# Teoria dos Grafos – Trabalho Pratrico I

**Autor: João Marcos Oliveira Lima**
**Professor: Willian P. Amorim**

[1](UFGD) UNIVERSIDADE FEDERAL DA GRANDE DOURADOS
Dourados – MS - -Brasil

[2](FACET) Faculdade de Ciências Exatas e Tecnologia
UFGD

{j.marcos.jomol@gmail.com

***Abstract.*** *O presente trabalho consiste em apresentar uma biblioteca capaz de realizar a representação de grafos por meio de lista de adjacência e matriz de adjacência.*
*Essa biblioteca deve ser capaz de não só representar mais também realizar um processo de busca por profundidade e em largura partindo de um nó raiz que será informado pelo usuário.*

## 1. Introdução

O trabalho foi proposto pela disciplina de teoria dos grafos ministrada pelo professor Willian P. Amorim o trabalho apresentava a proposta de desenvolver uma biblioteca que possa ser facilmente utilizadas em outros códigos, cujo a biblioteca deve ser capaz de representar um grafo por meio de lista de adjacência e matriz de adjacência, também a mesma deve ser capaz de realizar o processo de busca por meio de algoritmo de a profundidade e em largura.
A raiz do grafo deve ser recebida como entrada do usuário juntamente com o nome do arquivo onde se encontra grafo, o tipo de representação que o grafo usara e o tipo de busca que será aplicada.
A linguagem na qual foi escolhida para o desenvolvimento da biblioteca foi o C++ por ter mais afinidade e ser uma linguagem compilada e de baixo nível então acaba sendo mais rápida para ser executada.
A processo de saída da biblioteca será um arquivo de saída contendo o numero de vértice do grafos, numero de arestas, a media dos graus de cada vértices contido no grafo, e outro arquivo contendo o resultado da busca ou seja vértice que tá sendo visitado no momento iniciando pela raiz , e o nível onde o vértice se encontra no grafo.
Artigos e livros usados para estudos:
[Jurkiewicz 2009].
[Goldbarg 2012].
[Cormen 2009]

## 2. Desenvolvimento

O processo de desenvolvimento foi realizado por meio de uma classe denominada Grafo onde está contido todas as estruturas com as variáveis necessária e os métodos.
Para melhor facilita o entendimento foi dividindo o processo de desenvolvimento em quatro subsecção são elas :

### 2.1. Lista de Adjacência

A lista foi desenvolvida por meio de uma estrutura ¡lista¿ do C++ onde cada elementos da lista e um ponteiro que vai apontar para outras listas que serão usadas para armazenar seus vértices adjacentes, para isso e realizado um processo de busca no arquivo onde cada elementos adjacente a um vértice é acionado a sua lista que ele aponta.
O exemplo abaixo representa um grafo completo de 3 vértice (K3)onde os elementos entre o colchete são ponteiros para outra lista .

### 2.2. Matriz de Adjacência

A matriz foi implementada com uma matriz dinamicamente alocada de tamanho NxN onde N é o numero de vértice do grafo, usando o principio de adjacência quando um vértice tem adjacência com outro é adicionada o 1 na matriz de adjacência rede rente a linha e coluna pela qual os vértice se encontrando na matriz e caso o contrario aciona-se 0.
A vantagem e o acessado por ser um acesso sequencial onde para ver se dois vértices tem adjacência e só verifica diretamente na matriz as posições dos vértices, exemplo ver se o vértice p e q tem adjacência basta verifica se o elementos da Matriz[p][q] == 1 caso seja 1 tem adjacência caso o contrario não tem.
Desvantagem é a memoria se tornando inviável para casos onde são usados muitos vértices, gastando um quantidade absurda de memoria para representar casos onde não adjacência armazenando apenas 0 na matriz .

### 2.3. Busca por Profundidade

A busca em profundidade foi implementada para as duas representações tanto a matriz quanto na lista, o algoritmo é o mesmo, usando uma estrutura auxiliar pilha para a funcionalidade do algoritmo e passando o nó que será a raiz e a lista ou a matriz.
A busca foi implementada de maneira interativa e obedecendo o critério em que a busca sempre inicia da raiz e então vai empilhando na ordem do vértice adjacente mais a esquerda do gravo e ir desempilhando de maneira sempre pega o ultimo elemento inserido na pilha e assim até percorrer todos elementos do grafo.

### 2.4. Busca por Largura

A busca em largura também foi implementada para as duas representação do grafo matriz ou lista, a grande diferença entre a busca em largura e a busca por profundidade é que a busca por profundidade usa uma estrutura fila como auxiliar.
A busca em largura foi implementada de maneira interativa, obedecendo o critério de enfileirar os elementos adjacente a raiz e depois desenfileirar seus elementos até que a busca esteja completa por todo grafo, sempre pegando o primeiro elemento da fila empilhando e depois removendo da estrutura.

## 3. Dificuldade

A maior dificuldade encontrada foi a apresentação dos neveis no momento em que remove o vértice da estrutura no percurso da busca, mais esse problema foi contornado na busca em profundidade que o problema foi resolvido com uma variável nível pela qual

é incrementado quando a busca vai indo mais profunda e quando ela se retrocede ela é decrementado.
E na busca em largura foi declarado dois vetores de tamanho n onde um contém o antecessor dos vértices e outro para armazena o nível de um vértice i, onde o conteúdo de nível vai ser o conteúdo do vetor nível refente ao elemento do anterior +1, exemplo Nível[i] = Nível[Anterior[i]]+1.

## 4. Estudo de Caso

G(m,n) em que m é o numero de aresta e n o numero de vértice, em negrito seria o algoritmo melhor para determinado caso.

| | Mat. Adj. | List. Adj. |
|---|---|---|
| Memória | $O(n^2)$ | **O(m+n)** |
| Buscar todos os vizinhos de $v_i$ | O(n) | **O(n)** |
| Conferir adjacência de $v_i$ e $v_j$ | **O(1)** | O(n) |
| Visitar todas as arestas | $O(n^2)$ | **O(m)** |
| Calcular grau de um vértice | O(n) | **O(n)** |

**Figure 1. Representação das complexidade dos algoritmos em alguns casos.**

A maquin onde foi realizado os teste foi um notebook da ACER Aspire E15 Intel core i5-4210 1.7GHz turbo 2.7GHz, Memoria Ram de 4GB DDR3 L, HD 500GB. Tempo de execução:

Lista de Adjacência :
Busca por Profundidade :
Raiz : 57368 —-0.17s
Raiz : 11744 —-0.17s
Média de tempo : 0.17s

Busca por Largura :
Raiz :57368—-0.17s
Raiz :11744—-0.17s
Média de tempo :0.17s

Matriz de Adjacência :
Busca por Profundidade :
Raiz : 57368—-40.74s
Raiz : 11744——39.71s
Média de tempo : 40.00s
Busca por Largura :
Raiz : 57368—-38.57s
Raiz : 11744—-39.46s

Média de tempo : 39.00s

Memória de execução:

Lista de Adjacência :
Busca por Profundidade :
Raiz : 57368
Raiz : 11744
Média de Memória : 6.9MB

Busca por Largura :
Raiz : 57368
Raiz : 11744
Média de Memória : 7.9MB

Matriz de Adjacência :
Busca por Profundidade :
Raiz : 57368
Raiz : 11744
Média de Memória : 10.67MB

Busca por Largura :
Raiz : 57368
Raiz : 11744
Média de Memória : 10.50

## References


Cormen, T. H. (2009). *Introduction to algorithms*. MIT press.

Goldbarg, M. (2012). *Grafos: Conceitos, algoritmos e aplicações*. Elsevier Brasil.

Jurkiewicz, S. (2009). Grafos–uma introdução. *São Paulo: OBMEP*.