TRABALHO: ESTRUTURA DE DADOS II

Prof. Patrícia Aparecida Proença

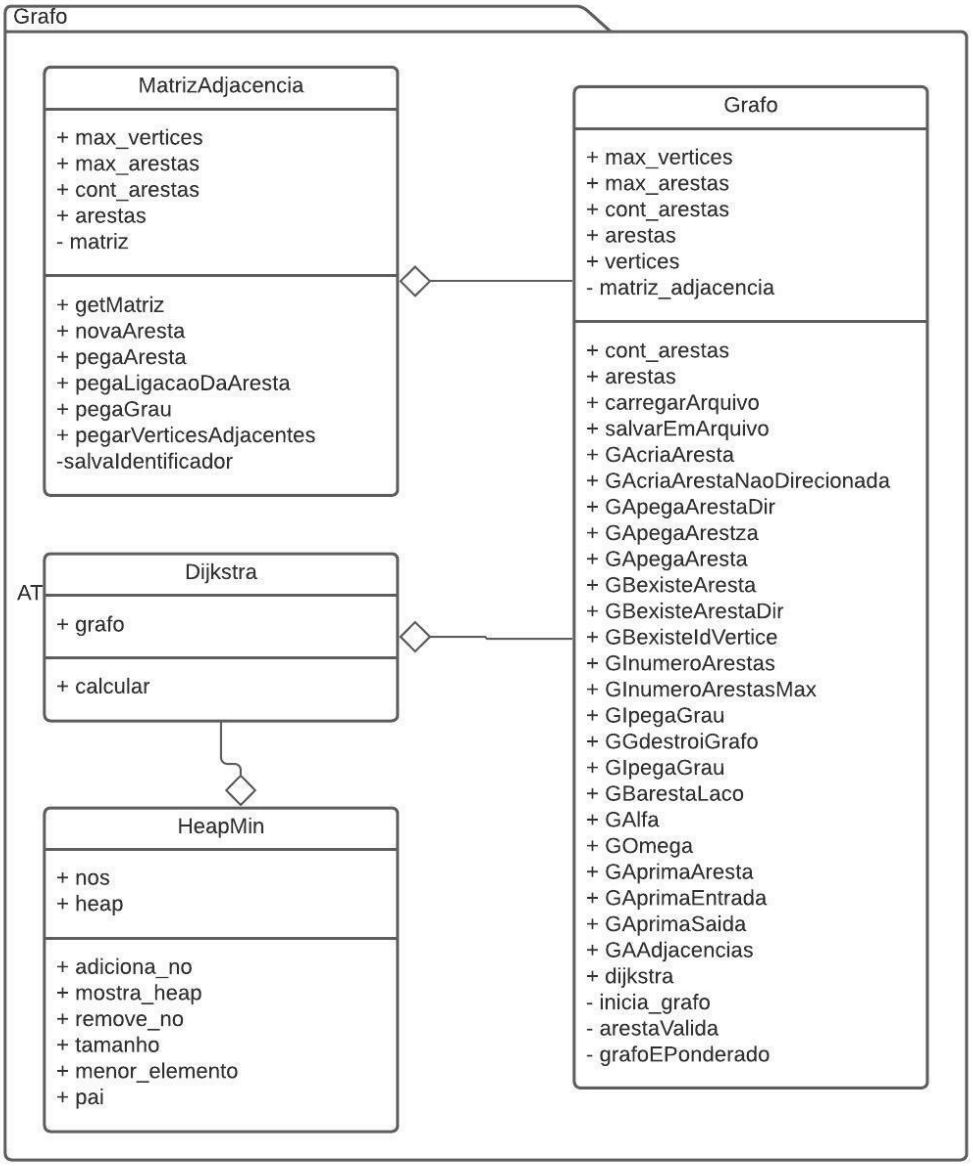
Alunos: Rafael Souza Bernardo - 0041154

Thiago Henrique Domingues - 0041149

1. Introdução

A teoria dos grafos e sua linguagem são usados em muitas áreas da Ciência da Computação, Matemática e Engenharia, uma vez que os gráficos são um bom modelo para muitos problemas fundamentais nessas áreas. Este trabalho tem como objetivo implementar os principais métodos básicos da estrutura de dados grafo.

2. Implementação



Para o processo de implementação foi optado pela linguagem de programação Python. Apesar da linguagem ser multiparadigmas, optamos em desenvolver utilizando a orientação a objetos. A orientação a objetos permite que haja uma reutilização do código criado, diminuindo o tempo de desenvolvimento, bem como o número de linhas de código.

3.1 Matriz de adjacência

Matriz de adjacência é uma matriz 2D de tamanho V x V, onde V é o número de vértices em um grafo, onde cada célula da matriz corresponde a uma aresta. Em um grafo sem pesos uma célula *matrizj[i] [j] = 1* indica que há uma aresta do vértice i ao vértice j.

3.2 Algoritmo de Dijkstra

Para a realização do cálculo do caminho escolhemos o a algoritmo de Dijkstra (E.W. Dijkstra) visto que é um dos mais populares. Seu funcionamento é bastante simples. Escolhido um vértice como raiz da busca, este algoritmo calcula o custo mínimo deste vértice para todos os demais vértices do grafo.

Este algoritmo utiliza uma fila de prioridade para armazenar os vértices vizinhos do vértice atual. Para representação desta fila utilizamos uma estrutura de Min-heap, onde cada pai é menor ou igual que qualquer de seus filhos. ais precisamente, um vetor *A[1..n]* é um min-heap se $A[\lfloor i / 2 \rfloor] \leq A[i]$ para $i = 2, \ldots, n$.

3. Testes

```
❱ python3 src/main.py
GRAFO 02

Lista de arestas:
        1 - (1, 2)
        2 - (1, 4)
        3 - (1, 5)
        4 - (2, 6)
        5 - (2, 7)
        6 - (3, 4)
        7 - (3, 5)
        8 - (3, 8)
        9 - (4, 5)
        10 - (5, 9)
        11 - (6, 10)
        12 - (7, 10)
        13 - (7, 11)
        14 - (8, 11)
        15 - (8, 12)
        16 - (9, 12)
        17 - (10, 11)
        18 - (10, 12)
        19 - (11, 12)

Lista de vertices: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
Qtde máxima de vertices 12
Qtde máxima de arestas 19
-------------------------------------------

Existe vértice com o id 0: False
Existe aresta direcionada (3,4) True
Existe aresta 6: False
Aresta 11 é um laço True
Grau do vertíce 4: 11
```

```
❯ python3 src/main.py
GRAFO 01

[[0 1 0 0 1 0 0]
 [0 0 1 0 1 0 0]
 [0 0 0 1 0 0 0]
 [0 0 0 0 1 1 0]
 [0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0]]

Lista de arestas:
        1 - (1, 2)
        2 - (1, 5)
        3 - (2, 3)
        4 - (2, 5)
        5 - (3, 4)
        6 - (4, 5)
        7 - (4, 6)

Lista de vertices: [1, 2, 3, 4, 5, 6]
Qtde máxima de vertices 7
Qtde máxima de arestas 7
---------------------------------------------

Existe o vértice 10: False
Existe aresta 6: False
Existe aresta direcionada (3,6) False
Aresta 4 é um laço True
Grau do vertíce 3: 2
```

```
❯ python3 src/main.py
GRAFO 03

Lista de arestas:
       1 - (1, 2)
       2 - (1, 7)
       3 - (2, 3)
       4 - (3, 1)
       5 - (3, 4)
       6 - (3, 6)
       7 - (4, 3)
       8 - (4, 5)
       9 - (5, 6)
       10 - (5, 9)
       11 - (7, 6)
       12 - (7, 11)
       13 - (8, 7)
       14 - (8, 10)
       15 - (9, 8)
       16 - (8, 11)
       17 - (11, 7)
       18 - (9, 6)
       19 - (10, 9)
       20 - (10, 5)

Lista de vertices: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Qtde máxima de vertices 11
Qtde máxima de arestas 20
---------------------------------------------

Existe vértice com o id 10: True
Existe aresta direcionada (3,4) True
Existe aresta 6: False
Pega aresta (5,6): 0
Aresta 11 é um laço True
Grau do vertíce 3: 11
Primeira aresta na estrela de entrada do vértice 10: 19
Primeira aresta na estrela de saída do vértice 10: 14
```

4. Conclusão

A implementação do trabalho transcorreu sem maiores problemas e os resultados ficaram dentro do esperado. A principal dificuldade encontrada foi relacionada à nomenclatura de alguns métodos que por serem muito parecidos acabou provocando certa confusão na hora de implementar. Um exemplo seria *GInumeroArestasMax* e *GInumeroArestas*.

5. Referências

MAIDA, João Paulo. **Teoria dos Grafos**: uma abordagem prática em java. São Paulo: Casa do Código, 2020.

JOSHI, Vaidehi. **Finding The Shortest Path, With A Little Help From Dijkstra**. 2017. Disponível em: https://link.medium.com/PHF5defi5hb. Acesso em: 16 jul. 2021.