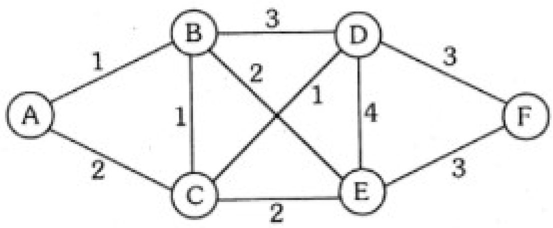
DATA STRUCTURES AND ALGORITHMS PROJECT-2

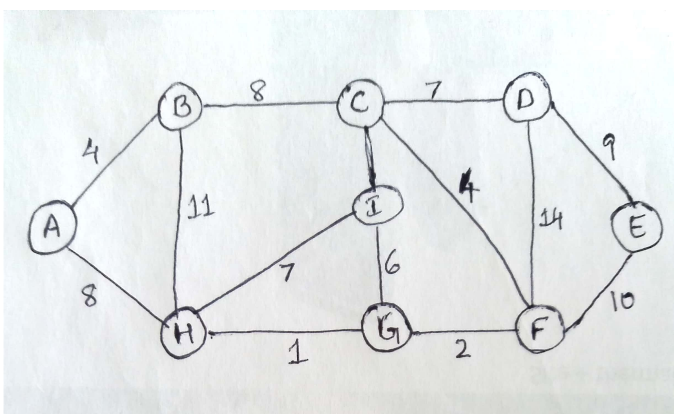
Submitted by - Nirja Sanjay Ayare and Manasi Prabhune

The following two algorithms are implemented in this project: Dijkstra's Algorithm and Kruskal's Algorithm.
The input was taken from an input text file which contains number of edges, vertices, the neighbouring vertices and the respective weights.
Three input graphs were constructed and the program was run on three different input files.
The following three graphs were taken:

First Graph:



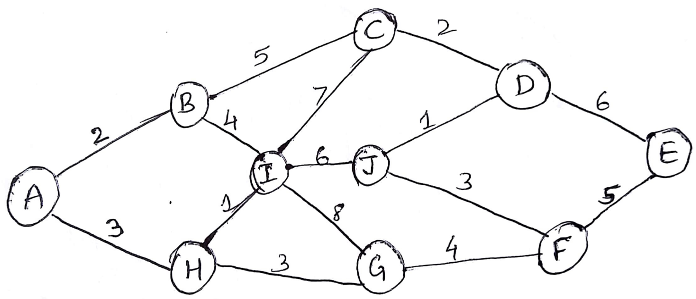
Second Graph:



Third Graph:

DATA STRUCTURES AND ALGORITHMS PROJECT-2

Submitted by - Nirja Sanjay Ayare and Manasi Prabhune



Problem 1:
Find the shortest path in undirected graphs for a given source node. You will print out each path and the corresponding path cost after execution.

For the shortest path algorithm we are using Dijkstra's algorithm for the implementation. We create an input file which contains number of edges, number of vertices, details of the connectivity of nodes and their weights. In the code, a function is written which reads the input from the text file in the format mentioned and then determines the shortest path from the given and source node and then displays the shortest paths and the respective cost. The time taken for the execution of each graph is also shown.

The output of the first graph is shown below:

```
Vertex          Distance              Path
A -> B          1                     B
A -> C          2                     C
A -> D          3                     C D
A -> E          3                     B E
A -> F          6                     C D F

The time taken for the execution of Dijkstra's Algorithm is: 16301 nanoseconds
```

The source node is A and the respective path's and distance to the path are shown. The time taken for execution is also displayed.

The output of the second graph is:

Vertex	Distance	Path
B -> A	4	A
B -> C	8	C
B -> D	15	C D
B -> E	22	C F E
B -> F	12	C F
B -> G	12	H G
B -> H	11	H
B -> I	10	C I

```
The time taken for the execution of Dijkstra's Algorithm is: 32602 nanoseconds
```

The source node considered here is B and the shortest distance(cost) from B to other nodes is calculated and the path taken is also shown. The time taken for the execution of the algorithm is displayed.

The output of the third graph is:

Vertex	Distance	Path
D -> A	9	C B A
D -> B	7	C B
D -> C	2	C
D -> E	6	E
D -> F	4	J F
D -> G	8	J F G
D -> H	8	J I H
D -> I	7	J I
D -> J	1	J

```
The time taken for the execution of Dijkstra's Algorithm is: 22942 nanoseconds
```

The source node considered here is D and the shortest distance(cost) from D to other nodes is calculated and the path taken is also shown.

The code calculates the shortest path and the cost (distance) required to take that path. Here a parent array is created which is updated regularly with the path and distances.
It works for both directed and undirected graphs. The code finds the shortest distance from source to all its vertices.

Two for loops are used in the Dijkstra() function. Once the graph is complete, the time complexity of the algorithm is of $O(n^2)$.

Problem 2:
Given a connected, undirected, weighted graph, find a spanning tree using edges that minimizes the total weight $w(T) = \sum_{(u,\ v)\in T} w(u,v)$. Use Kruskal or Prims algorithm to find the Minimum Spanning Tree (MST). You will print out the edges of the tree and total cost of your answer.

The output for the first graph is:

```
Given graph:
Source   Destination     Weight
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
Following are the edges in the constructed MST
Source   Destination     Weight
A               B        1
B               C        1
C               D        1
B               E        2
D               F        3
The minimum cost of Kruskal's Algortihm is: 8

The time taken for execution of Kruskal's Algorithm is 3368305 nanoseconds
```

The output of the second graph is:

```
Given graph:
Source   Destination     Weight
A               B       4
A               H       8
B               C       8
B               H       11
C               I       2
C               D       7
C               F       4
D               E       9
D               F       14
E               F       10
F               G       2
G               I       6
G               H       1
H               I       7
Following are the edges in the constructed MST
Source   Destination     Weight
G              H         1
C              I         2
F              G         2
A              B         4
C              F         4
C              D         7
A              H         8
D              E         9
The minimum cost of Kruskal's Algortihm is: 37

The time taken for execution of Kruskal's Algorithm is 3044093 nanoseconds
```

The output of the third graph is:

```
Given graph:
Source   Destination     Weight
A                B      2
A                H      3
B                C      5
B                I      4
C                I      7
C                D      2
D                E      6
D                J      1
E                F      5
F                J      3
F                G      4
G                I      8
G                H      3
H                I      1
I                J      6
Following are the edges in the constructed MST
Source   Destination     Weight
D               J        1
H               I        1
A               B        2
C               D        2
A               H        3
F               J        3
G               H        3
F               G        4
E               F        5
The minimum cost of Kruskal's Algortihm is: 24

The time taken for execution of Kruskal's Algorithm is 3062809 nanoseconds
```

For Kruskal's algorithm first the edges are sorted in ascending order of their weights.
The smallest edge is then picked and checked if it forms a cycle with the spanning tree formed till now.if not, that edge is included else it's discarded.
This is repeated till V-1 edges are present in the spanning tree.
In this code the Union Find algorithm is used that performs two useful operations:
Find: determining which subset a particular element is in. This is used for determining if two elements are in the same subset.
Union: Join two subsets into a single subset.
Union-Find Algorithm can be used to check whether an undirected graph contains cycle or not.

The time taken for execution of each graph is shown.

Time complexity: O(E log E) or O(E log V) time. After sorting the edges in ascending order, we iterate through all the edges and apply the union-find algorithm. The find and union operations can take maximum of O(log V) time. So overall complexity is O(E log E + E log V) time. The value of E can be atmost O(V2), so O(log V) are O(log E) same. Therefore, overall time complexity is O(E log E) or O(E log V).