Name: Vivek Vikram Pundkar

Roll no: 77

GR no: 11910860

Division: C

Batch: 3

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Prim's Algorithm**

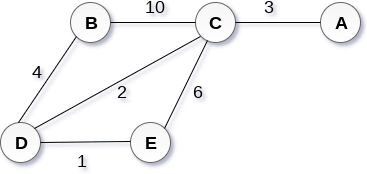
Prim's Algorithm is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

Prim's algorithm starts with the single node and explore all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

The algorithm is given as follows.

**Example:**

Construct a minimum spanning tree of the graph given in the following figure by using prim's algorithm.



Solution

Step 1 : Choose a starting vertex B.

Step 2: Add the vertices that are adjacent to A. the edges that connecting the vertices are shown by dotted lines.

Step 3: Choose the edge with the minimum weight among all. i.e. BD and add it to MST. Add the adjacent vertices of D

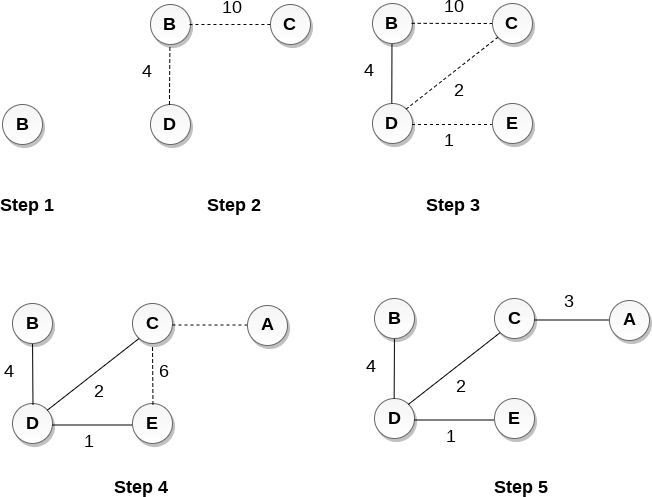
i.e. C and E.

Step 3: Choose the edge with the minimum weight among all. In this case, the edges DE and CD are such edges. Add them to MST and explore the adjacent of C i.e. E and A.

Step 4: Choose the edge with the minimum weight i.e. CA. We can't choose CE as it would cause cycle in the graph.

The graph produces in the step 4 is the minimum spanning tree of the graph shown in the above figure. The cost of MST will be calculated as;

cost(MST) = 4 + 2 + 1 + 3 = 10 units.



Algorithm

Step 1: Select a starting vertex

Step 2: Repeat Steps 3 and 4 until there are fringe vertices

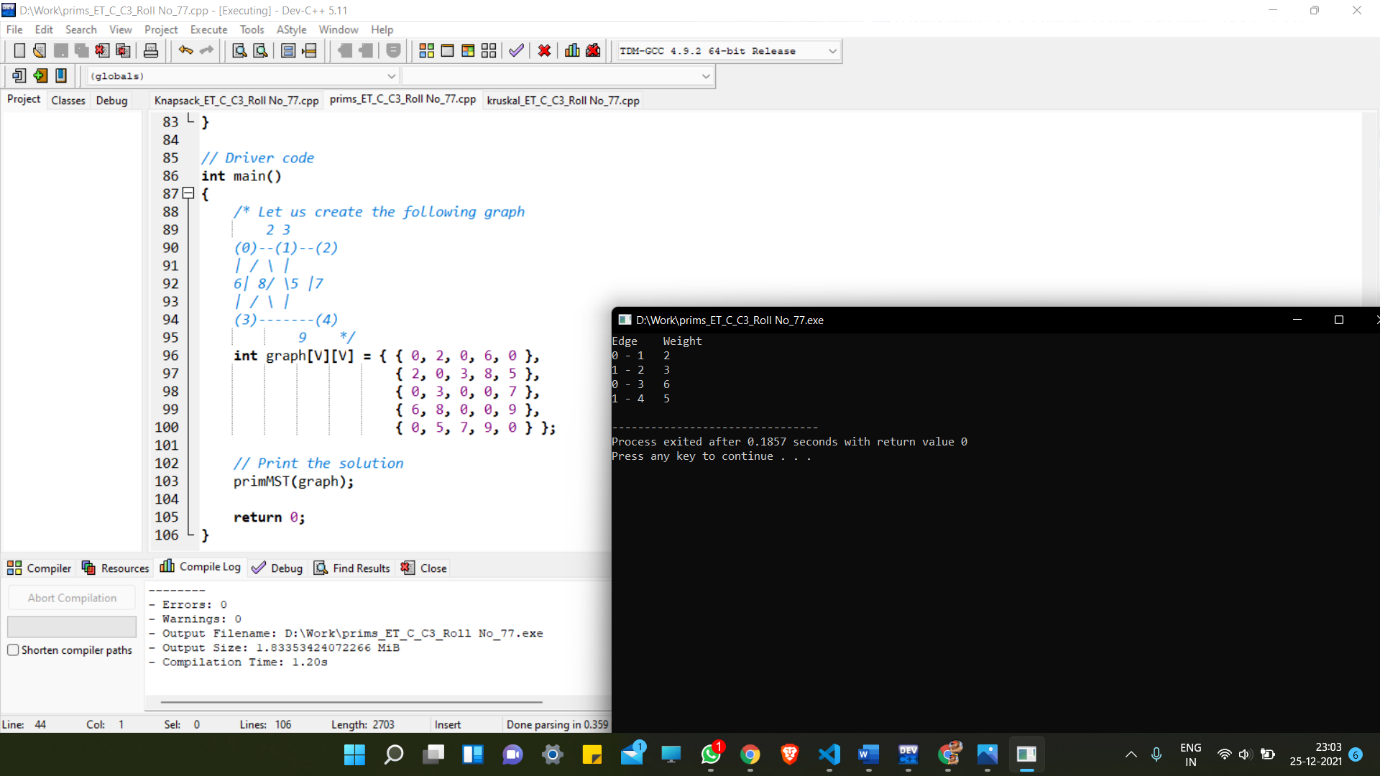
Step 3: Select an edge e connecting the tree vertex and fringe vertex that has minimum weight

Step 4: Add the selected edge and the vertex to the minimum spanning tree T [END OF LOOP]

Step 5: EXIT

**Time Complexity** of the above program is O(V^2). If the input graph is represented using adjacency list, then the time complexity of Prim’s algorithm can be reduced to O(E log V) with the help of binary heap.

**OUTPUT:**

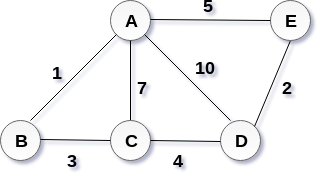


\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Kruskal's Algorithm**

Kruskal's Algorithm is used to find the minimum spanning tree for a connected weighted graph. The main target of the algorithm is to find the subset of edges by using which, we can traverse every vertex of the graph. Kruskal's algorithm follows greedy approach which finds an optimum solution at every stage instead of focusing on a global optimum.

**Example:**

Apply the Kruskal's algorithm on the graph given as follows.

Solution:

the weight of the edges given as :

Edge AE AD AC AB BC CD DE

Weight 5 10 7 1 3 4 2

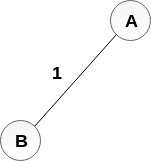
Sort the edges according to their weights.

Edge AB DE BC CD AE AC AD

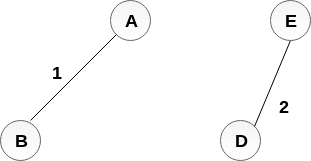
Weight 1 2 3 4 5 7 10

Start constructing the tree;

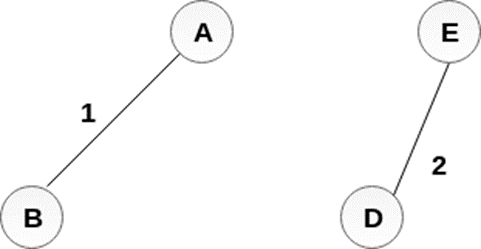
Add AB to the MST;

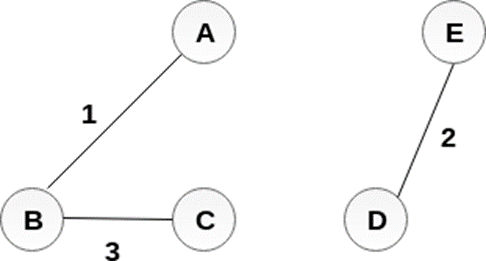


Add DE to the MST;



Add BC to the MST;





The next step is to add AE, but we can't add that as it will cause a cycle.

The next edge to be added is AC, but it can't be added as it will cause a cycle. The next edge to be added is AD, but it can't be added as it will contain a cycle. Hence, the final MST is the one which is shown in the step 4.

the cost of MST = 1 + 2 + 3 + 4 = 10.

The Kruskal's algorithm is given as follows.

Algorithm

Step 1: Create a forest in such a way that each graph is a separate tree.

Step 2: Create a priority queue Q that contains all the edges of the graph.

Step 3: Repeat Steps 4 and 5 while Q is NOT EMPTY

Step 4: Remove an edge from Q

Step 5:

IF the edge obtained in Step 4 connects two different trees, then Add it to the forest (for combining two trees into one tree).

ELSE

Discard the edge

Step 6: END

**Time Complexity:**

O(ElogE) or O(ElogV). Sorting of edges takes O(ELogE) time. After sorting, we iterate through all edges and apply the find-union algorithm. The find and union operations can take at most O(LogV) time. So overall complexity is O(ELogE + ELogV) time. The value of E can be at most O(V2), so O(LogV) is O(LogE) the same. Therefore, the overall time complexity is O(ElogE) or O(ElogV). Krushkals

**OUTPUT:**

