323.25 Project 5: Kruskal MST

Student: Jingshi Liu

Due Date: 3/27/2022

Algorithm Steps for the Project:

Step 0: inFile, outFile1, outFile2 <— open via args[] as given in the above
numNodes <— inFile
numSets <— numNodes
whichSet <— allocate space, size of numNodes + 1, set whichSet[i] to i, i from 1 to numNodes+1 edgeHead <—get a dummy edge <0,0,0, null>
mstHead <— get a dummy edge<0,0,0, null>
totalMSTCost <— 0

Step 1: <Ni, Nj, cost> <—read from inFile newEdge <—get a new edge (Ni, Nj, cost)

Step 2: insert (newEdge, edgeHead)
Step 3: printList (edgeHead, outFile2)
Step 4: repeat step 1 to step 3 while inFile is not empty
Step 5: nextEdge <—removeEdge (edgeHead)
Step 6: repeat Step 5 while whichSet [nextEdge.Ni] == whichSet [nextEdge.Nj] // Ni and Nj cannot be in the same set

Step 7: push (nextEdge, mstHead)

totalMSTCost += nextEdge.cost

merge2Sets (nextEdge,Ni, nextEdge.Nj)

numSets --

Step 8: printAry(whichSet)
Step 9: printList (edgeHead, outFile2) // with caption indicating which edge list you are printing.

printList (mstHead, outFile2) // with caption indicating which edge list you are printing.
Step 10: repeat step 5 – step 9 while numSets > 1
Step 11: printList (mstHead, outFile1)
Step 12: close all files.

Source Code:

```
package com.company;

import java.io.*;
import java.util.Scanner;



class Edge{
    int Ni, Nj, cost;
    Edge next;

    Edge(int n1, int n2, int cost){
        Ni = n1;
        Nj = n2;
        this.cost = cost;
        next = null;
    }

    String printEdge(){
        return "<" + Ni + ", " + Nj + ", " + cost + "> -> ";
    }
}



class KruskalMST{
    int num_nodes;
    int[] which_set;
    int num_sets;
    int total_MST_cost;
    Edge edge_head = new Edge(0,0,0);
```

```
Edge mst_head = new Edge(0,0,0);

KruskalMST(int numNodes){
    num_nodes = numNodes;
    which_set =  new int[num_nodes+1];
    for (int i = 0; i < which_set.length; i++) {
        which_set[i] = i;
    }
    num_sets = num_nodes;
}

void insert(Edge new_edge){
    Edge spot = edge_head;

    while(spot.next != null && spot.next.cost < new_edge.cost){
        spot = spot.next;
    }

    new_edge.next = spot.next;
    spot.next = new_edge;
}

Edge removeEdge(){
    if(edge_head.next == null)
        return null;

    Edge res = edge_head.next;
    edge_head.next = edge_head.next.next;
    res.next = null;

    return res;
```

```
}

void mergeSets(int Ni, int Nj){
    int set_of_Ni = which_set[Ni], set_of_Nj = which_set[Nj];

    if( set_of_Ni > set_of_Nj)
        for (int i = 0; i < which_set.length; i++) {
            if(which_set[i] == set_of_Ni)
                which_set[i] = set_of_Nj;
        }
    else
        for (int i = 0; i < which_set.length; i++) {
            if(which_set[i] == set_of_Nj)
                which_set[i] = set_of_Ni;
        }
}

void push(Edge nextEdge){
    nextEdge.next = mst_head.next;
    mst_head.next = nextEdge;
}

void printArray(BufferedWriter outFile) throws IOException {
    String res = "Which Set: ";

    for(int i: which_set)
        res += i + " ";
    outFile.write(res + "\n");
}
```

```
    void printList(BufferedWriter outFile, Edge head) throws
IOException {
        Edge pointer;
        String res = "";

        if(head == edge_head){
            pointer = edge_head;
            res += "Edge List: ";
        }else {
            pointer = mst_head;
            res += "MST List: ";
        }

        while(pointer != null){
            res += pointer.printEdge();
            pointer = pointer.next;
        }

        outFile.write(res + " null\n");
    }

}

public class Main {

    public static void main(String[] args) throws IOException {

         Scanner input = new Scanner(new File(args[0]));

        BufferedWriter outFile1 = new BufferedWriter(new
FileWriter(args[1], true));
```

```
        BufferedWriter outFile2 = new BufferedWriter(new
FileWriter(args[2], true));




        int num_nodes = input.nextInt();
        KruskalMST kruskalMST = new KruskalMST(num_nodes);


        outFile2.write("Reading:\n");


        int n1, n2, cost;
        try{
            while(input.hasNextLine()){
                n1 = input.nextInt();
                n2 = input.nextInt();
                cost = input.nextInt();


                kruskalMST.insert(new Edge(n1, n2, cost));
                kruskalMST.printList(outFile2, kruskalMST.edge_head);
            }
        }catch (Exception ignored){
        }



        outFile2.write("\n\nConstructing Minimum Spanning Tree:\n");
        Edge nextEdge;
        while(kruskalMST.num_sets > 1){
            do{
                nextEdge = kruskalMST.removeEdge();
            }while(nextEdge != null &&
kruskalMST.which_set[nextEdge.Ni] ==
kruskalMST.which_set[nextEdge.Nj]);
```

```
            if(nextEdge == null)
                break;

            kruskalMST.push(nextEdge);
            kruskalMST.total_MST_cost += nextEdge.cost;
            kruskalMST.mergeSets(nextEdge.Ni, nextEdge.Nj);
            kruskalMST.num_sets--;

            kruskalMST.printArray(outFile2);
            kruskalMST.printList(outFile2, kruskalMST.edge_head);
            kruskalMST.printList(outFile2, kruskalMST.mst_head);
            outFile2.write("\n");
        }

        kruskalMST.printList(outFile1, kruskalMST.mst_head);
        outFile1.write("Total cost is: " + kruskalMST.total_MST_cost);

       outFile1.close();
        outFile2.close();
    }
}
```

Output:

Data1:

6
1 2 6
1 3 1
1 4 3
2 3 5
3 4 2
2 5 4
3 5 6
3 6 3
4 6 2
5 6 6

OutFile1 for Data 1:

MST List: <0, 0, 0> —> <2, 3, 5> —> <2, 5, 4> —> <3, 4, 2> —> <4, 6, 2> —> <1, 3, 1> —> null
Total cost is: 14

OutFile2 (Debugging) for Data 2:

Reading:
Edge List: <0, 0, 0> —> <1, 2, 6> —> null
Edge List: <0, 0, 0> —> <1, 3, 1> —> <1, 2, 6> —> null
Edge List: <0, 0, 0> —> <1, 3, 1> —> <1, 4, 3> —> <1, 2, 6> —> null
Edge List: <0, 0, 0> —> <1, 3, 1> —> <1, 4, 3> —> <2, 3, 5> —> <1, 2, 6> —> null
Edge List: <0, 0, 0> —> <1, 3, 1> —> <3, 4, 2> —> <1, 4, 3> —> <2, 3, 5> —> <1, 2, 6> —> null
Edge List: <0, 0, 0> —> <1, 3, 1> —> <3, 4, 2> —> <1, 4, 3> —> <2, 5, 4> —> <2, 3, 5> —> <1, 2, 6> —> null
Edge List: <0, 0, 0> —> <1, 3, 1> —> <3, 4, 2> —> <1, 4, 3> —> <2, 5, 4> —> <2, 3, 5> —> <3, 5, 6> —> <1, 2, 6> —> null
Edge List: <0, 0, 0> —> <1, 3, 1> —> <3, 4, 2> —> <3, 6, 3> —> <1, 4, 3> —> <2, 5, 4> —> <2, 3, 5> —> <3, 5, 6> —> <1, 2, 6> —> null
Edge List: <0, 0, 0> —> <1, 3, 1> —> <4, 6, 2> —> <3, 4, 2> —> <3, 6, 3> —> <1, 4, 3> —> <2, 5, 4> —> <2, 3, 5> —> <3, 5, 6> —> <1, 2, 6> —> null
Edge List: <0, 0, 0> —> <1, 3, 1> —> <4, 6, 2> —> <3, 4, 2> —> <3, 6, 3> —> <1, 4, 3> —> <2, 5, 4> —> <2, 3, 5> —> <5, 6, 6> —> <3, 5, 6> —> <1, 2, 6> —> null

Constructing Minimum Spanning Tree:
Which Set: 0 1 2 1 4 5 6
Edge List: <0, 0, 0> —> <4, 6, 2> —> <3, 4, 2> —> <3, 6, 3> —> <1, 4, 3> —> <2, 5, 4> —> <2, 3, 5> —> <5, 6, 6> —> <3, 5, 6> —> <1, 2, 6> —> null
MST List: <0, 0, 0> —> <1, 3, 1> —> null

Which Set: 0 1 2 1 4 5 4
Edge List: <0, 0, 0> —> <3, 4, 2> —> <3, 6, 3> —> <1, 4, 3> —> <2, 5, 4> —> <2, 3, 5> —> <5, 6, 6> —> <3, 5, 6> —> <1, 2, 6> —> null
MST List: <0, 0, 0> —> <4, 6, 2> —> <1, 3, 1> —> null

Which Set: 0 1 2 1 1 5 1
Edge List: <0, 0, 0> —> <3, 6, 3> —> <1, 4, 3> —> <2, 5, 4> —> <2, 3, 5> —> <5, 6, 6> —> <3, 5, 6> —> <1, 2, 6> —> null
MST List: <0, 0, 0> —> <3, 4, 2> —> <4, 6, 2> —> <1, 3, 1> —> null

Which Set: 0 1 2 1 1 2 1
Edge List: <0, 0, 0> —> <2, 3, 5> —> <5, 6, 6> —> <3, 5, 6> —> <1, 2, 6> —> null
MST List: <0, 0, 0> —> <2, 5, 4> —> <3, 4, 2> —> <4, 6, 2> —> <1, 3, 1> —> null

Which Set: 0 1 1 1 1 1 1
Edge List: <0, 0, 0> —> <5, 6, 6> —> <3, 5, 6> —> <1, 2, 6> —> null
MST List: <0, 0, 0> —> <2, 3, 5> —> <2, 5, 4> —> <3, 4, 2> —> <4, 6, 2> —> <1, 3, 1> —> null

Data 2:

12
6 4 3
12 7 4
6 12 7
10 12 7
9 10 4
2 4 1
9 11 5
3 2 5
5 7 5
1 6 3
8 6 2
9 8 2
8 10 1
5 4 2
4 3 3
1 2 6
1 11 6
3 5 4
6 7 2

OutFile1 for Data 2:

MST List: <0, 0, 0> —> <9, 11, 5> —> <12, 7, 4> —> <6, 4, 3> —> <1, 6, 3> —> <4, 3, 3> —> <8, 6, 2> —> <9, 8, 2> —> <5, 4, 2> —> <6, 7, 2> —> <2, 4, 1> —> <8, 10, 1> —> null
Total cost is: 28

OutFile 2 for Data 2:
Reading:
Edge List: <0, 0, 0> —> <6, 4, 3> —> null
Edge List: <0, 0, 0> —> <6, 4, 3> —> <12, 7, 4> —> null
Edge List: <0, 0, 0> —> <6, 4, 3> —> <12, 7, 4> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <6, 4, 3> —> <12, 7, 4> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <2, 4, 1> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <2, 4, 1> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <9, 11, 5> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <2, 4, 1> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <3, 2, 5> —> <9, 11, 5> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <2, 4, 1> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <2, 4, 1> —> <1, 6, 3> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <2, 4, 1> —> <8, 6, 2> —> <1, 6, 3> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <2, 4, 1> —> <9, 8, 2> —> <8, 6, 2> —> <1, 6, 3> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <8, 10, 1> —> <2, 4, 1> —> <9, 8, 2> —> <8, 6, 2> —> <1, 6, 3> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <8, 10, 1> —> <2, 4, 1> —> <5, 4, 2> —> <9, 8, 2> —> <8, 6, 2> —> <1, 6, 3> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <8, 10, 1> —> <2, 4, 1> —> <5, 4, 2> —> <9, 8, 2> —> <8, 6, 2> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <8, 10, 1> —> <2, 4, 1> —> <5, 4, 2> —> <9, 8, 2> —> <8, 6, 2> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <8, 10, 1> —> <2, 4, 1> —> <5, 4, 2> —> <9, 8, 2> —> <8, 6, 2> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <8, 10, 1> —> <2, 4, 1> —> <5, 4, 2> —> <9, 8, 2> —> <8, 6, 2> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
Edge List: <0, 0, 0> —> <8, 10, 1> —> <2, 4, 1> —> <6, 7, 2> —> <5, 4, 2> —> <9, 8, 2> —> <8, 6, 2> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4>

—> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null

Constructing Minimum Spanning Tree:
Which Set: 0 1 2 3 4 5 6 7 8 9 8 11 12
Edge List: <0, 0, 0> —> <2, 4, 1> —> <6, 7, 2> —> <5, 4, 2> —> <9, 8, 2> —> <8, 6, 2> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
MST List: <0, 0, 0> —> <8, 10, 1> —> null

Which Set: 0 1 2 3 2 5 6 7 8 9 8 11 12
Edge List: <0, 0, 0> —> <6, 7, 2> —> <5, 4, 2> —> <9, 8, 2> —> <8, 6, 2> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
MST List: <0, 0, 0> —> <2, 4, 1> —> <8, 10, 1> —> null

Which Set: 0 1 2 3 2 5 6 6 8 9 8 11 12
Edge List: <0, 0, 0> —> <5, 4, 2> —> <9, 8, 2> —> <8, 6, 2> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
MST List: <0, 0, 0> —> <6, 7, 2> —> <2, 4, 1> —> <8, 10, 1> —> null

Which Set: 0 1 2 3 2 2 6 6 8 9 8 11 12
Edge List: <0, 0, 0> —> <9, 8, 2> —> <8, 6, 2> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
MST List: <0, 0, 0> —> <5, 4, 2> —> <6, 7, 2> —> <2, 4, 1> —> <8, 10, 1> —> null

Which Set: 0 1 2 3 2 2 6 6 8 8 8 11 12
Edge List: <0, 0, 0> —> <8, 6, 2> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
MST List: <0, 0, 0> —> <9, 8, 2> —> <5, 4, 2> —> <6, 7, 2> —> <2, 4, 1> —> <8, 10, 1> —> null

Which Set: 0 1 2 3 2 2 6 6 6 6 6 11 12
Edge List: <0, 0, 0> —> <4, 3, 3> —> <1, 6, 3> —> <6, 4, 3> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
MST List: <0, 0, 0> —> <8, 6, 2> —> <9, 8, 2> —> <5, 4, 2> —> <6, 7, 2> —> <2, 4, 1> —> <8, 10, 1> —> null

Which Set: 0 1 2 2 2 2 6 6 6 6 6 11 12
Edge List: <0, 0, 0> —> <1, 6, 3> —> <6, 4, 3> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
MST List: <0, 0, 0> —> <4, 3, 3> —> <8, 6, 2> —> <9, 8, 2> —> <5, 4, 2> —> <6, 7, 2> —> <2, 4, 1> —> <8, 10, 1> —> null

Which Set: 0 1 2 2 2 2 1 1 1 1 1 11 12
Edge List: <0, 0, 0> —> <6, 4, 3> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null

MST List: <0, 0, 0> —> <1, 6, 3> —> <4, 3, 3> —> <8, 6, 2> —> <9, 8, 2> —> <5, 4, 2> —> <6, 7, 2> —> <2, 4, 1> —> <8, 10, 1> —> null

Which Set: 0 1 1 1 1 1 1 1 1 1 1 11 12
Edge List: <0, 0, 0> —> <3, 5, 4> —> <9, 10, 4> —> <12, 7, 4> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
MST List: <0, 0, 0> —> <6, 4, 3> —> <1, 6, 3> —> <4, 3, 3> —> <8, 6, 2> —> <9, 8, 2> —> <5, 4, 2> —> <6, 7, 2> —> <2, 4, 1> —> <8, 10, 1> —> null

Which Set: 0 1 1 1 1 1 1 1 1 1 1 11 1
Edge List: <0, 0, 0> —> <5, 7, 5> —> <3, 2, 5> —> <9, 11, 5> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
MST List: <0, 0, 0> —> <12, 7, 4> —> <6, 4, 3> —> <1, 6, 3> —> <4, 3, 3> —> <8, 6, 2> —> <9, 8, 2> —> <5, 4, 2> —> <6, 7, 2> —> <2, 4, 1> —> <8, 10, 1> —> null

Which Set: 0 1 1 1 1 1 1 1 1 1 1 1 1
Edge List: <0, 0, 0> —> <1, 11, 6> —> <1, 2, 6> —> <10, 12, 7> —> <6, 12, 7> —> null
MST List: <0, 0, 0> —> <9, 11, 5> —> <12, 7, 4> —> <6, 4, 3> —> <1, 6, 3> —> <4, 3, 3> —> <8, 6, 2> —> <9, 8, 2> —> <5, 4, 2> —> <6, 7, 2> —> <2, 4, 1> —> <8, 10, 1> —> null