**Pontifícia Universidade Católica de Minas Gerais** 

**Curso:** Engenharia de Software

**Matéria:** Teoria dos Grafos e Computabilidade

**Professor(a):** Zenilton Kleber Gonçalves do Patrocínio Júnior  
**Alunos:**

Guilherme Gabriel Silva Pereira (705903)

Lucas Ângelo Oliveira Martins Rocha (689603)

**TRABALHO PRÁTICO N.01**

Nosso trabalho foi desenvolvido na linguagem Java, implementamos três estruturas de grafo, matriz de incidência, matriz de adjacência e lista de adjacência. Porém, a implementação das três estruturas foi apenas para testes, possibilitando entendê-las melhor e ajudar na decisão de para cada caso, qual irá se sair melhor na nossa visão.

Depois de muitos testes e análises, chegamos à conclusão que a lista de adjacência é a melhor para todos os casos, direcionado e ponderado, não-direcionado e ponderado, direcionado e não-ponderado e não-direcionado e não-ponderado.

Essa conclusão mencionada no parágrafo anterior se deve a uma série de motivos. Dentre eles, o fato de que matriz de adjacência fica espelhada caso tente representar as relações de um grafo não-direcionado, além de não ser possível representar os pesos de um empoderamento, com isso, a única estrutura que poderia ser bem representada seria um grafo direcionado e não-ponderado, porém, mesmo assim continuará com o problema de possuir muitas posições com valor zero ocupado espaço de memória.

A outra estrutura cogitada foi a matriz de incidência, porém, não foi escolhida por possuir algumas características. Tais características são que também não conseguem representar um os pesos das arestas sem mudar a estrutura original, pois seria possível representar os pesos do poderamento se na primeira linha colocasse os pesos de cada aresta, já que cada coluna representa uma aresta (relação entre os vértices). Diante disso, consideramos melhor não alterar a estrutura padrão de uma matriz de incidência para adicionar pesos. Além disso, essa matriz, também como a de adjacência, ocupa muito espaço em memória com posições possuindo zero, já que quanto mais vértices possuir, mais zeros cara coluna irá possuir.

Conclui-se que a lista de adjacência é a melhor para todos os casos, já que consegue guardar o peso dos arcos e dá suporte para grafo direcionado e não-direcionado. Com um bônus de que com a nossa implementação com dois vetores, não ocupa memória desnecessária como as matrizes, sem repetição de valores de vértices nem suas relações.