**Documentação do segundo Trabalho Prático da disciplina de Algoritmos I**

**Luis Gabriel Caetano Diniz – Matrícula: 2019075711**

Departamento de Ciência da Computação – Universidade Federal De Minas Gerais (UFMG)

Belo Horizonte – MG – Brazil

[lgcaetano@ufmg.br](mailto:lgcaetano@ufmg.br)

1. **Modelagem Computacional do Problema**

O problema proposto no trabalho consistia no desenvolvimento de um algoritmo que receberia as rotas de determinada companhia aérea e deveria imprimir na saída padrão quantas rotas adicionais eram necessárias para que qualquer um dos aeroportos atendidos pela companhia fosse alcançável a partir de qualquer outro.

Para resolver o problema, o modelamos usando grafos. Para cada entrada do programa, pode-se construir um grafo onde os vértices representam os aeroportos atendidos pela companhia enquanto as arestas representam as rotas oferecidas. Assim, podemos resolver o problema com o auxílio das diversas técnicas e algoritmos conhecidos sobre grafos.

1. **Estruturas de Dados e explicação do algoritmo**

**2.1 Classes e estruturas utilizadas**

Para podermos usar o conhecimento que já temos sobre grafos para solucionar o problema, precisamos de uma estrutura de dados para representá-los. Por esta razão, criamos a classe **Grafo**, que representará o conjunto de rotas e aeroportos disponibilizados. A classe possui um construtor, que inicializa uma matriz que será sua matriz de adjacência, um destrutor, uma função **contemAresta** que recebe dois números inteiros, que devem representar vértices, e retorna **true** caso exista uma aresta saindo do primeiro vértice em direção ao segundo, contém também uma função **contruirAresta** que, evidentemente, constrói uma aresta direcionada do vértice representado pelo primeiro parâmetro do método em direção ao segundo, um método denominado **dfs**, que executa o algoritmo DFS e retorna um vector de inteiros contendo os índices dos vértices que foram alcançados pelo algoritmo a partir do vértice inicial, além de receber um vector por referência (que utiliza para armazenar quais vértices já foram visitados pelo algoritmo) e uma pilha (stack) utilizada para colocar os vértices em ordem decrescente de acordo com o tempo de término no algoritmo DFS (algo essencial para a execução do algoritmo de Kosaraju). Além disso, a classe **Grafo** também possui o método **algoritmoDeKosaraju**, que executa o algoritmo de Kosaraju, retornando um vector de vectors de inteiros (algo próximo, mas não igual, a uma matriz de inteiros) contendo em cada linha os índices dos vértices que compõem um componente fortemente conectado