**CS323-22: Project 7A (CPP)**

**Yida Tao**

**Due date: Nov. 28, 2017**

Algorithm Steps:

Step 0: numNodes 🡨 get from input file

Allocate and initialize all the members in KruskalMST class,

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

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)

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) with inWhichSet(Nj) to see if there are equal 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

merge2Sets (Ni, Nj) // now, Ni, Nj are in the same set

numSets --

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

Step 7: printList(MSTofG) // print up to the first 10 edges of MSTofG) to argv[3] file

Step 8: repeat step 4 – step 7 until numSets is equal to 1.

Step 9: output the entire MSTofG and the totalCost to argv[2] with proper heading, one edge with cost per text line.

Step 10: close all files.

**Source Code**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class graphNode{

public:

int nodeID;

graphNode \*next;

graphNode(int i){

nodeID = i;

next = NULL;

}

~graphNode(){

delete this;

}

};

class undirectedEdge{

public:

int Ni;

int Nj;

int edgeCost;

undirectedEdge \*next;

undirectedEdge(){

Ni = 0;

Nj = 0;

edgeCost = 0;

next = NULL;

}

undirectedEdge(int i, int j, int c){

Ni = i;

Nj = j;

edgeCost = c;

next = NULL;

}

void printEdge(){

cout<< "<" << Ni << "," << Nj << "," << edgeCost <<">";

}

};

class KruskalMST{

public:

int numNodes;

int \*inWhichSet;

int numSets;

int totalMSTCost;

undirectedEdge \*MSTofG;

undirectedEdge \*edgeListHead;

KruskalMST(int num){

numNodes = num;

inWhichSet = new int[numNodes + 1];

numSets = numNodes;

totalMSTCost = 0;

MSTofG = new undirectedEdge();

edgeListHead = new undirectedEdge();

for(int i = 0; i < numNodes + 1; i++){

inWhichSet[i] = i;

}

}

~KruskalMST(){

undirectedEdge \*walker = edgeListHead;

while(walker != NULL){

walker = walker->next;

delete edgeListHead;

edgeListHead = walker;

}

walker = MSTofG;

while(walker != NULL){

walker = walker->next;

delete MSTofG;

MSTofG = walker;

}

delete[] inWhichSet;

}

void printList(undirectedEdge \*head){

undirectedEdge \*temp = head;

int counter = 0;

temp->printEdge();

while(temp->next != NULL && counter < 10){

temp = temp->next;

cout<<" --> ";

temp->printEdge();

counter++;

}

cout<<endl;

cout<<endl;

}

void printSet(){

cout<<"inWhichSet array: "<<endl;

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

cout<<i<<" is in : group "<< inWhichSet[i]<<endl;

}

cout<<endl;

}

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 \*walker = edgeListHead;

if(walker->next == NULL){

throw ("list is empty");

}

walker = walker->next;

edgeListHead->next = walker->next;

return walker;

}

void pushEdge(undirectedEdge \*edge){

undirectedEdge \*walker = MSTofG;

while(walker->next != NULL) {

walker = walker->next;

}

edge->next = walker->next;

walker->next = edge;

}

void merge2Sets(int i, int j){

if(inWhichSet[i] < inWhichSet[j]){

int temp = inWhichSet[j];

for(int k = 1; k < numNodes + 1; k++){

if(inWhichSet[k] == temp){

inWhichSet[k] = inWhichSet[i];

}

}

}

else{

int temp = inWhichSet[i];

for(int k = 1; k < numNodes + 1; k++){

if(inWhichSet[k] == temp){

inWhichSet[k] = inWhichSet[j];

}

}

}

}

};

int main(int argc, char \*argv[]) {

//file checking ----------------------------------------------------

if(argv[1]==NULL || argv[2]==NULL || argv[3]==NULL) {

cout<<"no parameter, require 1 input file, 2 output file"<<endl;

return 0;

}

ifstream infile;

infile.open(argv[1]);

if(!infile.is\_open()){

cout<<"cant find file"<<endl;

return 0;

}

infile.close();

streambuf \*console = cout.rdbuf();

ofstream out1;

out1.open(argv[2]);

ofstream out2;

out2.open(argv[3]);

//end file checking--------------------------------------------------

//set output to output file 1

cout.rdbuf(out1.rdbuf());

KruskalMST \*kmst = NULL;

undirectedEdge \*edge = NULL;

int i = 0;

int j = 0;

int c = 0;

infile.open(argv[1]);

// Step 0

infile >> i;

cout << "\*\*\* A Kruskal's MST of the input graph is given below: \*\*\*"<<endl;

cout << i <<endl;

kmst = new KruskalMST(i);

//debug

cout.rdbuf(out2.rdbuf());

//kmst->printList();

kmst->printSet();

//Step 1,2,3

while(infile >> i) {

cout.rdbuf(out1.rdbuf());

infile >> j;

infile >>c;

cout << i <<" "<< j <<" "<< c <<endl;

cout.rdbuf(out2.rdbuf());

edge = new undirectedEdge(i,j,c);

kmst->insertEdge(edge);

cout<<"edgeListHead --> ";

kmst->printList(kmst->edgeListHead);

}

infile.close();

cout<<endl;

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

while(kmst->numSets != 1){

//step 4,5

do{

edge = kmst->removeEdge();

}while(kmst->inWhichSet[edge->Ni] == kmst->inWhichSet[edge->Nj]);

//step 6

kmst->pushEdge(edge);

kmst->totalMSTCost += edge->edgeCost;

kmst->merge2Sets(edge->Ni, edge->Nj);

kmst->numSets --;

cout<<"------------------------------------------------------"<<endl;

cout<<"numset:" <<kmst->numSets<<endl;

kmst->printSet();

//step 7

cout<<"MSTofG --> ";

kmst->printList(kmst->MSTofG);

cout<<"edgeListHead --> ";

kmst->printList(kmst->edgeListHead);

}

//step 9

cout.rdbuf(out1.rdbuf());

cout<<endl;

cout<<"Final MSTofG:"<<endl;

kmst->printList(kmst->MSTofG);

cout<<"\*\*\* The total cost of a Kruskal's MST is: "<<kmst->totalMSTCost<<endl;

//step 10

cout.rdbuf(console);

cout<<"input: "<<argv[1]<<endl;

cout<<"output1: "<<argv[2]<<endl;

cout<<"output2: "<<argv[3]<<endl;

cout<<"done"<<endl;

out1.close();

out2.close();

delete edge;

delete kmst;

return 0;

}

**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**

\*\*\* A Kruskal'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

Final MSTofG:

<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>

\*\*\* The total cost of a Kruskal's MST is: 20

**Output 2**

inWhichSet array:

1 is in : group 1

2 is in : group 2

3 is in : group 3

4 is in : group 4

5 is in : group 5

6 is in : group 6

7 is in : group 7

8 is in : group 8

9 is in : group 9

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

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

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

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

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

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

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

edgeListHead --> <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>

edgeListHead --> <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>

edgeListHead --> <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>

edgeListHead --> <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>

------------------------------------------------------

numset:8

inWhichSet array:

1 is in : group 1

2 is in : group 2

3 is in : group 3

4 is in : group 2

5 is in : group 5

6 is in : group 6

7 is in : group 7

8 is in : group 8

9 is in : group 9

MSTofG --> <0,0,0> --> <2,4,1>

edgeListHead --> <0,0,0> --> <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> --> <1,2,6>

------------------------------------------------------

numset:7

inWhichSet array:

1 is in : group 1

2 is in : group 2

3 is in : group 3

4 is in : group 2

5 is in : group 5

6 is in : group 6

7 is in : group 6

8 is in : group 8

9 is in : group 9

MSTofG --> <0,0,0> --> <2,4,1> --> <6,7,2>

edgeListHead --> <0,0,0> --> <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>

------------------------------------------------------

numset:6

inWhichSet array:

1 is in : group 1

2 is in : group 2

3 is in : group 3

4 is in : group 2

5 is in : group 5

6 is in : group 6

7 is in : group 6

8 is in : group 8

9 is in : group 8

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

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

------------------------------------------------------

numset:5

inWhichSet array:

1 is in : group 1

2 is in : group 2

3 is in : group 3

4 is in : group 2

5 is in : group 5

6 is in : group 6

7 is in : group 6

8 is in : group 6

9 is in : group 6

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

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

------------------------------------------------------

numset:4

inWhichSet array:

1 is in : group 1

2 is in : group 2

3 is in : group 2

4 is in : group 2

5 is in : group 5

6 is in : group 6

7 is in : group 6

8 is in : group 6

9 is in : group 6

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

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

------------------------------------------------------

numset:3

inWhichSet array:

1 is in : group 1

2 is in : group 2

3 is in : group 2

4 is in : group 2

5 is in : group 5

6 is in : group 1

7 is in : group 1

8 is in : group 1

9 is in : group 1

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

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

------------------------------------------------------

numset:2

inWhichSet array:

1 is in : group 1

2 is in : group 1

3 is in : group 1

4 is in : group 1

5 is in : group 5

6 is in : group 1

7 is in : group 1

8 is in : group 1

9 is in : group 1

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

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

------------------------------------------------------

numset:1

inWhichSet array:

1 is in : group 1

2 is in : group 1

3 is in : group 1

4 is in : group 1

5 is in : group 1

6 is in : group 1

7 is in : group 1

8 is in : group 1

9 is in : group 1

MSTofG --> <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>

edgeListHead --> <0,0,0> --> <5,7,5> --> <3,2,5> --> <1,2,6>