Relatório 2o projecto ASA 2023/2024

Grupo: TP021
Alunos: Maria Ramos (105875) e Guilherme Campos (106909)

Descrição do Problema e da Solução

O objetivo do problema é determinar o pior cenário de propagação de uma doença, considerando a instantânea transmissão entre indivíduos que se conhecem, direta ou indiretamente. A solução proposta emprega uma adaptação do algoritmo de Tarjan (iterativo) para encontrar componentes fortemente ligados (SCC), permitindo a identificação de grupos sociais impactados, combinado com a busca em profundidade (DFS) para determinar a quantidade máxima de conexões dentro dos componentes identificados no grafo social, chegando finalmente ao pior cenário de propagação de uma doença.

Análise Teórica:

Leitura dos dados de entrada:

```
Initialize V and E
For each edge from 1 to E:
    Read the edge data
```

- Leitura de V e E: O(1) - Operações constantes.
- Loop de leitura de relações: O(E) – Linear em relação ao número de relações.

Complexidade total da leitura: O(E)

Organização dos dados data na lista de adjacências ou na matriz da função principal:

```
Initialize adjacency list with size V x V
For each edge from 1 to E:
    Add the edge to the adjacency list
```

- Inicialização da lista de adjacência: O(V^2) – Quadrático em relação às relações da lista.
- Loop para organizar as relações na lista de adjacência: O(E) Linear em relação ao número de relações.

Complexidade total de organizar dados: O(V^2 + E)

Aplicação do algoritmo iterativo Tarjan com DFS para encontrar componentes fortemente ligados:

```
For each vertex i from 1 to numNodes:
    If the vertex i is not discovered (disc[i] == -1):
        Call tarjanSCC(i)
```

- **Algoritmo Tarjan para SCCs:** O(V * (V + E)). A complexidade aumenta quando chamado para cada um dos vértices do grafo.

Relatório 2o projecto ASA 2023/2024

```
Resize sccGraph to hold scc_count elements
For each node u from 1 to numNodes:
    For each node v in the adjacency list of u:
        If the strongly connected component of u is not the same as the
strongly connected component of v:
            Add v to the strongly connected component graph of u
```

- **Construção do gráfico de SCC:** O(V + E).

```
Initialize a dynamic programming array dp with size scc_count and fill
it with -1
Initialize max_connections to 0
For each i from 0 to scc_count:
    Update max_connections to be max(max_connections, dfs(i, dp))
```

- Depth-First Search (DFS): O(V * (V + E)). A complexidade da DFS aumenta consoante o número de componentes fortemente ligados, sendo no pior dos casos todos os vértices do grafo.

Complexidade total da aplicação do algoritmo: O(V * (V + E))

Apresentação dos dados:

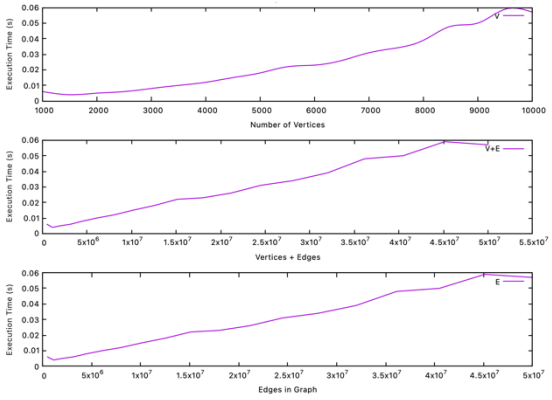
- Impressão do resultado: O(1) - Operações constantes.

Complexidade total da apresentação dos dados: O(1)

Complexidade global da solução: O(V * (V + E).

Avaliação Experimental dos Resultados:

Para avaliar o desempenho da nossa solução, utilizamos diferentes tamanhos de V + E, medindo o tempo de execução em segundos:



Os gráficos representam o desempenho do nosso algoritmo em relação ao aumento do input em três casos específicos. Os gráficos variam igualmente no tamanho de V+E, sendo realizada uma transformação nos eixos horizontal para f(O(V(V+E))), f(O(E)) e f(O(V)), respetivamente. Observamos uma relação linear com os tempos no eixo vertical, confirmando que nossa implementação está alinhada com a análise teórica da complexidade O(V(V+E)), uma vez que o gráfico formado é quase linear.