Assignment 11

Sayak Ghorai || BT21GCS004 || B2 || Design and Analysis of Algorithm

Q: Write a code to demonstrate Prim's Algorithm to find shortest path from source node to any other node.

Answer:

Approach: first make a graph with certain Edges and Vertecies and then add edges between Vertecies. That creates a Graph Structure. Now we can easily apply Prim's Algo in this structure to get a connected graph with minimum edge cost(Minimum Spanning Tree)

Code:

Main.java

```java
import java.lang.*;
2 usages
class MinSpanTree{
    8 usages
    private static final int V = 5;//No of vertices
    1 usage
    int minKey(int[] key, Boolean[] mstSet){
        int min = Integer.MAX_VALUE, min_index = -1;
        for (int v = 0; v < V; v++)
            if (!mstSet[v] && key[v] < min){
                min = key[v];
                min_index = v;
            }
        return min_index;
    }
    1 usage
    void printMinSpanTree(int[] parent, int[][] graph)
    {
        System.out.println("Edge \tWeight");
        for (int i = 1; i < V; i++)
            System.out.println(parent[i]+" - "+i+"\t"+graph[i][parent[i]]);
    }
    1 usage
    void primMST(int[][] graph){
        int[] parent = new int[V];
        int[] key = new int[V];
        Boolean[] mstSet = new Boolean[V];
        for (int i = 0; i < V; i++){
            key[i] = Integer.MAX_VALUE;
            mstSet[i] = false;
        }
        key[0] = 0;
        parent[0] = -1;
        for (int count = 0; count < V - 1; count++){
            int u = minKey(key, mstSet);
            mstSet[u] = true;
            for (int v = 0; v < V; v++)
                if (graph[u][v] != 0 && !mstSet[v]
                        && graph[u][v] < key[v]) {
                    parent[v] = u;
                    key[v] = graph[u][v];
                }
        }
        printMinSpanTree(parent, graph);
    }

    public static void main(String[] args)
    {
        MinSpanTree t = new MinSpanTree();
        int[][] graph = new int[][]
                {
                { 0, 3, 2, 8, 4 },
                { 3, 0, 3, 5, 8 },
                { 2, 3, 0, 1, 7 },
                { 8, 5, 1, 0, 5 },
                { 4, 8, 7, 5, 0 }
                };
        t.primMST(graph);
    }
}
```

Output:

```
/Users/sayakghorai/Desktop/DAA_Assignments/Assignment_11/Assignment_11/out/production/Assignment_11 MinSpanTree
Edge    Weight
0 - 1   3
0 - 2   2
2 - 3   1
0 - 4   4

Process finished with exit code 0
```

Analysis:

Using Adjacency list to show the graph, time complexity is $O(V^2)$ but using binary heap method, it can be reduced to O(E*logV).
Space complexity is O(V).