OCEAN SHIPPING AND URBAN DELIVERIES MANAGEMENT PROGRAM

DESENHO DE ALGORITMOS – 2º SEMESTRE 2022/2023

G14_3

PEDRO PAIXÃO – 202008467

NUNO PENAFORT- 202008405

ANTÓNIO CORREIRA – 201804832

PROBLEMA

Pretende-se implementar um sistema de gestão de informação referente a entregas.

Este sistema é capaz de gerir toda essa informação .

LEITURA DOS FICHEIROS

Para a leitura dos ficheiros, tivemos de usar duas formas, pois os “Real-World-Graphs” tinham um ficheiro para os nós e outro para as arestas, enquanto que os restantes só tinha um ficheiro. Por isso, quando se lemos os “Real-World-Graphs” usamos o readNodes e depois o readEdges2, para os outros, usamos o readEdges1, que dentro do método adicionamos os nós e as edges respetivas. Estas funções encontram-se no ficheiro FileReader.cpp.

MENU

Quando inicia-se o programa, damos ao utilizador a escolha que tipo de grafo quer analisar:

```
        Ocean Shipping and Urban Deliveries ManagementProgram

1)"Extra_Fully_Connected_Graphs"
2)"Real-world Graphs"
3)"Toy-Graphs"

0) Exit
```

Depois de escolhido, o utilizador pode escolher que grafo quer analisar:

```
        Ocean Shipping and Urban Deliveries ManagementProgram

1)"graph1"
2)"graph2"
3)"graph3"
```

Por fim, pode escolher como pretende analisar o grafo:

```
        Ocean Shipping and Urban Deliveries ManagementProgram

1) Backtracking Algorithm
2) Triangular Approximation Heuristic
3) Genetic Algorithm

0) Exit
```

T4.1 – BACKTRACKING ALGORITHM

DESCRIÇÃO:

- O Backtracking algorithm é uma abordagem que consiste em começar num vértice inicial e explorar todos os caminhos possíveis selecionando recursivamente o próximo vértice que ainda não foi visitado ao caminho atual.
- Sempre que escolhe um caminho que não tem “saída” ele retrocede, quando isto acontece, significa que não há vértices não visitados restantes a serem adicionados ao caminho atual.
- O algoritmo continua esse processo até que todos os vértices sejam visitados e um ciclo completo seja formado.

Nos “toy graphs” shipping e tourism o algoritmo mostra a distância mínima, o caminho e o tempo que demora (que é curto pois estes grafos são pequenos)

```
Minimum distance : 86.7
Optimal path : [0 -> 1 -> 11 -> 10 -> 12 -> 13 -> 3 -> 2 -> 4 -> 6 -> 9 -> 7 -> 8 -> 5 -> 0]

    Execution time: 0.579186 seconds
```

```
==================

Minimum distance : 2600
Optimal path : [0 -> 3 -> 2 -> 1 -> 4 -> 0]

    Execution time: 0.00019 seconds

==================
```

T4.1 – BACKTRACKING ALGORITHM

VANTAGENS E DESVANTAGENS:

- Devido ao algoritmo testar todas as possibilidades, a solução é sempre a melhor possível. No entanto, devido a isso, a complexidade temporal é O(2^n), o que faz com que não seje útil para os grafos grandes já que demoraria imenso tempo.
- Por causa disso, pusemos uma restrição de tempo em que se passar de 30 segundos o programa para de imediato a execução, tornando-se assim relativamente simples de perceber para que grafos é viável utilizar Backtracking

Já para o grafo com 25 arestas demora mais de 30 segundos

```
3                    1
"edges_25.csv"   "edges_100.csv"

file accepted

This service is no feasible for the given data

Process finished with exit code 0
```

T4.2 – TRIANGULAR APROXIMATION HEURISTIC

DESCRIÇÃO:

- Este algoritmo é o contrário do outro, o tempo de execução é rápido [O(|V|^2)], no entanto não fornece a solução ótima, mas sim uma solução próxima da ótima.
- Começa por selecionar um nó arbitrário como ponto de partida(por causa disso decidimos escolher o primeiro). Em seguida escolhe os dois nós mais próximos que ainda não foram visitados , formando um triângulo. Este algoritmo vai estendendo iterativamente o triângulo, selecionando o nó mais próximo não visitado em relação a um dos 3 vértices do triângulo.
- Este processo continua até todos os nós estarem visitados resultando num caminho que se aproxima da solução ótima.

```
"edges_700.csv"
==================

Optimal path (MST in preorder) : [0 -> 102 -> 505 -> 223 -> 640 -> 667 -> 166 -> 188 -> 327 -> 549 -> 53 -> 364
Minimum distance (at most twice the cost of the best solution) : 2.15383e+06

Execution time: 0.177671 seconds

==================
```

```
"edges_400.csv"
==================

Optimal path (MST in preorder) : [0 -> 66 -> 65 -> 139 -> 1 -> 210 -> 351 -> 19
Minimum distance (at most twice the cost of the best solution) : 1.66824e+06

Execution time: 0.0624882 seconds

==================
```

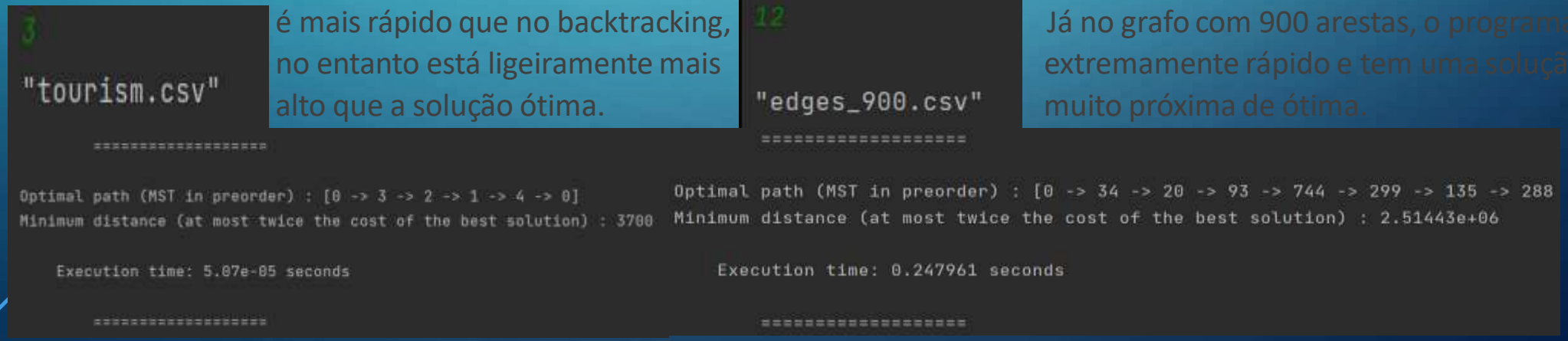
T4.2 – TRIANGULAR APROXIMATION HEURISTIC

VANTAGENS E DESVANTAGENS:

- Este algoritmo permite encontrar soluções com maior eficiência quando comparado com o backtracking.
- O algoritmo consegue descobrir uma solução próxima de ótima mesmo para grafos extremamente grandes, o que é bom pois descobrir a solução ótima não seria possível de qualquer forma para dados complexos com muitas possibilidades.
- No entanto convém recordar que como não obtém a solução ótima, significa que para grafos curtos poderá compensar usar o backtracking.

No tourism que é um grafo pequeno, o tempo ainda é mais rápido que no backtracking, no entanto está ligeiramente mais alto que a solução ótima.

Já no grafo com 900 arestas, o programa foi extremamente rápido e tem uma solução muito próxima de ótima.



T4.3- GENETIC ALGORITHM

DESCRIÇÃO:

- Este algoritmo consiste em simular o processo de evolução natural para encontrar uma solução ótima.
- Uma população de caminhos potenciais são gerados aleatoriamente (chamados cromossomos). Cada cromossomo representa uma possível solução para o TSP.
- Cada cromossoma é avaliado pela qualidade do caminho (ou seja pela distância total).
- Os caminhos com melhores distâncias são os que têm maior chance de serem escolhidos para a fase seguinte – o cruzamento.
- Nessa fase combina-se a informação de dois caminhos para criar um novo caminho

(de maneira a simular ao cruzamento de cromossomas pai para criar um cromossoma filho).

- De seguida introduz-se pequenas alterações aos caminhos novos de maneira a evitar que as soluções fiquem abaixo do ideal. A seguir os novos cromossomas substituem alguns dos caminhos pais de modo a ter um número de soluções constante.
- - O algoritmo itera por estas etapas até que o número de soluções seje igual ao número de nós do grafo. Retorna o caminho que tem a melhor solução.

T4.3- GENETIC ALGORITHM

VANTAGENS E DESVANTAGENS:

- Tal como qualquer outro algoritmo não garante uma solução ótima para o TSP, apenas uma solução próxima. Complexidade temporal é O(|V|^2).
- O desempenho e a solução do algoritmo são um pouco aleatórias em algumas instâncias, no entanto nos piores casos apresentava soluções parecidas ao Triangular Aproximation. Em média parecia ter tempos e soluções melhores quando comparando com o outro algoritmo.

Genetic Algorithm

```
12
"edges_900.csv"
```

```
====================

Path : [0 -> 352 -> 7 -> 867 -> 707 -> 887 -> 75 -> 424 -> 774 ->
Minimum distance : 1.88867e+06

    Execution time: 0.264066 seconds

====================
```

Triangular Algorithm

```
====================

Optimal path (MST in preorder) : [0 -> 34 -> 20 -> 93 -> 744 -> 299 -> 135 ->
Minimum distance (at most twice the cost of the best solution) : 2.51443e+06

    Execution time: 0.334963 seconds

====================
```

DIFICULDADES, ESFORÇO E AUTOAVALIAÇÃO

As principais dificuldades encontradas durante a elaboração do projeto foi principalmente relacionadas com *dataset*, tendo em conta que havia grafos com erros numa fase inicial e outros que não funcionariam para certos algoritmos devido a serem defeituosos os dados fornecidos.

Autoavaliação:
Pedro Paixão – 47.5%
Nuno Penafort – 47.5%
António Correia - 5%