

IFB – Campus Taguatinga  
Curso: ABI – Ciência da Computação  
Turma: A – 2017/02  
Disciplina: Teoria dos Grafos  
Professor: Raimundo Cláudio da Silva Vasconcelos  
Alunos: Rodrigo de Oliveira Freire - 161057600044  
Pedro Vinícius de Castro Boaron - 151057600032  
Felipe de Melo Pinto - 151057600082



## **Relatório – Trabalho 1**

Este relatório descreve a implementação de uma biblioteca destinada a manipular grafos dentro de um programa.

### **Decisões de Projeto**

Para tanto, o grupo fez uso da linguagem JAVA e implementou tal biblioteca na forma de um conjunto de classes, permitindo sua reusabilidade de maneira fácil e prática em quaisquer outras aplicações OO.

### **Implementação de funcionalidades**

A biblioteca foi implementada como a classe de nome **Grafo.java**. Abaixo, segue a descrição de seus atributos e métodos:

```

Grafo
  vert : ArrayList<Vertice>
  arquivo : ArrayList<Integer>
  gp : Grafo
  matriz : boolean[][]
  Grafo()
  searchVerticeRef(String) : Vertice
  addVertice(String) : void
  criaListaVertice(ArrayList<Integer>) : void
  addEdge(String, String) : void
  getNumVertices() : int
  getDegree(String) : int
  getAllAdjacents(String) : ArrayList<String>
  getSources() : ArrayList<String>
  getSinks() : ArrayList<String>
  showInfo() : void
  criaArray(String) : void
  lerArquivo(String) : FileReader
  saidaArquivo() : void
  getGrauVertice(Integer) : int
  diferente(String, String) : int
  maior(String, String) : boolean
  possibilidades(Grafo, String, ArrayList<Integer>) : boolean
  printaSequencia(Vertice) : void
  atribuiTamanhoSequencia() : void
  getVerticeInicial() : Vertice
  recebeVertice() : Integer
  percorrendoAdjacente(Grafo, Queue<Vertice>) : void
  geraMatriz(ArrayList<Integer>) : boolean[][]
  arq : BufferedReader

```

## Estudo de Caso 1

Para fins de demonstração das funcionalidades esperadas pela classe de manipulação de grafos, foi elaborado um programa cuja finalidade é responder algumas questões referentes ao grafo representado pelo arquivo **collaboration\_graph.txt**. Eis as questões e suas respectivas respostas:

- Desempenho em termos de quantidade de memória utilizada das duas representações do grafo (matriz e lista de adjacências):

R: Estes são os números obtidos:

Matriz – aproximadamente 6.23GB de memória RAM

Lista – aproximadamente 6.24GB de memória RAM

- Desempenho em termos de tempo de execução para busca em largura das duas representações do grafo:

R: Estes são os números obtidos:

Matriz – 5 segundos

Lista – 1 minuto e 21 segundos

- Quantidade, maior e menor componentes conexos do grafo:

R: Estes são os números obtidos:

Quantidade de componentes: 7 componentes

Tamanho do maior componente: 4

Tamanho do menor componente: 2

## Estudo de Caso 2

O programa anteriormente elaborado também utiliza a biblioteca de manipulação de grafos para responder algumas questões referentes a outro grafo, contido no arquivo **as\_graph.txt**. Eis as questões e suas respectivas respostas:

- Gráfico explicativo do maior e menor graus do grafo, e comparativo com o maior grau possível de um grafo:

R: Estes são os dados obtidos:

Em comparação com o maior grau possível, conclui-se que .

- Quantidade, maior e menor componentes conexos do grafo:

R: Estes são os números obtidos:

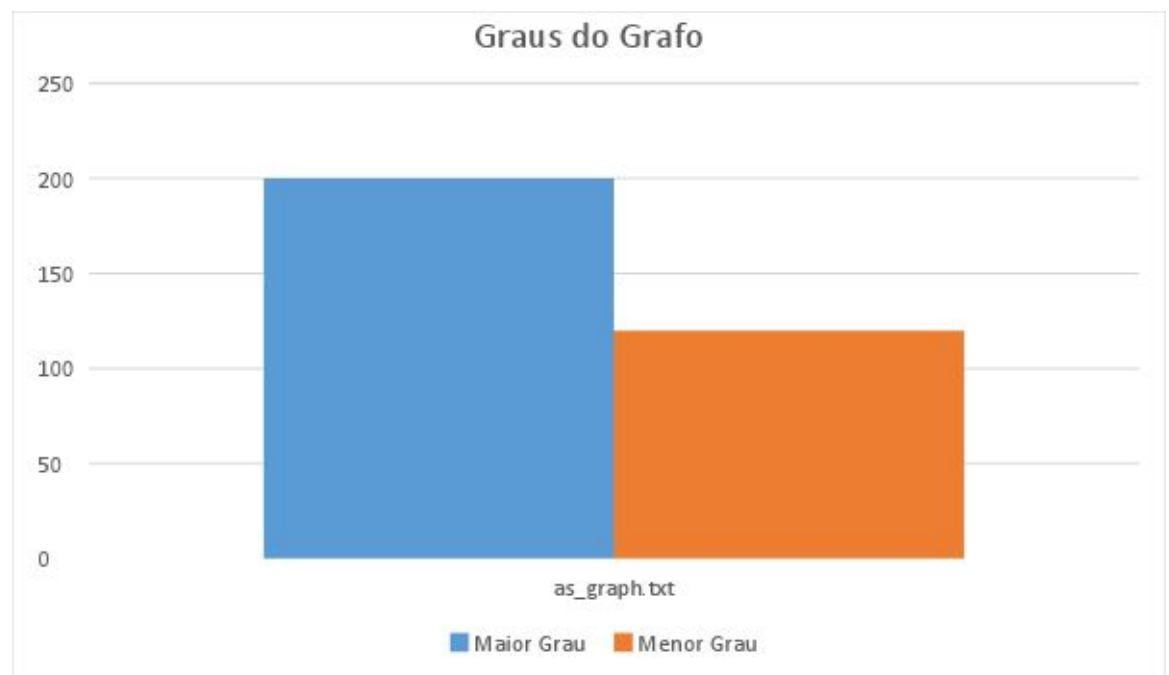
Quantidade de componentes: XXX

Tamanho do maior componente: XXX

Tamanho do menor componente: XXX

- Valor da distância do vértice 1 para qualquer outro:

R: Estes são os números obtidos:



Distância para o vértice XXX: XXX

Distância para o vértice XXX: XXX

Distância para o vértice XXX: XXX

Distância para o vértice XXX: XXX

Conclusão: XXX .

- Diâmetro do grafo (utilizando busca em largura):

R: O diâmetro do grafo é de 4 .