SIDHANT CHAKU

22BDS0123

DAA LABTASK

Q1)

SINGLE SOURCE SHORTEST PATH

Sidhant Chakr
22BDS0123
DAA - Lab Task

Single Source Shortest path

Algorithm :-

1 Initialise : → Create an empty set to keep track of visited nodes.

→ Assign a dist value to every node in the graph, Initially, set the dist to the source node as 0 & all other nodes as ∞

→ Create a priority queue to store nodes & their tentative distances.

2. Main Loop :- While PQ is not empty :
→ Extract the node w the min. distance from the PQ. Mark it as visited
→ For each neighbour of the current node :
→ Calculate tentative distance from the source to this neighbour through current node.
→ If tentative dist. < prev recorded dist.

3. Termination :- Once all reachable nodes hv been visited, algorithm terminates.

4. Output :- The algo. outputs the shortest path from the source node to every other node in the graph.

This algo. guarantees finding the shortest path from the source node to all other nodes in the graph, provided the graph doesn't contain -ve - weight edges

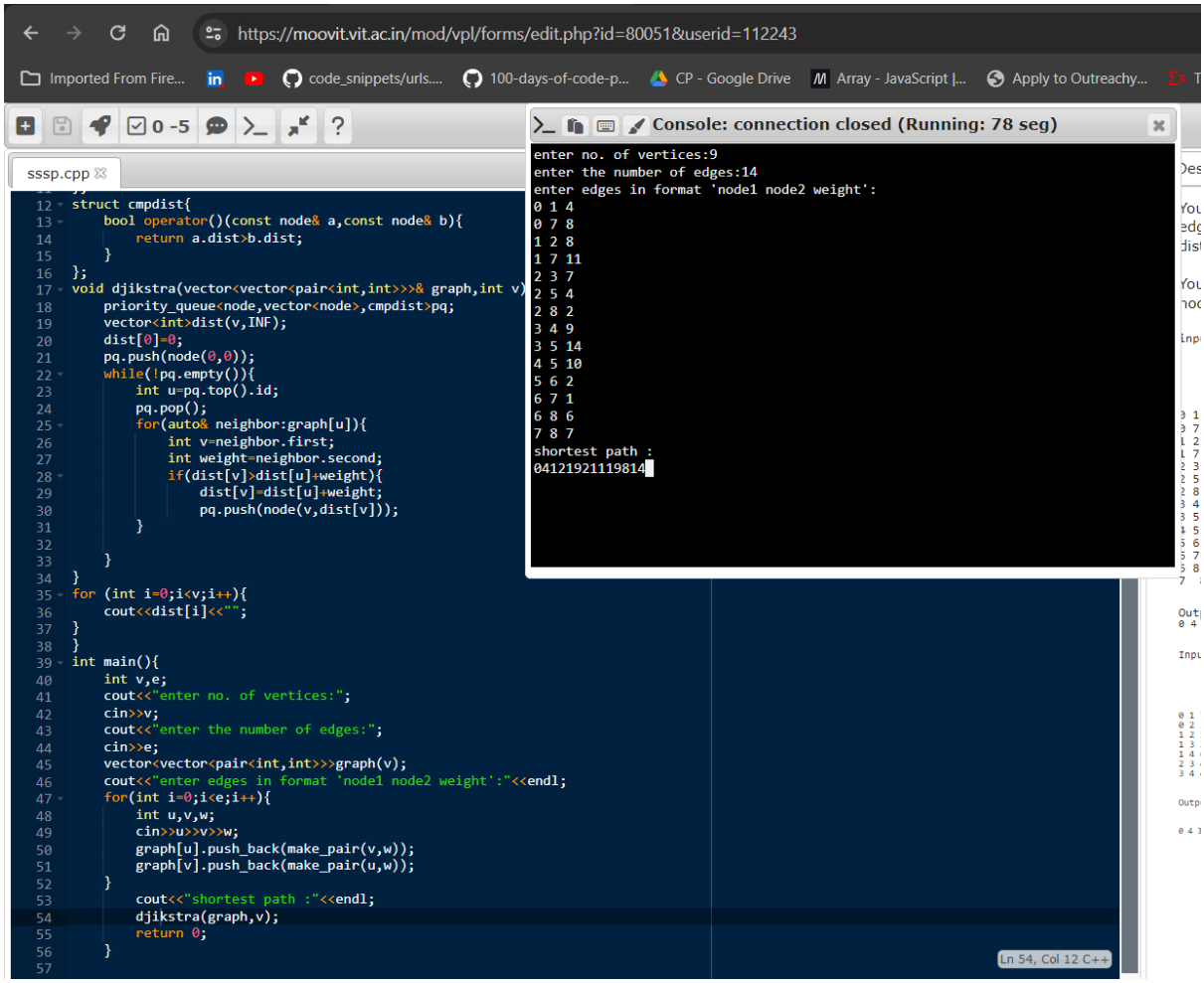
PROGRAM CODE:

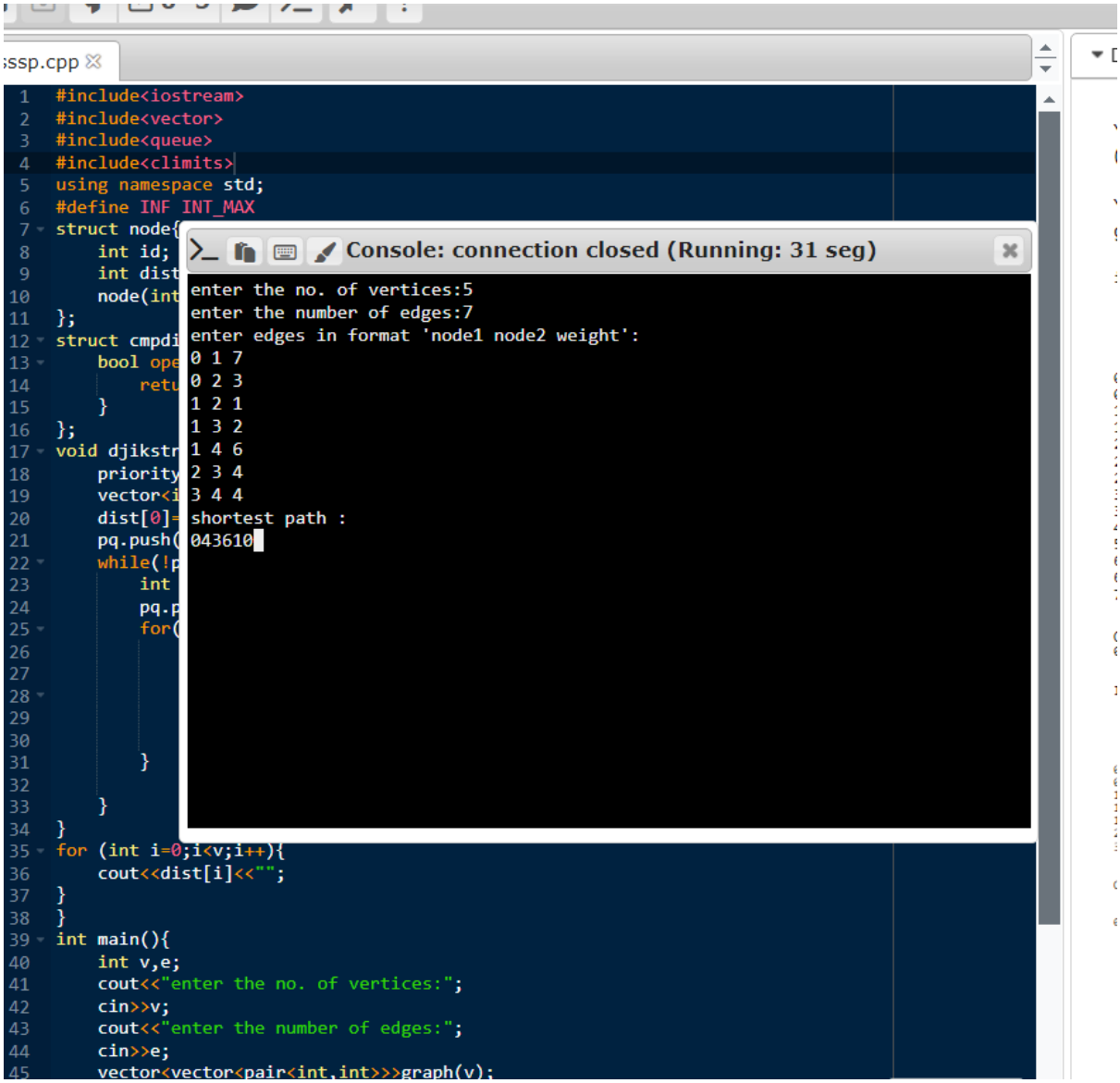
```
#include<iostream>
#include<vector>
#include<queue>
#include<climits>
using namespace std;
#define INF INT_MAX
struct node{
    int id;
    int dist;
    node(int id,int dist):id(id),dist(dist){}
};
struct cmpdist{
    bool operator()(const node& a,const node& b){
        return a.dist>b.dist;
    }
};
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:


https://moovit.vit.ac.in/mod/vpl/forms/edit.php?id=80051&userid=112243
Imported From Fire...
code_snippets/urls....
100-days-of-code-p...
CP - Google Drive
Array - JavaScript |...
Apply to Outreachy...
0 -5
sssp.cpp
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 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;
}
Ln 54, Col 12 C++
Console: connection closed (Running: 78 seg)
enter no. of vertices:9
enter the number of edges:14
enter edges in format 'node1 node2 weight':
0 1 4
0 7 8
1 2 8
1 7 11
2 3 7
2 5 4
2 8 2
3 4 9
3 5 14
4 5 10
5 6 2
6 7 1
6 8 6
7 8 7
shortest path :
04121921119814



Q2)

KMP STRING MATCHING

② KMP string matching

1. Preprocess the pattern : → Analyse the pattern to find any prefixes that are also suffixes within the pattern itself.
   → Construct a prefix function that indicates the length of the longest proper prefix.

2. Matching Process : → start traversing the text from L to R.
   → Compare characters of text with char. of patt.
   → If a mismatch occurs at 'j' in the pattern
   → If a match is found, continue comparing subsequent until either the pattern is fully matched or a mismatch occurs.

3. Termination : → The algo terminates when either the pattern is fully matched against the text or the end of the text is reached.

4. Output : →
   If a match is found, the algo outputs the index in the text where the pattern starts. Otherwise, pattern is not found in the text.

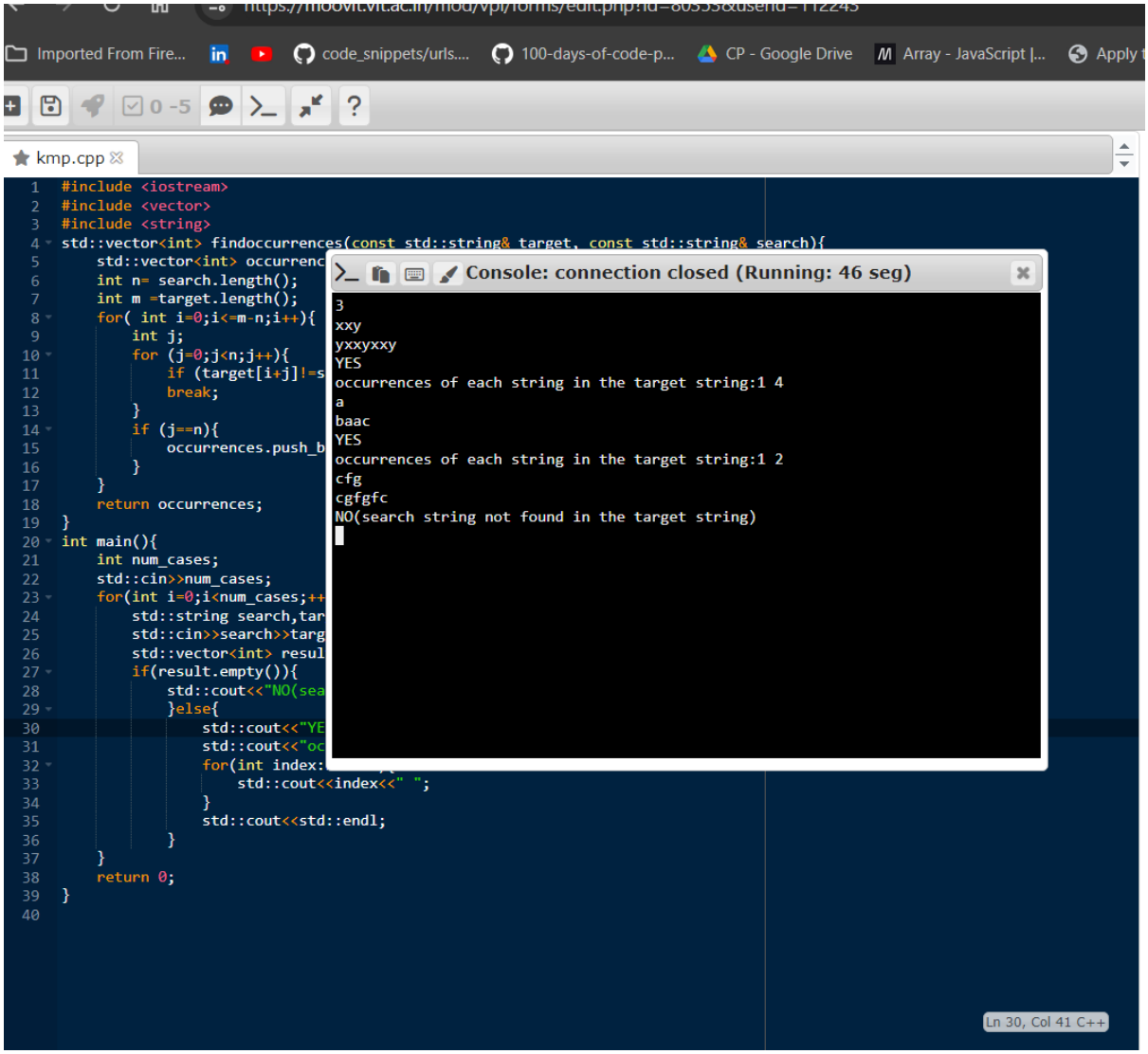
KMP algo achieves efficient string matching with a linear time complexity relative to the size of the text and the pattern.

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:


Imported From Fire...
code_snippets/urls....
100-days-of-code-p...
CP - Google Drive
Array - JavaScript |...
kmp.cpp
#include <iostream>
#include <vector>
#include <string>
std::vector<int> findoccurrences(const std::string& target, const std::string& search){
std::vector<int> occurrenc
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]!=s
break;
}
if (j==n){
occurrences.push_b
}
}
return occurrences;
}
int main(){
int num_cases;
std::cin>>num_cases;
for(int i=0;i<num_cases;++
std::string search,tar
std::cin>>search>>targ
std::vector<int> resul
if(result.empty()){
std::cout<<"NO(sea
}else{
std::cout<<"YE
std::cout<<"oc
for(int index:
std::cout<<index<<" ";
}
std::cout<<std::endl;
}
}
return 0;
}
Console: connection closed (Running: 46 seg)
3
xxy
yxxyxxy
YES
occurrences of each string in the target string:1 4
a
baac
YES
occurrences of each string in the target string:1 2
cfg
cgfgfc
NO(search string not found in the target string)
Ln 30, Col 41 C++