**Descrição do Problema e da Solução**

O objetivo do problema é desenvolver um algoritmo para encontrar o número mínimo de mudanças de linha necessárias para viajar entre duas quaisquer estações da rede de metro dada. Esta rede de metro é definida como um multi-grafo não-dirigido com N vértices, onde cada vértice representa uma estação de metro, e com M arestas, onde cada aresta uma secção de uma linha, isto é, uma ligação entre duas estações da rede de metro. Além disso, existem L linhas diferentes na rede. Assim sendo, recebemos como entrada N estações de metro que juntas fazem M ligações entre si em L linhas.

A solução passa por fazer um multi-grafo não-dirigido, em que cada vértice representa uma linha de metro, e cada aresta representa uma ligação entre duas linhas. Isto é, caso duas linhas tenham pelo menos uma estação em comum, então estas estarão ligadas no nosso grafo. Deste modo, o nosso grafo terá L vértices e E arestas, em que o E representa o número de ligações entre linhas. Após a construção deste grafo, percorremos o mesmo com uma BFS L vezes, que nos retornará o número máximo de mudanças de linhas que alguém fará para ir de uma estação para outra, já que a distância entre dois vértices no nosso grafo corresponde ao número de mudanças de linha necessárias para ir de uma estação na primeira linha para uma estação na segunda linha.

**Análise Teórica**

* **Leitura dos dados de entrada e processamento dos dados:** simples leitura do input, com ciclos a depender de M (número de ligações da rede de metro), em que se cria a linha Z, caso ainda não tenha sido criada, e adiciona-se as estações X e Y ao set de estações que estão inseridas na linha, com complexidade O(). Caso a linha Z já tenha sido criada, obtém-se o ponteiro para a mesma, percorrendo uma lista com todas as linhas até encontrarmos a pretendida e adiciona-se as estações X e Y ao set de estações que estão inseridas na linha. Logo, O().
* **Construção do grafo:** verifica quais linhas têm estações em comum, através de um loop a depender quadraticamente de L, em que percorre a lista de estações que pertencem à lista X e verifica se pertence também à Y. Caso pertença, adiciona a estação Y ao vetor de estações ligadas da linha X e vice-versa. Logo, O().
* **Aplicação do algoritmo indicado para calcular o valor pedido:** utilização do algoritmo BFS para percorrer o grafo através de um loop a depender de L. Este grafo tem L vértices e E arestas, em que o E corresponde ao número de ligações entre linhas, pelo que será igual a , já que cada linha pode estar conectada a L-1 linhas. Logo, O().
* **Complexidade global da solução:** deste modo, a solução tem uma complexidade global de O().

**Avaliação Experimental dos Resultados**

Tendo em conta que o main() terá complexidade O(), para verificar este resultado, gerou-se testes em que se aumentava o tamanho de N em 10000 instâncias de 10000 até 30000, para M e L constantes. Além disso, aumentava-se o M em 300000 instâncias deste 300000 até 900000 para N e L constantes. Por fim, aumentava-se o L em 300 instâncias deste 300 até 900 para M e N constantes. Deste modo, gerou-se 27 testes para comprovar a complexidade teórica calculada.

Assim, construiu-se um gráfico com a complexidade teórica dos dados no eixo XX e com o tempo correspondente para cada N, M e L no eixo YY. Ao analisar o gráfico, observa-se uma complexidade linear entre a complexidade teórica e os tempos registados, confirmando que a implementação está de acordo com a complexidade teórica.

\*Testes corridos num Macbook M1 Pro com 16GB de RAM