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

void dijkstras(int source, vector<pair<int, int>> adj[], int n)
{
    priority_queue<pair<int, int>> pq; // priority queue to store
the nodes with their distances
    vector<int> distTo(n, INT_MAX);    // array for storing
shortest paths, initialized with INT_MAX

    distTo[source] = 0;
    pq.push(make_pair(0, source)); // inserting pair of (distance,
node) in the priority queue

    while (!pq.empty())
    {
        int dist = pq.top().first;
        int prev = pq.top().second;
        pq.pop();

        vector<pair<int, int>>::iterator it;
        for (it = adj[prev].begin(); it != adj[prev].end(); it++)
        {
            int next = it->first;
            int nextDist = it->second;
            if (distTo[next] > distTo[prev] + nextDist) //Greedy
Condition
            {
                distTo[next] = distTo[prev] + nextDist;
                pq.push(make_pair(distTo[next], next));
            }
        }
    }

    cout << "The distances from source, " << source << " are :
\n";
    for (int i = 0; i < n; i++)
    {
        cout << "Node " << i << ": " << distTo[i] << endl;
    }
    cout << "\n";
}

int main()
{
    int n, m, source; // n = number of nodes, m = number of edges,
source = source node

    cout << "Enter the number of nodes: ";
    cin >> n;
    cout << "Enter the number of edges: ";
    cin >> m;
    vector<pair<int, int>> adj[n]; // adjacency list for the graph

    int a, b, wt;
    for (int i = 0; i < m; i++)
    {
        cout << "Enter the edge " << i << " (<first_node>
<second_node> <weight>): ";
        cin >> a >> b >> wt;
        adj[a].push_back(make_pair(b, wt));
        adj[b].push_back(make_pair(a, wt));
    }

    cout << "Enter the source node: ";
    cin >> source;

    dijkstras(source, adj, n);
}
```

---------- Output ----------

```
Enter the number of nodes: 6
Enter the number of edges: 8
Enter the edge 0 (<first_node> <second_node> <weight>): 0 1 4
Enter the edge 1 (<first_node> <second_node> <weight>): 0 2 4
Enter the edge 2 (<first_node> <second_node> <weight>): 2 1 2
Enter the edge 3 (<first_node> <second_node> <weight>): 1 3 3
Enter the edge 4 (<first_node> <second_node> <weight>): 1 4 6
Enter the edge 5 (<first_node> <second_node> <weight>): 1 5 1
Enter the edge 6 (<first_node> <second_node> <weight>): 3 4 2
Enter the edge 7 (<first_node> <second_node> <weight>): 5 4 3
Enter the source node: 0
The distances from source, 0 are :
Node 0: 0
Node 1: 4
Node 2: 4
Node 3: 7
Node 4: 8
Node 5: 5
```