Lab Report-07

(Prim's Algorithm)

CSE-2212 (Design and Analysis of Algorithms Lab)

Submitted By:

Name: Eyasir Ahamed
Exam Roll: 413
Class Roll: 15
Registration No: 202004017

Submitted To:

Sharad Hasan
Ex. Lecturer
Dept. of CSE
Sheikh Hasina University, Netrokona

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SHEIKH HASINA UNIVERSITY
NETROKONA, BANGLADESH

#7_Prim's Algorithm

Problem Definition

Given a weighted undirected graph represented as an adjacency list and the number of vertices V, the problem is to find the sum of weights of all edges in the Minimum Spanning Tree (MST).

Formal Statement of Algorithm

- Initialize a priority queue pq to store pairs of integers (weight, node) sorted by weight in ascending order.
- Initialize a vector vis to mark visited nodes.
- Push the pair (0, 0) into the priority queue to start the algorithm from node 0 with weight 0.
- Initialize sum to 0 to store the sum of edge weights.
- While the priority queue is not empty:
    - Pop the top element it from the priority queue.
    - Extract the node node and weight wt from it.
    - If the node is already visited, continue to the next iteration.
    - Mark the node as visited.
    - Add wt to sum.
    - Iterate through all adjacent nodes adjNode of node:

        - If adjNode is not visited, push the pair (edW, adjNode) into the priority queue, where edW is the weight of the edge between node and adjNode.
- Return sum as the sum of all edge weights in the MST.

Complexity Analysis of Algorithm

- Time Complexity:
    - O(E log V), where E is the number of edges and V is the number of vertices.
    - Each edge and vertex can be visited at most once in the worst case, and the priority queue operations take O(log V) time.
- Space Complexity:
    - O(V) for the priority queue and vis vector.
    - O(E) for the adjacency list.
    - Overall space complexity: O(V + E).

Actual Code and Output

```cpp
#include <bits/stdc++.h>
using namespace std;

int spanningTree(int V, vector<vector<int>> adj[]) {
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;

    vector<int> vis(V, 0);
    pq.push({0, 0});
    int sum = 0;

    while (!pq.empty()) {
        auto it = pq.top();
        pq.pop();
        int node = it.second;
        int wt = it.first;

        if (vis[node] == 1) continue;
        vis[node] = 1;
        sum += wt;

        for (auto it : adj[node]) {
            int adjNode = it[0];
            int edW = it[1];
            if (!vis[adjNode]) {
                pq.push({edW, adjNode});
            }
        }
    }
    return sum;
}

int main() {
    int V = 5;
    vector<vector<int>> edges = {{0, 1, 2}, {0, 2, 1}, {1, 2, 1}, {2, 3, 2}, {3, 4, 1}, {4, 2, 2}};
    vector<vector<int>> adj[V];

    for (auto it : edges) {
        vector<int> tmp(2);
        tmp[0] = it[1];
        tmp[1] = it[2];
        adj[it[0]].push_back(tmp);

        tmp[0] = it[0];
        tmp[1] = it[2];
        adj[it[1]].push_back(tmp);
    }

    int sum = spanningTree(V, adj);
    cout << "The sum of all the edge weights: " << sum << endl;

    return 0;
}
```

```
The sum of all the edge weights: 5
[Finished in 1.2s]
```