Relatório 2º projeto ASA 2023/2024

Grupo: AL004

Aluno(s): Francisca Almeida (105901) e José Frazão (106943)

Descrição do Problema e da Solução

No âmbito da resolução do projeto para auxiliar o Professor João Caracol na análise do pior cenário de propagação de doenças transmissíveis em Portugal, estabelecemos uma analogia entre a solução desse problema e a solução do caminho mais longo num Grafo Direcionado Acíclico (DAG). Nesse contexto, consideramos os indivíduos como vértices e as relações entre eles como arcos.

Para abordar essa questão, implementamos o algoritmo de Kosajaru-Sharir. O processo envolve duas travessias em profundidade (DFS). Na primeira, calculamos a ordem topológica para cada vértice, com ênfase no tempo de finalização. Na segunda iteração, ordenando o grafo pelo tempo de descoberta final, determinamos para cada vértice a sua Componente Fortemente Conexa (SCC). Ao mesmo tempo, sempre que encontramos um arco entre SCCs distintas (representadas por a e b), atualizamos dinamicamente o valor do número máximo de saltos que podem ser dados até b. Isso permite a construção em tempo real da solução desejada para o problema de propagação de doenças.

Análise Teórica

- **Leitura dos dados de entrada**: simples leitura do input, um pedido inicial do número de pessoas e o número de relações e um ciclo a depender linearmente de V *(número de relações entre indivíduos)*.
    - Logo, O(V).
- **Processamento da instância**: inicializamos o grafo com um Loop que percorre o número de vértices.
    - Logo, O(V).
- **Aplicação do algoritmo indicado:** Este algoritmo divide-se em duas DFS iterativas, ambas com complexidade O(V+E). Na segunda DFS, enquanto é executada, calcula diretamente o maior caminho da SCC, acrescentando uma complexidade adicional de O(V).
    - A complexidade total destas etapas é, portanto, O(V+E).

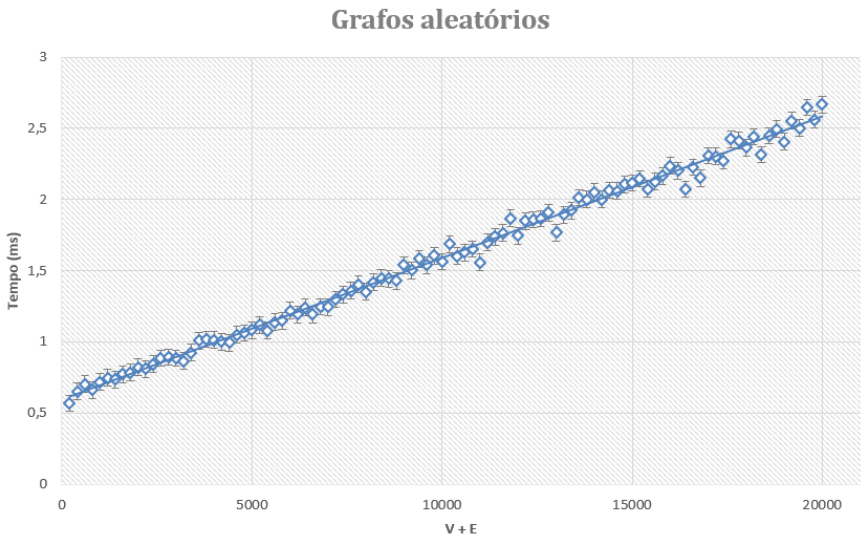
```
DFS(G)
    for each v in G.V
        v.color = white
    end for
    let S be a new stack
    time := 1
    for each v in G.V
        if v.color == white
            S.push(v)
            while !S.empty()
                u = S.top()
                if u.color ≠ white
                    if u.color == gray
                        u.color = black
                        u.f = time + 1
                        time = time + 1
                    end if
                S.pop()
                continue
                end if
            end while
            for each w in G.Adj[u]
                if w.color ≠ black
                    S.push(w)
                end if
            end for
            u.color = gray
        end if
    end for
```

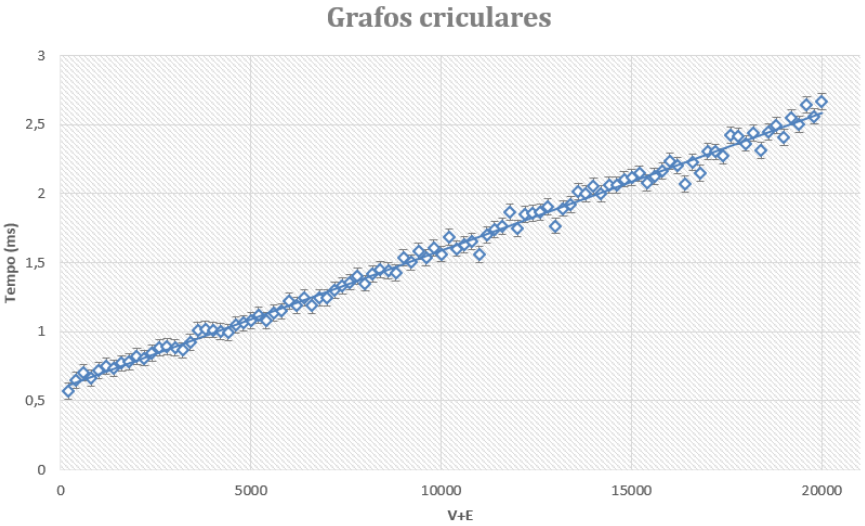
- **Apresentação dos dados**: Iteramos por um vetor das SCCs que, no pior dos casos, possui V elementos e procuramos o valor máximo deste.
    - Logo, O(V).

Complexidade global da solução: O(V+E)

Avaliação Experimental dos Resultados

Para avaliar o desempenho do algoritmo, elaboramos dois conjuntos de gráficos, um empregando grafos circulares e o outro com grafos aleatórios. Foram realizadas 100 amostras para cada conjunto, variando o número de nós e arestas de forma incremental. Iniciamos com um conjunto de dados contendo 100 pessoas e 100 relações, aumentando progressivamente em incrementos de 100 até atingir 10.000.





Como antecipado, observamos nos gráficos uma relação linear entre o pior caso e o tempo.