Relatório 2º projeto ASA 2023/2024

Grupo: AL061
Alunos: Miguel Casimiro Barbosa (106064) e Diogo Miguel dos Santos Almada (106630)

Descrição do Problema e da Solução

O problema reside em encontrar uma solução para o maior número de saltos que uma doença pode fazer entre indivíduos, isto é, o maior caminho que se consegue encontrar num grafo dirigido, tendo em conta que indivíduos que se conhecem mutuamente de forma direta ou indireta ficam infectados instantaneamente, ou seja, o número de saltos entre vértices de um mesmo SCC não aumenta. Para tal, começamos por ler o input e colocar as adjacências dos vértices em duas listas de adjacências que representam o grafo e o grafo transposto (consideramos índices de 0 a V-1 para não ocupar memória desnecessária). No total, são feitas duas DFSs: na primeira, é construída uma stack com a ordem decrescente de tempos de fim da DFS; na segunda, usamos a stack mencionada agora para percorrer o grafo transposto de uma source para uma sync do grafo original e, usando as adjacências de cada vértice (do grafo transposto), descobrir o maior número de saltos possíveis no grafo, respeitando as condições do problema descrito.

Ao fazer a segunda DFS, que encontra os SCCs à medida que visita os vértices da pilha inversa e colocando-os a cinzento, examinamos os vértices adjacentes aos mesmos. Se o vértice for branco significa que pertence ao mesmo SCC e deve ser adicionado à stack da DFS, se for preto significa que pertence a outro SCC já percorrido. Nesse caso deve verificar-se se o valor de saltos do SCC a ser percorrido tem um novo máximo e atualizá-lo caso necessário, sendo uma unidade superior ao valor do SCC já percorrido, este último passo faz uso de um vetor com o número máximo de saltos até cada vértice. No final da travessia por um SCC, com auxílio de uma stack inicializada para cada SCC, os vértices desse SCC são marcados de preto e o número máximo de saltos até aos mesmos (igual para todos os vértices do mesmo SCC) é atualizado para garantir que todos possuem o mesmo valor máximo. É devolvido no final o número máximo de saltos no grafo, valor que vai sendo atualizado numa variável sempre que um SCC acaba de ser visitado.

Análise Teórica

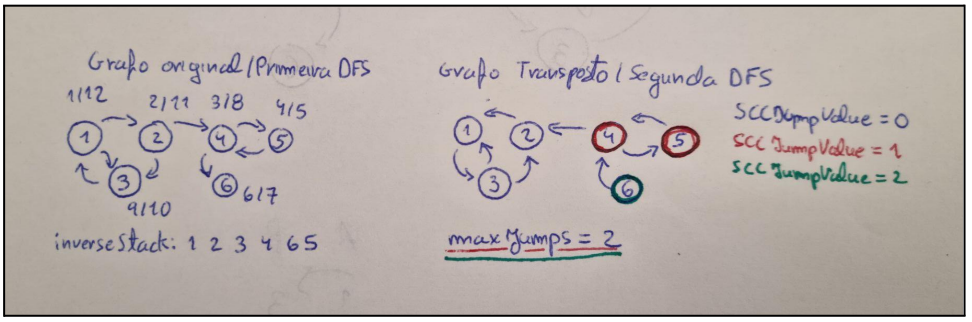
- Leitura dos dados de entrada: leitura do input correspondente ao número de vértices e ao número de arestas. Logo **O(1)**, irrelevante para a complexidade geral;
- Leitura dos dados de entrada: leitura do input correspondente à aresta criada por dois vértices e inserção da mesma na lista de adjacências no grafo e no grafo transposto, com ciclo a depender linearmente de E (número de arestas). Logo, **O(E)**;
- Chamar a DFS iterativa (a começar no vértice 1) para encontrar os tempos de fim dos vértices:
    - No primeiro *for*, percorrer todos os vértices do grafo, apenas executando o que está dentro do ciclo caso o vértice ainda não tenha sido visitado (ou seja, quando este ainda está a branco). Logo, **O(V)**;
    - Percorrer o segundo ciclo *while*, que acaba quando a stack fica vazia, ou seja, quando temos de recomeçar a travessia noutro vértice ou quando já não há mais vértices para visitar. Este ciclo itera sobre os vértices no máximo E vezes.
      Logo, **O(E)**;
    - Dentro do segundo ciclo, percorrer as adjacências (no ciclo *for*) do vértice em questão e dar *push* aos mesmos, caso estes ainda não tenham sido visitados (brancos), dando *push* para a stack dos tempos de fim da DFS caso já tenham sido visitados anteriormente, indicando que o tempo de fim foi definido. Logo, **O(E)**;
    - Assim, a complexidade da primeira DFS é **O(V + E)**.

Relatório 2º projeto ASA 2023/2024

Grupo: AL061
Alunos: Miguel Casimiro Barbosa (106064) e Diogo Miguel dos Santos Almada (106630)

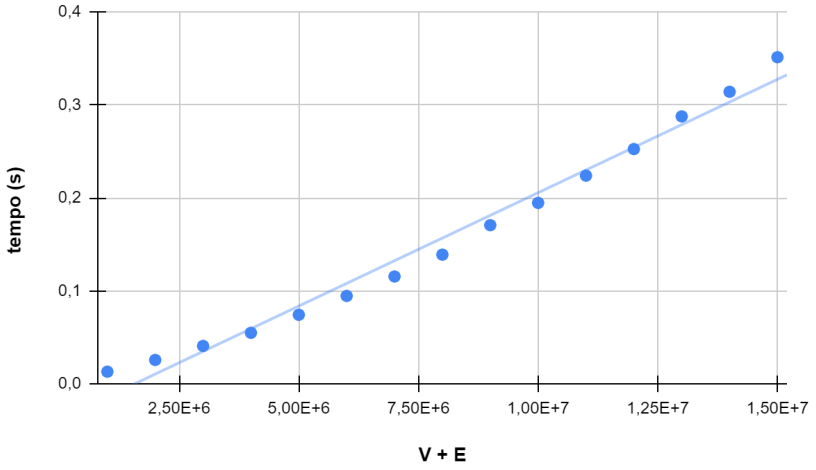
- Chamar a segunda DFS iterativa, para encontrar o maior número de saltos entre SCCs. (*Parte iterativa da DFS explicada anteriormente na primeira DFS)*:
    - *While* externo com V iterações, pois percorre a stack construída na primeira DFS. Internamente, o código apenas é executado o número de vértices que existirem no grafo devido a cada vértice só ser visitado apenas uma vez (quando é branco/ não visitado). Logo, **O(V)**;
    - Como cada vértice é visitado apenas uma vez, as suas adjacências são examinadas apenas uma vez no loop *for* interior, pelo que as arestas serão percorridas no máximo uma vez na DFS. Logo, **O(E)**.
    - Último while (após terminar de visitar um SCC), que percorre a pilha greyVerts (todos os vértices do SCC) e marca os vértices a preto (não voltam a ser visitados). Como cada vértice é visitado apenas uma vez, ao longo do primeiro ciclo while todos os vértices estarão nesta pilha uma e uma só vez independentemente do número de SCCs. Logo, **Θ(V)**.
    - Logo a complexidade da segunda DFS simplificada também é **O(V + E)**.

- Análise geral ignorando inicializações: **O(E) + O(V + E) + O(V + E)**, simplificando, **O(V + E)**.



Avaliação Experimental dos Resultados

A regressão linear abaixo foi construída colocando o eixo dos XX a variar com a quantidade prevista na análise teórica e o eixo dos YY a variar com o tempo calculado pela execução do algoritmo.



Como se pode averiguar, é observada uma relação linear entre a complexidade teórica prevista e os tempos registados, confirmando que a implementação está de acordo com a análise teórica.