Übung 7: Graphen

Abgabetermin: 07.05.2013

Name: _____ **Matrikelnummer:** _____

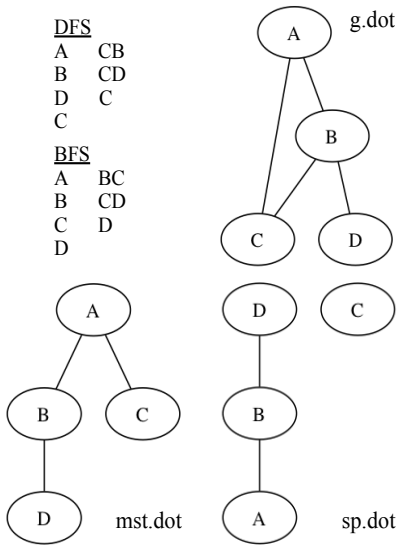
Gruppe: □ **G1** Di 10:15 □ **G2** Di 11:00 □ **G3** Di 12:45

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Punkte
Aufgabe 1	24	□	Java-Programm, Testfälle und Ergebnisse	Java-Programm	□	

Aufgabe 1: DFS, BFS, Minimal Spanning Tree, Shortest Path (24 Punkte)

Gegeben ist ein Graph der aus Knoten (vertex) und Kanten (edge) besteht. Kanten können ungerichtet oder gerichtet sein, und können ein Kantengewicht haben. Implementieren Sie in der Klasse *Graphs* folgende Algorithmen: Depth-First-Search, Breadth-First-Search, Minimal Spanning Tree und Shortest Path. Folgendes Beispiel zeigt, wie der Graph verwendet wird und was ausgegeben werden soll:

```
Graph g = new Graph();
Vertex a = new Vertex('A', true);  g.addVertex(a);
Vertex b = new Vertex('B', false); g.addVertex(b);
Vertex c = new Vertex('C', false); g.addVertex(c);
Vertex d = new Vertex('D', false); g.addVertex(d);

g.addEdge(new Edge(a, b, 0, Edge.Kind.Undirected));
g.addEdge(new Edge(b, c, 0, Edge.Kind.Undirected));
g.addEdge(new Edge(b, d, 0, Edge.Kind.Undirected));
g.addEdge(new Edge(c, a, 0, Edge.Kind.Undirected));

Out.open("g.dot");
Out.println(DotMaker.makeDotForGraph(g));
Out.close();

Graphs.depthFirstSearch(g);
Graphs.breadthFirstSearch(g);

Graphs.makeMinimalSpanningTree(g, a);
Out.open("mst.dot");
Out.println(DotMaker.makeDotForGraph(g));
Out.close();

Graphs.makeShortestPath(g, a, d);
Out.open("sp.dot");
Out.println(DotMaker.makeDotForGraph(g));
Out.close();
```



Verwenden Sie die Vorgabeklassen aus dem Paket *at.jku.ssw* mit folgenden Schnittstellen (alles *public*):

```
class Graph {
  final List vertices;
  List edges;
  Graph() {...}

  void addVertex(Vertex v) {...}
  void addEdge(Edge e) {...}
  Vertex getRoot() {...}

  Edge[] getEdges(Vertex v) { ... }
  Edge getEdge(Vertex start, Vertex end) {...}
}
class VertexPriorityQueue {
  static VertexPriorityQueue
    createMinWeightPriorityQueue() {...}
  static VertexPriorityQueue
    createDistancePriorityQueue() {...}
  void offer(Vertex vertex) {...}
  Vertex poll() {...}
  int size() {...}
  void print() {...}
  boolean contains(Vertex v) {...}
}
```

```
class Vertex {
  final char value;
  final boolean isRoot;
  boolean marked;
  Vertex dad;
  int minWeight = 0;
  int distance = 0;
  Vertex(char value, boolean isRoot) {...}
}
class Edge {
  enum Kind { Directed, Undirected }
  final Vertex start;
  final Vertex end;
  final int weight;
  final Kind kind;
  Edge(Vertex start, Vertex end,
       int weight, Kind kind) {...}
}
```

Implementieren Sie die Klasse *Graphs* mit folgender Schnittstelle:

```
class Graphs {
  public static void depthFirstSearch(Graph graph) {...}
  public static void breadthFirstSearch(Graph graph) {...}
  public static void makeMinimalSpanningTree(Graph graph, Vertex start) {...}
  public static void makeShortestPath(Graph graph, Vertex from, Vertex to) {...}
}
```

Implementierungshinweise:

a) Die Methoden *depthFirstSearch* und *breadthFirstSearch* durchlaufen den Graphen und geben jeden besuchten Knoten und dazu den aktuellen Stack- bzw. Queueinhalt aus (siehe Abbildung auf Seite 1). Verwenden Sie die Klassen *ArrayStack* bzw. *ArrayQueue*, die Sie in Übung 2 implementiert haben.

b) Die Methode *makeMinimalSpanningTree* modifiziert den übergebenen Graphen so, dass nur mehr die Kanten des MST enthalten sind. Erzeugen Sie die benötigte Prioritätswarteschlange mit der Methode *createMinWeightPriorityQueue* aus der Vorgabeklasse *VertexPriorityQueue*.

c) Die Methode *makeShortestPath* modifiziert den übergebenen Graphen so, dass nur mehr die Kanten des kürzesten Pfades enthalten sind. Erzeugen Sie die benötigte Prioritätswarteschlange mit der Methode *createDistancePriorityQueue* aus der Vorgabeklasse *VertexPriorityQueue*.

d) Die Methode *DotMaker.makeDotForGraph(Graph graph)* erzeugt GraphViz-Bilder Ihrer Graphen. DotMaker zeichnet Kanten mit *kind == Edge.Kind.Directed* als gerichtete Kanten und beschriftet Kanten mit *weight != 0* mit dem Kantengewicht.

e) Testen Sie Ihre Implementierung mit der Vorgabedatei *GraphTest.java* und vergleichen Sie Ihre Ergebnisse mit *GraphTest.Output.txt*.

Abzugeben ist: Java-Programm und Testergebnisse