

SIDHANT CHAKU

22BDS0123

DAA LABTASK

Q1)

SINGLE SOURCE SHORTEST PATH

Sidhant Chakr
22BDS0123
DAA - Lab Task

Single Source Shortest path

Algorithm:-

1. Initialise:- \rightarrow Create an empty set to keep track of visited nodes.
 \rightarrow Assign a dist value to every node in the graph, Initially, set the dist. to the source node as 0 & all other nodes as ∞
 \rightarrow Create a priority queue to store nodes & their tentative distances.
2. Main Loop:- While PQ is not empty:
 \rightarrow Extract the node w the min. distance from the PQ. Mark it as visited.
 \rightarrow For each neighbour of the current node:
 \rightarrow Calculate tentative distance from the source to this neighbour through current node.
 \rightarrow If tentative dist. < prev. recorded dist.
3. Termination:- Once all reachable nodes have 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 dijkstra(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... [in](#) [code_snippets/urls...](#) [100-days-of-code-p...](#) [CP - Google Drive](#) [M Array - JavaScript \[...\]](#) [Apply to Outreachy...](#) [T](#)

0 - 5

🔍 ?

sssp.cpp

```
12 - struct cmpdist{
13 -     bool operator()(const node& a,const node& b){
14 -         return a.dist>b.dist;
15 -     }
16 - };
17 - void djikstra(vector<vector<pair<int,int>>>& graph,int v)
18 - {
19 -     priority_queue<node,vector<node>,cmpdist>pq;
20 -     vector<int>dist(v,INF);
21 -     dist[0]=0;
22 -     pq.push(node(0,0));
23 -     while(!pq.empty()){
24 -         int u=pq.top().id;
25 -         pq.pop();
26 -         for(auto& neighbor:graph[u]){
27 -             int v=neighbor.first;
28 -             int weight=neighbor.second;
29 -             if(dist[v]>dist[u]+weight){
30 -                 dist[v]=dist[u]+weight;
31 -                 pq.push(node(v,dist[v]));
32 -             }
33 -         }
34 -     }
35 -     for (int i=0;i<v;i++){
36 -         cout<<dist[i]<<" ";
37 -     }
38 - }
39 - int main(){
40 -     int v,e;
41 -     cout<<"enter no. of vertices:";
42 -     cin>>v;
43 -     cout<<"enter the number of edges:";
44 -     cin>>e;
45 -     vector<vector<pair<int,int>>>graph(v);
46 -     cout<<"enter edges in format 'node1 node2 weight':"<<endl;
47 -     for(int i=0;i<e;i++){
48 -         int u,v,w;
49 -         cin>>u>>v>>w;
50 -         graph[u].push_back(make_pair(v,w));
51 -         graph[v].push_back(make_pair(u,w));
52 -     }
53 -     cout<<"shortest path : "<<endl;
54 -     djikstra(graph,v);
55 -     return 0;
56 - }
57
```

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

Des
You
ad
dis
You
loc
Inp
3 1
3 7
1 2
1 7
2 3
2 5
2 8
3 4
3 5
3 6
3 7
3 8
7 :
Out:
0 4
Inp:
0 1
0 2
1 2
1 3
1 4
2 3
3 4
Outp:
0 4 3

Ln 54, Col 12 C++

```
1 #include<iostream>
2 #include<vector>
3 #include<queue>
4 #include<climits>
5 using namespace std;
6 #define INF INT_MAX
7 struct node{
8     int id;
9     int dist;
10    node(int id, int dist): id(id), dist(dist) {}
11 };
12 struct cmpdist{
13     bool operator()(const node& a, const node& b){
14         return a.dist > b.dist;
15     }
16 };
17 void djikstra(int s, vector<vector<pair<int, int>>>& graph, vector<int>& dist, vector<int>& pq){
18     priority_queue<node, vector<node>, cmpdist> pq;
19     dist[s] = 0;
20     pq.push(node(s, 0));
21     while(!pq.empty()){
22         int u = pq.top().id;
23         pq.pop();
24         for(int v = 0; v < graph[u].size(); v++){
25             int w = graph[u][v].first;
26             int weight = graph[u][v].second;
27             if(dist[w] > dist[u] + weight){
28                 dist[w] = dist[u] + weight;
29                 pq.push(node(w, dist[w]));
30             }
31         }
32     }
33 }
34
35 for (int i=0; i<v; i++){
36     cout<<dist[i]<<" ";
37 }
38
39 int main(){
40     int v, e;
41     cout<<"enter the no. of vertices:";
42     cin>>v;
43     cout<<"enter the number of edges:";
44     cin>>e;
45     vector<vector<pair<int, int>>>graph(v);
```

Console: connection closed (Running: 31 seg)

enter the no. of vertices:5
enter the number of edges:7
enter edges in format 'node1 node2 weight':
0 1 7
0 2 3
1 2 1
1 3 2
1 4 6
2 3 4
3 4 4
shortest path :
0 4 3 6 1 0

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 K
→ 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 also 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"<<std::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:

https://moovit.vit.ac.in/mod/vpl/forums/edit.php?id=80535&userid=112245

Imported From Fire... code_snippets/urls... 100-days-of-code-p... CP - Google Drive Array - JavaScript |... Apply

★ kmp.cpp

```
1 #include <iostream>
2 #include <vector>
3 #include <string>
4 std::vector<int> findoccurrences(const std::string& target, const std::string& search){
5     std::vector<int> occurrences;
6     int n = search.length();
7     int m = target.length();
8     for(int i=0; i<=m-n; i++){
9         int j;
10        for(j=0; j<n; j++){
11            if (target[i+j] != search[j])
12                break;
13        }
14        if (j==n){
15            occurrences.push_back(i);
16        }
17    }
18    return occurrences;
19 }
20 int main(){
21     int num_cases;
22     std::cin>>num_cases;
23     for(int i=0; i<num_cases; i++){
24         std::string search, target;
25         std::cin>>search>>target;
26         std::vector<int> result = findoccurrences(target, search);
27         if(result.empty()){
28             std::cout<<"NO(search string not found in the target string)\n";
29         }else{
30             std::cout<<"YES\n";
31             std::cout<<"occurrences of each string in the target string:";
32             for(int index: result){
33                 std::cout<<index<<" ";
34             }
35             std::cout<<std::endl;
36         }
37     }
38     return 0;
39 }
40
```

Console: connection closed (Running: 46 seg)

```
3
xxxxy
yxyxyxy
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
cfgfgfc
NO(search string not found in the target string)
```

Ln 30, Col 41 C++