Relatório

Projeto Prático nº2

Algoritmos e Estruturas de Dados

Licenciatura em Engenharia Informática

2016/2017

João Moreira - 2015230374

Ricardo Tavares - 2014230130

Introdução

Neste segundo projeto prático procuramos idealizar e construir um gerador de cidades e uma estrutura de dados baseada em grafos que permita ler e manipular o mapa gerado.

Nas páginas que se seguem encontra-se a explicação de cada uma das estruturas e uma descrição geral do trabalho realizado, bem como os resultados obtidos e respetiva análise.

Interface:

A interface é um simples menu que nos permite, através da introdução de um numerário obter as opções, de leitura do mapa, cálculo da distância mais curta, impressão para o ecrã do grafo e sair do programa.

Opção 1 – Para gerar o mapa, o programa pede o numero da tarefa e o numero de cidades (a não ser que a tarefa escolhida seja a 3, explicaremos melhor na página 3), sendo que envia estes dados para o Gerador de Cidades implementado no ficheiro Gerador.py, sendo que após estas serem geradas, são carregas para o nosso grafo, estando assim prontas para manipulação.

Opção 2 – Após termos o mapa lido calculamos a menor distância possível a percorrer, sendo que isto é feito através da função bfs_paths(), que leva o grafo em utilização, o inicio e a ultima cidade como variáveis continuas.

Após isso imprime-se o caminho ideal e o tempo que o programa demorou a calcular.

Opção 3 – Aqui apenas fazemos a impressão dos vértices, sendo que aparece a Cidade X e as cidades às quais ela está conectada bem como a sua distância a todas elas.

Opção 0 – A opção de saída é apenas um "break" do ciclo "while true".

Gerador.py

O gerador funciona com base em 3 parâmetros: o nome do ficheiro, o numero de cidades e a tarefa em questão.

O nome do ficheiro é necessário para que o leitor e o gerador estejam sincronizados e para que não existam erros em termos de leitura.

O numero de cidades é necessário para que se criem apenas o numero de cidades necessárias.

A tarefa implica uma escolha, no gerador, sendo que ele irá optar por criar as cidades com a mesma distância entre semelhantes, como por exemplo, de A até B e B até A ser a distância igual, ou isto não se aplicar. Para alternar entre estas opções tem de se escolher tarefa 1 ou 3, ou 2 no ficheiro Run.py quando se está a correr o programa.

Vertex.py

Neste ficheiro encontramos as ligações dos vértices do grafo, sendo que aqui estão as funções chamadas para adicionar vizinhos, imprimir para o ecrã onde está conectado cada vértice, encontrar as conexões, o Id e o peso de cada um dos vértices.

Graph.py

Este ficheiro é a base do programa. Aqui estão as funções que nos permitem manipular e criar o grafo que irá ser utilizado. O grafo acede aos vértices constantemente daí termos funções como: getVertex(), calcDist(),…

Algumas das funções que se encontram aqui foram obtidas de slides das aulas teóricas das aulas de AED.

As funções mais importantes neste ficheiro são a de adicao de vértices, o cálculo da distância entre vértices, o adicionar de Edge's, o encontrar o numero de vértices e por fim o Breadth-First Search, sendo este um algoritmo que adaptamos de modo a encontrarmos o caminho mais rápido entre cidades. Este algoritmo tem por base, manter um queue de caminhos até chegar ao final.

Run.py

Aqui se encontra a interface do programe e de maneira geral o acesso aos comandos implementados.

Primeiramente, este ficheiro tem um ciclo while infinito até que se envie no terminal a opção 0.

Para além da opção de saída, temos três outras opções que nos permitem assim correr o programa.

A primeira tem haver com o gerar e carregar do mapa pretendido. O utilizador pode escolher qual a tarefa em questão das 3.

- Na primeira o programa pede o número de cidades que o utilizador pretende gerar, sendo que estas terão distâncias todas randomizadas umas das outras, menos quando se trata de distâncias entre cidades iguais, por exemplo: uma cidade A que tenha distância 10 até à cidade B, terá igual distância da cidade B até A.

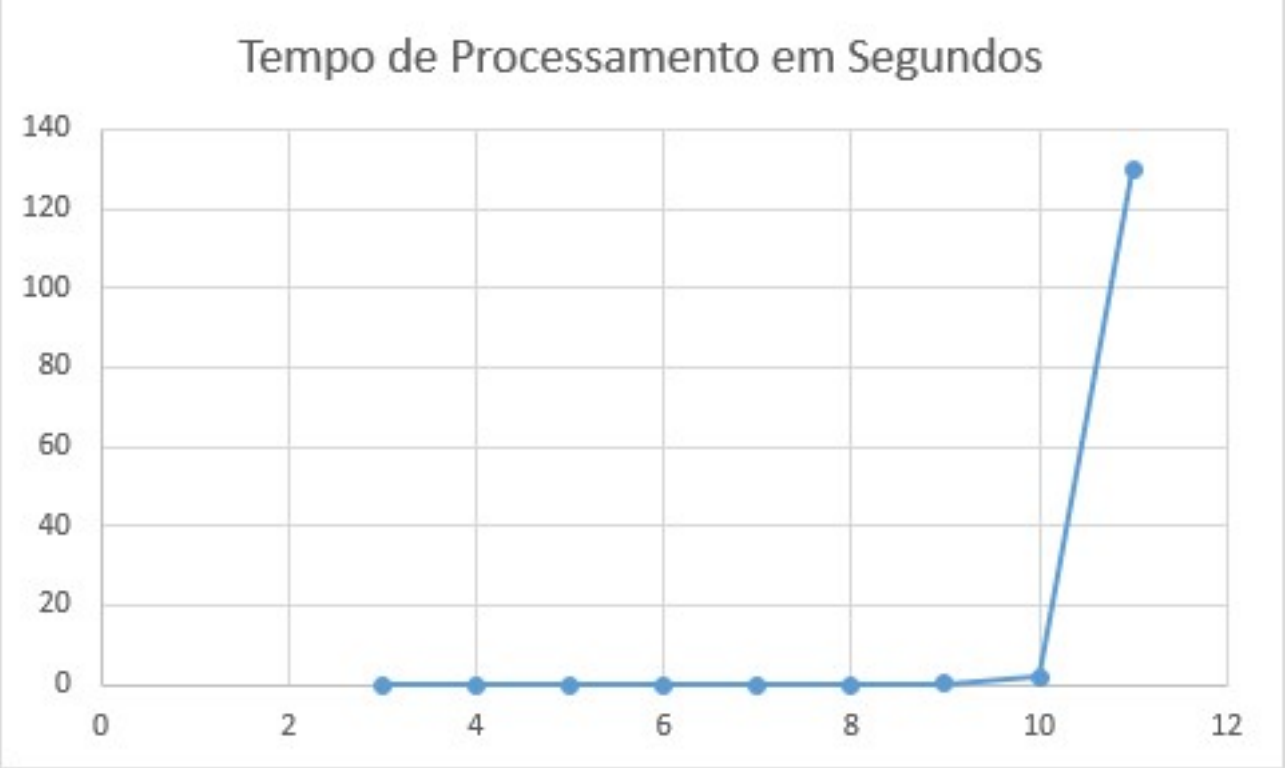
O programa obtém os dados que necessita para a função gerador e gera assim as cidades. Apos isso, começa um ciclo de leitura em que lê e manda para o grafo as informações das cidades.

- A segunda opção é em tudo igual à primeira tirando que as distâncias entre cidades podem ser diferentes, por exemplo, se da cidade A até à cidade B temos 10 de distância, da cidade B a A podemos ter 12 (ou outro valor qualquer).

- A terceira opção coloca por default o numero de cidades a 11, visto que é o máximo que o programa aguenta em menos de 30min.

A segunda opção do menu inicial é a de calcular o caminho mais curto, sendo que temos também implementado o tempo que demora a correr este processo através da função time().

Um gráfico demonstrativo do tempo que o programa demora a correr o numero de cidades entre 3 e 11 inclusive(no eixo dos yy's encontramos o tempo em segundos, no eixo dos xx's encontramos o numero de cidades):



NumCidades	3	4	5	6	7	8	9	10	11
Tempo em segundos	Aprox. 0	Aprox. 0	0,00049998	0,0005	0,0025	0,14	0,118	2,03	130,14

Como podemos concluir, existe uma subida abruta das 10 para as 11 cidades.

A terceira opção é para mostrar o grafo e a sua organização, mostra-nos as ligações e o peso de cada vértice.

Conclusão

Após interpretação dos tempos e análise dos resultados, concluímos que o algoritmo deveria aguentar mais cidades e ter maior eficiência, mas tendo em conta que não foi muito otimizado, é normal que esteja um pouco longe das espectativas.

Concluímos que o nosso algoritmo não é ótimo para a resolução do problema visto que não aguenta com grande numero de cidades.

Percentagem de trabalho:

João Moreira – 50%

Ricardo Tavares – 50%