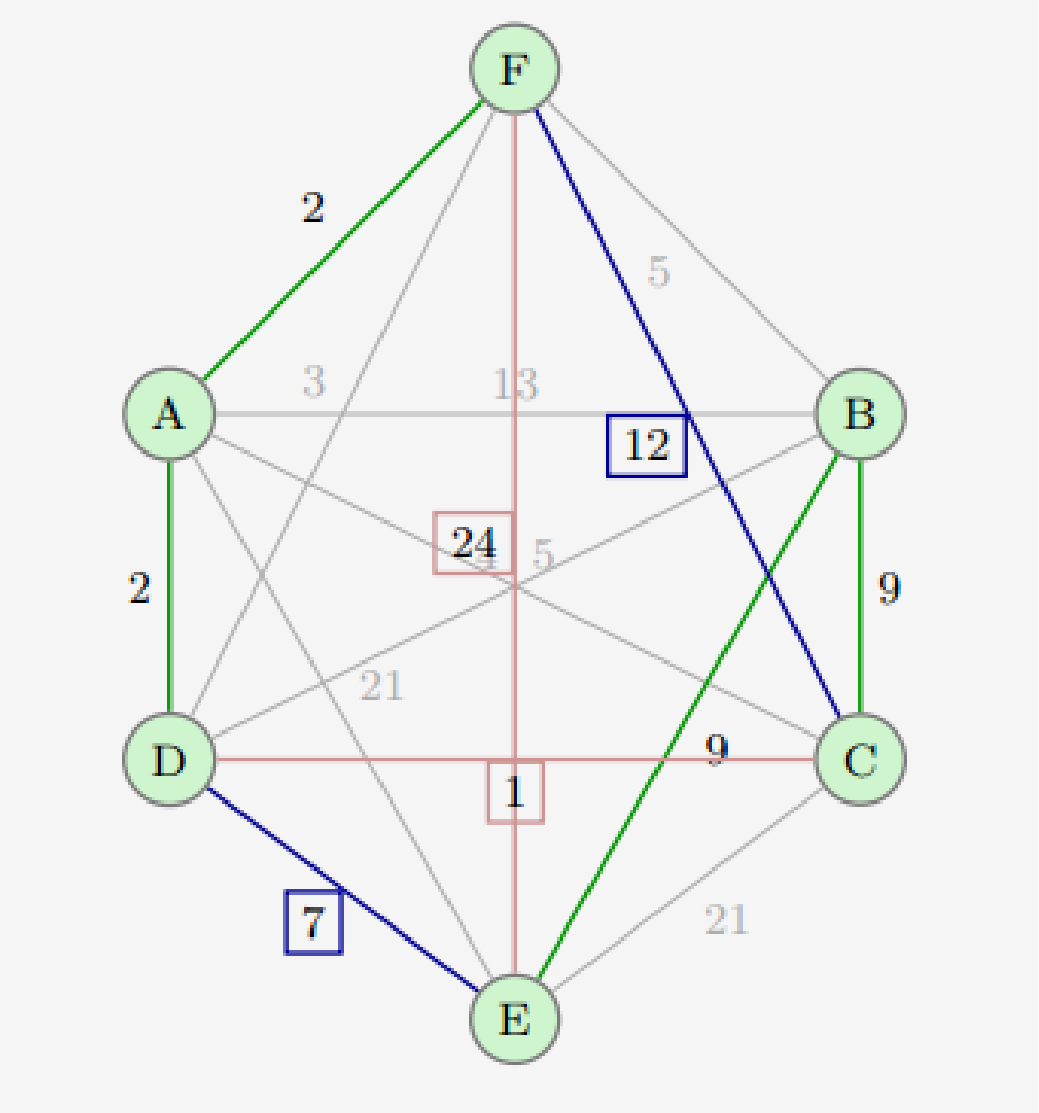
Não é necessário efetuar trocas de edges.

Vamos selecionar o próximo edge, **BC**.O edge não-adjacente a este é **AD**.



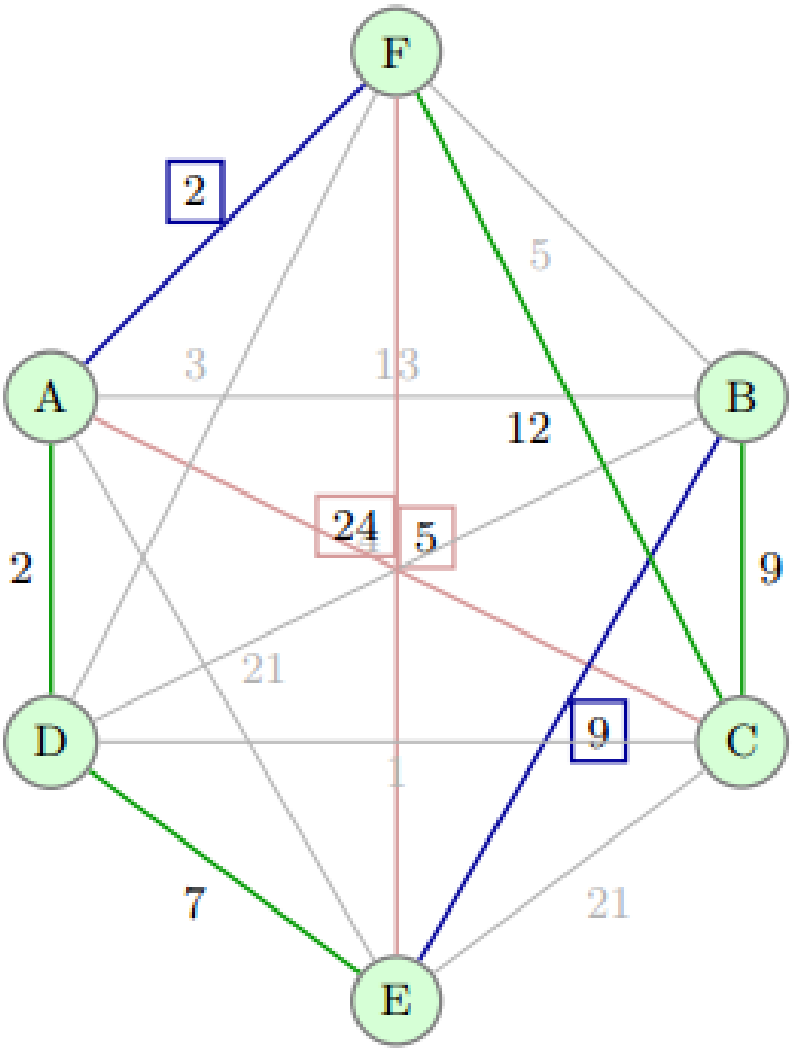
Este cenário não garante alterações ao caminho.

Agora vamos considerar o edge **CF**.O edge não-adjacente a este é **DE**

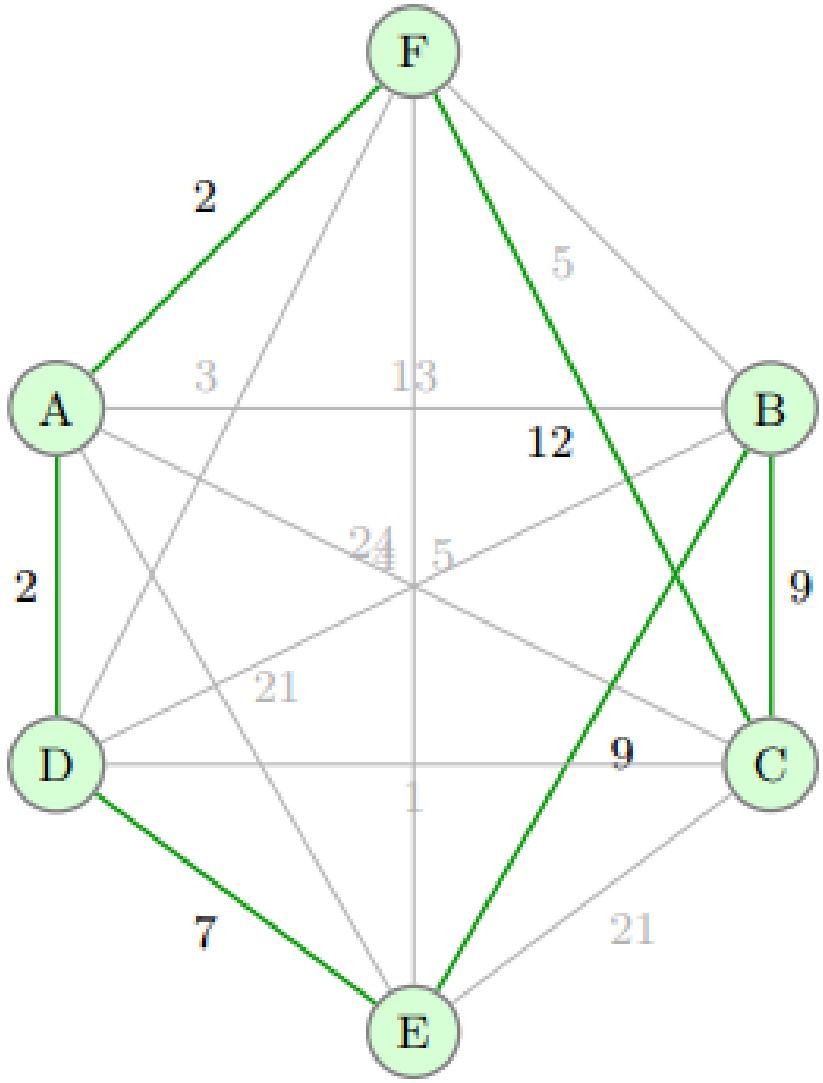


Aqui, não é necessário efetuar alterações ao caminho visto que a soma da distancia dos edges possíveis de trocar é inferior quando comparado com os edges existentes.

O proximo edge a ser considerado é **FA**. O edge não adjacente é **EB**.



Este também é um caso em que não existe necessidade de se efetuar trocas de edges.



Finalmente chegamos a uma solução optimizada. A solução apresenta um custo de 41, efetivamente inferior ao custo da solução inicial.

2.2 Intersecções

As intersecções estão a ser detectadas com a definição de intersecção de dois segmentos de recta, que nos diz que tendo dois segmentos de recta, **AB** e **CD**, existe intersecção se **A** estiver de um lado de **CD** e **B** do outro, e **C** estiver de um lado de **AB** e **D** do outro.

3 Enquadramento

O trabalho descrito neste relatório foi realizado recorrendo à linguagem de programação ANSI C, assim como os recursos disponibilizados na unidade curricular.

3.1 Motivação

A principal motivação para a realização deste trabalho, resulta da importância em implementar o algoritmo 2-Opt ao problema Travel Salesman, assim como demonstrar os conhecimentos alcançados na disciplina de Algoritmia e Estrutura de Dados.

3.2 Objectivos

Pretende-se através deste trabalho, atingir uma solução válida para o Travel Salesman Problem, implementando o algoritmo 2-Opt.

4 Conclusões

A implementação do algoritmo 2-Opt ao TSP desenvolvida, para além de permitir os requisitos pedidos no enunciado do trabalho prático, permite também a possível implementação de outros algoritmos,pois sendo modular torna-se mais escalável, entre outras funcionalidades.
De frisar que devido à liberdade proporcionada, quer na sua forma de desenvolvimento quer na implementação permitiu desta forma aguçar a curiosidade para o uso de diversos algoritmos para a solução do Travel Salesman Problem.
Foi sem duvida um desafio interessante, mas que por limitação de tempo,

deixa ainda uma larga margem para melhoramentos.

5 Bibliografia

Referências

[1] https://en.wikipedia.org/wiki/2-opt.

[2] https://en.wikipedia.org/wiki/travelling salesman problem.

[3] Doutor Faroq AlTam. *Aulas Teórico-Práticas Algoritmia e Estruturas de Dados 2º ano, 2º semestre da Licenciatura em Engenharia Informática do Instituto Superior Manuel Teixeira Gomes.* ISMAT, 2016-2017.

6 Anexos

Ficheiro "relatorio.pdf"e "eps.c, eps.h, tsp.c, tsp.h, file.c, file.h, main.c", assim como as pastas "tspdata, results", compactado num ficheiro "trabalho.zip".

Não existem quaisquer códigos ou listagens adicionais.