Name: Abhishek Anand

Enr. No- 12019002004061

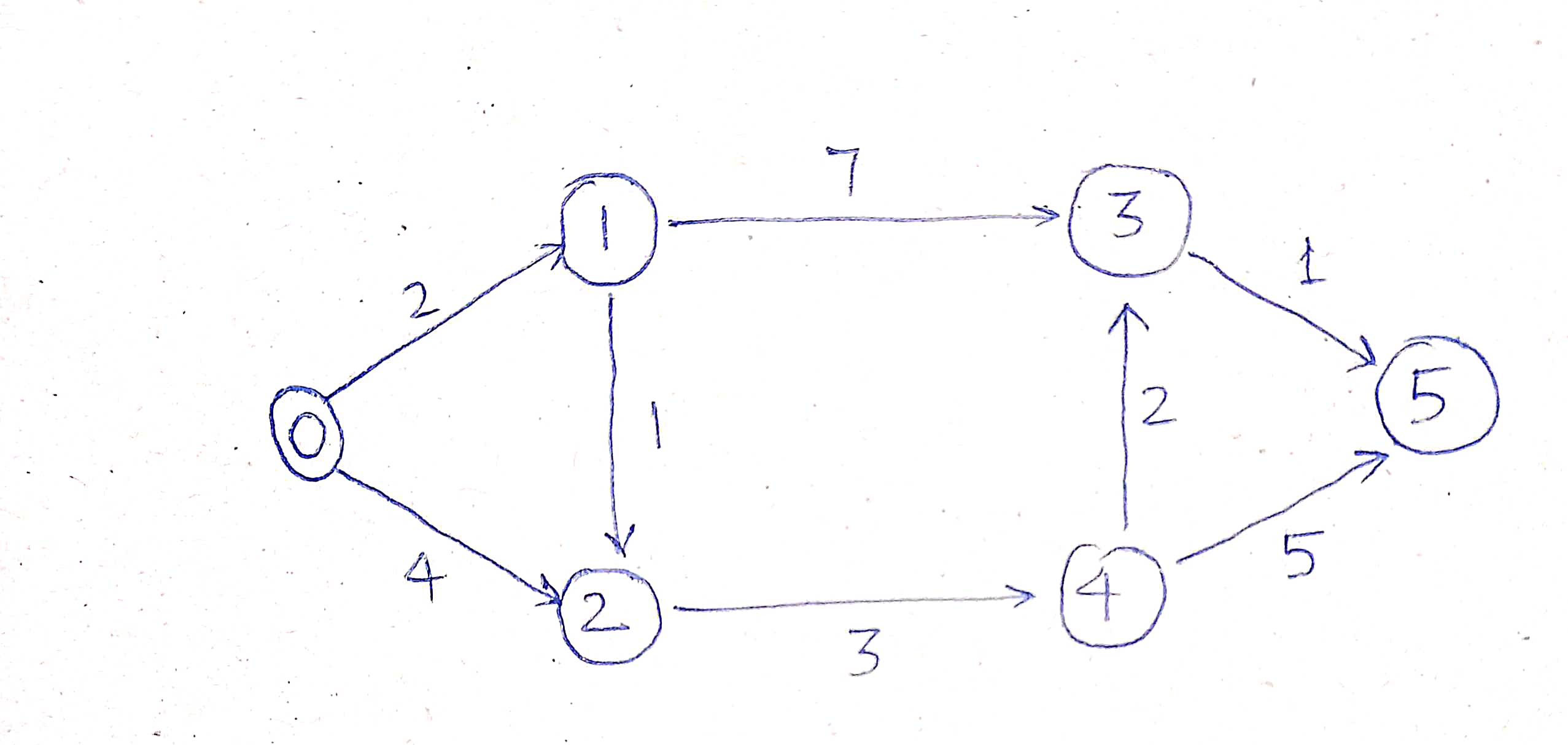
Roll-57

Section –A

Subject- Computer Networks Lab

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Single-Source Shortest Path Problem**

****

**Approach**

The algorithm computes for each vertex u the distance to u

from the start vertex v, that is, the weight of a shortest path

between v and u.

the algorithm keeps track of the set of vertices for which the

distance has been computed, called the cloud C

Every vertex has a label D associated with it. For any vertex u,

D[u] stores an approximation of the distance between v and

u. The algorithm will update a D[u] value when it finds a

shorter path from v to u.

When a vertex u is added to the cloud, its label D[u] is equal

to the actual (final) distance between the starting vertex v

and vertex u.

**Code:-**

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int main(){

vector<pair<int,int>> graph[6]={{{1,2},{2,4}},{{2,1},{3,7}},{{4,3}},{{5,1}},{{3,2},{5,5}},{}};

priority\_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>> pq;

vector <int> dist(6,100000000);

dist[0]=0;

pq.push({0,0});

while(!pq.empty()){

int a=pq.top().first;

int b=pq.top().second;

pq.pop();

for(auto it : graph[a]){

if(b+it.second < dist[it.first]){

dist[it.first]= b+it.second;

pq.push({it.first,dist[it.first]});

}

}

}

for(auto it : dist) {

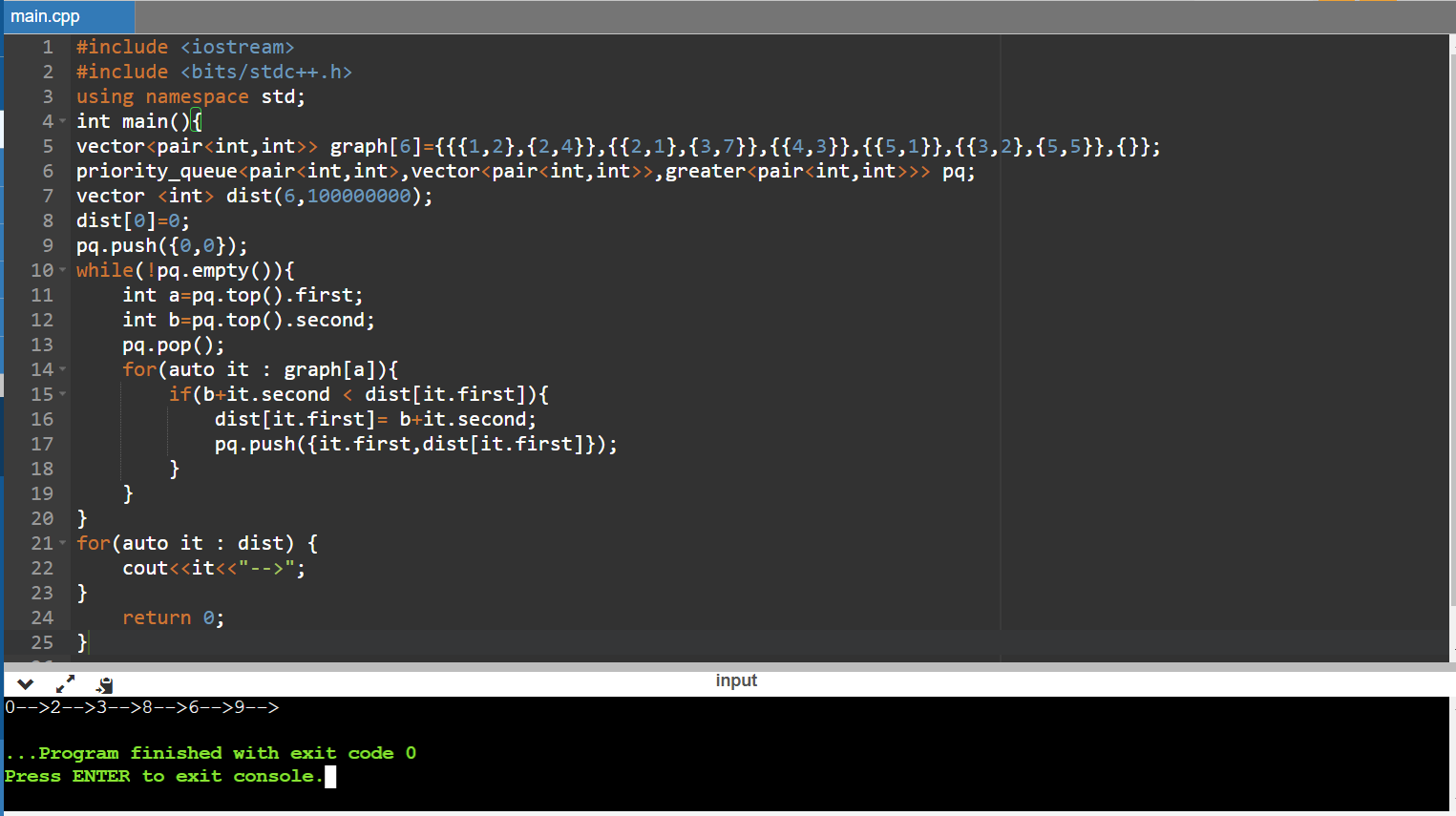
cout<<it<<"-->";

}

return 0;

}

**Input & Output Screenshot:-**

****