**Descrição do Problema e da Solução (Necessita revisão gramatical)**

Problema: Assumindo que os alunos partilham resumos, formas de estudo e soluções com os seus amigos, o professor da disciplina em questão considerou que a nota dos alunos devia ter em conta as notas dos seus amigos. Então, foi-nos proposto que desenvolvêssemos um algoritmo otimizado que alterasse as notas previamente obtidas pela nota máxima das relações sociais de cada aluno.

Solução: Para representar o grafo e as suas conexões utilizámos listas de adjacências. A nossa solução baseia-se em utilizar o algoritmo de Tarjan para identificar as SCC do grafo e garantir que os filhos independentes (se existirem) são verificados antes dos seus predecessores. Criámos uma função que, no momento do Pop do algoritmo de Tarjan, atua de modo a alterar a nota máxima dos vértices das SCCs, se necessário.

A função verifica dois casos: Se o vértice for independente e o seu outdegree for diferente de 0, (como o algoritmo de Tarjan garante que as suas ligações já foram verificadas), trocamos a sua nota com a nota máxima dos filhos, se necessário; No caso de estarmos na presença de um vértice independente com outdegree igual 0 (o aluno não tem relações sociais), não acontece nada, porque não é possível a sua nota vir a ser alterada; No caso de ser um SCC, obtém o máximo das notas dos vértices pertencentes à SCC (juntamente com os seus vértices-filho) e troca para a maior obtida.

**Perguntar ao stor se podemos manipular este relatório a nossa merce, i.e., se podemos adicionar um subtópico em cada pagina como temos.**

**Análise Teórica**

1. Leitura dos dados de entrada: nesta função temos dois for’s: um para colocar no grafo os vértices (armazenando o espaço necessário), e outro para fazer as conexões entre os vértices. Assim, temos um for com complexidade Θ(V) e outro com complexidade Θ(E), pelo que no final a complexidade da leitura é Θ(V+E). ??????? (acho que sim)

parseCommandLine(){

Leitura()

Erros()

inicializaGrafo() //O(1)

for i=1 to numVertices: //O(V)

novoVertice(Nota)

for i=1 to numArestas: //O(E)

conexao(v1,v2)

}

compare (list<int> scc){

if tamanho[scc]==1 e temConexões():

for x conexões[scc] //O(V) ?

nota = max(x,elemento[scc]);

else if tamanho[scc]==1 e !temConexões():

return;

else{

max = scc[1] //Primeiro elemento

for x scc: //O(V+E)

If Nota[x] > max:

max = Nota[x]

for u conexões[x]

If Nota[u] > max:

max = Nota[u]

for v scc: //O(V)

if Nota[v] < max:

Nota[v] = max;

}

}

1. Exemplo:
2. Processamento do grafo para fazer alguma coisa. Logo, O(??)
3. Aplicação do algoritmo de Tarjan. Como dito no resumo da solução, utlizamos esse algoritmo para identificar as SCCs do gráfico e garantir que os filhos independentes, se existirem, são verificados antes dos seus predecessores.

Chamadas à função tarjan recursivamente → O(V)

Lista de adjacências de cada vértice é verificada apenas uma vez → O(E)

Logo, a aplicação do Tarjan → O(V+E)

1. Transformação dos dados com uma dada finalidade. O(?Y?Y?)
2. Apresentação dos dados. Neste passo, como vemos em baixo, temos um único for onde vai percorrer todos os vértices e apresentar a sua nota. Assim sendo, é Θ(V)

output(){

for i=1 to numVertices: //O(V)

printf(Nota[i]);

}

1. No final de toda esta análise, podemos concluir que:

Complexidade global da solução: O(V+E)

**Avaliação Experimental dos Resultados**

**Descrição do tipo experiências feitas e gráfico demonstrativo da avaliação de tempos associados**

Efetuámos vários testes, com grafos gerados pelo gerador fornecido pelo professor. Gerámos grafos, tal que V+E = 100000, 250000, 500000, 750000 e 1000000. Como pode ser visto na tabela e no gráfico a seguir representados, calculámos a média do tempo para cada instância e obtivémos o gráfico correspondente aos valores obtidos experiencialmente.

**Gerar vários grafos de tamanho incremental e cálculo dos tempos para cada instância. Gerar o gráfico do tempo em função do tamanho do grafo de entrada como exemplificado abaixo.**

|  |  |
| --- | --- |
| **V+E** | **Média de Milissegundos** |
| **100 000** | **57** |
| **250 000** | **151** |
| **500 000** | **316** |
| **750 000** | **486** |
| **1 000 000** | **653** |

Fig.1 – Tabela de complexidade da solução

Fig.2 – Gráfico de complexidade da solução

**Concluir se o gráfico gerado está concordante com a análise teórica prevista.**

Como podemos ver pela análise do gráfico, a linha obtida é uma linha quase linear. Uma vez que O(V+E) é uma função linear, podemos concluir que nos aproximámos bastante do resultado pretendido/previsto pela analise teórica previamente feita.