**Program 4: Minimum Spanning Trees and Shortest Path**

Due Dates: Part 1 July 27, 2021

Part1 & 2 August 3, 2021

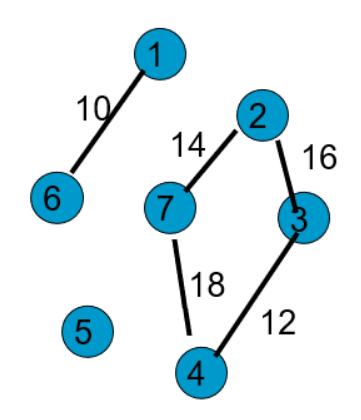
**Overview:**

The problem to modify the program you have already written to create the MST – perfective type of program maintenance!

In the previous program you created a graph from an input file then used a min-heap to implement Prim’s Algorithm to create the MST.

There was the assumption that the input graph was connected, however, in reality there are many graphs that are not connected

Thus the need to modify your program to handle non-connected graphs – in this case, each part of the graph would have a separate MST. Also, the program would need to handle, for example, where the graph has vertices and no edges (e.g. 4 vertices with no edges is 4 non-connected sub-graphs).



3 MSTs

5 cost 0 (since no edge)

1 – 6 cost 10

2 – 7

2 – 3 cost 42

3 – 4

You are to also implement the Dijkstra’s algorithm for a shortest path(s) requests against the full graph. You may have multiple shortest path scenarios with an unconnected graph like the above.

File input is the same format as the previous program. The program will process a graph from the input file creating the MST(s), the user will then enter which vertex to create the shortest path from and the program will generate the shortest path table (some vertices may be unreachable!).

The user can then select from a menu to enter:

* a destination request - program will return the path and cost or if no path exists
* a new starting vertex - program generates new shortest path table
* exit to next graph in input file or end program if no more graphs in input file

You need to perform edits on the user input, examples:

* was the vertex entered a valid vertex for that graph?
* was the selection a valid menu option?

Output file will show what the user input and the output that was presented on the screen.

Input Format

Input and output names are to be entered by the user.

Line 1: Two space-separated integers: X , the number of nodes in the graph, and Y , the number of edges.

Lines 2 . . . Y+1 :

Line i contains three space-separated numbers describing an edge:

si and ti , the IDs of the two nodes involved, and wi , the weight of the edge.

Sample input (**input file name format is short#.dat where this is file short1.dat**)

6 9

0 1 1

1 3 5

3 0 3

3 4 1

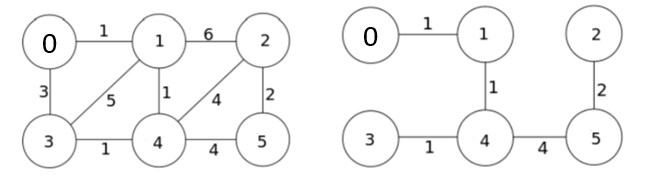
1 4 1

1 2 6

5 2 2

2 4 4

5 4 4



Graph MST

Line 1: 6 nodes (0 – 5) and 9 edges to follow line 2 - 10

Line 2: 0 1 1 describes edge between 0 and 1 of weight 1

Line 3: 1 3 5 describes edge between 1 and 3 of weight 5

…

The MST sum is 1+1+1+4+2 = 9

Output to Screen and File (output file name format: **short#.out**)

Echo print the input (format appropriately similar to previous program)

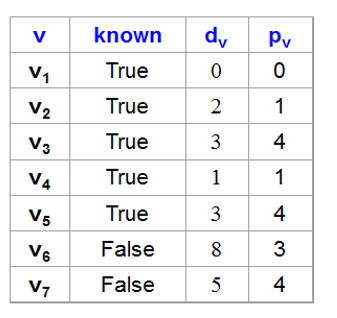
Print the adjacency list for the full graph

Print the adjacency list for the MST(s)

After receiving starting vertex for shortest path show shortest path table created

- if number of nodes < 10 print each iteration shortest path table of Dijkstra’s algorithm

Example shortest path table:



Start the Program 4 turn in template with Part 1 & Part 2 template from Program 3!!!!!!!

Assumptions

Input format is correct

Input line 1 numbers are correct

Example Exception Handling **(for full credit you must include additional edits)**

File handling (file does not exist,empty) stop program

Error if vertices on edges do not exist error message to screen and file

continue next input

REQUIRED FILES/TESTS:

**Input file does not exist or is empty – error message to screen and output file - stop program**

# Input file (**short1.dat**)

6 9

0 1 1

1 3 5

3 0 3

3 4 1

1 4 1

1 2 6

5 2 2

2 4 4

5 4 4

User requests shortest path start vertex of 4

Display iterations of Dijkstra’s algorithm (number of vertices < 10) to show it is calculation properly

User enters destination vertex of 2

Display shortest path and distance or no path

User enters destination vertex of 0

Display shortest path and distance or no path

User requests shortest path start vertex of 0

Display iterations of Dijkstra’s algorithm (number of vertices < 10) to show it is calculation properly

User enters destination vertex of 2

Display shortest path and distance or no path

User enters destination vertex of 3

Display shortest path and distance or no path

User enters destination vertex of 2

Display shortest path and distance or no path

User enters destination vertex of 0

Display shortest path and distance or no path

# Input file (**short2.dat**)

5 4 print out “5 vertices, 4 edges

0 1 1  print “add edge 0 1 with weight 1” (edge added message)

1 3 5 print edge added message

3 0 3 print edge added message

2 4 1  print edge added message

Print out the adjacency list for full graph and MSTs along with the sum for each MST as this graph is not connected

User requests shortest path start vertex of 4

Display iterations of Dijkstra’s algorithm (number of vertices < 10) to show it is calculation properly

User enters destination vertex of 2

Display shortest path and distance or no path

User enters destination vertex of 0

Display shortest path and distance or no path

User requests shortest path start vertex of 0

Display iterations of Dijkstra’s algorithm (number of vertices < 10) to show it is calculation properly

User enters destination vertex of 0

Display shortest path and distance or no path

User enters destination vertex of 31

Display error “Destination path vertex entered: 31 is not a valid vertex”

User enters destination vertex of 2

Display shortest path and distance or no path

User enters destination vertex of 3

Display shortest path and distance or no path

User requests shortest path start vertex of -1

Display error “Shortest path starting vertex entered: -1 is not a valid vertex”

Input File (**short3.dat**)

6 16 [messaging details not listed]

6 16

0 1 1

1 3 5

-3 1 4 invalid source vertex

3 0 3

3 4 1

6 1 6 invalid source vertex

1 4 1

1-3 4 invalid destination vertex

2 4 0 invalid weight

1 2 6

5 2 2

2 4 -3 invalid weight

2 4 4

5 14 5 invalid destination vertex

5 4 4

15 3 7  invalid source vertex

User requests shortest path start vertex of 1

Display iterations of Dijkstra’s algorithm (number of vertices < 10) to show it is calculation properly

User enters destination vertex of 2

Display shortest path and distance or no path

User enters destination vertex of 5

Display shortest path and distance or no path

User requests shortest path start vertex of 1

Display iterations of Dijkstra’s algorithm (number of vertices < 10) to show it is calculation properly

User enters destination vertex of 0

Display shortest path and distance or no path

User enters destination vertex of 3

Display shortest path and distance or no path

Input File (**short4.dat**) (spacing to show what would be considered separate graphs)

0 0

-1 3

0 1 1

1 2 1

2 3 4

6 -4

5 0 (now a valid graph with no edges!!!!!!!)

User requests shortest path start vertex of 4

Display iterations of Dijkstra’s algorithm (number of vertices < 10) to show it is calculation properly

User enters destination vertex of 2

Display shortest path and distance or no path

User enters destination vertex of 0

Display shortest path and distance or no path

User requests shortest path start vertex of 0

Display iterations of Dijkstra’s algorithm (number of vertices < 10) to show it is calculation properly

User enters destination vertex of 0

Display shortest path and distance or no path

User enters destination vertex of 3

Display shortest path and distance or no path

0 5

0 1 3

1 2 6

2 3 5

3 4 6

4 5 3

5 3 (now a valid graph!)

0 3 5

0 4 6

1 2 4

User requests shortest path start vertex of 4

Display iterations of Dijkstra’s algorithm (number of vertices < 10) to show it is calculation properly

User enters destination vertex of 2

Display shortest path and distance or no path

User enters destination vertex of 0

Display shortest path and distance or no path

User requests shortest path start vertex of 1

Display iterations of Dijkstra’s algorithm (number of vertices < 10) to show it is calculation properly

User enters destination vertex of 0

Display shortest path and distance or no path

User enters destination vertex of 3

Display shortest path and distance or no path

You **MUST** submit **three** additional test files