Relatório 2º projeto ASA 2024/2025

Grupo: AL077

Aluno(s): Alexandre Delgado (109441) e Madalena Yang (110206)

Descrição da Solução

A solução proposta consiste em transformar as linhas de metro em vértices de um grafo e as mudanças de linhas em arcos desse grafo.

A partir do input recebido, criamos uma estrutura do tipo *vector<set<int>>* em que cada índice representa uma linha de metro e tem um set de estações pertencentes a essa linha. Depois, construímos o nosso grafo de linhas. Para isso, optámos por ter uma lista de adjacências representada por um vetor de vetores (*vector<vector<int>>*), onde cada posição corresponde a um vértice (linha) e os valores associados são todos os vértices adjacentes a esse vértice (linhas conectadas a essa linha). Por exemplo, (Linha) 1: 2 3
(Linha) 2: 1
(Linha) 3: 1 4 5
(Linha) 4: 3 5
(Linha) 5: 3 4

Para calcular o índice de conectividade, aplicamos o algoritmo BFS a todos os vértices do grafo das linhas, guardando o maior valor de distância mínima encontrado em cada execução.

O algoritmo BFS utilizado foi ligeiramente modificado. Utiliza na mesma uma queue, mas à medida que um vértice não visitado é explorado, guarda-se a distância desse vértice ao vértice source. O algoritmo BFS gera o caminho mais curto entre o vértice source e qualquer outro vértice do grafo. Assim, o último valor adicionado ao vetor das distâncias vai ser a distância entre o vértice source e o vértice mais distante dele através do caminho mais curto.

Existem dois casos em que o output esperado é -1. Um dos casos é, do nº de estações dado, haver alguma estação que não tem nenhuma ligação. Esse caso é detetado quando recebemos o input. Outro caso é ter linhas isoladas, ou seja, partes do grafo serem disconectadas, esse caso é apanhado na execução do algoritmo BFS.

Apresentamos o pseudocódigo da função que construímos o grafo das linhas:

```
buildLinesGraph()
  // metro_lines é o vector<set<int>>
  for i = 1 to l                                   // O(l)
    for j = i + 1 to l                             // O(l)
      for each station ∈ metro_lines[i]            // O(n)
        if (metro_lines[j].find(station) exists)   // O(log n)
            linesGraph[i].push_back(j);            // O(1)
            linesGraph[j].push_back(i);            // O(1)
            break;
```

Relatório 2º projeto ASA 2024/2025

Grupo: AL077

Aluno(s): Alexandre Delgado (109441) e Madalena Yang (110206)

Análise teórica

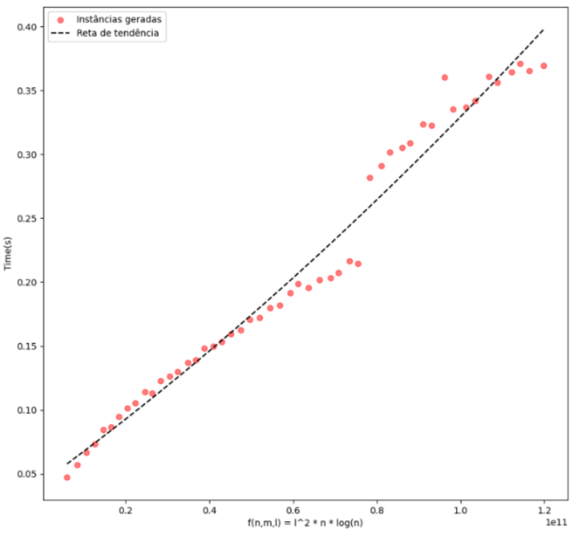
- **Leitura dos dados de entrada:** ciclo a depender linearmente de m (nº de ligações recebidas) inserindo o input num *vector<set<int>>*. Complexidade: $O(m+1) \times O(2\log n) \approx O(m) \times O(\log n) = \boldsymbol{O(m\log n)}$.
  - Verificação se todas as estações estão na mesma linha: percorremos todas as linhas e verificamos se algum dos *set<int>* tem tamanho n (*size()* é $O(1)$). Complexidade: $\boldsymbol{O(l)}$.
  - Verificação se faltam estações no input: temos um vetor de boolean com tamanho n inicializado a false. Sempre que recebemos informação de uma dada estação x, alteramos o valor da estação x para true. Complexidade: $\boldsymbol{O(n)}$.
- **Construção do grafo:** olhando para o pseudocódigo acima apresentado, a complexidade é: $O(l \times l \times n \times \log n) = \boldsymbol{O(l^2 n \log n)}$.
- **Cálculo do índice de conectividade:** aplicamos o algoritmo BFS para todos os vértices do grafo de linhas. Complexidade: $\boldsymbol{O(l)}$.
  - **Aplicação do algoritmo BFS:** o algoritmo tem complexidade $O(V+E)$, onde V é o nº de linhas de metro (l) e E é o nº de conexões entre as linhas, ou seja, nº de arcos do grafo.

> Pela teoria dos grafos, o nº máximo de arcos de um grafo com L vértices é $\boldsymbol{\frac{L(L-1)}{2}}$

Logo, $O(l) \times O(l+E) = O(l^2 + lE) \approx O\left(l^2 + l\left(\frac{l(l-1)}{2}\right)\right) \approx O(l^2 + l^3) \approx \boldsymbol{O(l^3)}$

Deste modo, $\boldsymbol{O(m\log n + l + n + l^2 n\log n + l^3) = O(m\log n + l^2 n\log n)}$, considerando $\boldsymbol{l \ll n}$, $O(l) < O(n) < O(l^2 n \log n)$ e $O(l^3) < O(l^2 n\log n)$ e tendo em conta que $\boldsymbol{m \le l \times n}$, $\boldsymbol{O(ln\log n) < O(l^2 n\log n)}$. Conclui-se então que a complexidade que domina é a da construção do grafo. Assim, a complexidade global da solução é $\boldsymbol{O(l^2 n \log n)}$.

Avaliação Experimental dos Resultados



Após ter gerado várias instâncias de tamanho incremental, obteve-se o seguinte gráfico: no eixo das abcissas, está a função correspondente à complexidade prevista; no eixo das ordenadas, o tempo em segundos.

Observa-se que, existem instâncias com desvios em relação à regressão linear, o que é esperado, pois o tempo de execução é influenciado também por outros fatores externos.

Assim, como os dados aproximam-se a uma reta linear, conclui-se que a implementação está de acordo com a análise teórica.