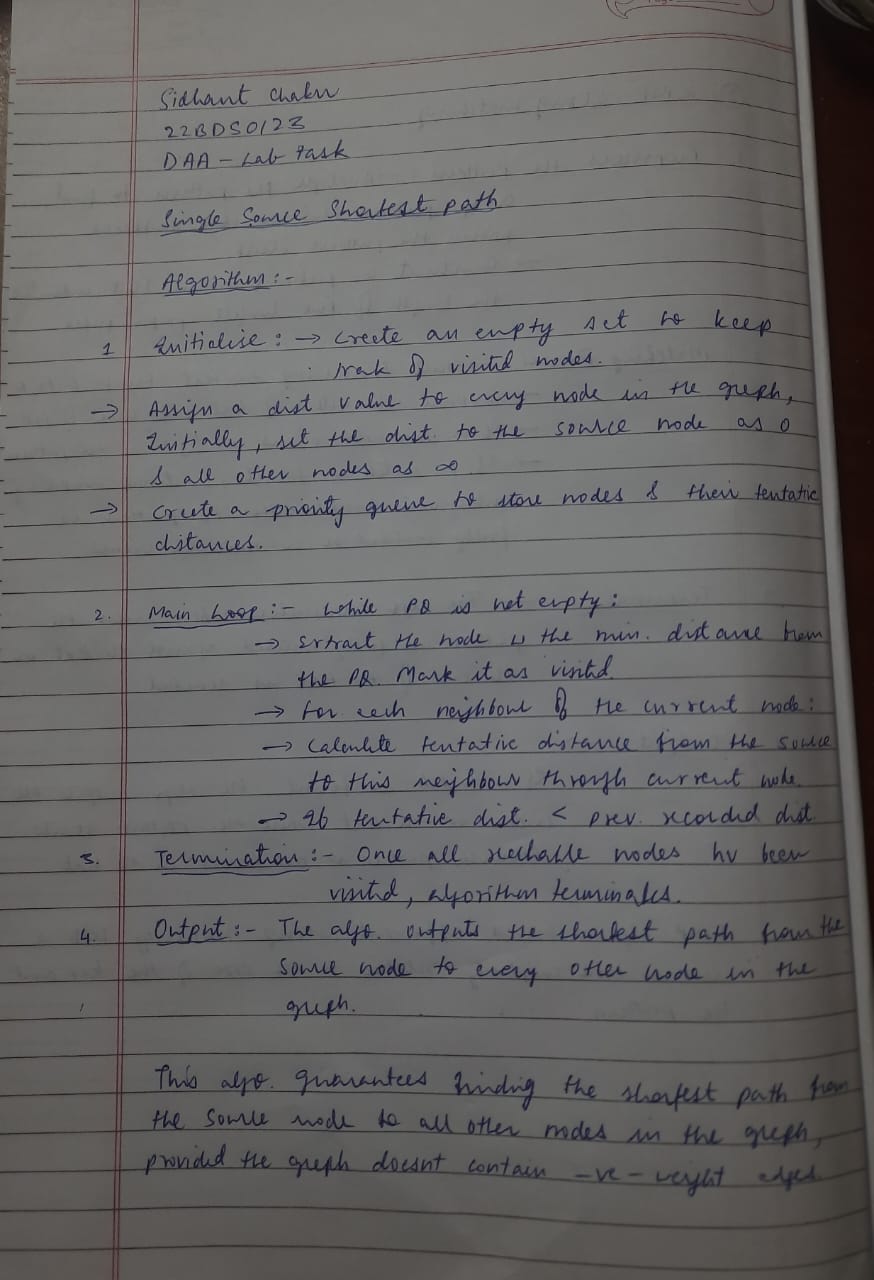
SIDHANT CHAKU

22BDS0123

DAA LABTASK

Q1)

SINGLE SOURCE SHORTEST PATH



PROGRAM CODE:

#include<iostream>

#include<vector>

#include<queue>

#include<climits>

using namespace std;

#define INF INT\_MAX

struct node{

int id;

int dist;

node(int id,int dist):id(id),dist(dist){}

};

struct cmpdist{

bool operator()(const node& a,const node& b){

return a.dist>b.dist;

}

};

void djikstra(vector<vector<pair<int,int>>>& graph,int v){

priority\_queue<node,vector<node>,cmpdist>pq;

vector<int>dist(v,INF);

dist[0]=0;

pq.push(node(0,0));

while(!pq.empty()){

int u=pq.top().id;

pq.pop();

for(auto& neighbor:graph[u]){

int v=neighbor.first;

int weight=neighbor.second;

if(dist[v]>dist[u]+weight){

dist[v]=dist[u]+weight;

pq.push(node(v,dist[v]));

}

}

}

for (int i=0;i<v;i++){

cout<<dist[i]<<"";

}

}

int main(){

int v,e;

cout<<"enter the no. of vertices:";

cin>>v;

cout<<"enter the number of edges:";

cin>>e;

vector<vector<pair<int,int>>>graph(v);

cout<<"enter edges in format 'node1 node2 weight':"<<endl;

for(int i=0;i<e;i++){

int u,v,w;

cin>>u>>v>>w;

graph[u].push\_back(make\_pair(v,w));

graph[v].push\_back(make\_pair(u,w));

}

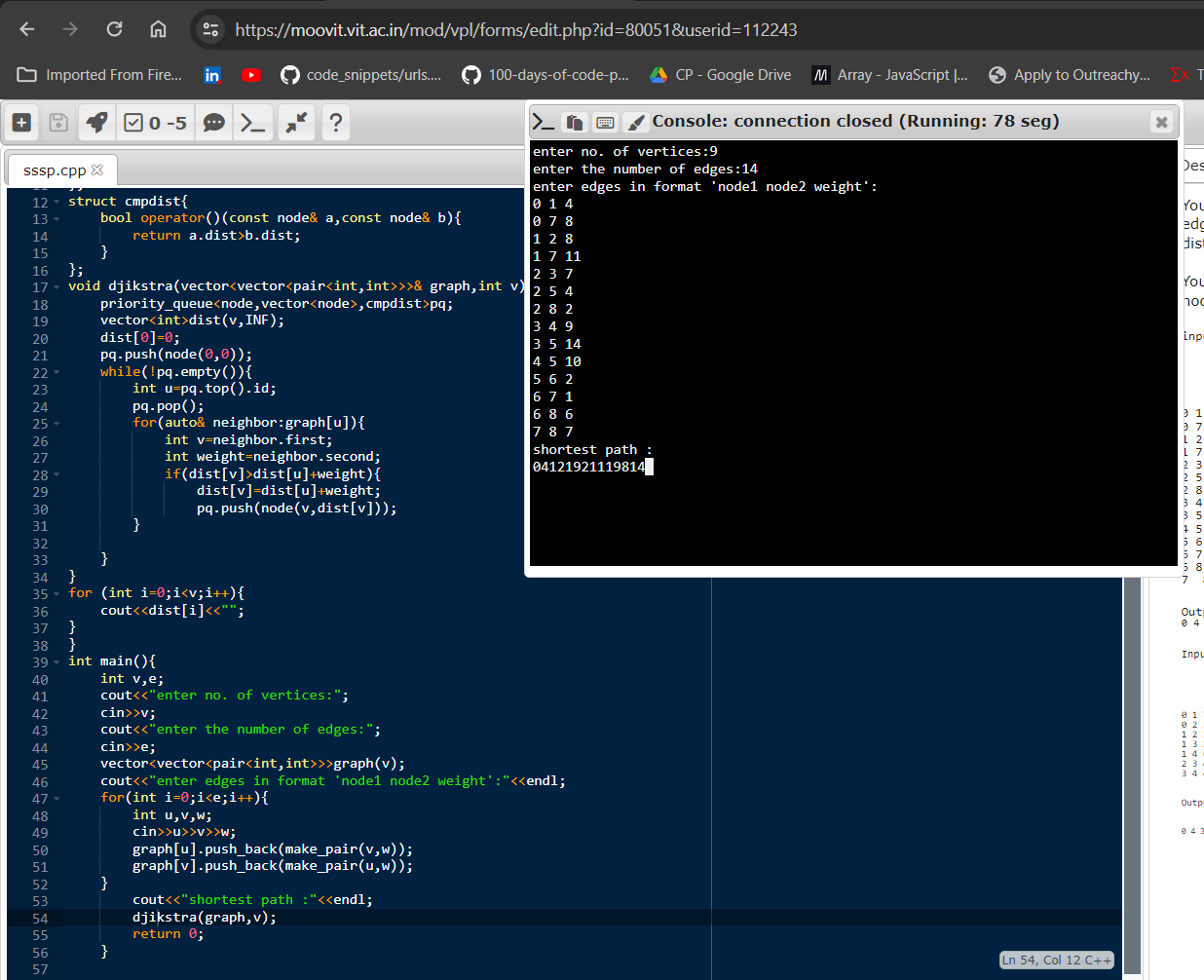
cout<<"shortest path :"<<endl;

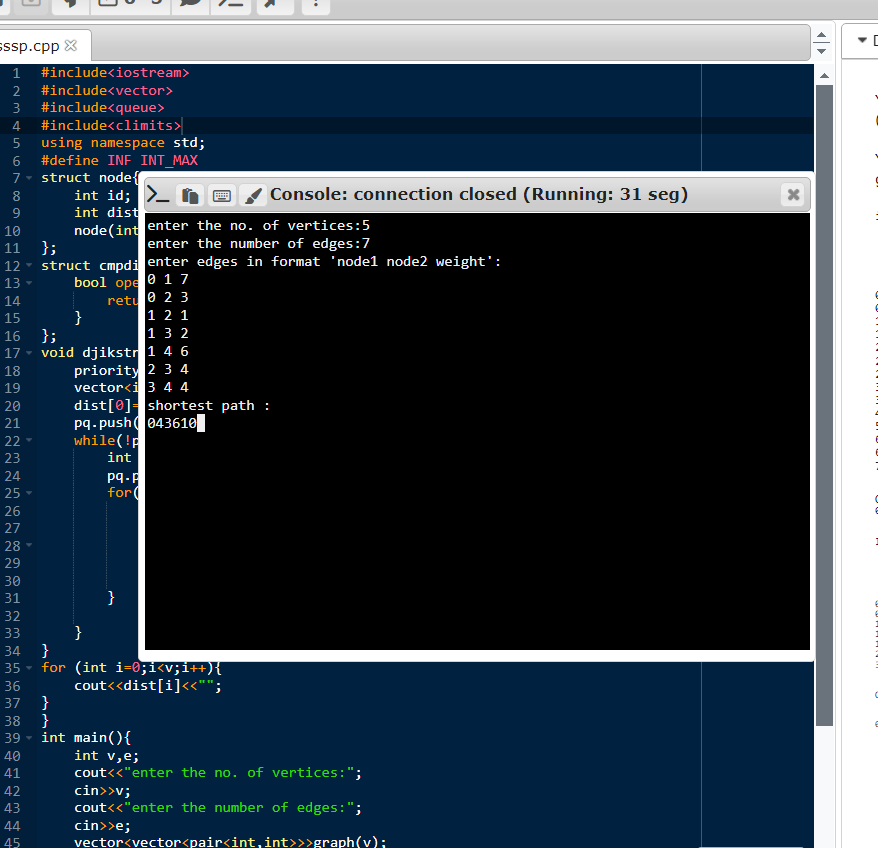
djikstra(graph,v);

return 0;

}

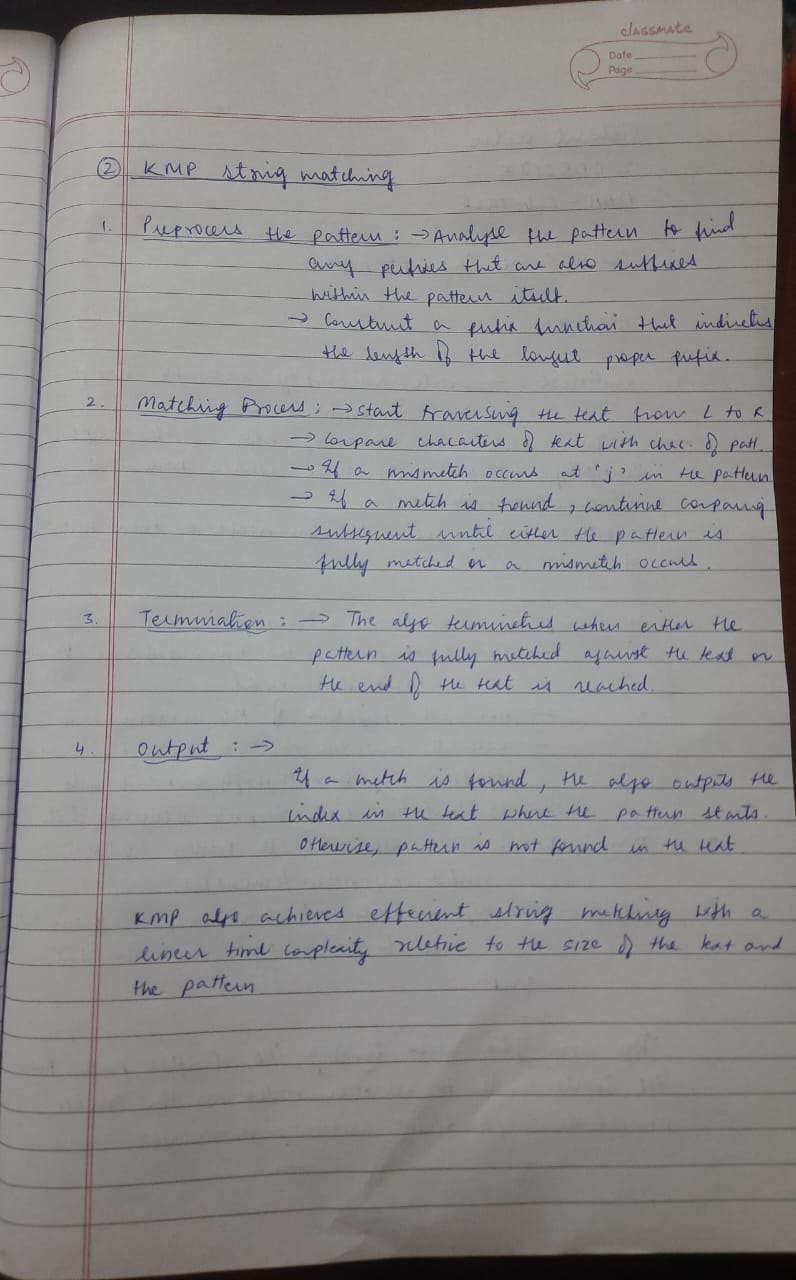
OUTPUT:





Q2)

KMP STRING MATCHING



INPUT CODE:

#include <iostream>

#include <vector>

#include <string>

std::vector<int> findoccurrences(const std::string& target, const std::string& search){

std::vector<int> occurrences;

int n= search.length();

int m =target.length();

for( int i=0;i<=m-n;i++){

int j;

for (j=0;j<n;j++){

if (target[i+j]!=search[j])

break;

}

if (j==n){

occurrences.push\_back(i);

}

}

return occurrences;

}

int main(){

int num\_cases;

std::cin>>num\_cases;

for(int i=0;i<num\_cases;++i){

std::string search,target;

std::cin>>search>>target;

std::vector<int> result= findoccurrences(target,search);

if(result.empty()){

std::cout<<"NO(search string not found in the target string)"<<std::endl;

}else{

std::cout<<"YES"<<stdśss::endl;

std::cout<<"occurrences of each string in the target string:";

for(int index:result){

std::cout<<index<<" ";

}

std::cout<<std::endl;

}

}

return 0;

}

OUTPUT:

