## CSCI 230 PA 12 Submission

## Due Date: ##/##/2022 Late (date and time):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Name(s): Nero Li

Exercise 1 -- need to submit source code and I/O  
 -- check if completely done ✔️ ; otherwise, discuss issues below

Source code below:

**exercise\_1.cpp:**

/\* Program: PA\_12\_exercise\_1

Author: Nero Li

Class: CSCI 230

Date: 05/31/2022

Description:

Implement one MST algorithm -- either Prim-Jarnik Algorithm or

Kruskal Algorithm. Try a small graph below and print out the MST

and total cost.

I certify that the code below is my own work.

Exception(s): N/A

\*/

#include <iostream>

#include <queue>

#include "AdjacencyListGraph.h"

#include "HeapPriorityQueue.h"

using namespace std;

typedef pair<int, Vertex \*> PQP;

class isLess

{

public:

bool operator()(PQP a, PQP b)

{

return a.first < b.first;

}

};

void printMST(AdjacencyListGraph G, map<Vertex \*, Edge \*> parent, Vertex \*src, int level)

{

cout << src->getElement() << endl;

for (auto i : parent)

{

if (i.second != NULL)

{

if (G.opposite(i.first, i.second) == src)

{

for (int p = 0; p < level; ++p)

{

if (p == level - 1)

cout << "-";

else

cout << " ";

}

printMST(G, parent, i.first, level + 1);

}

}

}

}

void PrimJarnikMST(AdjacencyListGraph G, Vertex \*s)

{

HeapPriorityQueue<PQP, isLess> Q;

map<Vertex \*, int> distance;

map<Vertex \*, Edge \*> parent;

map<Vertex \*, PQP> locator;

map<Vertex \*, bool> visited;

for (auto v : G.getVertices())

{

if (v == s)

distance[v] = 0;

else

distance[v] = INT\_MAX;

parent[v] = NULL;

PQP l;

l.first = distance[v];

l.second = v;

Q.insert(l);

locator[v] = l;

visited[v] = false;

}

while (!Q.empty())

{

PQP l = Q.min();

Q.removeMin();

Vertex \*u = l.second;

// cout << u->getElement() << endl;

visited[u] = true;

for (auto e : G.incomingEdges(u))

{

Vertex \*z = G.opposite(u, e);

int r = e->getElement();

if (r < distance[z] && !visited[z])

{

distance[z] = r;

parent[z] = e;

Q.replace(locator[z], r);

// cout << z->getElement() << ", " << r << ": " << Q.min().second->getElement() << endl;

}

}

}

printMST(G, parent, s, 1);

int cost{0};

for (auto i : parent)

if (i.second != NULL)

cost += i.second->getElement();

cout << "Total cost: " << cost << endl;

}

int main()

{

AdjacencyListGraph G;

Vertex \*A = G.insertVertex("A");

Vertex \*B = G.insertVertex("B");

Vertex \*C = G.insertVertex("C");

Vertex \*D = G.insertVertex("D");

Vertex \*E = G.insertVertex("E");

G.insertEdge(A, B, 3);

G.insertEdge(A, D, 5);

G.insertEdge(A, E, 5);

G.insertEdge(B, C, 4);

G.insertEdge(C, D, 2);

G.insertEdge(D, E, 5);

G.insertEdge(C, E, 3);

cout << "Original Graph:\n";

G.print();

cout << endl;

cout << "MST:\n";

PrimJarnikMST(G, A);

cout << endl;

cout << "Modified by: Nero Li\n";

return 0;

}

Input/output below:

Original Graph:

Vertex A

3 adjacencies:(B, 3) (D, 5) (E, 5)

Vertex B

2 adjacencies:(A, 3) (C, 4)

Vertex C

3 adjacencies:(B, 4) (D, 2) (E, 3)

Vertex D

3 adjacencies:(A, 5) (C, 2) (E, 5)

Vertex E

3 adjacencies:(A, 5) (D, 5) (C, 3)

MST:

A

-B

-C

-D

-E

Total cost: 12

Modified by: Nero Li

Exercise 2 (with extra credit) -- need to submit source code and I/O  
 -- check if completely done ✔️ ; otherwise, discuss issues below

Source code below:

**exercise\_2.cpp:**

/\*

Input/output below:

Modified by: Nero Li

Answer for Question 1:

Ans

Answer for Question 2:

Ans