Grupo: AL092 | Alunos: Mariana Carvalho, 109974; Beatriz Teixeira, 110263

Relatório do 2º projeto ASA 2024/2025

DESCRIÇÃO DA SOLUÇÃO

A solução desenvolvida para o estudo da rede de transporte urbano da Caracolândia utiliza um algoritmo baseado no algoritmo BFS para calcular o índice de conectividade do metro (número máximo de mudanças de linha necessárias para viajar entre quaisquer duas estações).

Assim, contruimos um grafo onde um **nó** representa uma estação de metro e uma **aresta** representa uma conexão entre duas estações, associada a uma linha. Deste modo, a solução verifica inicialmente a conectividade do grafo e, posteriormente, determina o número máximo de mudanças de linha através da análise do grafo das linhas do metro.

ANÁLISE TEÓRICA DA SOLUÇÃO PROPOSTA

```
BFS(graph, start_station, end_station):
    Create queue ← Empty Queue
    Create visited ← Empty Set
    Enqueue start_station into queue
    Add start_station to visited
    While queue is not empty:
        current_station ← Dequeue from queue
        If current_station = end_station:
            Return "Path found"
        For each neighbor_station in graph[current_station]:
            If neighbor_station is not in visited:
                Add neighbor_station to visited
                Enqueue neighbor_station into queue
    Return -1
```

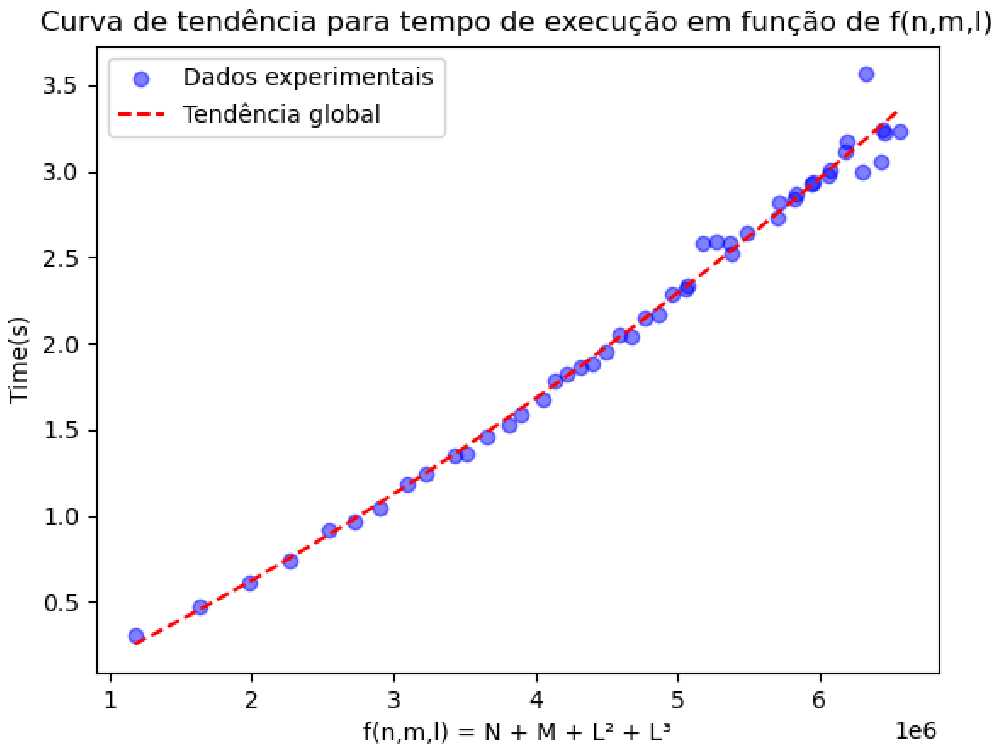
- Construção do grafo de linhas: todas as conexões **C** são percorridas. Posteriormente, para cada estação, verificamos todas as combinações possíveis de pares de linhas que compartilham essa estação. No pior caso, uma estação pertencerá a todas as linhas da rede. Complexidade: **$O(C + L^2)$**, com **C** (número de conexões) e **L** (número de linhas do metro).

```
For startLine := 1 to numLines:
    For endLine := (startLine + 1) to numLines:
        changes ← minLineChangesPath(startLine, endLine, line_graph)
        If changes = (numLines - 1):
            Return changes
        If changes > maxChanges:
            maxChanges ← changes
Return maxChanges
```

- BFS: para cada par de linhas **(L^2)** todas as estações **E** e todas as arestas **A** do grafo são percorridas, logo Complexidade do BFS entre linhas: **$O(L^2 * (L+A))$**, ou seja **$O(L^3)$**.
- DSU para verificar se um grafo é conexo: A DSU (Disjoint Set Union) é usada para verificar a conectividade do grafo de estações. Processamos todas as **E** arestas (conexões entre estações) para realizar operações de união (union) ou encontrar o representante de um conjunto (find). Cada operação de união ou busca tem um custo amortizado de **$O(\alpha(V))$**, onde **$\alpha(V)$** é a inversa da função de Ackermann. Esta função cresce extremamente lentamente e pode ser considerada quase constante para valores práticos de **V** (número de estações). Complexidade: **O(E)**.

Avaliando a complexidade global, temos a complexidade de construção do grafo mais a do DSU, mais a do cálculo da variável metro connectivity – **$O(L^3)$**. Temos então a seguinte expressão **$O(E+C+L^2+L^3)$**, que pode ser simplificada por **$O(L^3)$**.

AVALIAÇÃO DA SOLUÇÃO PROPOSTA



Este gráfico mostra a curva de tendência para o tempo de execução de uma função f(n,m,l), onde n representa o número de estações, m o número de conexões e l o número de linhas. O tempo de execução é representado no eixo Oy em segundos, enquanto os parâmetros n, m e l estão no eixo Ox. A curva ilustra como o tempo de execução varia à medida que os parâmetros aumentam. Assim, espera-se que o tempo de execução aumente com o crescimento de n, m e l, refletindo a complexidade do algoritmo em relação ao tamanho da entrada.