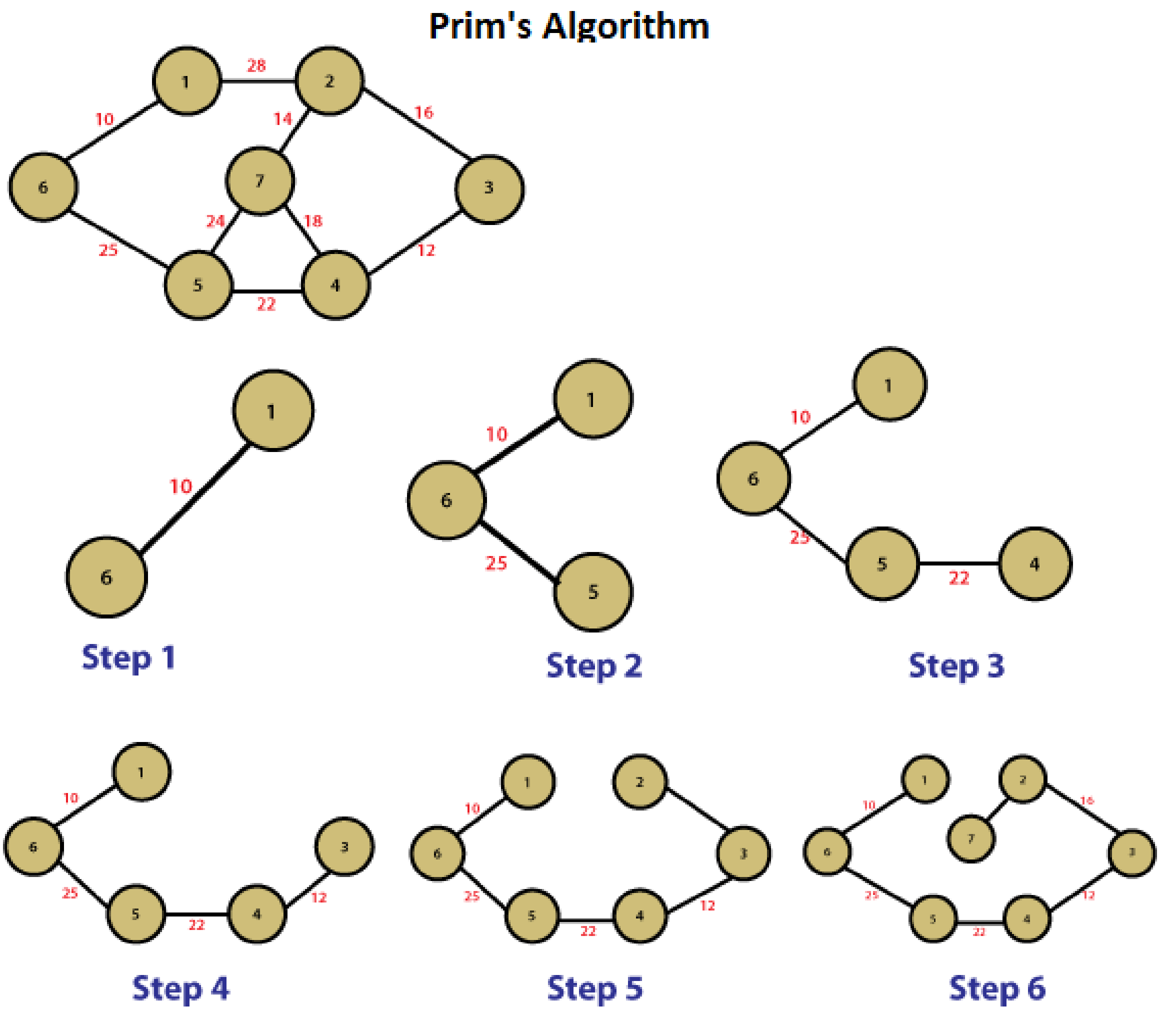
Name: Achyut Shukla Batch: AA – A1 PRN: 20070122005

AA LAB ASSIGNMENT | PRIM'S ALGORITHM

TITLE: Analysis, Proof of Analysis and Implementation of **Prim's MST Algorithm**

MECHANISM

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.
2. Grow the tree by adding the cheapest edge possible that connects the tree to a vertex not yet in the tree.
3. Repeat step 2 until all vertices are in the tree.



ALGORITHM / PSEUDOCODE

Input: Graph represented as a set of vertices V and edges E
Output: Tree T

Function Prim *(G= (V, E)):*
T = tree with a single vertex, arbitrarily chosen from G
vis = {T}
pq = priority queue of edges adjacent to the initial vertex

while *pq* ***is not*** *empty:*
u, v = cheapest edge in pq
if v ***not in*** *vis:*
add (u, v) to T
add v to vis
add all edges adjacent to v to pq

return *T*

IMPLEMENTATION

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

const int INF = 1e9;

int main() {
    int n, m;
    cin >> n >> m;

    vector<pair<int,int>> adj[n];
    for(int i=0; i<m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        u--, v--;
        adj[u].push_back({v, w});
        adj[v].push_back({u, w});
    }

    vector<int> dist(n, INF);
    vector<bool> vis(n, false);
    vector<int> parent(n, -1);
```

```
    priority_queue<pair<int,int>, vector<pair<int,int>>,
    greater<pair<int,int>>> pq;
    pq.push({0, 0});
    dist[0] = 0;

    while(!pq.empty()) {
        int u = pq.top().second;
        pq.pop();

        if(vis[u]) continue;
        vis[u] = true;

        for(auto edge: adj[u]) {
            int v = edge.first, w = edge.second;
            if(!vis[v] && w < dist[v]) {
                dist[v] = w;
                parent[v] = u;
                pq.push({dist[v], v});
            }
        }
    }

    for(int i=1; i<n; i++) {
        cout << i+1 << " " << parent[i]+1 << "\n";
    }

    return 0;
}
```

```
4 5
1 2 1
1 3 3
1 4 4
2 3 2
3 4 5


MST:
2 1
3 2
4 1
PS F:\#3 SIT\6. SEM 6\Advance Algo\Lab>
```

T(n) ANALYSIS WITH PROOF

The time complexity of the Prim's Algorithm is **O((V+E)log(V))** because each edge is inserted in the priority queue only once and insertion in priority queue take logarithmic time.

SPACE COMPLEXITY ANALYSIS

The space complexity of Prim's algorithm is **O(V+E).** This is because we need to store the visited set, the priority queue, and the MST set. The visited set and the priority queue each take O(V) space, and the MST set takes O(E) space.

ADVANTAGES / DISADVANTAGES

ADVANTAGE	DISADVANTAGES
Guaranteed to find the minimum spanning tree	Can be slower than other algorithms for dense graphs
Efficient for sparse graphs	Requires a connected graph
Easy to implement	Not as widely known as other algorithms

REAL LIFE APPLICATIONS:

Prim's algorithm is used in many real-life applications, such as **network design, transportation planning, and resource allocation**. For example, it can be used to design a communication network with minimum cost, or to plan the construction of roads between cities with minimum cost.

OPTIMIZATIONS AND ADVANCEMENTS:

One optimization is to use a Fibonacci heap instead of a binary heap as the priority queue. This can reduce the time complexity to O(E + V log V). Another optimization is to use a data structure called a disjoint-set data structure to keep track of the connected components of the MST. This can reduce the time complexity to O(E log V). Additionally, there are other algorithms for finding the MST, such as Kruskal's algorithm and Boruvka's algorithm, which may be more efficient in some cases.