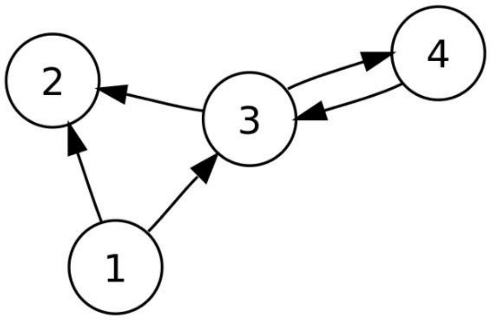
Graph Algorithms



DOCUMENTATION REPORT

PROGRAMMING PROJECT 2

ITCS 6114 – Algorithms and Data Structures

DEPARTMENT OF COMPUTER SCIENCE

SUBMITTED TO
Dewan T. Ahmed, Ph.D.

SUBMITTED BY:

Shreeya Gupta **Sai Kumar Thallada**

801136226 **801154715**

UNC CHARLOTTE
College of Computing and Informatics

Graph Algorithms

Problem1:

Find shortest path tree in both directed and undirected weighted graphs for a given source vertex. Assume there is no negative edge in your graph. You will print each path and path cost for a given source.

Dijkstra's Algorithm:

It's an algorithm for finding the shortest paths between nodes in graph. Shortest path tree is created with given source as root. 2 sets are maintained one contains vertices included shortest path while other includes vertices not yet included in shortest path tree, at each step a vertex that is not yet included and has a minimum distance from the source.

Data Structure Used:

HashMap is used for mapping to be done, for instance to map vertex names to Vertex objects, built from a set of Edges. Implementation of Dijkstra's algorithm has been done using a binary heap.

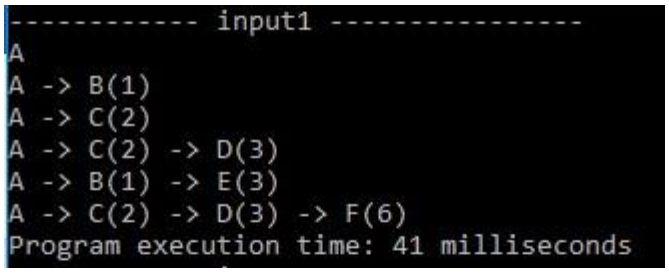
Run Time:

Dijkstra's Algorithm run time
input1.txt: 41 ms
input2.txt: 47 ms
input3.txt: 44 ms
input4.txt: 60 ms

Sample Input:

a) input1.txt :

```
6 10 U
A B 1
A C 2
B C 1
B D 3
B E 2
C D 1
C E 2
D E 4
D F 3
E  F 3
A
```

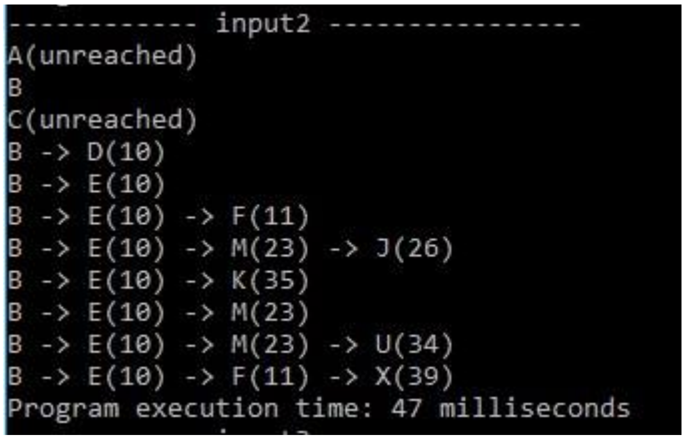


```
------------ input1 ----------------
A
A -> B(1)
A -> C(2)
A -> C(2) -> D(3)
A -> B(1) -> E(3)
A -> C(2) -> D(3) -> F(6)
Program execution time: 41 milliseconds
```

b) input2.txt :

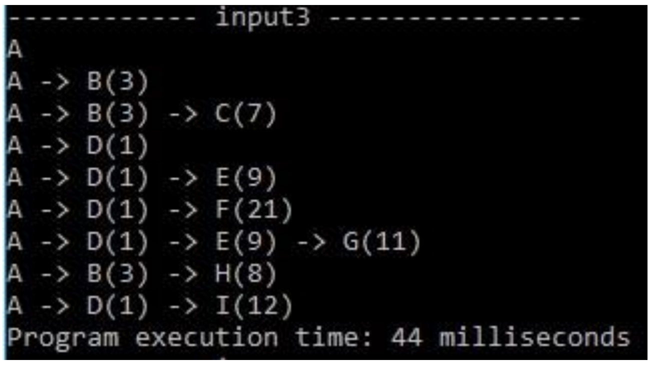
```
11 18 D
A B 10
A C 20
B D 10
C E 33
B E 10
C D 20
D E 20
D F 2
E  F 1
E  K 25
E M 13
F M 22
F  X 28
K J 4
M U 11
```

M J 3
X U 10
J U 15
B


```
------------ input2 ----------------
A(unreached)
B
C(unreached)
B -> D(10)
B -> E(10)
B -> E(10) -> F(11)
B -> E(10) -> M(23) -> J(26)
B -> E(10) -> K(35)
B -> E(10) -> M(23)
B -> E(10) -> M(23) -> U(34)
B -> E(10) -> F(11) -> X(39)
Program execution time: 47 milliseconds
```

c) input3.txt :

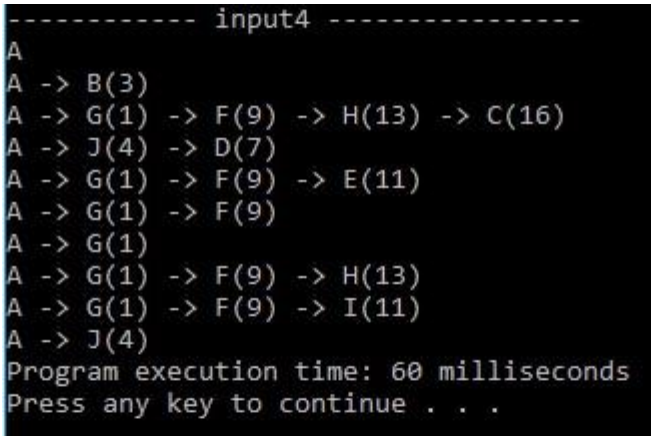
9 16 D
A B 3
A C 8
A D 1
B C 4
B H 5
C D 9
C H 3
C I 6
D I 11
D E 8
D G 12
D F 20
H I 10
E F 17
E G 2
G F 15
A



```
------------ input3 ----------------
A
A -> B(3)
A -> B(3) -> C(7)
A -> D(1)
A -> D(1) -> E(9)
A -> D(1) -> F(21)
A -> D(1) -> E(9) -> G(11)
A -> B(3) -> H(8)
A -> D(1) -> I(12)
Program execution time: 44 milliseconds
```

d) input4.txt :

```
10 15 U
A B 3
A J 4
A G 1
B D 10
D J 3
D H 11
J G 6
G F 8
G E 14
F H 4
F I 2
F E 2
H I 6
I  E 1
H C 3
A
```



```
------------ input4 ----------------
A
A -> B(3)
A -> G(1) -> F(9) -> H(13) -> C(16)
A -> J(4) -> D(7)
A -> G(1) -> F(9) -> E(11)
A -> G(1) -> F(9)
A -> G(1)
A -> G(1) -> F(9) -> H(13)
A -> G(1) -> F(9) -> I(11)
A -> J(4)
Program execution time: 60 milliseconds
Press any key to continue . . .
```

Problem 2 :

To find the minimum spanning tree of a connected, undirected, weighted graph, Kruskal's algorithm has been implemented.

Minimum Spanning Tree

In a connected, undirected weighted graph a tree that connects all the vertices together and is also a subgraph of the given graph and is also a tree is a spanning tree. There are many different spanning trees possible for a single graph. A spanning tree with weight less than or equal to the weight of every other spanning tree is a minimum spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

Kruskal's Algorithm

In this algorithm there is a forest where each vertex is a single node tree. It finds a safe edge to add to the growing forest by finding, of all the edges that connect any two trees in the forest, an edge (u, v) of least weight. This algorithm is a greedy algorithm as at each step it adds an edge of least possible weight to the forest. At the end of algorithm, we are left with a one cloud that encompasses the Minimum Spanning Tree and also a tree which is the Minimum Spanning Tree.

Data Structures Used:

We have used array and HashMap as data Structures. In the HashMap we are mapping the nodes (vertices) and in the Array List we are taking a list of edges.

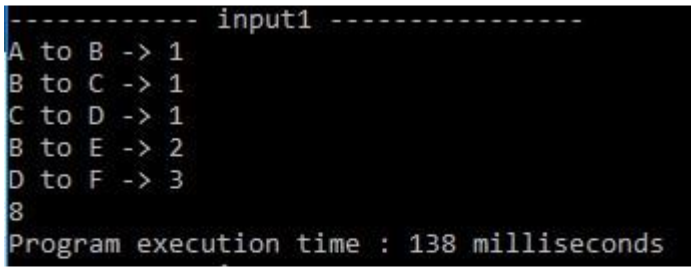
Run time:

Kruskal's Algorithm run time
input1.txt: 138ms
input2.txt: -
input3.txt: -
input4.txt: 147 ms

__Sample Input:__

a) input1.txt :

```
6 10 U
A B 1
A C 2
B C 1
B D 3
B E 2
C D 1
C E 2
D E 4
D F 3
E  F 3
A
```

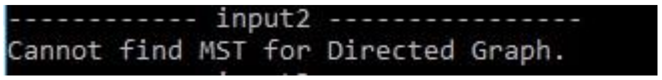


```
------------ input1 ----------------
A to B -> 1
B to C -> 1
C to D -> 1
B to E -> 2
D to F -> 3
8
Program execution time : 138 milliseconds
```

b) input2.txt :

```
11 18 D
A B 10
A C 20
B D 10
C E 33
B E 10
C D 20
D E 20
D F 2
E  F 1
E  K 25
E M 13
F M 22
F  X 28
K J 4
M U 11
```

M J 3
X U 10
J U 15
B

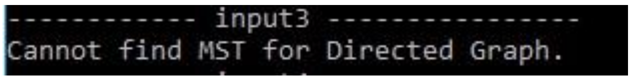


```
----------- input2 ---------------
Cannot find MST for Directed Graph.
```

c) input3.txt : 9 16 4

A B 4
A C 4
A D 4
B C 4
B H 4
C D 4
C H 4
C I 4
D I 11 D E 4

D G 12
D F 20
H I 10
E F 17
E G 2
G F 15
A

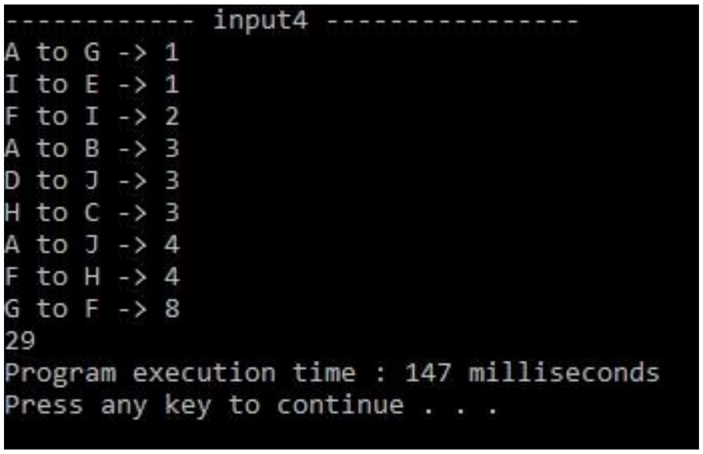


```
----------- input3 ---------------
Cannot find MST for Directed Graph.
```

d) input4.txt :

10 15 U
A B 3
A J 4
A G 1
B D 10
D J 3
D H 11
J G 6
G F 8

```
G E 14
F H 4
F I 2
F E 2
H I 6
I  E 1
H C 3
A
```



```
------------ input4 ----------------
A to G -> 1
I to E -> 1
F to I -> 2
A to B -> 3
D to J -> 3
H to C -> 3
A to J -> 4
F to H -> 4
G to F -> 8
29
Program execution time : 147 milliseconds
Press any key to continue . . .
```