Licenciatura em Engenharia Informática

ESINF 2023/2024

Relatório Trabalho 3/ Projeto Integrador ESINF

Autores:

Rafael Rocha - 1201260@isep.ipp.pt

Marcelo Ramos - 1221638@isep.ipp.pt

Dephane Cabral - 1221636@isep.ipp.pt

Tiago Correia - 1211742@isep.ipp.pt

Docentes:

Fátima Rodrigues - mfc@isep.ipp.pt

Nuno Filipe Malheiro - nfm@isep.ipp.pt

Índice

Diagrama de Classes

Análise de Complexidade das Funcionalidades Implementadas

US EI01:

USEI02:

USEI03:

USEI04:

```
8 usages
public static ReturnData calculateMinimumSpanningTreeWithKruskal(Graph<Local, Integer> graph) {

    if (graph.isDirected()) return null;

    Graph<Local, Integer> minimumSpanningTree = new MapGraph<>( directed: false);

    for (Local vertex : graph.vertices()) {
        minimumSpanningTree.addVertex(vertex);
    }

    List<Edge<Local, Integer>> edgeList = new ArrayList<>();

    for (Edge<Local, Integer> edge : graph.edges()) {
        edgeList.add(edge);
    }

    // Ordenar
    edgeList.sort(Comparator.comparing(Edge::getWeight));

    // armazana o total dos pesos
    int totalWeight = 0;

    for (Edge<Local, Integer> edge : edgeList) {

        LinkedList<Local> connectedVerteces = Algorithms.DepthFirstSearch(minimumSpanningTree, edge.getVOrig());

        if (!connectedVerteces.contains(edge.getVDest())) {

            minimumSpanningTree.addEdge(edge.getVOrig(), edge.getVDest(), edge.getWeight());
            totalWeight += edge.getWeight();
        }
    }
    return new ReturnData(minimumSpanningTree, totalWeight);
}
```

O algoritmo usado para encontrar a árvore geradora de custo mínimo foi o algoritmo de Kruskal.

1. **Criação da árvore inicial:** Iteração sobre os vértices do grafo original: O(V), onde V é o número de vértices. Adição de vértices à árvore: O (1) para cada vértice.
2. **Criação da lista de arestas:** Iteração sobre todas as arestas do grafo original: O(E), onde E é o número de arestas.
3. **Ordenação da lista de arestas:** Ordenação da lista de arestas usando o algoritmo de ordenação rápido (quicksort ou mergesort): O (E log E).
4. **Iteração sobre a Lista de Arestas Ordenada:** Iteração sobre todas as arestas ordenadas: O(E). Para cada aresta, realiza uma busca em profundidade (DFS) para verificar se a adição da aresta cria um ciclo na árvore atual. A busca em profundidade tem uma complexidade de O (V + E), mas neste caso, considerando o grafo é uma árvore, a complexidade é O(V). Adição de arestas à árvore: O (1) para cada aresta.

Resumindo a complexidade do algoritmo implementado é de aproximadamente O (E log V).

Melhorias Possíveis

Tentar melhorar a complexidade dos algoritmos desenvolvidos. O algoritmo desenvolvido para encontrar a árvore geradora do custo mínimo adiciona vértices que geram ciclos e isso seria remover esse comportamento.