



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Name:	RITESH SHETTY
Roll No:	54
Class/Sem:	SE/IV
Experiment No.:	6
Title:	Prim's Algorithm.
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 6

Title: Prim's Algorithm.

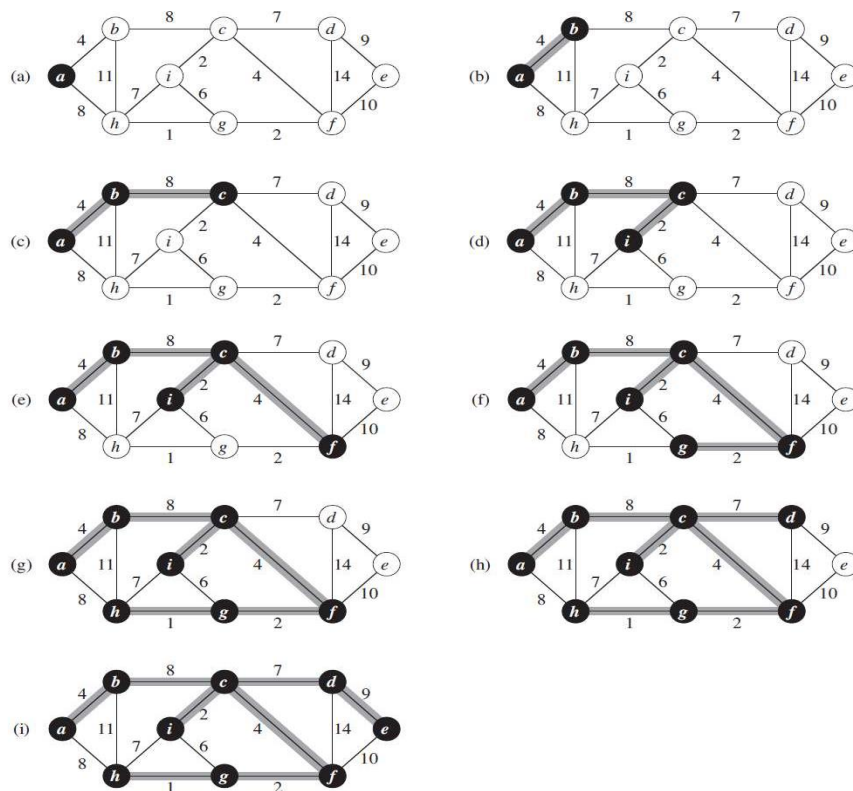
Aim: To study and implement Prim's Minimum Cost Spanning Tree Algorithm.

Objective: To introduce Greedy based algorithms

Theory:

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. 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. The algorithm operates by building this tree one vertex at a time, from an arbitrary starting vertex, at each step adding the cheapest possible connection from the tree to another vertex.

Example:





Algorithm and Complexity:

```
1  Algorithm Prim( $E, cost, n, t$ )
2  //  $E$  is the set of edges in  $G$ .  $cost[1 : n, 1 : n]$  is the cost
3  // adjacency matrix of an  $n$  vertex graph such that  $cost[i, j]$  is
4  // either a positive real number or  $\infty$  if no edge  $(i, j)$  exists.
5  // A minimum spanning tree is computed and stored as a set of
6  // edges in the array  $t[1 : n - 1, 1 : 2]$ .  $(t[i, 1], t[i, 2])$  is an edge in
7  // the minimum-cost spanning tree. The final cost is returned.
8  {
9      Let  $(k, l)$  be an edge of minimum cost in  $E$ ;
10      $mincost := cost[k, l]$ ;
11      $t[1, 1] := k$ ;  $t[1, 2] := l$ ;
12     for  $i := 1$  to  $n$  do // Initialize near.
13         if  $(cost[i, l] < cost[i, k])$  then  $near[i] := l$ ;
14         else  $near[i] := k$ ;
15      $near[k] := near[l] := 0$ ;
16     for  $i := 2$  to  $n - 1$  do
17     { // Find  $n - 2$  additional edges for  $t$ .
18         Let  $j$  be an index such that  $near[j] \neq 0$  and
19          $cost[j, near[j]]$  is minimum;
20          $t[i, 1] := j$ ;  $t[i, 2] := near[j]$ ;
21          $mincost := mincost + cost[j, near[j]]$ ;
22          $near[j] := 0$ ;
23         for  $k := 1$  to  $n$  do // Update  $near[ ]$ .
24             if  $((near[k] \neq 0) \text{ and } (cost[k, near[k]] > cost[k, j]))$ 
25                 then  $near[k] := j$ ;
26     }
27     return  $mincost$ ;
28 }
```

Time Complexity is $O(n^2)$, Where, n = number of vertices **Theory:**

Implementation:

```
#include <limits.h>
#include <stdbool.h>
#include <stdio.h>
#define V 5
int minKey(int key[], bool mstSet[])
```



```
{
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (mstSet[v] == false && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}

int printMST(int parent[], int graph[V][V])
{
    printf("Edge \tWeight\n");
    for (int i = 1; i < V; i++)
        printf("%d - %d \t%d \n", parent[i], i,
            graph[i][parent[i]]);
}

void primMST(int graph[V][V])
{
    int parent[V];
    int key[V];
    bool mstSet[V];

    for (int i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = false;

    key[0] = 0;

    parent[0] = -1;

    for (int count = 0; count < V - 1; count++) {
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
int u = minKey(key, mstSet);
mstSet[u] = true;
for (int v = 0; v < V; v++)
    if (graph[u][v] && mstSet[v] == false
        && graph[u][v] < key[v])
        parent[v] = u, key[v] = graph[u][v];
}
printMST(parent, graph);
}
int main()
{
    int graph[V][V] = { { 0, 2, 0, 6, 0 },
                        { 2, 0, 3, 8, 5 },
                        { 0, 3, 0, 0, 7 },
                        { 6, 8, 0, 0, 9 },
                        { 0, 5, 7, 9, 0 } };

    primMST(graph);

    return 0;
}
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Output:

Edge	Weight
0 - 1	2
1 - 2	3
0 - 3	6
1 - 4	5

Conclusion: Implementing Prim's algorithm has proven to be effective in generating minimum spanning trees, efficiently connecting all nodes in a graph while minimizing total edge weight. This experiment underscores the algorithm's practical applicability in optimizing network connectivity, demonstrating its importance in various real-world scenarios.