**Descrição do Problema e da Solução**

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 três casos:

* **Se o SCC possuir apenas um elemento:**

1. E o seu outdegree for diferente de 0:

-O algoritmo de Tarjan garante que as suas ligações já foram verificadas previamente, logo troca a sua nota com a nota máxima dos filhos, se necessário;

1. E o seu outdegree igual 0 (o aluno não tem relações sociais):

-Não acontece nada, porque não é possível a sua nota vir a ser alterada;

* **Se o SCC possuir mais do que um elemento:**

-Obtém o máximo das notas dos vértices pertencentes à SCC (juntamente com os seus vértices-filho) e troca para a maior obtida;

**Análise Teórica**

1. Leitura dos dados de entrada: na função que lê o input, temos dois for’s:

- Um para colocar no grafo os vértices (armazenando o espaço necessário) → Θ(V);

- Outro para fazer as conexões entre os vértices → Θ(E).

Assim, complexidade da leitura dos dados de entrada é Θ(V+E).

1. Aplicação do algoritmo de Tarjan:

- Inicialização dos valores de tempo de descoberta e do valor de low de cada vértice → Θ(V);

- Chamada de uma função recursiva auxiliar que encontra os SCCs do grafo → Θ(V);

- Lista de adjacências de cada vértice percorrida 1 vez → Θ(E);

- Chamada da função que troca o valor das notas das SCCs → O(V);

Assim, a complexidade da aplicação do algoritmo de Tarjan é 3\*O(V) + O(E) = O(V+E).

1. Aplicação da função que troca a nota das SCCs:

-Se a SCC possuir apenas um elemento e o seu outdegree for igual a 0, a função retorna → Θ(1);

- Se a SCC possuir apenas um elemento e o seu outdegree for diferente de 0, percorre a sua lista de adjacências e troca a sua nota para a maior nota dos vértices-filho → Θ(E);

- Se a SCC possuir mais do que um elemento, percorre para cada vértice a sua lista de adjacências para obter a nota máxima → Θ(V + E);

- Percorre todos os vértices da SCC para alterar a nota para a máxima obtida previamente, se necessário → Θ(V);

Assim, a complexidade desta função é 2\*O(V) + 2\*O(E) = O(V+E).

1. Apresentação dos dados:

- Percorre os vértices do grafo e retorna para a linha de comandos a nota de cada um → Θ(V);

Concluímos assim que, teoricamente, a complexidade global da nossa solução é O(V+E).

**Avaliação Experimental dos Resultados**

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 obtivemos o gráfico correspondente aos valores obtidos experiencialmente.

**(ms)**

**(V+E)**

Fig.1 – Tabela de complexidade da solução Fig.2 – Gráfico de complexidade da solução

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

Após uma análise cuidada do gráfico obtido, concluímos que a nossa solução aproxima--se à análise teórica previamente feita, uma vez que o gráfico corresponde a uma função linear.