**ALGORITHM AND DATA STRUCTURES ITCS 6114**

**PROJECT 2 :**

**Single source shortest path and Minimum spanning tree**

**BY : KOWSHIK PRASAD NAVILUR**

**SRIKANTH NARAYANA MURTHY**

**DIJKSTRA’S ALGORITHM**

Dijkstra's Algorithm solves the single source shortest path problem in **O((E + V)logV)** time, which can be improved to **O(E + VlogV)** when using a heap.

Queue data structure is used in implementation.

ALGORITHM

1. Set the distance to the source to 0 and the distance to the remaining vertices to infinity.
2. Set the **current** vertex to the source.
3. Mark the **current** vertex as visited.
4. For all vertices adjacent to the **current** vertex, set the distance from the source to the **adjacent** vertex equal to the minimum of its present distance and the **sum** of the **weight of the edge** from the current vertex to the adjacent vertex and the distance from the source to the **current** vertex.
5. From the set of **unvisited vertices**, arbitrarily set one as the new **current** vertex, provided that there exists an edge to it such that it is the minimum of all edges from a vertex in the set of **visited vertices** to a vertex in the set of **unvisited vertices**. To reiterate: The new current vertex must be unvisited and have a minimum weight edges from a visited vertex to it. This can be done trivially by looping through all visited vertices and all adjacent unvisited vertices to those visited vertices, keeping the vertex with the minimum weight edge connecting it.
6. Repeat steps 3-5 until all vertices are flagged as visited.

**PRIMS ALGORITHM**

**1)** Create a set mstSet that keeps track of vertices already included in MST.

**2)** Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.

**3)** While mstSet doesn’t include all vertices  
 **a)** Pick a vertex u which is not there in mstSet and has minimum key value.  
 **b)** Include u to mstSet.  
 **c)** Update key value of all adjacent vertices of u.

To update the key values, iterate through all adjacent vertices. For every adjacent vertex v, if weight of edge u-v is less than the previous key value of v, update the key value as weight of u-v

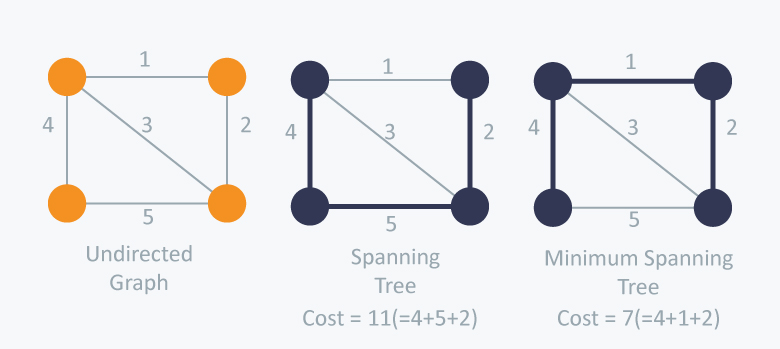
The idea of using key values is to pick the minimum weight edge from cut. The key values are used only for vertices which are not yet included in MST, the key value for these vertices indicate the minimum weight edges connecting them to the set of vertices included in MST.

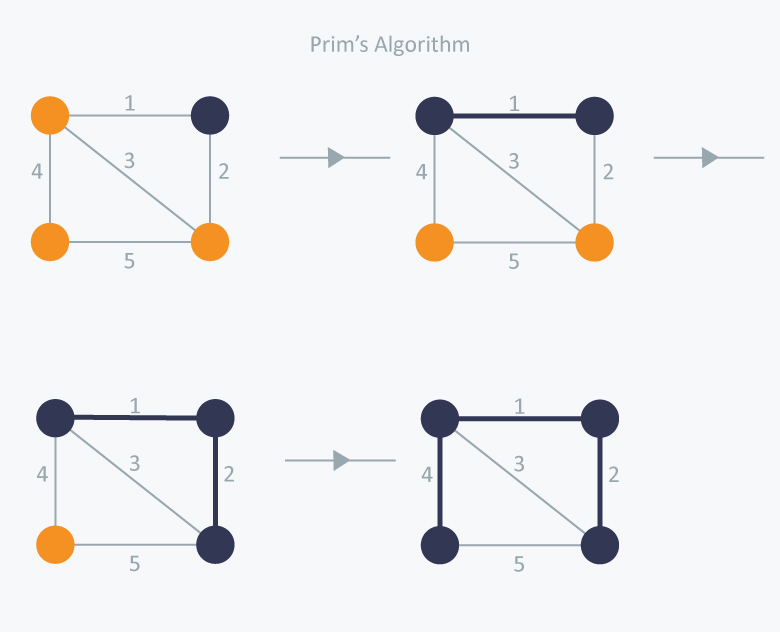
Ideally, the time complexity of the Prim’s Algorithm is O((V+E)logV).

However, In this implementation the time complexity is *O*(|V|2) as we are using adjacency matrix where V is the number of vertices in graph.

First V represents every vertex in the graph that is being looped through in the while loop and the second V represents every other vertex that the first vertex may be connected to in the graph via an edge.

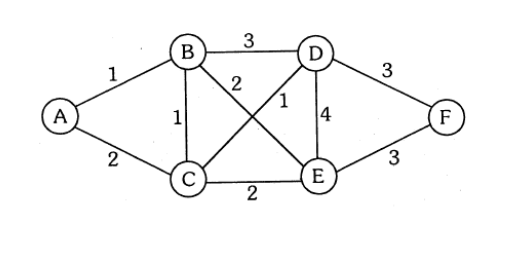
PRIMS EXAMPLE



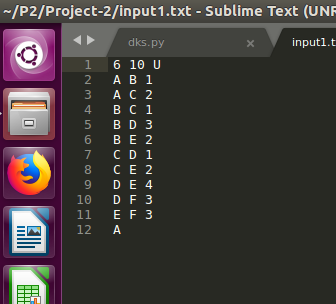


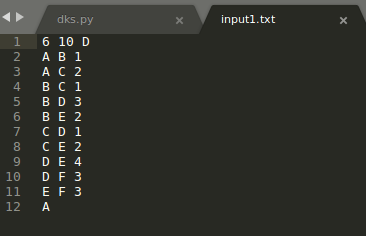
**OUTPUTS**

INPUT GRAPH



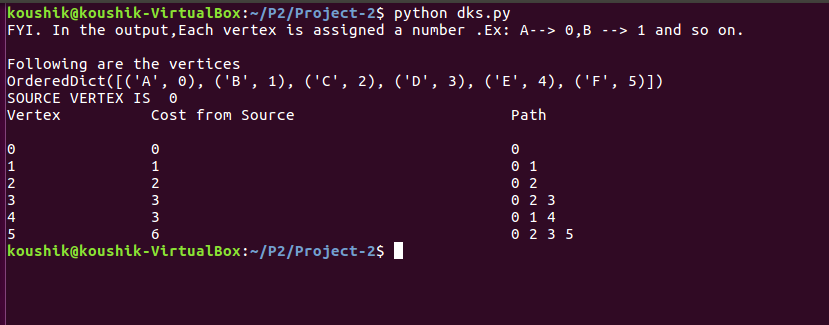
INPUT FILE



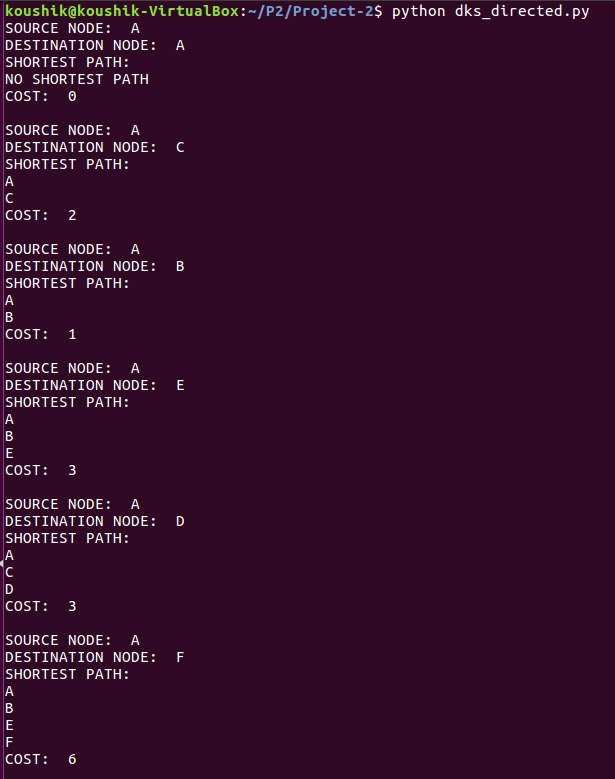


OUTPUT

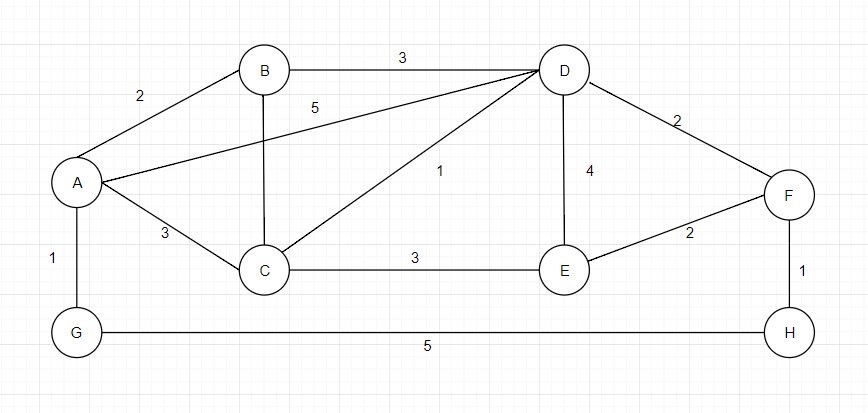
Undirected Graph Output



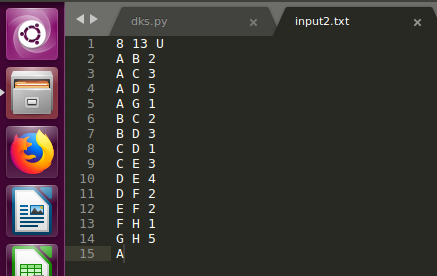
Directed Graph Output

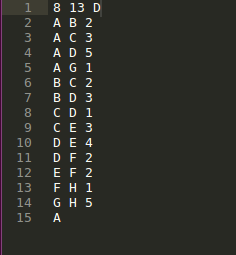


**Example 2**



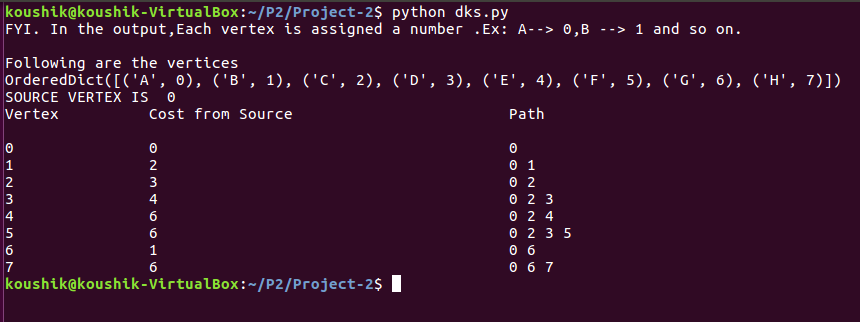
INPUT



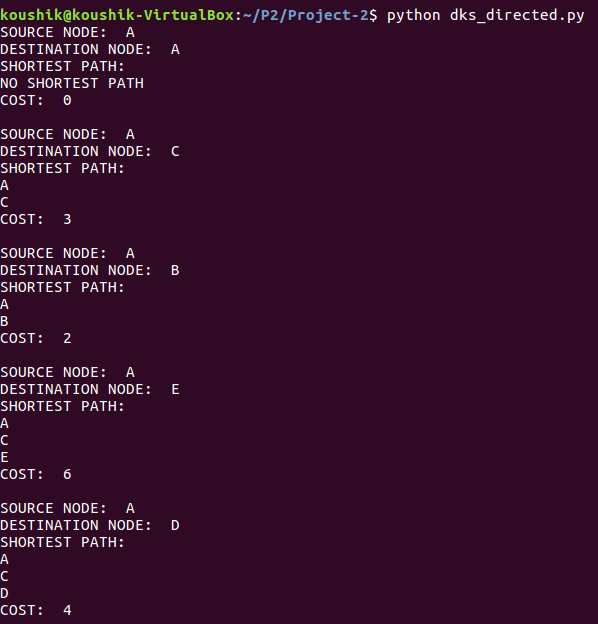


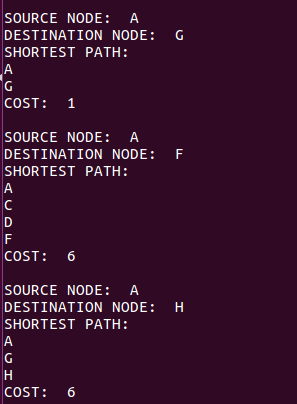
OUTPUT

Undirected Graph Output



Directed Graph Output

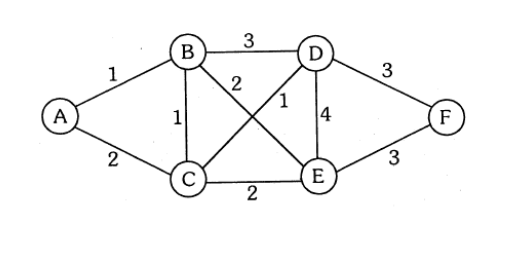




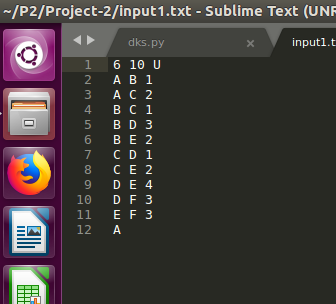
PRIMS ALGORITHM

Example 1

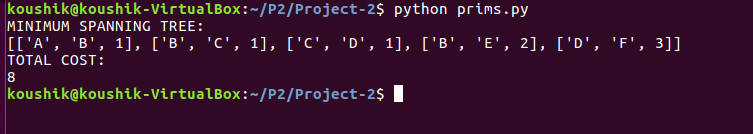
Input Graph



INPUT FILE

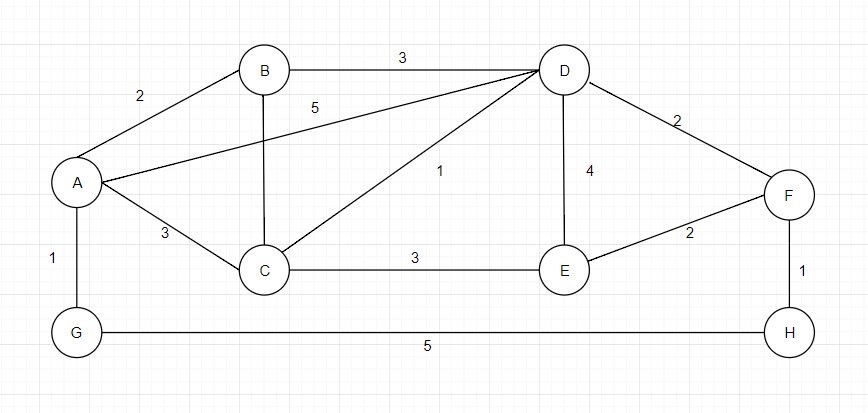


OUTPUT

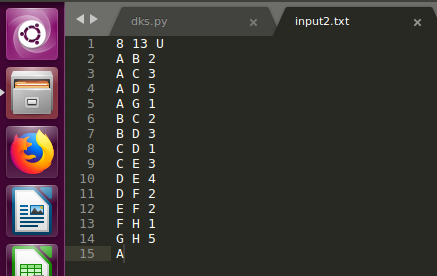


Example 2

Input Graph



Input File



Output

