Algorithms & Data Structures Project 2

Graph Algorithms: Single-source Shortest Path and Minimum Spanning Tree (MST)

Submitted To:

Dr Dewan Tanvir Ahmed

Submitted By:

Ankit Pandita

Jinraj Jain

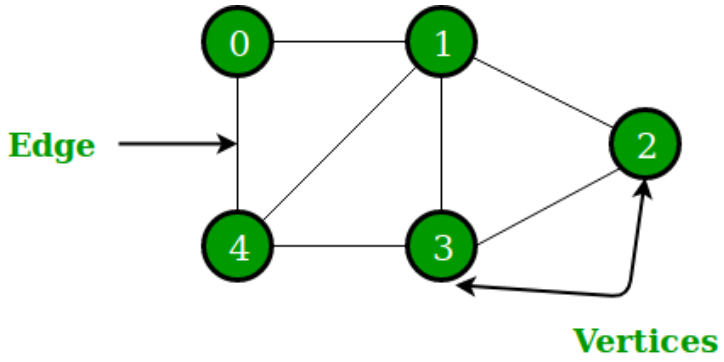
Avijit Jaiswal

GRAPH

A **Graph** is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph.

More formally a Graph can be defined as,

A Graph consists of a finite set of vertices(or nodes) and set of Edges which connect a pair of nodes.



Graphs are used to solve many real-life problems. Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network.

Graphs are also used in social networks like linkedIn, Facebook. For example, in Facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender, locale etc.

Undirected graphs have edges that do not have a direction. The edges indicate a two-way relationship, in that each edge can be traversed in both directions.

Directed graphs have edges with direction. The edges indicate a one-way relationship, in that each edge can only be traversed in a single direction.

DIJKSTRA'S ALGORITHM

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks.

For a given source node in the graph, the algorithm finds the shortest path between that node and every other.

It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined.

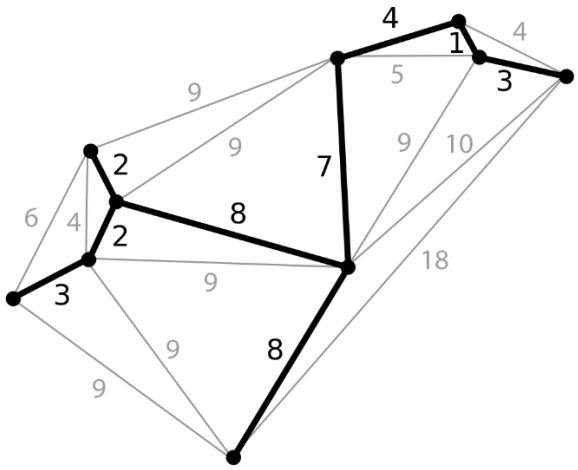
For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road (for simplicity, ignore red lights, stop signs, toll roads and other obstructions), Dijkstra's algorithm can be used to find the shortest route between one city and all other cities.

Runtime: Dijkstra's Algorithm has a runtime of O(E log V), where E is the number of edges and V is the number of vertices.

MINIMUM SPANNING TREE

A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

That is, it is a spanning tree whose sum of edge weights is as small as possible. More generally, any edge-weighted undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of the minimum spanning trees for its connected components.



KRUSKAL'S ALGORITHM

Kruskal's algorithm is a minimum-spanning-tree algorithm which finds an edge of the least possible weight that connects any two trees in the forest.

It is a greedy algorithm in graph theory as it finds a minimum spanning tree for a connected weighted graph adding increasing cost arcs at each step.

This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.

If the graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree for each connected component).

Runtime: Kruskal's Algorithm has a time complexity of O(E log E).

PROGRAM STRUCTURE:

The programming language used is Java.

The project consists of 3 source files:

Main.java: Consists of the main driving function for the program.

ShortestPath.java: Consists of the implementation for Dijkstra's shortest path algorithm.

MinimumSpanningTree.java: Consists of the implementation for Kruskal's Minimum Spanning tree algorithm.

Input is provided in the form of text files.

Data Structure Used:

Shortest Path: One-dimensional and two-dimensional array

Minimum Spanning Tree: Arrays and Union

INPUT and OUTPUT:

Implement Dijkstra's algorithm for shortest path for UnDirected graphs.

Case 1 Input

9 14 U

A B 4

A H 8

B C 8

B H 11

C D 7

C F 4

C I 2

D E 9

D F 14

E F 10

F G 2

G I 6

G H 1

H I 7

C

Output

Number of vertices = 9

Number of edges = 14

Selected Graph is Undirected.

Applying Dijkstra's Algorithm:

	A	B	C	D	E	F	G	H	I
A	0	4	0	0	0	0	0	8	0
B	4	0	8	0	0	0	0	11	0

C	0	8	0	7	0	4	0	0	2
D	0	0	7	0	9	14	0	0	0
E	0	0	0	9	0	10	0	0	0
F	0	0	4	14	10	0	2	0	0
G	0	0	0	0	0	2	0	1	6
H	8	11	0	0	0	0	1	0	7
I	0	0	2	0	0	0	6	7	0

Source of graph is C

Shortest Path from source:

C -> B -> A = 12

C -> B = 8

C -> D = 7

C -> F -> E = 14

C -> F = 4

C -> F -> G = 6

C -> F -> G -> H = 7

C -> I = 2

Total time taken = 21900 ns

Case 2 Input

9 17 U

A B 3

A G 5

A H 7

B C 7

B H 1

C D 1

C H 2

C I 2

D E 5

D I 3

E I 2

E F 4

F G 2

F H 3

F I 3

G H 3

H I 1

H

Output

Number of vertices = 9

Number of edges = 17

Selected Graph is Undirected.

Applying Dijkstra's Algorithm:

	A	B	C	D	E	F	G	H	I
A	0	3	0	0	0	0	5	7	0
B	3	0	7	0	0	0	0	1	0
C	0	7	0	1	0	0	0	2	2
D	0	0	1	0	5	0	0	0	3
E	0	0	0	5	0	4	0	0	2
F	0	0	0	0	4	0	2	3	3
G	5	0	0	0	0	2	0	3	0
H	7	1	2	0	0	3	3	0	1
I	0	0	2	3	2	3	0	1	0

Source of graph is H

Shortest Path from source:

H -> B -> A = 4

H -> B = 1

H -> C = 2

H -> C -> D = 3

H -> I -> E = 3

H -> F = 3

H -> G = 3

H -> I = 1

Total time taken = 25800 ns

Implement Dijkstra's algorithm for shortest path for Directed graphs.

Case 1 input

8 15 D

A B 2

A C 6

A D 7

B E 5

B F 12

B C 3

C D 2

C G 3

D C 1

D G 2

E C 3

F E 1

G F 5

H G 4

H D 2

A

Output

Number of vertices = 8

Number of edges = 15

Selected Graph is Directed.

Applying Dijkstra's Algorithm:

	A	B	C	D	E	F	G	H
A	0	2	6	7	0	0	0	0
B	0	0	3	0	5	12	0	0
C	0	0	0	2	0	0	3	0
D	0	0	1	0	0	0	2	0
E	0	0	3	0	0	0	0	0
F	0	0	0	0	1	0	0	0
G	0	0	0	0	0	5	0	0
H	0	0	0	2	0	0	4	0

Source of graph is A

Shortest Path from source:

A -> B = 2

A -> B -> C = 5

A -> D = 7

A -> B -> E = 7

A -> B -> C -> G -> F = 13

A -> B -> C -> G = 8

A -> H = 2147483647

Total time taken = 19600 ns

Case 2 Input

9 15 D

A B 2

A E 10

A G 45

B C 3

B F 25

B H 45

C D 4

C I 45

D E 5

D H 39

E F 6

F G 5

G H 4

H I 3

I A 2

A

Output

Number of vertices = 9

Number of edges = 15

Selected Graph is Directed.

Applying Dijkstra's Algorithm:

	A	B	C	D	E	F	G	H	I
A	0	2	0	0	10	0	45	0	0
B	0	0	3	0	0	25	0	45	0
C	0	0	0	4	0	0	0	0	45
D	0	0	0	0	5	0	0	39	0
E	0	0	0	0	0	6	0	0	0
F	0	0	0	0	0	0	5	0	0
G	0	0	0	0	0	0	0	4	0
H	0	0	0	0	0	0	0	0	3
I	2	0	0	0	0	0	0	0	0

Source of graph is A

Shortest Path from source:

A -> B = 2

A -> B -> C = 5

A -> B -> C -> D = 9

A -> E = 10

A -> E -> F = 16

A -> E -> F -> G = 21

A -> E -> F -> G -> H = 25

A -> E -> F -> G -> H -> I = 28

Total time taken = 21900 ns

Implement kruskal's algorithm for Minimum Spanning Trees.

Case 1 Input

9 14 U

A B 3

A H 7

B C 8

B H 12

C D 8

C F 5

C I 3

D E 8

D F 13

E F 12

F G 3

G I 5

G H 1

H I 6

C

Number of vertices = 9

Number of edges = 14

Applying Kruskal's Algorithm:

A -> B = 3

A -> H = 7

B -> C = 8

B -> H = 12

C -> D = 8

C -> F = 5

C -> I = 3

D -> E = 8

D -> F = 13

E -> F = 12

F -> G = 3

G -> I = 5

G -> H = 1

H -> I = 6

Minimum Spanning Tree:

G -> H = 1

A -> B = 3

C -> I = 3

F -> G = 3

C -> F = 5

A -> H = 7

C -> D = 8

D -> E = 8

Total Cost = 38

Total time taken = 1148500 ns

Case 2 Input

9 14 U

A B 4

A H 8

B C 8

B H 11

C D 7

C F 4

C I 2

D E 9

D F 14

E F 10

F G 2

G I 6

G H 1

H I 7

C

Number of vertices = 9

Number of edges = 14

Applying Kruskal's Algorithm:

A -> B = 4

A -> H = 8

B -> C = 8

B -> H = 11

C -> D = 7

C -> F = 4

C -> I = 2

D -> E = 9

D -> F = 14

E -> F = 10

F -> G = 2

G -> I = 6

G -> H = 1

H -> I = 7

Minimum Spanning Tree:

G -> H = 1

C -> I = 2

F -> G = 2

A -> B = 4

C -> F = 4

C -> D = 7

A -> H = 8

D -> E = 9

Total Cost = 37

Total time taken = 991900 ns

Case 3 Input

9 17 U

A B 3

A G 5

A H 7

B C 7

B H 1

C D 1

C H 2

C I 2

D E 5

D I 3

E I 2

E F 4

F G 2

F H 3

F I 3

G H 3

H I 1

H

Output

Number of vertices = 9

Number of edges = 17

Applying Kruskal's Algorithm:

A -> B = 3

A -> G = 5

A -> H = 7

B -> C = 7

B -> H = 1

C -> D = 1

C -> H = 2

C -> I = 2

D -> E = 5

D -> I = 3

E -> I = 2

E -> F = 4

F -> G = 2

F -> H = 3

F -> I = 3

G -> H = 3

H -> I = 1

Minimum Spanning Tree:

B -> H = 1

C -> D = 1

H -> I = 1

C -> H = 2

E -> I = 2

F -> G = 2

A -> B = 3

F -> H = 3

Total Cost = 15

Total time taken = 1035800 ns

Case 4 Input

9 14 U

A B 6

A H 9

B C 10

B H 12

C D 8

C F 6

C I 3

D E 7

D F 12

E F 9

F G 3

G I 6

G H 3

H I 5

C

Number of vertices = 9

Number of edges = 14

Applying Kruskal's Algorithm:

A -> B = 6

A -> H = 9

B -> C = 10

B -> H = 12

C -> D = 8

C -> F = 6

C -> I = 3

D -> E = 7

D -> F = 12

E -> F = 9

F -> G = 3

G -> I = 6

G -> H = 3

H -> I = 5

Minimum Spanning Tree:

C -> I = 3

F -> G = 3

G -> H = 3

H -> I = 5

A -> B = 6

D -> E = 7

C -> D = 8

A -> H = 9

Total Cost = 44

Total time taken = 969300 ns