Relatório 1º projecto ASA 2019/2020

Alunos: Mónica Jin (92532) e Margarida Moreira (93881)
Grupo: al010

Descrição do Problema: O problema apresentado tem como objetivo a previsão de notas dos alunos tendo em conta as conexões que eles têm entre eles, i.e. um aluno fica com a nota mais alta da rede de conexões que tem.

Descrição da Solução: Na solução proposta(desenvolvida em C++), a rede de conexões é representada através de um grafo dirigido, onde os alunos são os vértices e as conexões são os arcos dirigidos, e.g. (u,v) indica que o aluno u tem v como amigo. No código submetido, escolhemos representar o grafo sob a forma de uma lista de ajacências. Para tal, usámos um vetor de listas unidirecionais, pois estas permitem otimizar o espaço usado.
Para resolver o problema baseámo-nos no algoritmo de Tarjan implementando modificações com o objetivo de atualizar a nota de cada aluno para a mais alta possível, tendo em conta as suas conexões.
A escolha do algoritmo de Tarjan permitiu-nos que os vértices fossem atualizados por ordem topológica inversa, i.e. é-nos garantido que o(s) vértice(s) que terminámos de visitar não tem conexões com os vértices que ainda não terminaram de ser visitados. Esta propriedade é útil uma vez que temos a garantia que vértices acabados de serem visitados não terão de ser atualizados posteriormente. Este algoritmo também localiza os componentes fortemente ligados, o que nos permite colocar a mesma nota em todos os vértices de cada componente.

Análise Teórica

Algorithm 1 Tarjan Modificado

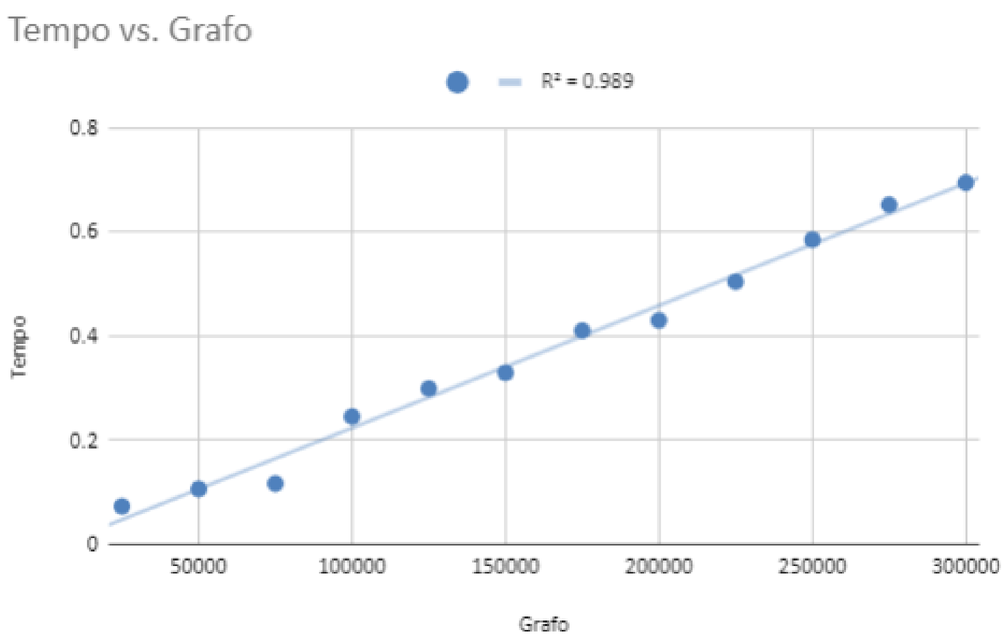
```
procedure TARJAN(G, numV)                        ▷ Complexidade do Tarjan: O(V+E)
    for each vertex u ∈ V[G] do                  ▷ Inicialização: O(V)
        disc ← lows ← −1
        onStack ← false
    for each vertex u ∈ V[G] do                  ▷ Chamadas ao Visit: O(V)
        if disc[u] = -1 then
            Visit(u)
        print grades[u]

procedure VISIT(u)                               ▷ Complexidade do Visit: O(V+E)
    disc ← lows ← time
    time ← time + 1
    Push(stack, u)
    for v ∈ Adj[u] do                            ▷ Análise da lista de adjacências: O(E)
        grades[u] ← max(grades[u], grades[v])
        if if disc[v] = -1 then
            Visit(v)
            grades[u] ← max(grades[u], grades[v])
        if onStack[v] then
            lows[u] = min(lows[u], lows[v])
    if disc[u] = lows[u] then
        then repeatv ← Pop(stack)                ▷ Pop de todos os vértices de um scc: O(V)
        grades[u] ← max(grades[u], grades[v])
```

Relatório 1º projecto ASA 2019/2020

Alunos: Mónica Jin (92532) e Margarida Moreira (93881)
Grupo: al010

Análise da Complexidade
Leitura de input: O(V) (Leitura das notas) + O(E) (Leitura das conexões)
Tarjan: O(V+E) (Dentro do tarjan fazemos o output das notas finais)
Complexidade Total: O(V+E)

Avaliação Experimental dos Resultados
Para comprovar a complexidade temporal mencionada anteriormente, O(V+E), realizámos vários testes com grafos de diferentes tamanhos(entre 25000 a 300000) medindo o tempo de execução do algoritmo apresentado. Os dados obtidos encontram-se no gráfico abaixo.



Conclusão
Como se pode observar no gráfico, o tempo cresce linearmente em função do aumento do tamanho grafo e o valor de R^2 é aproximadamente igual a 1, o que comprova que a complexidade temporal da solução proposta é de facto O(V+E), i.e. linear.