**CS323-22: Project 7B (JAVA)**

**Yida Tao**

**Due date: Nov. 30, 2017**

Algorithm Steps:

Step 0: numNodes 🡨 get from input file

Allocate and initialize all the members in PrimMST class accordingly

Step 1: <Ni, Nj, edgeCost> 🡨 read from input file

newEdge 🡨 create an undirectedEdge and fill with <Ni, Nj, cost>

insertEdge ( newEdge, edgeListHead) // Using insertion sort to insert newEdge into the linked list of edgeListHead

Step 2: printList (edgeListHead) //See the output format in the above

Step 3: repeat step 1 to step 2 until the input file is empty

Step 4: nextEdge 🡨 removedEdge (edgeListHead)

If Ni and Nj are in the same set // check inWhichSet(Ni) and inWhichSet(Nj)

then discards nextEdge

Step 5: repeat step 4 until Ni and Nj are in different sets

Step 6: pushEdge(nextEdge, MSTofG) // push nextEdge in the front of MSTofG

totalMSTCost += the cost of nextEdge

if Ni is in setA, move2SetA(Nj, setA) // move Nj from setB to setA

else move2SetA(Ni, setA) // move Ni from setB to setA

printSet(inWhichSet) // print the inWhichSet array to argv[3]

Step 7: printList(MSTofG) // print up to the first 10 edges of MSTofG) to argv[2] file

Step 8: repeat step 4 – step 7 until setB is empty. // inWhichSet are all 1

Step 9: output the entire MSTofG and the totalCost to outfile1, argv[1], with proper heading, one edge with cost per text line.

Step 10: close all files.

**Source Code**

**public** **class** graphNode {

**int** nodeID;

graphNode next;

graphNode(**int** i){

nodeID = i;

next = **null**;

}

}

**public** **class** undirectedEdge {

**int** Ni;

**int** Nj;

**int** edgeCost;

undirectedEdge next;

undirectedEdge(){

Ni = 0;

Nj = 0;

edgeCost = 0;

}

undirectedEdge(**int** i, **int** j, **int** cost){

Ni = i;

Nj = j;

edgeCost = cost;

next = **null**;

}

**void** printEdge(){

System.***out***.print("<" + Ni + "," + Nj + "," + edgeCost + ">");

}

}

**public** **class** PrimMST {

**int** numNodes;

**int** inWhichSet[];

undirectedEdge edgeListHead;

undirectedEdge MSTofG;

**int** totalMSTCost;

PrimMST(**int** num){

numNodes = num;

inWhichSet = **new** **int**[numNodes + 1];

edgeListHead = **new** undirectedEdge();

MSTofG = **new** undirectedEdge();

totalMSTCost = 0;

inWhichSet[1] = 1;

**for**(**int** i = 2; i < numNodes + 1; i++){

inWhichSet[i] = 2;

}

}

**void** insertEdge(undirectedEdge newedge){

undirectedEdge walker = edgeListHead;

**while**(walker.next != **null** && walker.next.edgeCost < newedge.edgeCost){

walker = walker.next;

}

newedge.next = walker.next;

walker.next = newedge;

}

undirectedEdge removeEdge(undirectedEdge head){

undirectedEdge walker = head;

undirectedEdge r;

**while**(walker.next != **null** && inWhichSet[walker.next.Ni] == inWhichSet[walker.next.Nj]){

walker = walker.next;

}

**if**(walker.next == **null**){

**throw** **new** NullPointerException("no vaild edge in the graph");

}

r = walker.next;

walker.next = walker.next.next;

**return** r;

}

**void** pushEdge(undirectedEdge newedge){

undirectedEdge walker = MSTofG;

**while**(walker.next != **null**){

walker = walker.next;

}

newedge.next = walker.next;

walker.next = newedge;

}

**void** move2SetA(**int** i){

inWhichSet[i] = 1;

}

**void** printSet(){

System.***out***.println("inWhichSet array:");

**for**(**int** i = 1; i < numNodes + 1; i++){

System.***out***.print(inWhichSet[i] + " ");

}

System.***out***.println();

}

**void** printList(undirectedEdge newedge){

undirectedEdge walker = newedge;

**int** counter = 0;

System.***out***.print("listHead --> " );

walker.printEdge();

**while**(walker.next != **null** && counter++ < 10){

walker = walker.next;

System.***out***.print(" --> ");

walker.printEdge();

}

System.***out***.println();

}

**boolean** isSetBEmpty(){

**boolean** b = **true**;

**for**(**int** i = 2; i < numNodes + 1; i++){

**if**(inWhichSet[i] == 2){

**return** **false**;

}

}

**return** b;

}

}

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** project7b {

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

FileOutputStream fos1 = **new** FileOutputStream(args[1]);

FileOutputStream fos2 = **new** FileOutputStream(args[2]);

Scanner scan = **null**;

PrimMST pmst = **null**;

undirectedEdge newedge = **null**;

**int** x = 0;

**int** y = 0;

**int** c = 0;

**int** counter = 0;

//output to output1

System.*setOut*(**new** PrintStream(fos1));

**try** {

scan = **new** Scanner(**new** File(args[0]));

//step 0

x = scan.nextInt();

pmst = **new** PrimMST(x);

System.***out***.println("\*\*\* The Prim’s MST of the input graph is given below: \*\*\*");

System.***out***.println(x);

//output to output2

System.*setOut*(**new** PrintStream(fos2));

pmst.printList(pmst.edgeListHead);

//step 1,2,3

**while**(scan.hasNextInt()){

//output to output1

System.*setOut*(**new** PrintStream(fos1));

x = scan.nextInt();

y = scan.nextInt();

c = scan.nextInt();

System.***out***.println(x + " " + y + " " + c);

//output to output2

System.*setOut*(**new** PrintStream(fos2));

newedge = **new** undirectedEdge(x,y,c);

pmst.insertEdge(newedge);

pmst.printList(pmst.edgeListHead);

}

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **finally**{

**if**(scan != **null**) scan.close();

}

System.*setOut*(**new** PrintStream(fos1));

System.***out***.println();

System.***out***.println("List MSTofG:");

//Step 4,5,6,7,8

**while**(!pmst.isSetBEmpty()){

//step 4,5

newedge = pmst.removeEdge(pmst.edgeListHead);

//step 6

pmst.pushEdge(newedge);

pmst.totalMSTCost += newedge.edgeCost;

**if**(pmst.inWhichSet[newedge.Ni] == 1){

pmst.move2SetA(newedge.Nj);

}

**else**{

pmst.move2SetA(newedge.Ni);

}

System.*setOut*(**new** PrintStream(fos2));

pmst.printSet();

//step 7

System.*setOut*(**new** PrintStream(fos1));

pmst.printList(pmst.MSTofG);

}

System.***out***.println();

System.***out***.println("Final MSTofG:");

pmst.printList(pmst.MSTofG);

System.***out***.println("\*\*\* The total Cost of the Prim’s MST is: " + pmst.totalMSTCost);

//finish up

**try** {

fos1.close();

fos2.close();

System.*setOut*(**new** PrintStream(**new** FileOutputStream(FileDescriptor.***out***)));

} **catch** (IOException e) {

e.printStackTrace();

}

System.***out***.println("Done");

}

}

**Input**

9

6 4 3

2 4 1

3 2 5

5 7 5

1 6 3

8 6 2

9 8 2

4 3 3

1 2 6

3 5 4

6 7 2

**Output 1**

\*\*\* The Prim¡¯s MST of the input graph is given below: \*\*\*

9

6 4 3

2 4 1

3 2 5

5 7 5

1 6 3

8 6 2

9 8 2

4 3 3

1 2 6

3 5 4

6 7 2

List MSTofG:

listHead --> <0,0,0> --> <1,6,3>

listHead --> <0,0,0> --> <1,6,3> --> <6,7,2>

listHead --> <0,0,0> --> <1,6,3> --> <6,7,2> --> <8,6,2>

listHead --> <0,0,0> --> <1,6,3> --> <6,7,2> --> <8,6,2> --> <9,8,2>

listHead --> <0,0,0> --> <1,6,3> --> <6,7,2> --> <8,6,2> --> <9,8,2> --> <6,4,3>

listHead --> <0,0,0> --> <1,6,3> --> <6,7,2> --> <8,6,2> --> <9,8,2> --> <6,4,3> --> <2,4,1>

listHead --> <0,0,0> --> <1,6,3> --> <6,7,2> --> <8,6,2> --> <9,8,2> --> <6,4,3> --> <2,4,1> --> <4,3,3>

listHead --> <0,0,0> --> <1,6,3> --> <6,7,2> --> <8,6,2> --> <9,8,2> --> <6,4,3> --> <2,4,1> --> <4,3,3> --> <3,5,4>

Final MSTofG:

listHead --> <0,0,0> --> <1,6,3> --> <6,7,2> --> <8,6,2> --> <9,8,2> --> <6,4,3> --> <2,4,1> --> <4,3,3> --> <3,5,4>

\*\*\* The total Cost of the Prim¡¯s MST is: 20

**Output 2**

listHead --> <0,0,0>

listHead --> <0,0,0> --> <6,4,3>

listHead --> <0,0,0> --> <2,4,1> --> <6,4,3>

listHead --> <0,0,0> --> <2,4,1> --> <6,4,3> --> <3,2,5>

listHead --> <0,0,0> --> <2,4,1> --> <6,4,3> --> <5,7,5> --> <3,2,5>

listHead --> <0,0,0> --> <2,4,1> --> <1,6,3> --> <6,4,3> --> <5,7,5> --> <3,2,5>

listHead --> <0,0,0> --> <2,4,1> --> <8,6,2> --> <1,6,3> --> <6,4,3> --> <5,7,5> --> <3,2,5>

listHead --> <0,0,0> --> <2,4,1> --> <9,8,2> --> <8,6,2> --> <1,6,3> --> <6,4,3> --> <5,7,5> --> <3,2,5>

listHead --> <0,0,0> --> <2,4,1> --> <9,8,2> --> <8,6,2> --> <4,3,3> --> <1,6,3> --> <6,4,3> --> <5,7,5> --> <3,2,5>

listHead --> <0,0,0> --> <2,4,1> --> <9,8,2> --> <8,6,2> --> <4,3,3> --> <1,6,3> --> <6,4,3> --> <5,7,5> --> <3,2,5> --> <1,2,6>

listHead --> <0,0,0> --> <2,4,1> --> <9,8,2> --> <8,6,2> --> <4,3,3> --> <1,6,3> --> <6,4,3> --> <3,5,4> --> <5,7,5> --> <3,2,5> --> <1,2,6>

listHead --> <0,0,0> --> <2,4,1> --> <6,7,2> --> <9,8,2> --> <8,6,2> --> <4,3,3> --> <1,6,3> --> <6,4,3> --> <3,5,4> --> <5,7,5> --> <3,2,5>

inWhichSet array:

1 2 2 2 2 1 2 2 2

inWhichSet array:

1 2 2 2 2 1 1 2 2

inWhichSet array:

1 2 2 2 2 1 1 1 2

inWhichSet array:

1 2 2 2 2 1 1 1 1

inWhichSet array:

1 2 2 1 2 1 1 1 1

inWhichSet array:

1 1 2 1 2 1 1 1 1

inWhichSet array:

1 1 1 1 2 1 1 1 1

inWhichSet array:

1 1 1 1 1 1 1 1 1