# Descrição do Problema e da Solução

O problema colocado consistia em perceber qual o número de “saltos” que um vírus consegue dar dentro de uma rede de pessoas que se conhecem. Para solucionar a questão decidimos construir um grafo dirigido para representar a rede de pessoas, com arcos de x para y se x conhece y. Começamos por percorrer o grafo através de um algoritmo iterativo da DFS de forma a obtermos os tempos de fim (que são guardados do maior para o menor). De seguida, voltamos a realizar uma DFS iterativa (percorrendo os vértices de forma decrescente dos tempos de fim obtidos anteriormente), mas desta vez sobre o grafo transposto.

À medida que visitamos um determinado vértice calculamos também qual o maior caminho que podemos fazer até esse vértice. Para tal, verificamos os seus adjacentes e obtemos o maior dos caminhos até eles. O maior caminho até determinado vértice será então o valor anterior + 1.

# Análise Teórica

Na fase inicial é feita uma leitura dos dados de entrada através de um loop que depende de forma linear do número m de relações. É também construído o grafo e o seu transposto. Logo esta etapa é O(m).

De seguida é efetuada uma DFS iterativa que está dividida em duas funções: a função principal DFS que possui um loop inicial que coloca todos os vértice como não visitados e um segundo loop (loop 2) que percorre os vértices do grafo que ainda não foram fechados e chama a segunda função, DFSVisit, que possui uma stack onde inicialmente apenas se encontra o vértice a ser visitado e um loop que apenas termina quando a stack estiver vazia, dentro deste loop a função percorre as adjacências do vértice inicial (loop 3) e, caso estas não tenham sido visitadas adiciona-as à stack procedendo à sua visita na iteração seguinte. A complexidade da função DFS é então dominada pelos loops 1, 2 e 3, onde o loop 1 tem complexidade O(n), uma vez que percorre todos os vértices do grafo, o mesmo pode ser dito para o loop 2. A soma de todas as iterações do loop 3 corresponde ao número de relações de indivíduos (m). Logo esta etapa tem complexidade O(n+n+m) ≡ O(n + m).

Em seguida, é efetuada nova DFS, desta vez sobre o grafo transposto, utilizando a ordem decrescente dos tempos de fim (obtidos na etapa anterior). Nesta DFS, verificamos qual é o maior dos caminhos até cada adjacente do vértice a ser visitado, guardando este valor. Quando todos os adjacentes do vértice foram visitados, o valor do maior caminho será então o valor guardado + 1. Como todas estas operações são O(1) a DFS mantém a sua complexidade. Logo esta etapa é O(n+m).

Por fim é apresentado o caminho máximo entre os demais o que corresponde a uma complexidade O(1).

Conclui-se então que a complexidade global é O(n+m).

DFSVisit (G, source)

let S be a new stack with source

while S not empty

let v = S.top()

if v.color = BLACK

S.pop()

continue

time++;

if v.color = WHITE

for u ∈ G.adj[v]

S.push(u)

v.color = GREY

else if v.color = GREY

v.color = BLACK

S.pop()

v.time = time

else

DFS(G)

for each v ∈ G.V

v.color = WHITE

for each v ∈ G.V

if v.color = WHITE

DFSVisit(G, v)

**Notação :** WHITE = vizinhos por explorar, GREY = já explorei os vizinhos, BLACK = já fechei; *time* é uma variável global

# Avaliação Experimental dos Resultados

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **M** | **Complexidade** | **Tempo (s)** |
| 0,00E+00 | 0,00E+00 | 0,00E+00 | 0 |
| 1,20E+05 | 1,20E+05 | 2,40E+05 | 0,072 |
| 2,00E+05 | 2,00E+05 | 4,00E+05 | 0,14 |
| 3,00E+05 | 3,00E+05 | 6,00E+05 | 0,212 |
| 4,00E+05 | 4,00E+05 | 8,00E+05 | 0,292 |
| 5,00E+05 | 5,00E+05 | 1,00E+06 | 0,379 |
| 6,00E+05 | 6,00E+05 | 1,20E+06 | 0,447 |
| 7,00E+05 | 7,00E+05 | 1,40E+06 | 0,521 |
| 8,00E+05 | 8,00E+05 | 1,60E+06 | 0,601 |
| 9,00E+05 | 9,00E+05 | 1,80E+06 | 0,693 |
| 1,00E+06 | 1,00E+06 | 2,00E+06 | 0,803 |
| 1,10E+06 | 1,10E+06 | 2,20E+06 | 0,894 |
| 1,25E+06 | 1,25E+06 | 2,50E+06 | 1,024 |
| 1,50E+06 | 1,50E+06 | 3,00E+06 | 1,153 |

De forma a verificar a complexidade real da solução proposta para o problema realizamos testes onde calculamos o tempo necessário para obter o valor do número máximo de saltos para determinados n e m.

Analisando o resultado dos testes, verificamos que os tempos de execução variam de forma linear em relação à complexidade teórica estimada, o que confirma que a implementação está de acordo com a análise teórica.