Start with creating the classes Vertex, Edge, WeightedGraph.

The class Vertex has the following private fields:

int id;

double key;

Vertex predecessor;

ArrayList <Vertex> adj;

ArrayList <Double> weights.

The reason for the last field: the Prim’s algorithm (pseudocode) contains the line

|  |
| --- |
|  |

There is no place in the vertex to hold the edges starting from this vertex; searching in the list of edges is inefficient. That’s why you will grow two lists (adjacent vertices and weights of corresponding edges) simultaneously.

Apparently you will need constructor(s), accessors, mutators: getKey, setKey, attachVertex, attachWeight, setPredecessor etc. Also, toString.

You need also the method compareTo which compares vertex keys. It is needed for proper use of PriorityQueue.

The class Edge has the obvious private fields:

Vertex v1;

Vertex v2;

double weight;

Constructor(s), accessors, toString.

The class WeightedGraph:

private Vertex[] vertices;

private ArrayList <Edge> edges; // it may be easier to handle than just an array

constructor(s), accessors, mutators.

The method print;

important: fillFromFile. This method may be defined either in this class, or in a separate file.

It reads the file showing weight lists and initializes the graph.

If you want, you can create a constructor

public WeightedGraph (String filename) { … }

When all this job is done and you can create a graph and print it, you can start implementing Prim’s algorithm, following the pseudocode in the slide 8 of Lec12bPrim.

The method may be

Edge [] prim (Graph g, int rootIndex) // for choosing a vertex which serves as root).

Use the Java class PriorityQueue from java.util.

After you update v.key, you should call Q.remove(v); Q,add(V); to make sure the queue is proper.

As the last line, you allocate an array of edges with the size vertices.length – 1, and fill it with new edges (v, v.p)