UNIVERSIDADE D COIMBRA

Relatório do Problema 3
LPA 2019/20

Finding Connections

Trabalho realizado por:
Maria Paula de Alencar Viegas - 2017125592 - uc2017125592@student.uc.pt

1) **Qual a complexidade temporal da implementação?**

Considerando V o número de vértices e E o número de edges:
O algoritmo dos pontos de articulação tem complexidade O (| V | + | E |).
O algoritmo de Floyd-Warshall tem complexidade O (| V | ^ 3).
O algoritmo de Kruskal tem complexidade O (| E | · log | V |).

2) **Quais algoritmos e estruturas foram implementadas? Qual o propósito delas?**

Foi criada uma classe *Graph* que armazena:

o número total de vértices *V*;

uma matriz de adjacência *cost* para armazenar o custo das ligações entre vértices;

uma matriz de adjacência *min_cost_servers* para armazenar o menor custo das ligações entre os servidores;

um array dinâmico *adj* de listas de vértices adjacentes*;*

um inteiro *n_servers* que armazena o número total de servidores (pontos articulados) no grafo;

um array dinâmico de bools *servers* que indica qual vértice é servidor;

um inteiro *shortest* que armazena a quantidade mínima de cabo necessária para conectar a rede.

um inteiro *tree_cable* que armazena o total de cabo necessário para uma topologia de árvore.

Foi utilizado o algoritmo para calcular os pontos articulados que utiliza árvore DFS baseado no algoritmo nos power points e no algoritmo obtido em https://www.geeksforgeeks.org/articulation-points-or-cut-vertices-in-a-graph/. Com esse algoritmo, obtemos os pontos de articulações, que são os servidores do problema proposto, e podemos contar quantos existem.

Também foi utilizado o algoritmo Floyd Warshall disponibilizado nos power points e em https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/. Assim, obtemos uma matriz com o menor comprimento de cabo necessário para conectar um par de equipamentos. Com essa matriz, podemos calcular a quantidade mínima de cabo para conectar os servidores da rede.

Por fim, foi utilizado o algoritmo de Kruskal disponibilizado nos power points da cadeira e em https://www.geeksforgeeks.org/kruskals-algorithm-simple-implementation-for-adjacency-matrix/. Ele permite achar a árvore mínima de expansão (Minimum Spanning Tree) e assim podemos achar a quantidade necessária de cabo para a árvore do grafo.

3) **A implementação é válida para qualquer input?**

A implementação é válida para qualquer input desde que obedeça o formato estabelecido no enunciado. Foi definida um valor INF para representar não-ligações nas matrizes de adjacência que é desconsiderado nos cálculos dos custos.

4) Quais técnicas de aceleração foram implementadas?

Uma técnica de aceleração implementada foi guardar a matriz *min_cost_servers* com os custos das conexões entre os servidores. Dessa forma, não é preciso percorrer a matriz inteira novamente no algoritmo de Kruskal.

Também pode ser considerado uma técnica de aceleração a verificação da quantidade de servers antes de executar os algoritmos de Kruskal e Floyd Warshall, visto que só devem ser executados quando há mais de um servidor. Sem essa verificação, temos Time Limit Exceeded no Mooshak.