Caleb Song HW 5 Algorithms

Documentation: EI with Dr. Hadfield 3/11 and 3/8

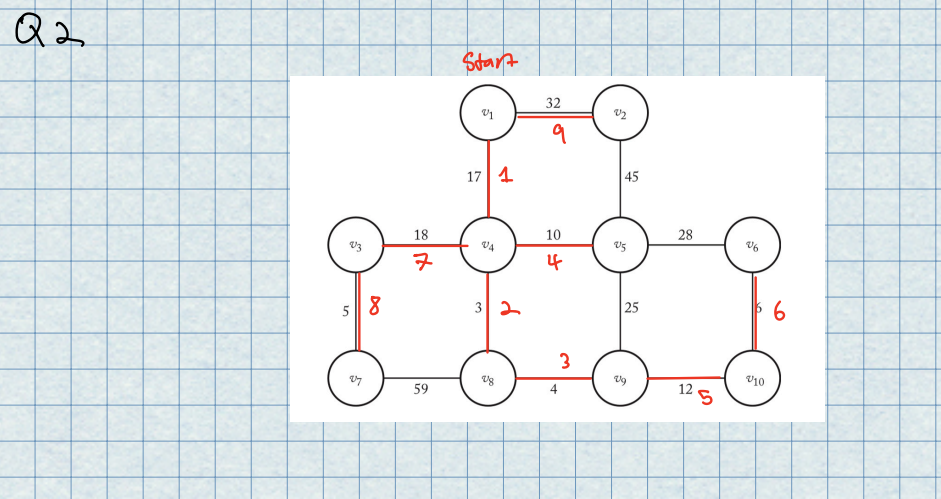
1. Prims

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* prim() - Implementation of Prim's algorithm for computing  
 \* the minimum cost spanning tree.  
 \*  
 \* Note:  
 \* - EdgeSet is a subclass of LinkedList<Edge> and thus has an  
 \* add(Edge) method. The Edge class has a three-parameter  
 \* constructor: Edge(node1,node2,weight)  
 \*  
 \* - Algorithm 4.1 in the N&N text uses 1-based indexing for arrays  
 \* so you'll need to adapt the algorithm for the 0-based indexing  
 \* of Java  
 \*  
 \* @param numNodes - number of nodes in the graph  
 \* @param weights - adjacency matrix with weights for the graph  
 \* @return - Edge set with edges for the MCST  
 \*/*private static EdgeSet prim( int numNodes, int[][] weights ) {  
  
 EdgeSet mcstEdges = new EdgeSet();  
 int min;  
 int vnear = 0;  
 int[] nearest = new int[numNodes];  
 int[] distance = new int[numNodes];  
 //initialize all values in array to have 0 as nearest node and have weight be distance of edge to v1  
 for (int i =1; i< numNodes; i++){  
 nearest[i] = 0;  
 distance[i] = weights[0][i];  
 }  
 for (int i = 0; i < numNodes-1; i++){  
 min = *INF*;  
 //find minimum distance node  
 for (int j = 1; j < numNodes; j++){  
 if (distance[j] >= 0 && distance[j] < min){  
 min = distance[j];  
 vnear = j;  
 }  
 }  
 //add edge of minimum distance that is connected to current nodes to mcst  
 mcstEdges.addEdge(vnear,nearest[vnear], distance[vnear]);  
 //set distance to -1 as it was already travelled to  
 distance[vnear] = -1;  
 //go through a reintialize nearest based on the new node added  
 for (int k = 1; k < numNodes; k++){  
 if (weights[k][vnear] < distance[k]){  
 distance[k] = weights[k][vnear];  
 nearest[k] = vnear;  
 }  
 }  
 }  
  
 return mcstEdges; // return the MCST edges  
  
} // end prim() method

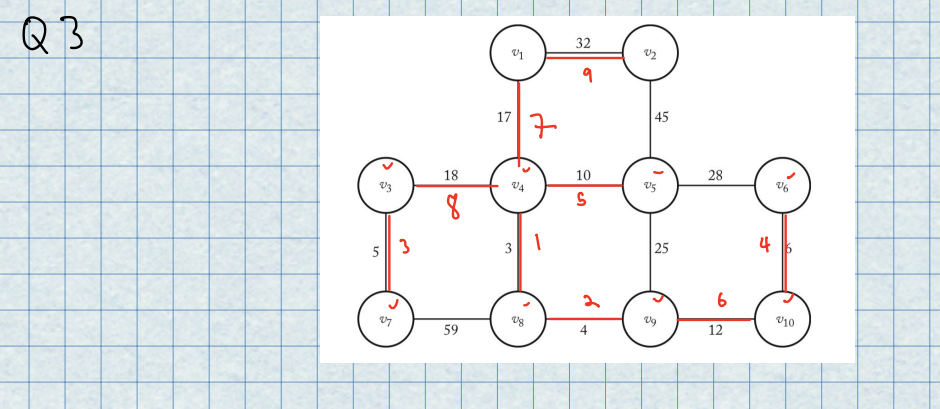
Kruskal

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* kruskal() - Implementation of Kruskall's algorithm for computing  
 \* the minimum cost spanning tree.  
 \*  
 \* Notes:  
 \* - Helper methods: initial(), find(), and merge() from N&N  
 \* Algorithm 4.2 are provided below this. They work on an int  
 \* array mapping a node (via the array index) to a subset number  
 \* (as the array value - subset[i] = j means node i is in subset j).  
 \*  
 \* - EdgeSet has a sort() method which will come in handy.  
 \*  
 \* - You can setup and use an iterator to traverse through an edge  
 \* set as EdgeSet is a subclass of LinkedList<Edge>.  
 \*  
 \* @param numNodes - number of nodes (vertices) in the graph  
 \* @param numEdges - number of edges in the graph  
 \* @param edges - edge set for all edges in the graph  
 \* @return - edge set with just edges for the MCST  
 \*/*private static EdgeSet kruskal( int numNodes, int numEdges,  
 EdgeSet edges ) {  
 //initalize mcst to return  
 EdgeSet mcst = new EdgeSet();  
 //sort edges in increasing order  
 edges.sort();  
 int node1;  
 int node2;  
 Edge edge;  
 //create a subset for nodes  
 int[] subsets = *initial*(numNodes);  
 int subsetFor1;  
 int subsetFor2;  
 int i = 0;  
 while (mcst.size() < numNodes-1){  
 //get lowest weight edge  
 edge = edges.get(i);  
 //get nodes that connect them  
 node1 = edge.getNode1();  
 node2 = edge.getNode2();  
 //find what subsets those nodes are in  
 subsetFor1 = *find*(subsets,node1);  
 subsetFor2 = *find*(subsets,node2);  
 //if they are not in the same subset merge the subset and then add the edge to the mcst  
 if (subsetFor1 != subsetFor2){  
 *merge*(subsets, subsets[node1],subsets[node2]);  
 mcst.addEdge(node1,node2,edge.getWeight());  
 }  
 //increase to go to next edge  
 i++;  
 }  
  
 return mcst; // return the resulting MCST  
  
} // end kruskal() method

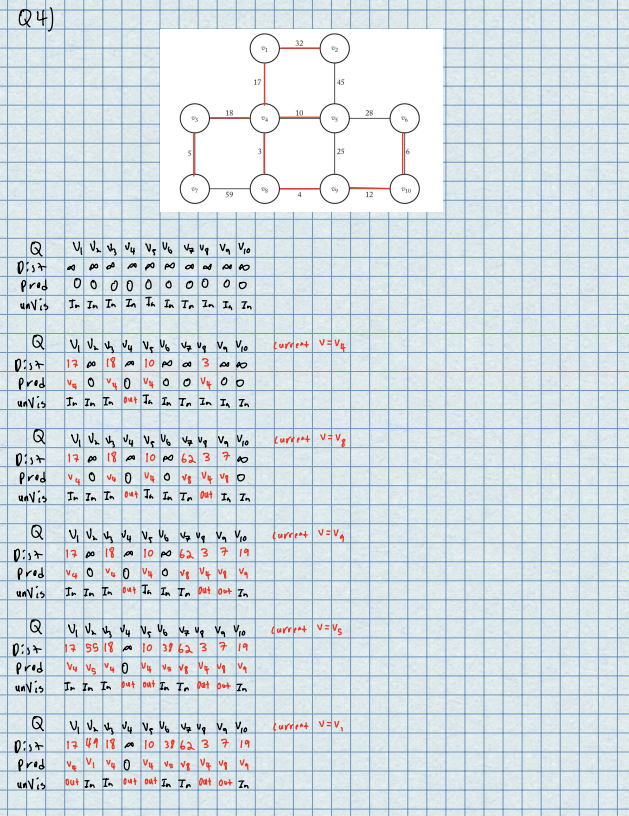
2)

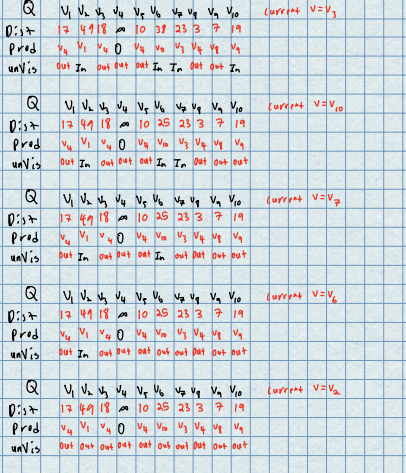


3)

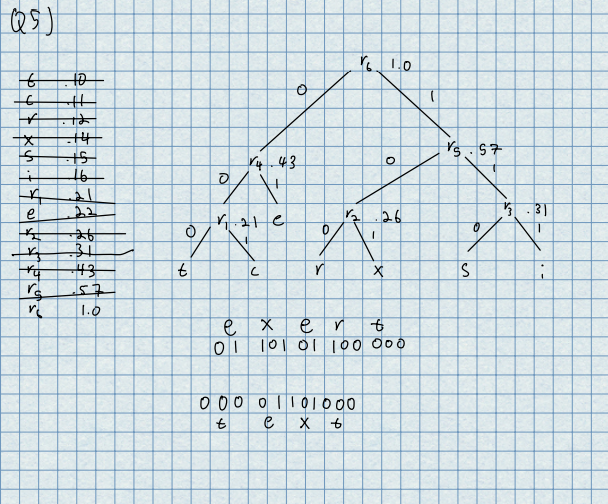


4)

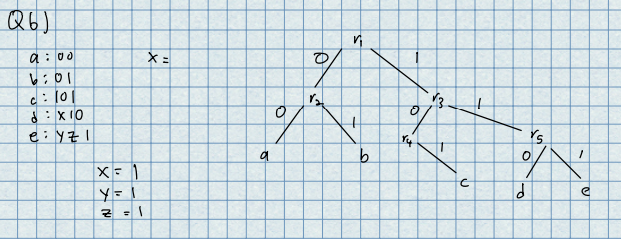




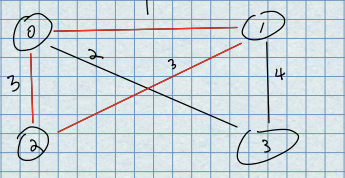
5)



6)



7)

 Example that doesn’t work

private static EdgeSet greedy( int numNodes, int[][] weights ){  
 EdgeSet mcst = new EdgeSet();  
 boolean[] visited = new boolean[numNodes];  
 //initialize all visited array to false  
 for (int i = 0; i < numNodes; i++){  
 visited[i]= false;  
 }  
 //start at 0 node  
 int sourceNode = 0;  
 int nextNode = 0;  
 int minEdge;  
 while (mcst.size() < numNodes-1){  
 minEdge = *INF*;  
 //go nodes edges and find min cost edge and node that connects to it  
 for (int i = 0; i <numNodes-1; i++){  
 if (weights[sourceNode][i] < minEdge && sourceNode != i && visited[i] == false){  
 minEdge = weights[sourceNode][i];  
 nextNode = i;  
 }  
 }  
 //set the node you are going to true  
 visited[nextNode] = true;  
 //set weights of where you just came from false  
 weights[sourceNode][nextNode] = *INF*;  
 weights[nextNode][sourceNode] = *INF*;  
 //add edge to MCST  
 mcst.addEdge(sourceNode,nextNode,minEdge);  
 //set next node to source node and run through loop  
 sourceNode = nextNode;  
  
 }  
 return mcst;  
}