MINI PROJECT USING C LANGUAGE :

PRIMS ALGORITHM

Jawaharlal Nehru University
School of Computer and Systems Sciences

Submitted By : Shreya Pal
Enrollment no. : 22/10/JC/032

Submitted To : Dr. Piyush Pratap Singh

Topics Discussed :

- History of Prims Algorithm
- Basic Concepts:

  a. Spanning Trees

  b. Weighted Graphs

  c. Minimum Spanning Tree

  d. Algorithms for Minimum Spanning Tree

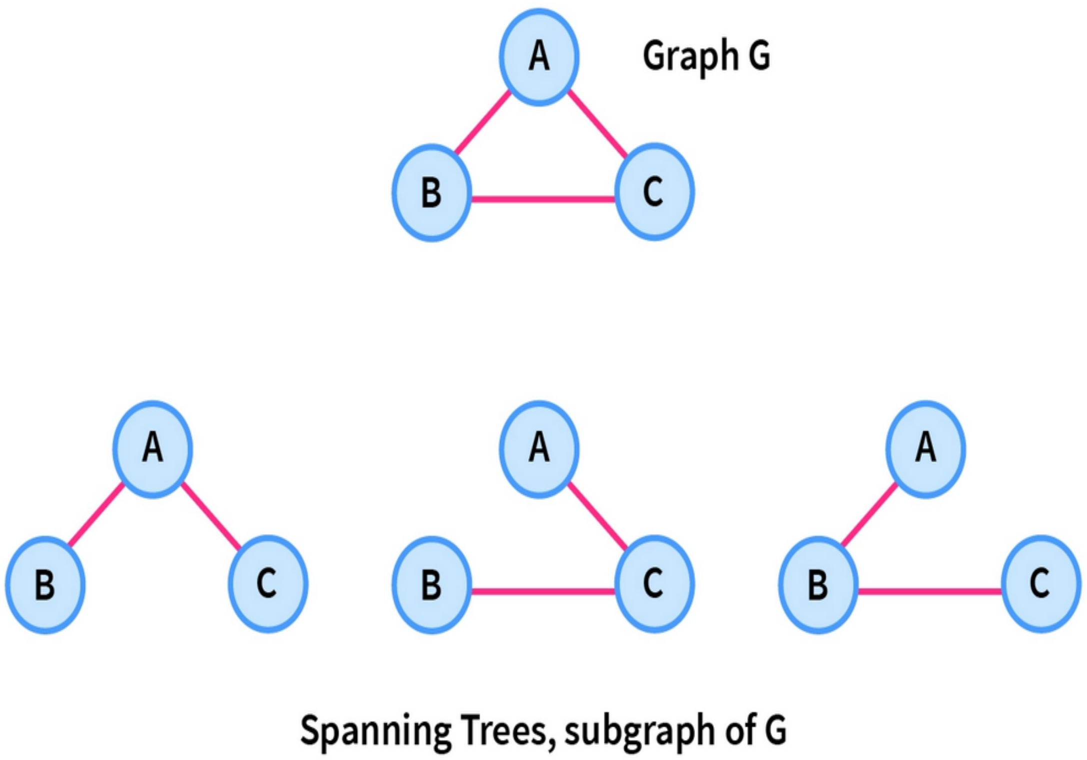
  e. Applications of Minimum Spanning Tree
- Prims Algorithm
- Data Structure of Prims Algorithm
- Code
- Output

History of Prims Algorithm :

- Prim's algorithm is a greedy algorithm(a greedy algorithm is a procedure that makes an optimal choice at each of its steps) that finds a minimum spanning tree for a weighted undirected graph.
- Developed in 1930 by Czech Mathematician Vojtěch Jarník and later rediscovered and republished by computer scientist Robert C. Prim in 1957 and E.W. Dijkstra in 1959.
- It is sometimes also called as the Prim-Jarn´ık algorithm.

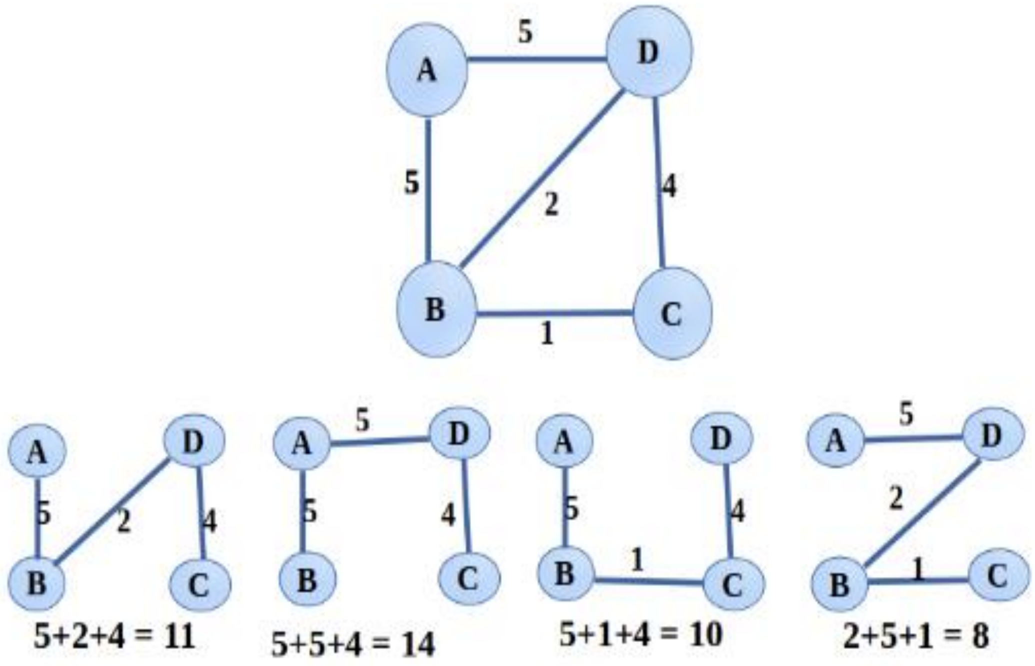
Basic Concepts :

- Spanning Tree : Let G=(V,E) be a simple undirected graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G.



Basic Concepts Continued:

- <u>Weighted Graph :</u> A weighted graph is a graph , in which edge has a weight (some real number).
- <u>Minimum Spanning Tree :</u> A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.



Spanning Tree

Basic Concepts Continued :

- <u>Algorithm for Minimum Spanning Tree :</u>

  The two widely used famous algorithms are :

  a. Prim’s algorithm

  b. Kruskal’s algorithm

- <u>Applications of Minimum Spanning :</u>

  a. Design of a Network (Telephone Network, Computer Network, Electronic Circuitry, etc.)

  b. Travelling Salesman Problem

  c. Airline Routes

  d. Study of Molecular Bonds in Chemistry

  e. Cartography

Prims Algorithm :

- **procedure** *Prim*(*G*: weighted connected undirected graph with n vertices)
- *T* := a minimum-weight edge
- **for** *i* := 1 **to** n − 2
- *e* := an edge of minimum weight incident to a vertex in T and
  not forming a simple circuit in T if added to T
- *T* := *T* with *e* added
- **return** *T* {*T* is a minimum spanning tree of *G*}

Data Structure of Prims Algorithm :
(for an Example problem)

- <u>Weighted Adjacency Matrix :</u>

	0	1	2	3	4	5	6	7
0	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	25	∞	∞	∞	5	∞
2	∞	25	∞	12	∞	∞	∞	10
3	∞	∞	12	∞	8	∞	∞	∞
4	∞	∞	∞	8	∞	16	∞	14
5	∞	∞	∞	∞	16	∞	20	18
6	∞	5	∞	∞	∞	20	∞	∞
7	∞	∞	10	∞	14	18	∞	∞

- The Prims Algorithm begins by choosing any edge with smallest weight, putting it into the spanning tree.
- T:

0	1					
1	6					
	0	1	2	3	4	5

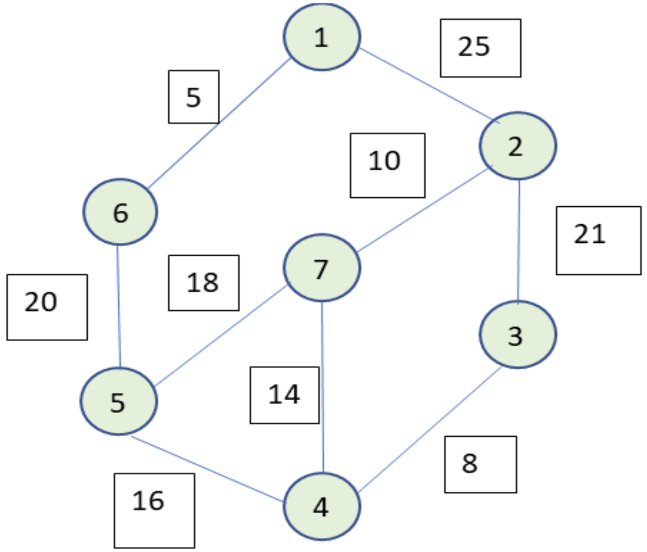
- Initial Steps:
- Track :

∞	∞	∞	∞	∞	∞	∞	∞
0	1	2	3	4	5	6	7

∞	0	∞	∞	∞	∞	0	∞
0	1	2	3	4	5	6	7

∞	0	1	6	6	6	0	6
0	1	2	3	4	5	6	7

- **Weighted Graph:**



- **Repeating Steps:**
- 1st Complete Repetition:

0	1	5				
1	6	6				
	0	1	2	3	4	5

- Track:

∞	0	1	6	6	0	0	6
0	1	2	3	4	5	6	7

∞	0	1	6	5	0	0	5
0	1	2	3	4	5	6	7

- 2nd Complete Repetition:

0	1	5	4			
1	6	6	5			
	0	1	2	3	4	5

- Track:

∞	0	1	4	0	0	0	4
0	1	2	3	4	5	6	7

- 3rd Complete Repetition :

0	1	5	4	3		
1	6	6	5	4		
	0	1	2	3	4	5

- Track:

∞	0	3	0	0	0	0	4
0	1	2	3	4	5	6	7

- 4th Complete Repetition:

0	1	5	4	3	2	
1	6	6	5	4	3	
	0	1	2	3	4	5

- Track:

∞	0	0	0	0	0	0	2
0	1	2	3	4	5	6	7

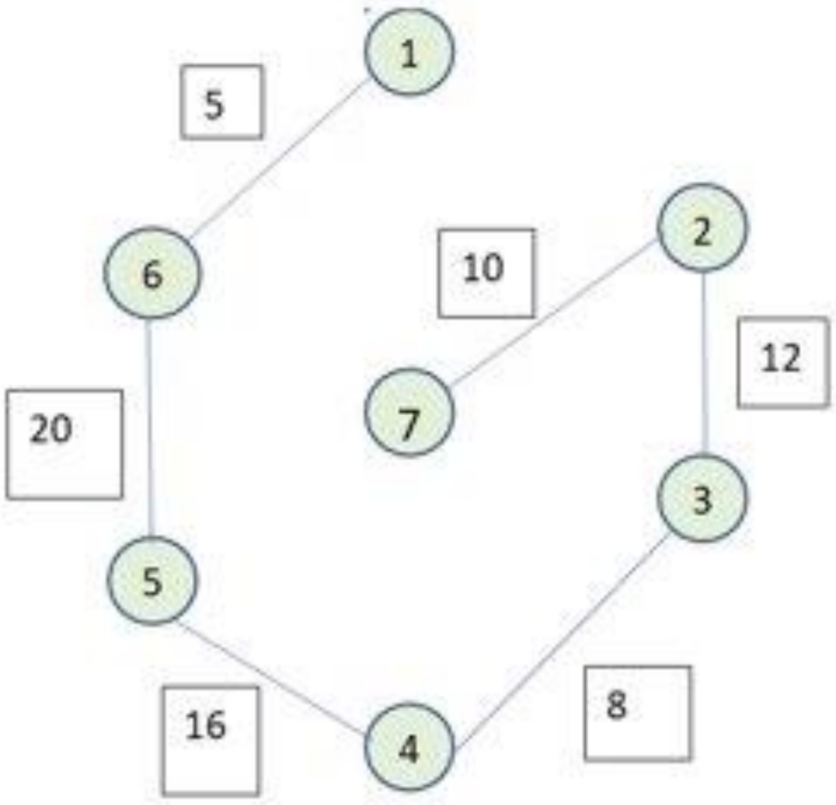
- Last Step:

0	1	5	4	3	2	7
1	6	6	5	4	3	2
	0	1	2	3	4	5

- Track:

∞	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7

- Procedure Ends and we obtain the following Minimum Spanning Tree:



Code :

```c
#include <stdio.h>
#define V 8
#define I 32767

void PrintMST(int T[][V-2], int G[V][V]){
    printf("\nMinimum Spanning Tree Edges (w/ cost)\n");
    int sum=0;
    for (int i =0; i<V-2; i++){
        int c = G[T[0][i]][T[1][i]];
        printf( "[ %d ]---[ %d] cost:%d ",T[0][i] ,T[1][i], c );
      printf("\n");
        sum += c;
    }
    printf("\n");
    printf("Total cost of Minimum Spanning Tree:%d \n ",sum);
}

void PrimsMST(int G[V][V], int n){
    int u;
    int v;
    int min =I;
    int track [V];
    int T[2][V-2] = {0};

```

```
// Initial: Find min cost edge
for (int i =1; i<V; i++){
    track[i] = I;  // Initialize track array with INFINITY
    for (int j =i; j<V; j++){
        if (G[i][j] < min){
            min = G[i][j];
            u = i;
            v = j;
        }
    }
}
T[0][0] = u;
T[1][0] = v;
track[u] = track[v] = 0;

// Initialize track array to track min cost edges
for (int i =1; i<V; i++){
    if (track[i] != 0){
        if (G[i][u] < G[i][v]){
            track[i] = u;
        } else {
            track[i] = v;
```

```
        // Repeat
        for (int i=1; i<n-1; i++){
            int k;
            min = I;
            for (int j=1; j<V; j++){
                if (track[j] != 0 && G[j][track[j]] < min){
                    k = j;
                    min = G[j][track[j]];
                }
            }
            T[0][i] = k;
            T[1][i] = track[k];
            track[k] = 0;

            // Update track array to track min cost edges
            for (int j=1; j<V; j++){
                if (track[j] != 0 && G[j][k] < G[j][track[j]]){
                    track[j] = k;
                }
            }
        }
        PrintMST(T, G);
    }
```

```
int main() {

    int cost [V][V]= {
                {I, I, I, I, I, I, I, I},
                {I, I, 25, I, I, I, 5, I},
                {I, 25, I, 12, I, I, I, 10},
                {I, I, 12, I, 8, I, I, I},
                {I, I, I, 8, I, 16, I, 14},
                {I, I, I, I, 16, I, 20, 18},
                {I, 5, I, I, I, 20, I, I},
                {I, I, 10, I, 14, 18, I, I},
    };

    int n = sizeof(cost[0])/sizeof(cost[0][0]) - 1;

    PrimsMST(cost, n);

    return 0;
}
```

Output:

```
> make -s
> ./main

Minimum Spanning Tree Edges (w/ cost)
[ 1 ]---[ 6] cost:5
[ 5 ]---[ 6] cost:20
[ 4 ]---[ 5] cost:16
[ 3 ]---[ 4] cost:8
[ 2 ]---[ 3] cost:12
[ 7 ]---[ 2] cost:10

Total cost of Minimum Spanning Tree:71
 >
```

Thank You !