1 Descrição do Problema e da Solução

O problema apresentado foi o de encontrar o **número máximo** de saltos que uma doença pode fazer numa rede de interações sociais (mútuas ou não) entre um conjunto de indivíduos. Contabiliza-se como **salto da doença entre dois indivíduos** apenas se estes não se conhecerem mutuamente (direta ou indiretamente) isto é, caso um conjunto de indivíduos se conheça mutuamente, considera-se que a doença se propaga instantaneamente, pelo que não se contabilizam saltos.

Na conceção da solução, optou-se por representar a rede social como um grafo dirigido, que pode ser cíclico ou não, onde cada vértice corresponde a um indivíduo e cada ligação dirigida simboliza que o indivíduo na origem conhece o indivíduo no destino. Desta forma, num grafo cíclico, um conjunto de indivíduos que se conhece mutuamente corresponderá a uma SCC (*Strongly Connected Components*) e o cálculo do maior número de saltos que a doença pode efetuar passa por determinar o **maior caminho** possível entre as SCC do grafo. Para identificar as SCC no grafo dirigido, recorreu-se a uma versão iterativa do algoritmo de **Kosaraju-Sharir**. Neste algoritmo, começa-se por fazer uma procura em profundidade-primeiro (DFS) de modo a ordenar os vértices por tempo de descoberta, guardando-os numa pilha. Através desta pilha, iremos fazer uma segunda DFS no grafo transposto de modo a identificar as SCC do grafo. A motivação por detrás deste algoritmo baseia-se no facto de as SCC de um grafo e do seu transposto serem as mesmas: após a primeira DFS, os vértices ficam ordenados em *post-order*, o que significa que qualquer vértice v que seja visitado seja por ordem numérica, seja por ser vizinho de um outro vértice u que já foi visitado, será adicionado à pilha antes de u, logo, se houver um caminho direto de u para v, então v estará na pilha antes de u (a menos que sejam da mesma SCC). Desta forma, ao realizar a segunda DFS no grafo transposto pela ordem dos vértices indicada na pilha, cada conjunto de vértices descobertos em cada visita corresponderão a uma componente fortemente ligada. Ao criar os grafos, todos os vértices apresentam *streak* igual a zero. Após identificar a SCC v no final de uma visita durante a segunda DFS, procede-se ao cálculo do respetivo *streak* iterando por todas as suas n SCC vizinhas. Em cada iteração i de 0 a $n-1$, caso o *streak* do vizinho e seja superior ao de v, atribui-se-lhe esse valor de *streak* incrementado em um, de acordo com a seguinte equação:

$$v_{streak} = \begin{cases} 0 & n = 0 \\ \underset{i<n}{max}\{e^i_{streak}, v_{streak}\} + 1 & n > 0 \end{cases} \quad (1)$$

O valor do maior número de saltos max_{streak} (inicialmente a zero) é atualizado após o cálculo de v_{streak} de acordo com a expressão:

$$max_{streak} = \underset{i<n}{max}\{max_{streak}, v_{streak}\} \quad (2)$$

Quando a segunda DFS termina, terá sido determinado o maior número de saltos possíveis na rede.

2 Análise Teórica

Sendo V o número de indivíduos e E o número de relações:

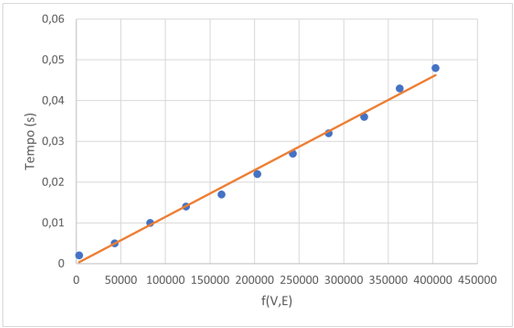
- Leitura dos dados de entrada: leitura do número de indivíduos (vértices) V e do número de relações (arestas) E e leitura das E relações num ciclo de complexidade linear e respetiva construção do grafo. Logo, $O(E)$.
- Criação do grafo transposto: exige visitar todos os vértices e identificar todas as ligações. Logo, $\Theta(V + E)$.
- Primeira DFS (por ordem numérica): realiza uma procura em profundidade primeira no grafo original, visitando todos os vértices e adjacentes, guardando-os por tempo de descoberta na pilha (no topo ficando o último vértice descoberto). Sendo que cada vértice e vizinhos (arestas) são visitados uma vez apenas, a complexidade é $O(V + E)$.

- Segunda DFS: realiza uma procura em profundidade primeira no grafo transposto, visitando todos os vertices com início no topo da pilha e identificando as componentes fortemente ligadas. Depois, calcula-se o maior *streak* por cada componente identificada em $O(E)$. Sendo que cada vertice e vizinhos (arestas) são visitados uma vez apenas, a complexidade será $O(V + E)$.
- Cálculo do maior *streak* de cada componente fortemente ligada: realizado durante a segunda DFS, é necessário visitar todas as componentes adjacentes, identificar aquela com *streak* máximo e incrementá-lo em 1. Se o *streak* for o maior calculado até ao momento, atualiza-se o valor do *streak* máximo. Logo, $O(E)$.
- Apresentação do número máximo de saltos (isto é, o *streak* máximo calculado) é realizada em tempo real, isto é, $\Theta(1)$.

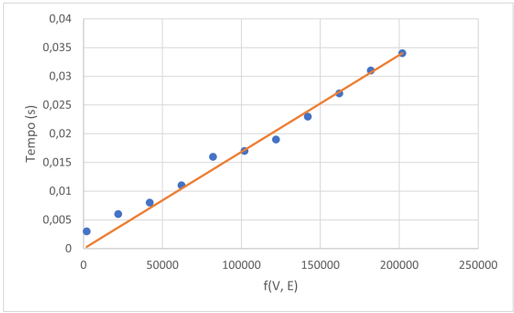
Como tal, de acordo com a análise teórica, a complexidade geral $f(V, E)$ do problema proposto é $O(V + E)$, pois são os termos dominantes de todas as complexidades analisadas.

3 Avaliação Experimental dos Resultados

Os seguintes gráficos de Excel representam, respetivamente, os resultados experimentais obtidos para mais de 10 instâncias geradas por cada um dos geradores de grafos fornecidos para valores incrementais de V e E. O primeiro gráfico (a) traduz os resultados obtidos para instâncias de grafos aleatórios, enquanto que o (b) traduz os resultados utilizando instâncias de grafos exclusivamente circulares. Cada valor de tempo correspondente foi obtido realizando a média entre o menor e o maior valor obtidos para mais de 10 execuções por instância:



(a) Resultados obtidos para o gerador 1.



(b) Resultados obtidos para o gerador 2.

Para ambos os conjuntos de instâncias geradas, verifica-se que o tempo de execução cresce linearmente em função de $f(V, E)$. Como tal, a complexidade temporal do problema corresponde, de um modo geral, à complexidade prevista na análise teórica.

4 Referências

- Kosaraju-Sharir's Algorithm (last accessed on 20th December 2023)