Relatório 2º projeto ASA 2024/2025

Grupo: AL070

Aluno(s): Ana Santos (109260) e Francisco Mendonça (109264)

Descrição da Solução

O problema baseia-se em, dada uma rede de metro, calcular o pior caso do número mínimo de mudanças de linha necessárias para viajar entre quaisquer duas estações de metro presentes na rede.

A nossa solução para o problema consiste na transformação da rede de metro num grafo bipartido onde, no qual, as estações estão diretamente ligadas às respetivas linhas. Após a sua construção aplicamos o algoritmo BFS na primeira estação, que não seja uma interseção, de cada linha. Com o algoritmo é calculada a distância entre as estações de linhas diferentes, isto é, o número mínimo de mudanças de linha, sendo retornado o máximo valor destas distâncias. Se alguma estação não for visitada significa que não está conectada a nenhuma linha.

Pseudo-código da construção do grafo:

```
graph()
| let m be the number of connections
| let n be the number of stations
|   for int i = 0 to m
|   | let lines be line + n -1
|   |  graph[station X - 1].push(lines)
|   |  graph[station Y - 1].push(lines)
|   |  graph[lines].push[station X ]
|   |  graph[lines].push[station Y]
```

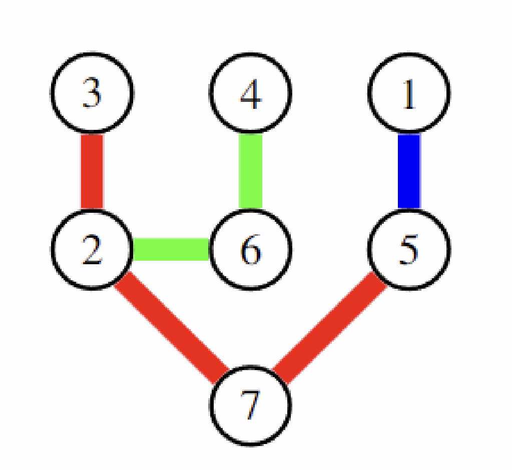


Fig. 1 – Exemplo 2 do projeto

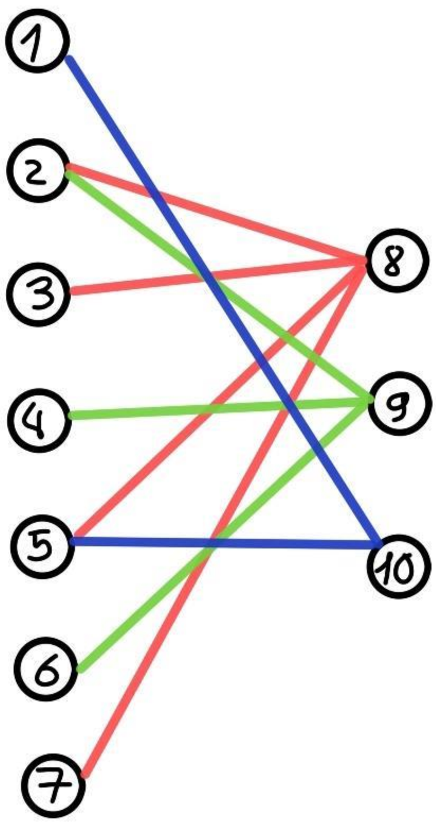


Fig. 2 – Representação do exemplo 2 no nosso grafo

Relatório 2º projeto ASA 2024/2025

Grupo: AL070

Aluno(s): Ana Santos (109260) e Francisco Mendonça (109264)

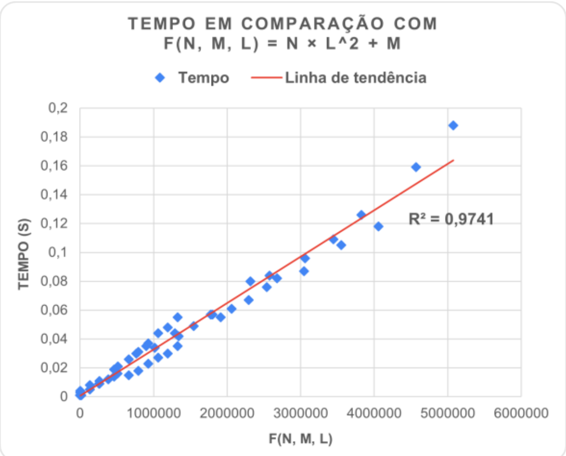
Análise Teórica da Solução Proposta

- Leitura dos dados e construção do grafo: leitura simples do input, com um ciclo a depender linearmente do número de conexões, M, que lê as conexões ao mesmo tempo que constrói o grafo. Logo a complexidade é $O(M)$.
- Aplicação do algoritmo BFS para cálculo do número máximo de mudanças de linha: visitamos todas a estações e linhas a partir de uma estação inicial, em cada estação passamos pelas suas linhas e em cada linha pelas suas estações, logo $(N \times L) + (L \times N) = 2NL$. No fim passamos por todas as estações de modo a verificar se todas foram visitadas, logo N. Assim a complexidade de BFS é $O(2NL + N) \Rightarrow O(N(2L + 1)) \Rightarrow O(N \times L)$. Como aplicamos a BFS a uma estação por linha para verificar o valor máximo para qualquer ponto, aplicamos o algoritmo L vezes. Logo $O(N \times L) \times O(L) \Rightarrow O(N \times L^2)$
- Complexidade global da solução: $O(N \times L^2) + O(M) \Rightarrow O(N \times L^2 + M)$

Avaliação Experimental da Solução Proposta

Para a parte experimental utilizamos o gerador de instâncias fornecido de forma a testar a análise de complexidade teórica. Geramos 50 instâncias de tamanho incremental $f(N, M, L)$ com diferentes combinações de N (25-50000), M (40-75000) e L (5-100) e medimos o tempo de execução.

Para a análise teórica $O(N \times L^2 + M) \Rightarrow f(N, M, L) = N \times L^2 + M$, o seguinte gráfico representa o tempo (eixo dos YYs) em função de $f(N, M, L)$ (eixo dos XXs).



Ao analisar o gráfico, é possível verificar que a análise teórica da complexidade do nosso algoritmo está correta visto que é possível estabelecer uma relação linear entre a complexidade teórica prevista e os tempos registados, R^2 é praticamente 1, o que significa que a nossa função representa muito bem a complexidade do código na prática.