Centro Federal de Educação Tecnológica de Minas Gerais
CEFET-MG

Laboratório de Algoritmos e Estruturas de Dados II

Prática VI:
Implementação em JAVA do TAD JAGM

Data de entrega: 12/05/2019

Illyana Guimarães de Avelar

2019

•Código da Classe JAGM:

```
package Pratica06;
public class JAGM {
  private int antecessor[];
  private double p[];
  private JGrafo grafo;

  public JAGM (JGrafo grafo){
    this.grafo = grafo;
  }
  public void obterAgm(int raiz)throws Exception{
    int n = this.grafo.numVertices();//quantidade de vertices
    this.p = new double[n];//peso de vertices
    int vs[] = new int[n+1];//vertices
    boolean itensHeap[] = new boolean[n];
    this.antecessor = new int[n];
    for (int u = 0; u < n; u ++){
      this.antecessor[u] = -1;
      p[u] = Double.MAX_VALUE;//infinito
      vs[u+1] = u;//Heap indireto a ser construido
      itensHeap[u] = true;
    }
    p[raiz] = 0;
    JHeap heap = new JHeap (p, vs);//instancia heap
    heap.constroi();//controi heap
    while (!heap.vazio ()){//enquanto houverem elementos
      int u = heap.retiraMin();//retira o menor caminho
      itensHeap[u] = false;
      if (!this.grafo.listaAdjVazia(u)){//se há elementos na lista de adjacência
        JGrafo.Aresta adj = grafo.primeiroListaAdj(u);
        while (adj != null) {
          int v = adj.v2 ();
          if (itensHeap[v] && (adj.peso () < this.peso (v))) {
            antecessor[v] = u; heap.diminuiChave (v, adj.peso ());
          }
          adj = grafo.proxAdj (u);
        }
      }
    }
  }
  public int antecessor(int u){
    return this.antecessor[u];
  }
  public double peso(int u){
    return this.p[u];
  }
  public void imprime(){
    for (int u = 0; u < this.p.length; u++)
      if (this.antecessor[u] != -1)
        System.out.println ("(" +antecessor[u]+ "," +u+ ") -- p:" + peso (u));
  }
  //calcula a soma dos valores das arestas da AGM
  public double total(){
      double total = 0;
```

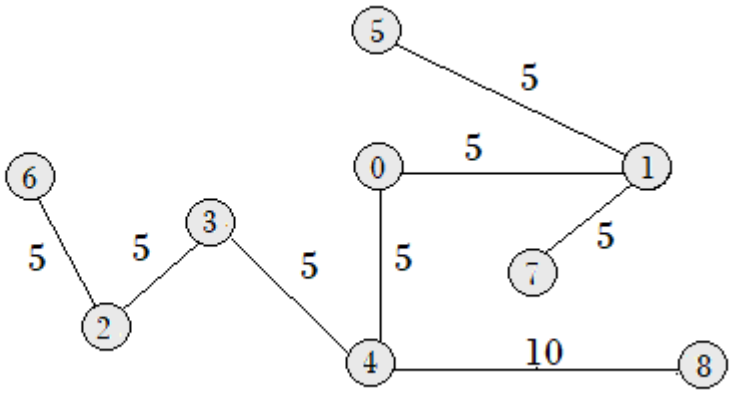
```
    for(double i : this.p)
       if(i!=Double.MAX_VALUE)
          total = total + i;
    return total;
  }
}
```

•Algoritmo Implementado:

Foi utilizado o *Algoritmo de Prim* para encontrar a AGM. Há um conjunto S onde uma aresta segura de peso mínimo é adicionada até que todos os vértices estejam na árvore.
A árvore começa por um vértice qualquer e cresce até que gere todos os vértices em V, e cada passo, uma aresta leve é adicionada à árvore, conectando-a a um vértice que ainda não está em S.
Quando o algoritmo termina, as arestas em S formam uma AGM (Arvore Geradora Mínima). O grafo e a raiz são entradas para o algoritmo; Os vértices residem em uma fila de prioridades mínima Q baseada em um campo que se refere ao peso mínimo das arestas incidentes em um vértice.

•Resultado obtido nos grafos:

```
(0,1) -- p:5.0
(3,2) -- p:5.0
(4,3) -- p:5.0
(0,4) -- p:5.0
(1,5) -- p:5.0
(2,6) -- p:5.0
(1,7) -- p:5.0
(4,8) -- p:10.0
Peso da AGM: 45.0
```




```
(3,1) -- p:2.0
(4,2) -- p:2.0
(2,3) -- p:3.0
(7,4) -- p:2.0
(7,5) -- p:4.0
(8,6) -- p:4.0
(8,7) -- p:3.0
Peso da AGM: 20.0
```