Algorithms

Topics: Shortest Path Algorithms

Shortest Path Algorithms

TUTORIAL **PROBLEMS**

The shortest path problem is about finding a path between 2 vertices in a graph such that the total sum of the edges weights is minimum.

This problem could be solved easily using **(BFS)** if all edge weights were (1), but here weights can take any value. Three different algorithms are discussed below depending on the use-case.

Bellman Ford's Algorithm:

Bellman Ford's algorithm is used to find the shortest paths from the source vertex to all other vertices in a weighted graph. It depends on the following concept: Shortest path contains at most $n - 1$ edges, because the shortest path couldn't have a cycle.

So why shortest path shouldn't have a cycle ?
There is no need to pass a vertex again, because the shortest path to all other vertices could be found without the need for a second visit for any vertices.

Algorithm Steps:

- The outer loop traverses from $0 : n - 1$.
- Loop over all edges, check if the next node distance > current node distance + edge weight, in this case update the next node distance to "current node distance + edge weight".

This algorithm depends on the relaxation principle where the shortest distance for all vertices is gradually replaced by more accurate values until eventually reaching the optimum solution. In the beginning all vertices have a distance of "Infinity", but only the distance of the source vertex = 0, then update all the connected vertices with the new distances (source vertex distance + edge weights), then apply the same concept for the new vertices with new distances and so on.

Implementation:

Assume the source node has a number (0):

```
    vector <int> v [2000 + 10];
    int dis [1000 + 10];

    for(int i = 0; i < m + 2; i++){

        v[i].clear();
        dis[i] = 2e9;
    }

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

        scanf("%d%d%d", &from , &next , &weight);

        v[i].push_back(from);
        v[i].push_back(next);
        v[i].push_back(weight);
   }

    dis[0] = 0;
    for(int i = 0; i < n - 1; i++){
        int j = 0;
        while(v[j].size() != 0){

            if(dis[ v[j][0]  ] + v[j][2] < dis[ v[j][1] ] ){
                dis[ v[j][1] ] = dis[ v[j][0]  ] + v[j][2];
            }
            j++;
        }
    }
```

A very important application of Bellman Ford is to check if there is a negative cycle in the graph,

Time Complexity of Bellman Ford algorithm is relatively high $O(V \cdot E)$, in case $E = V^2$, $O(E^3)$. Let's discuss an optimized algorithm.

Dijkstra's Algorithm

Dijkstra's algorithm has many variants but the most common one is to find the shortest paths from the source vertex to all other vertices in the graph.

Algorithm Steps:

- Set all vertices distances = infinity except for the source vertex, set the source distance = 0.
- Push the source vertex in a min-priority queue in the form (distance , vertex), as the comparison in the min-priority queue will be according to vertices distances.
- Pop the vertex with the minimum distance from the priority queue (at first the popped vertex = source).
- Update the distances of the connected vertices to the popped vertex in case of "current vertex distance + edge weight < next vertex distance", then push the vertex with the new distance to the priority queue.
- If the popped vertex is visited before, just continue without using it.
- Apply the same algorithm again until the priority queue is empty.

Implementation:

Assume the source vertex = 1.

```
#define SIZE 100000 + 1

vector < pair < int , int > > v [SIZE];   // each vertex has all the
connected vertices with the edges weights
int dist [SIZE];
bool vis [SIZE];

void dijkstra(){
                                                            // set the vertices
distances as infinity
    memset(vis, false , sizeof vis);            // set all vertex as
unvisited
    dist[1] = 0;
    multiset < pair < int , int > > s;          // multiset do the job as
a min-priority queue

    s.insert({0 , 1});                          // insert the source node
with distance = 0

    while(!s.empty()){

        pair <int , int> p = *s.begin();        // pop the vertex with the
minimum distance
        s.erase(s.begin());
        int x = p.s; int wei = p.f;
        if( vis[x] ) continue;                    // check if the popped
vertex is visited before
         vis[x] = true;

        for(int i = 0; i < v[x].size(); i++){
            int e = v[x][i].f; int w = v[x][i].s;
            if(dist[x] + w < dist[e]  ){            // check if the next
vertex distance could be minimized
                dist[e] = dist[x] + w;
                s.insert({dist[e],  e} );           // insert the next
vertex with the updated distance
            }
        }
    }
}
```

Time Complexity of Dijkstra's Algorithm is $O(V^2)$ but with min-priority queue it drops down to $O(V + E\ log\ V)$.

However, if we have to find the shortest path between all pairs of vertices, both of the above methods would be expensive in terms of time. Discussed below is another alogorithm designed for this case.

Floyd–Warshall's Algorithm

Floyd–Warshall's Algorithm is used to find the shortest paths between between all pairs of vertices in a graph, where each edge in the graph has a weight which is positive or negative. The biggest advantage of using this algorithm is that all the shortest distances between any 2 vertices could be calculated in $O(V^3)$, where V is the number of vertices in a graph.

The Algorithm Steps:

For a graph with N vertices:

- Initialize the shortest paths between any 2 vertices with Infinity.
- Find all pair shortest paths that use 0 intermediate vertices, then find the shortest paths that use 1 intermediate vertex and so on.. until using all N vertices as intermediate nodes.
- Minimize the shortest paths between any 2 pairs in the previous operation.
- For any 2 vertices (i, j) , one should actually minimize the distances between this pair using the first K nodes, so the shortest path will be: $min(dist[i][k] + dist[k][j], dist[i][j])$.

$dist[i][k]$ represents the shortest path that only uses the first K vertices, $dist[k][j]$ represents the shortest path between the pair k, j. As the shortest path will be a concatenation of the shortest path from i to k, then from k to j.

```
for(int k = 1; k <= n; k++){
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= n; j++){
            dist[i][j] = min( dist[i][j], dist[i][k] + dist[k][j] );
        }
    }
}
```

Time Complexity of Floyd–Warshall's Algorithm is $O(V^3)$, where V is the number of vertices in a graph.

Contributed by: omar khaled abdelaziz abdelnabi

Did you find this tutorial helpful? YES NO

TEST YOUR UNDERSTANDING

Shortest Path Problem

You are given *2* integers (N, M), *N* is the number of vertices, *M* is the number of edges. You'll also be given a_i , b_i, w_i where a_i and b_i represents an edge from a vertex a_i to a vertex b_i and w_i respresents the weight of that edge.

Task is to find the shortest path from source vertex (vertex number *1*) to all other vertices (v_i) where ($2 \leq i \leq N$).

Input:

First line contains two space separated integers, (N, M) Then *M* lines follow, each line has *3* space separated integers a_i , b_i, w_i.

Output:
Print the shortest distances from the source vertex (vertex number *1*) to all other vertices (v_i) where ($2 \leq i \leq N$). Print "10^9" in case the vertex "v_i" can't be reached form the source vertex.
Leave a space between any *2* printed numbers.

Constraints:
$(1 \leq N \leq 10^4)$
$(1 \leq M \leq 10^6)$
$(1 \leq a_i, b_i \leq N)$
$(1 \leq w_i \leq 1000)$

SAMPLE INPUT

```
5 5
1 2 5
1 3 2
3 4 1
1 4 6
3 5 5
```

SAMPLE OUTPUT

```
5 2 3 7
```

Enter your code or Upload your code as file. Save C (gcc 5.4.0)

```
/*
// Sample code to perform I/O:
#include <stdio.h>

int main(){
    int num;
    scanf("%d", &num);                          // Reading input from STDIN
    printf("Input number is %d.\n", num);       // Writing output to STDOUT
}

// Warning: Printing unwanted or ill-formatted data to output will cause the test cases t
*/

// Write your code here

```

1:1

Provide custom input

COMPILE & TEST SUBMIT

COMMENTS (125)

SORT BY: Relevance

Login/Signup to Comment

?

Gvs Akhil 2 years ago
I believe that the test case data is wrong for the sample problem. Could someone please check the data again? Thanks.
0 votes ● Reply ● Message ● Permalink

Akash Gupta 2 years ago
listen Akhil, everything is fine and perfect .you just need to check your algorithm again.
1 vote ● Reply ● Message ● Permalink

BestSinceDay1 2 years ago
input is surely wrong
0 votes ● Reply ● Message ● Permalink

BestSinceDay1 2 years ago
because i have copy pasted the algorithm given and still getting wrong answer.
0 votes ● Reply ● Message ● Permalink

Satyam Shekhar 2 years ago
test cases are correct... i think you probably made a undirected graph instead of directed one.... I was doing the same mistake.
41 votes ● Reply ● Message ● Permalink

Tài Nguyễn a year ago
thank you very much !
0 votes ● Reply ● Message ● Permalink

Abhinesh Garhwal a year ago
Thanks finally accepted
1 vote ● Reply ● Message ● Permalink

Abhishek Saharn a year ago
life saver :)
0 votes ● Reply ● Message ● Permalink

Sharvin Jondhale a year ago
thank you very much ! I was stuck on it for so long.
0 votes ● Reply ● Message ● Permalink

Puneet Kumar a year ago
arre satyam tum to guru nikale
2 votes ● Reply ● Message ● Permalink

Anmol Saxena a year ago
Thanks
0 votes ● Reply ● Message ● Permalink

Hemant Mangwani a year ago
```
// TAKE GRAPH AS DIRECTED ONE NOT UNDIRECTED
#include <bits/stdc++.h>
#define ll long long
#define T ll t; cin>>t; while(t--)
using namespace std;
#define speed ios_base::sync_with_stdio(0)
#define endl "\n"
vector<pair<int,int>>adj[1000001];
ll c=1000000000;
```

```
bool visited[1000001];
ll dist[1000001];
void intilization(int n)
{
for(ll i=0;i<=n;i++)
{
visited[i]=0;
dist[i]=c;
}
}
void dijkstras()
{
ll st=1;
//cout<<"starting= ";
//cin>>st;
dist[st]=0;
multiset<pair<int,int>>s;
s.insert({0,st});
while(!s.empty())
{
ll from,to,cost;
pair<int,int> p = *s.begin();
s.erase(s.begin());
from = p.second;
if(visited[from]) continue;
visited[from]=1;
for(ll i=0;i<adj[from].size();i++)
{
cost=adj[from][i].first;
to=adj[from][i].second;
if(dist[from]+cost<dist[to])
{
dist[to]=dist[from]+cost;
s.insert({dist[to],to});
}
}
}
}
int main()
{
speed;
ll nodes,edges,i;
cin>>nodes>>edges;
intilization(nodes);
for(i=0;i<edges;i++)
{
ll from,to,cost;
cin>>from>>to>>cost;
adj[from].push_back({cost,to});
// adj[to].push_back({cost,from});
}
dijkstras();
// for(i=1;i<=nodes;i++)
// {
// cout<<visited[i]<<" ";
// }
// cout<<endl;
for(i=2;i<=nodes;i++)
{
cout<<dist[i]<<" ";
}
cout<<endl;
}
```

▲ 4 votes ● Reply ● Message ● Permalink

Ravi Rahul 2 years ago
don't predefine the maximum number of vertices as 10^4 as given in question's constraints. make the number of vertices as 10^6 or as user defined and it won't give wrong answer.
▲ 1 vote ● Reply ● Message ● Permalink

Bhargav kular 2 years ago
In python 3 it shows "end of file while reading a line". Please help
▲ 1 vote ● Reply ● Message ● Permalink

Konstantin Yovkov Edited 2 years ago
It's because the input is wrong - m is 1000000, but there are actually 999200 edges given....
▲ 11 votes ● Reply ● Message ● Permalink

Nayan Jyoti Das a month ago
Hi Konstantin!! It is also showing me the same. If what u are saying is true, then what should I do? abandon the problem or there is any other solution for this?
▲ 0 votes ● Reply ● Message ● Permalink

BHAGWATI MALAV a year ago
I used java, passed sample test case, getting TLE while submitting, even if i use (e+v)logv
▲ 10 votes ● Reply ● Message ● Permalink

Akshay Bhatt a year ago
Same here
▲ 0 votes ● Reply ● Message ● Permalink

Utkarsh Gupta a year ago
Likewise
▲ 0 votes ● Reply ● Message ● Permalink

Vignesh 5 months ago
i am getting wrong answer. the same solution is accepted in hackerrank.
▲ 0 votes ● Reply ● Message ● Permalink

Suraj Jha 3 months ago
solution is correct(i got same solution using both bellman ford and dijkstra) , make sure to take only 999200 input lines . As of TLE we have some options 1) don't search for element in the PQ as its O(N) (assuming heap) just add it without checking 2) write your own heap and keep another array to keep track of position on heap so that searching can be O(1)
▲ 0 votes ● Reply ● Message ● Permalink

Ravi sharma a year ago
int e = v[x][i].f; int w = v[x][i].s;
this line in the algorithm is wrong
▲ 5 votes ● Reply ● Message ● Permalink

Nitin Rathodiya a year ago
No that is correct.
Because implementation of vector was like this vector<vertex,weight> not same as multiset in which implementation is as multiset<weight,vertex> .
▲ 2 votes ● Reply ● Message ● Permalink

Shubham Singhal a year ago

_/_

▲ 0 votes • Reply • Message • Permalink

Nitesh Chaudhry a year ago

Yes. INTERCHANGE the first and second
In the given function, edge(e) is at second position and weight at first .

▲ 0 votes • Reply • Message • Permalink

Mandava Desik Edited a year ago

The Algorithm for Dijkstra mentioned above misses a small point of updating all the distances to Infinite so pls modify it.

▲ 6 votes • Reply • Message • Permalink

Aishwarya Rastogi 2 years ago

can we use set instead of multiset in the implementation of djkstras algorithm

▲ 1 vote • Reply • Message • Permalink

Ajay Verma a year ago

yes we can!

▲ 3 votes • Reply • Message • Permalink

Shashank Gupta a year ago

Have you tried this ? I don't think we can. Set can't keep duplicate keys/values which I guess is a possibility in case of Dijkstra's algo.

▲ 1 vote • Reply • Message • Permalink

Pradeep Gehlot Edited a year ago

in case of this question set compare Whole pair ,i don't think any pair with same vertex and weight insert duplicate in set ,you are right set in not allow duplicate ,may be wieght duplicate but vertex different everytym.

▲ 0 votes • Reply • Message • Permalink

Utkarsh Gupta a year ago

Bellman ford in java exceeded time limit so I implemented Dijkstra and again the time limit exceeded. To further optimise the code, I implemented a binary heap as a priority queue and still the time limit is exceeding. Apparently, a lot of of people are getting the same result when submitting the code in java.

▲ 4 votes • Reply • Message • Permalink

Shruti Gupta 10 months ago

The first line of the dataset says there are 1000000 edges, but there are only 999200 edges, hence the TLE error in JAVA

▲ 4 votes • Reply • Message • Permalink

Prachi Prakash Mahapatro a year ago

Hey the time complexity of the Bellmen ford algol is O(VE) where E = V^2 so the time complexity will be O(V^3), I think there is a typo as it shows O(E^3).

▲ 3 votes • Reply • Message • Permalink

Abhinav Gautam a year ago

I think author forgot to set all the distance infinity in djkstras.
There is a comment line though.

▲ 1 vote • Reply • Message • Permalink

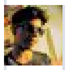
akshat sharma a year ago

?

can you upload your answer here , with same form of dijkstra algo as shown , mine answer is only print -1,see

```
#include <bits/stdc++.h>
using namespace std;
#define SIZE 100000 + 1
vector < pair < int , int > > v [SIZE]; // each vertex has all the connected vertices with the edges weights
int dist [SIZE];
bool vis [SIZE];
void dijkstra(){
// set the vertices distances as infinity
memset(vis, false , sizeof (vis)); // set all vertex as unvisited
dist[1] = 0;
multiset < pair < int , int > > s; // multiset do the job as a min-priority queue
s.insert({0 , 1}); // insert the source node with distance = 0
while(!s.empty()){
pair <int , int> p = *s.begin(); // pop the vertex with the minimum distance
s.erase(s.begin());
int x = p.second; int wei = p.first;
if( vis[x] ) continue; // check if the popped vertex is visited before
vis[x] = true;
for(int i = 0; i < v[x].size(); i++){
int e = v[x][i].first; int w = v[x][i].second;
if(dist[x] + w < dist[e] ){ // check if the next vertex distance could be minimized
dist[e] = dist[x] + w;
s.insert({dist[e], e} ); // insert the next vertex with the updated distance
}
}
}
}

int main()
{
// memset(vis,false,sizeof(vis));
memset(dist,INT_MAX,sizeof(dist));
int nodes,edges;
cin>>nodes;
cin>>edges;
for(int i=0;i<edges;i++)
{
int x,y,z;
cin>>x>>y>>z;
v[x].push_back({y,z});

}
dijkstra();
for(int i=2;i<=nodes;i++)
{
cout<<dist[i]<<" ";
}
return 0;
}
```

0 votes • Reply • Message • Permalink



Abhinav Gautam a year ago

```
//ABHINAV GAUTAM (gotham13121997)
#include <bits/stdc++.h>
using namespace std;
#define getC(n) scanf("%c",&n)
```

```
#define getL(n) scanf("%ld",&n)
#define getLL(n) scanf("%lld",&n)
#define getD(n) scanf("%lf",&n)
#define getS(n) scanf("%s",n)
#define getI(a) scanf("%d", &a)
#define getII(a,b) scanf("%d%d", &a, &b)
#define getIII(a,b,c) scanf("%d%d%d", &a, &b, &c)
#define getIIII(a,b,c,d) scanf("%d%d%d%d", &a, &b, &c, &d)
#define FOR(i,L,R) for (int i = L; i < R; i++)
#define FORE(i,L,R) for (int i = L; i <= R; i++)
#define FORD(i,L,R) for (int i = L; i > R; i--)
#define FORDE(i,L,R) for (int i = L; i >= R; i--)
#define foreach(v, c) for( typeof( (c).begin()) v = (c).begin(); v != (c).end(); ++v)
#define whileT int T; getI(T); while(T--)
#define mem(a,x) memset(a,x,sizeof(a))
#define abs(x) (x<0?(-x):x)
#define maX(a,b) ( (a) > (b) ? (a) : (b))
#define miN(a,b) ( (a) < (b) ? (a) : (b))
#define ALL(c) (c).begin(),(c).end()
#define PRESENT(c,x) ((c).find(x) != (c).end())
#define ll long long
#define ull unsigned long long
#define ui unsigned int
#define us unsigned short
#define fr first
#define se second
#define pb push_back
#define mp make_pair
typedef vector<int> vi;
typedef vector<vi> vvi;
typedef pair<int,int> pii;
typedef vector<pii> vpii;
typedef queue<int> qi;
int main()
{
int n,m,x,y,w;
getII(n,m);
vector<pair<int,int> > v[n+1];
ll dis[n+1];
bool vis[n+1];
while(m--)
{
getIII(x,y,w);
v[x].pb(mp(w,y));
}
FORE(i,0,n+1)
{
vis[i]=false;
dis[i]=1000000000;
}
dis[1]=0;
multiset<pii> s;
s.insert({0,1});
while(!s.empty())
{
pii p=*s.begin();
s.erase(s.begin());
int x=p.se;
if(vis[x])
continue;
vis[x]=true;
for(vector<pair<int,int> > ::iterator it=v[x].begin();it!=v[x].end();++it)
{
int e=it->second;
int w=it->first;
if(dis[x]+w<dis[e])
{
dis[e]=dis[x]+w;
s.insert({dis[e],e});
}
}
}
FORE(i,2,n)
{
cout<<dis[i]<<" ";
}
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

akshat sharma a year ago

thanks i understood my problem.

▲ 0 votes ● Reply ● Message ● Permalink

gabe_3491 a year ago

memset(dist,INT_MAX,sizeof(dist)); //it is not valid..
memset just fill the 1 byte...you have to assign INT_MAX by using for loop..

▲ 2 votes ● Reply ● Message ● Permalink

Naragoni Sairam Edited a year ago

Here is the solution....

```
#include <bits/stdc++.h>
using namespace std;
int dist[100005];
int vis[100005];
vector< pair<int,int> > v[100005];
void dijkstra()
{
multiset< pair< int,int > > s;
s.insert({0,1});
while(!s.empty())
{
pair< int,int > p = *s.begin();
s.erase(s.begin());
int e = p.second;
if(vis[e])
continue;
vis[e]=1;
for(int i=0;i<v[e].size();i++)
{
int x = v[e][i].first;
int y = v[e][i].second;
if(dist[e]+y<dist[x])
{
dist[x] = dist[e]+y;
s.insert({dist[x],x});
}
}
}
}
```

```
int main()
{
int m,n,x,y,w;
cin>>n>>m;
for(int i=0;i<=n+3;i++)
{
dist[i]=1e9;
vis[i]=0;
}
dist[0]=0;
dist[1]=0;
for(int i=0;i<m;i++)
{
cin>>x>>y>>w;
v[x].push_back(make_pair(y,w));
}
dijkstra();
for(int i=2;i<=n;i++)
{
cout<<dist[i]<<" ";
}
return 0;
}
```

▲ 2 votes ● Reply ● Message ● Permalink

EKJOT KAUR a year ago

```
#include <bits/stdc++.h>
using namespace std;
int main(){
int n,m;
cin>>n>>m;
int dis[n];
for(int i=1;i<=n;i++)
dis[i]=1e9;
dis[1]=0;
vector < pair<int, int> > v[n+1];
bool visited[n+1];
memset(visited,false, sizeof(visited));
for(int i=0;i<m;i++){
int a,b,w;
cin>>a>>b>>w;
v[a].push_back(make_pair(b,w));
}
multiset< pair< int , int > >s;
s.insert({0,1});
while(!s.empty()){
pair< int, int>p=*s.begin();
s.erase(s.begin());
//cout<<p.second <<" is the node"<<endl;
if(visited[p.second]==true)
continue;
visited[p.second]=true;
for(int i=0;i<v[p.second].size();i++){
int vertice=v[p.second][i].first;
int weight=v[p.second][i].second;
if(visited[vertice]==true)
continue;
if(dis[vertice] > (dis[p.second]+weight))
dis[vertice]=dis[p.second]+weight;
s.insert({dis[vertice],vertice});
}
}
for(int i=2;i<=n;i++)
cout<<dis[i]<<" ";
}
```

My code logic is exactly the same.. still getting TLE..

0 votes • Reply • Message • Permalink

Naragoni Sairam a year ago

No Idea !! Might be problem with the compiler ... idk
Your logic is correct though

0 votes • Reply • Message • Permalink

Rahul Khandelwal a year ago

Using (E+V)logV time complexity method still getting TLE :(

0 votes • Reply • Message • Permalink

Rohit a year ago

Probably you are pushing same node again and again in the priority queue. This can be avoided by using boolean visited[] array

0 votes • Reply • Message • Permalink

EKJOT KAUR a year ago

I tried that but still getting TLE

0 votes • Reply • Message • Permalink

Rohit a year ago

Refer to my code , you can find in the comments

0 votes • Reply • Message • Permalink

Manveer Singh 7 months ago

I think you should push even same node again, because may be the distance associated with that node has reduced. Duplicate nodes will be taken care of by min priority queue by taking less distance into consideration.

1 vote • Reply • Message • Permalink

Vaibhav Khulbe Edited a year ago

hey, the dijkstra algorithm code is wrong.
int e = v[x][i].f; int w = v[x][i].s;
in this line the values of f and s need to be swapped. i.e. the value of w should be v[x][i].f and that of e should be v[x][i].s.

it was a wonderful tutorial thanks a lot

1 vote • Reply • Message • Permalink

Raghav Ravi Prakash Edited a year ago

time limit exceeded after submission. Please help. Here's my code snippet in java.

```
ArrayList<Pair>[] adjacencyList = new ArrayList[N+1];
for(int i = 1; i <= N; i++)
adjacencyList[i] = new ArrayList<Pair>();
public void updateList(int sourceNode, int destNode, int weight)
{
Pair pairNode = new Pair(destNode,weight);
adjacencyList[sourceNode].add(pairNode);
}
```

```
public void dijkstra()
{
Comparator<Pair> weightCompare = new PriorityQueueWeight();
PriorityQueue<Pair> pq = new PriorityQueue<Pair>(1000000,weightCompare);

boolean[] visited = new boolean[N+1];
Arrays.fill(visited,1,visited.length,false);

int[] distance = new int[N+1];
distance[1] = 0; // distance between root node and itself is 0
Arrays.fill(distance,2,distance.length,1000000000);

pq.offer(new Pair(1,0)); // root node is 1.
while(!pq.isEmpty())
{
Pair pair = (Pair) pq.poll();
int sourceNode = pair.node;

if(visited[sourceNode])
continue;

visited[sourceNode] = true;
for(int i = 0; i < adjacencyList[sourceNode].size(); i++)
{
pair = (Pair) adjacencyList[sourceNode].get(i);
int destNode = pair.node;
int weight = pair.weight;

if((distance[sourceNode] + weight) < distance[destNode])
{
distance[destNode] = distance[sourceNode] + weight;
pq.offer(new Pair(destNode,distance[destNode]));
}
}
}

displayShortestDistances(distance);
}
private void displayShortestDistances(int[] distance)
{
for(int i = 2; i <= N; i++)
System.out.print(distance[i] + " ");
System.out.println();
}
```

▲ 1 vote ● Reply ● Message ● Permalink

Nitesh Chaudhry a year ago

//Dijkstr's algo
WORKING CODE :::

```
#include<bits/stdc++.h>
using namespace std;
#define SIZE 100000 + 1
vector < pair < int , int > > v [SIZE]; // each vertex has all the connected vertices with the edges weights
int dist [SIZE];
bool vis [SIZE];
void dijkstra(int n){
for(int i=0;i<=n;i++) dist[i]=1000000000; // set the vertices distances as infinity
memset(vis, false , sizeof vis); // set all vertex as unvisited
dist[1] = 0;
multiset < pair < int , int > > s; // multiset do the job as a min-priority queue
s.insert({0 , 1}); // insert the source node with distance = 0
while(!s.empty()){
pair <int , int> p = *s.begin(); // pop the vertex with the minimum distance
s.erase(s.begin());
int x = p.second; int wei = p.first;
if( vis[x] ) continue; // check if the popped vertex is visited before
vis[x] = true;
for(int i = 0; i < v[x].size(); i++){
int e = v[x][i].second; int w = v[x][i].first;
if(dist[x] + w < dist[e] ){ // check if the next vertex distance could be minimized
dist[e] = dist[x] + w;
s.insert({dist[e], e} ); // insert the next vertex with the updated distance
}
}
}
}
int main(){

int n,m,from,to,w;
cin>>n>>m; //Nodes and edges.
for(int i=0;i<m;i++)
{
cin>>from>>to>>w;
v[from].push_back(make_pair(w,to)); // Creating required D.S.
}
/*for(int i=1;i<n;i++){
for(int j=0;j<v[i].size();j++){
cout<<v[i][j].first<<" "<<v[i][j].second<<"\t";
}
}*/ //This was intended
dijkstra(n); //N is passed so the fn. knows the max dist where the dist has to be put 10e9
for(int i=2;i<=n;i++) cout<<dist[i]<<" ";
return 0;
}
```

▲ 1 vote ● Reply ● Message ● Permalink

Shiwang Gupta a year ago

priority queue would be more efficient in djisktra algorithm

▲ 1 vote ● Reply ● Message ● Permalink

Endou Mamoru a year ago

Can anyone tell me what will be the problem if i wrote s.insert({0, e}) instead of s.insert({dist[e], e}) in dijkstra .I know we have to insert vertex e in the queue but the dist[e] seems to perform no function?

▲ 1 vote ● Reply ● Message ● Permalink

Sahil Aggarwal a year ago

Problem statement should be updated to mention that edges are directed.

▲ 1 vote ● Reply ● Message ● Permalink

Codematters a year ago

if i use " priority_queue<PII, vector<PII>, greater<PII> > " (typedef pair<int,int> PII) in Dijkstra's Algorithm instead of multiset, the time complexity will still be O(V+ElogV). Right?

▲ 1 vote ● Reply ● Message ● Permalink

Vedant Gawade Edited 10 months ago

1: The time complexity for Bellman Ford is wrong i guess, it should be O(V*E) = O(V*V^2) = O(V^3), and also check required for reporting -ve cycle is not given in code, its incomplete!
2: Time Complexity of Dijkstra's algorithm depends on how the priority Q is implemented.
If array is used to implement the pri-Q : O(V^2)
If binary min heap is used : O(Vlog V + Elog V)
If fibonacci heap is used : O(Vlog V + E)
Refer MIT OCW: https://www.youtube.com/watch?v=2E7MmKv0Y24

1 vote ● Reply ● Message ● Permalink

Sudhanshu Shekhar 3 months ago

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;typedef pair < ll,ll > pll;
ll N,M; // n nodes m edgges
void diji(ll dist[],vector < pair < ll,ll > > adj[],ll x,bool vis[])
{
for(ll i=0;i<N+1;i++)
dist[i] = 1e9;
dist[x] = 0;
priority_queue < pll , vector < pll > , greater < pll > > q;
q.push(make_pair(0,x));
while(!q.empty())
{
pll p = q.top();
q.pop();
ll a = p.second;ll wt = p.first;
if(vis[a]==true)
continue;
vis[a]=true;
for(ll i=0;i<adj[a].size();i++)
{
ll e = adj[a][i].second , w = adj[a][i].first;
if(dist[a]+w<dist[e])
{
dist[e] = w + dist[a];
q.push(make_pair(dist[e],e));
}
}
}
return;
}
int main()
{
cin>>N>>M;
vector < pair < ll,ll > > adj[2*N];
for(ll i=0;i<M;i++)
{
ll x,y,w;
cin>>x>>y>>w;
adj[x].push_back(make_pair(w,y));
}
ll dist[N+1];bool vis[2*N];memset(vis,false,sizeof(vis));
diji(dist,adj,1,vis);
for(ll i=2;i<N+1;i++)
cout<<dist[i]<<" ";
return 0;
}
```

1 vote ● Reply ● Message ● Permalink

UdayNbausj 2 months ago

CPP solution
https://ideone.com/zDlVVx

▲ 1 vote ● Reply ● Message ● Permalink

Shreemoyee Sarkar 2 months ago

In the Dijkstra's algorithm implementation, multiset has been used. However I believe, its better to use a priority_queue with an underlying vector container since a priority_queue uses heap at the backend and popping the minimum element from a heap is O(n) process. On the contrary, multiset uses balanced BST which has O(logn) deletions.

▲ 1 vote ● Reply ● Message ● Permalink

Sumit Kumar Pradhan 2 years ago

It's showing time limit exceeded for all my submissions. I have used set for retrieving the node with least weight as shown in tutorial but still that didn't work. Please can someone help me??? Shall I post my code here???

▲ 0 votes ● Reply ● Message ● Permalink

Utsav Goel 2 years ago

What's wrong with this code? I added this after while loop in dijkstra function.
code---
```
while(!s.empty())
{
pair< int,int > u=*s.begin();
s.erase(s.begin());
cout<<u.first;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Vivek Singh 2 years ago

it's a directed graph!!!!!!!!!

▲ 0 votes ● Reply ● Message ● Permalink

akshat sharma a year ago

how you determine graph is directed? i cant made choice between from reading the question, please help me

▲ 0 votes ● Reply ● Message ● Permalink

Vivek Singh a year ago

"where ai and bi represents an edge from a vertex ai 'to' a vertex bi" this line is the key

▲ 0 votes ● Reply ● Message ● Permalink

Milind Ghiya Edited a year ago

I feel inputs must be wrong ....i hav assumed directed graph here and also dijkstra code for undirected got accepted on hackerrank!
here , runtime error for input mismatch is shown which is unexpected

▲ 0 votes ● Reply ● Message ● Permalink

Parth Pratim Chatterjee Edited a year ago

I am having trouble solving this problem. My Program is showing a Time Limit Exceeded error. What changes should i make in my code.
```
#include <iostream>
#include <vector>
using namespace std;
std::vector <std::vector<int> > dis(10001, std::vector<int>(10002, 1000));
long long int nodes,edges,t,t2,t3,t4,k,i,j;
int main()
```

```
{
cin >> nodes;
cin >> edges;
for(t=1;t<=nodes;t++)
{
dis[t][t] = 0;
}
for(t=1;t<=edges;t++)
{
cin >> t2 >> t3 >> t4;
dis[t2][t3] = t4;
}
for(k=1;k<=nodes;k++)
{
for(i=1;i<=nodes;i++)
{
for(j=1;j<=nodes;j++)
{
if(dis[i][j] > dis[i][k] + dis[k][j])
dis[i][j] = dis[i][k] + dis[k][j];
}
}
}
for(t2=2;t2<=nodes;t2++)
{
cout << dis[1][t2] << " ";
}
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Rohit a year ago

use a visited[] boolean array which make sure not to visit same node more than once

▲ 0 votes ● Reply ● Message ● Permalink

Vicky rana a year ago

there is a blunder mistake in question: they should have mentioned the question is of directed graph

▲ 0 votes ● Reply ● Message ● Permalink

Rohit a year ago

number of edges is not 1000000,take it as 999200 else you will get NZEC error

▲ 0 votes ● Reply ● Message ● Permalink

Mrinal Verma a year ago

can someone explain how multiset is working in dijkstra algorithim

▲ 0 votes ● Reply ● Message ● Permalink

Keyur Chaudhary a year ago

Use the Dijkstra's Algorithm for Adjacency List Representation -- it will definetly work

▲ 0 votes ● Reply ● Message ● Permalink

Rohit Edited a year ago

MY CODE IS WORKING

```
#include <bits/stdc++.h>
#define ii pair<long,long>
#define mm make_pair
using namespace std;
class prioritize
```

```
{
public:
bool operator()(const ii &a,const ii &b)
{
return a.second>b.second;
}
};
int main()
{
long n,m,x,y,w;
cin>>n>>m;
vector<ii> v[n+1];
bool mark[n+1];
memset(mark,false,sizeof(mark));
for(int i=0;i<m;i++)
{
cin>>x>>y>>w;
v[x].push_back(mm(y,w));
}
int dis[n+1];
for(int i=1;i<=n;i++)dis[i]=INT_MAX;
priority_queue<ii,vector<ii>,prioritize> pq;
pq.push(make_pair(1,dis[1]=0));
while(!pq.empty())
{
int p=pq.top().first;
pq.pop();
if(mark[p])continue;
mark[p]=true;
for(vector<ii> ::iterator it=v[p].begin();it!=v[p].end();++it)
{
int k=it->first;
int j=it->second;
if(!mark[k] && dis[k]>dis[p]+j)
{
dis[k]=dis[p]+j;
pq.push(mm(k,dis[k]));
}
}
}
for(long i=2;i<=n;i++)
{
if(dis[i]!=INT_MAX)cout<<dis[i]<<" ";
else cout<<"1000000000 ";
}
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Ajay Verma a year ago

a thing to notice in the very starting that BFS can be used with weight other than 1 also iff each should be equal, please modify it...!
And thanks a lot for this nice tutorial..

▲ 0 votes ● Reply ● Message ● Permalink

akshat sharma a year ago

can somebody tell me how dijkstra work differently in directed and undirected graphs , i cant able to get a intuituion of dijkstra in directed graph?

▲ 0 votes ● Reply ● Message ● Permalink

Pradeep Gehlot Edited a year ago

just do it on paper in both cases.
3 3
1 2 5
2 3 8
3 1 3
after that analysis your result.

0 votes • Reply • Message • Permalink

Sneha Yadav Edited a year ago

in bellman ford algo
for(int i = 0; i < n - 1; i++){
int j = 0;
while(v[j].size() != 0){
if(dis[v[j][0]] + v[j][2] < dis[v[j][1]]){
dis[v[j][1]] = dis[v[j][0]] + v[j][2];
}
j++;
}
why is i loop for ?
as with j we are reaching those from ,to and distance elements then why we are making j loop as there are only 3 elements there ?

0 votes • Reply • Message • Permalink

SATYAJEET BEHERA a year ago

i loop is for visiting all the nodes and j loop is for visiting all the edges as
u said three elements are in the each edge vector we don't have to loop through it or u can

0 votes • Reply • Message • Permalink

Sneha Yadav a year ago

thanks

0 votes • Reply • Message • Permalink

Tuấn Nguyễn Minh a year ago

Anyone here, can explain to me.. why my code is wrong???....
https://ideone.com/Q086zm

0 votes • Reply • Message • Permalink

Barun Kumar Acharya a year ago

Above in bellman ford's algo the complexity should be o(v3) right as e = v2

0 votes • Reply • Message • Permalink

gabe_3491 a year ago

if you are getting a wrong ans..then u must me taking it as a undirected graph..correct it..

0 votes • Reply • Message • Permalink

gabe_3491 a year ago

dijkstra's working fine but bellman give tle??any suggestion to improve bellman?

0 votes • Reply • Message • Permalink

ROHIT KUMAR SINHA a year ago

Why this problem is giving TLE when I'm trying to solve it by min heap using priority_queue() stl even if the program is correct??

0 votes • Reply • Message • Permalink

Mohammad Zaid a year ago

if(vis[x]) continue;
// check if the popped vertex is visited before
what's its use,

0 votes • Reply • Message • Permalink

Alok Kumar a year ago

My question is why we use multiset instead of priority_queue. Would any one explain how multiset is faster than priority_queue while both have insert operation in logn time.
For this question to be accepted you have to make directed graph.
Here are my both solution one with priority_queue another one with multiset.

```
==========
solution with priority_queue
=======================
#include <bits/stdc++.h>
#define ll long long
#define pb push_back
#define mp make_pair
#define pll pair<ll,ll>
#define ff first
#define ss second
using namespace std;
const int MAX = 1e6 + 5;
const int MAX_DIST=1e9+7;
bool visited[MAX];
ll dist[MAX];
vector <pll> adj[MAX];
void init()
{
for(ll i=0;i<MAX;i++)
{
dist[i]=1e9;
visited[i]=false;
}
}
void dijkstra(ll source)
{
//memset(dist,MAX_DIST,sizeof(dist));//set distance to maximum for each vertex
init();
//memset(visited,false,sizeof(visited));//set visited array to false
priority_queue<pll, vector<pll>, greater<pll> > Q;
dist[source]=0;
Q.push(mp(0, source));
while(!Q.empty())
{
// Select the edge with minimum weight
pll t = Q.top();
Q.pop();
ll x = t.ss;
// Checking for cycle
if(visited[x])
continue;
visited[x] = true;
for(ll i = 0;i < adj[x].size();++i)
{
ll y = adj[x][i].ss; ll weight=adj[x][i].ff;
if(dist[y]>dist[x]+weight)
{
dist[y]=dist[x]+weight;
Q.push(mp(dist[y],y));
}
}
}
}
//complexity O((v+e)logV)
int main()
{
ll nodes, edges, x, y;
ll weight, minimumCost;
cin >> nodes >> edges;
for(ll i = 0;i < edges;++i)
{
cin >> x >> y >> weight;
adj[x].pb(mp(weight, y));
//adj[y].pb(mp(weight, x));
}
// Selecting 1 as the source node from where all distances are to be calculated
dijkstra(1);
for(int i=2;i<=nodes;i++)
cout<<dist[i]<<" ";
return 0;
}
```

0 votes • Reply • Message • Permalink

Alok Kumar a year ago

Solution with multiset

```
====================================
#include <bits/stdc++.h>
#define ll long long
#define pb push_back
#define mp make_pair
#define pll pair<ll,ll>
#define ff first
#define ss second
using namespace std;
const int MAX = 1e6 + 5;
const int MAX_DIST=1e9+7;
bool visited[MAX];
ll dist[MAX];
vector <pll> adj[MAX];
void init()
{
for(ll i=0;i<MAX;i++)
{
dist[i]=1e9;
visited[i]=false;
}
}
void dijkstra(ll source)
{
//memset(dist,MAX_DIST,sizeof(dist));//set distance to maximum for each vertex
init();
//memset(visited,false,sizeof(visited));//set visited array to false
//priority_queue<pll, vector<pll>, greater<pll> > Q;
multiset<pll> Q;
dist[source]=0;
Q.insert(mp(0, source));
while(!Q.empty())
{
// Select the edge with minimum weight
pll t = *Q.begin();
Q.erase(Q.begin());
ll x = t.ss;
// Checking for cycle
if(visited[x])
continue;
visited[x] = true;
for(ll i = 0;i < adj[x].size();++i)
{
ll y = adj[x][i].ss; ll weight=adj[x][i].ff;
if(dist[y]>dist[x]+weight)
{
dist[y]=dist[x]+weight;
Q.insert(mp(dist[y],y));
}
}
}
}
//complexity O(V+ElogV)
int main()
{
ll nodes, edges, x, y;
ll weight, minimumCost;
cin >> nodes >> edges;
for(ll i = 0;i < edges;++i)
{
cin >> x >> y >> weight;
adj[x].pb(mp(weight, y));
//adj[y].pb(mp(weight, x));
}
// Selecting 1 as the source node from where all distances are to be calculated
dijkstra(1);
for(int i=2;i<=nodes;i++)
cout<<dist[i] <<" ";
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Alok Kumar a year ago

I think Tutorial for Bellman Ford is to be given in full so that one can implement it.
I have implemented Bellman Ford using C++ STL as follows

```
===================================================================
#include<bits/stdc++.h>
#define ll long long
#define pll pair<ll,ll>
#define plll pair<ll,pll>
#define mp make_pair
#define pb push_back
#define ff first
#define ss second
#define all(x) (x).begin(),(x).end()
#define foreach(v,it) for(__typeof((v).begin())it=(v).begin();it!=(v).end();it++)
#define N 100005
using namespace std;
const ll MAX_DIST=1e9+5;
vector<pll> adj[N];
bool visited[N];
ll parent[N];
ll dist[N];
ll nodes,edges;
void initialize(ll n)
```

```
{
for(int i = 0;i <= n;++i)
dist[i] = MAX_DIST;
}
ll bellmanFord(ll src)
{
//step 1: Initialize distance from src to all other nodes to inifinity.
initialize(nodes);//initialize distance of all vertices to max distance value
dist[src]=0;
parent[src]=0;
//step 2:Relax all edges |V|-1 times as because a simple shortest path from src to any other vertex
// can have at most |V|-1 vertex
for(ll i=1;i<nodes;i++)//O(|V-1|)
{
foreach(adj[i],it)//O(|E|)
{
ll a,b,weight;
b=(*it).ss;
weight=(*it).ff;
if(dist[i]==MAX_DIST) continue;
if( dist[i]+weight<dist[b])//checking for relaxation
{
dist[b]=dist[i]+weight;
parent[b]=i;
}
}
}
//step 3: check is there any negative cycle
for(ll i=1;i<nodes;i++)
{
foreach(adj[i],it)
{
ll a,b,weight;
b=(*it).ss;
weight=(*it).ff;
if(dist[i]==MAX_DIST) continue;
if( dist[i]+weight<dist[b])//checking for relaxation
{
//cout<<"negative cycle exist "<<endl;
return i;
}
}
}
return -1;
}
void printPath(int s, int d){
if(s==d)
cout<<d<<" ";
else
{
printPath(s,parent[d]);
cout<<d<<" ";
}
}
int main()
{
cin>>nodes>>edges;
ll a,b,weight;
ll times=edges;
while(times--)
{
cin>>a>>b>>weight;
adj[a].pb(mp(weight,b));
}
ll start;
cout<<"Enter start vertex "<<endl;
cin>>start;
ll response=bellmanFord(start);
if(response==-1)
{
cout<<"No negative cycle "<<endl;
for(ll i=2;i<=nodes;i++)
{
cout<<"path : ";
printPath(start,i);
cout<<endl;
cout<<"distance from "<<start<<" to "<<i<<" is : "<<dist[i]<<endl;
}
}
else
{
cout<<"Negative cycle exist print cycle "<<endl;
printPath(response,parent[response]);
}
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Alok Kumar a year ago

Bellman Ford Algorithms using STL

```
========================================
#include<bits/stdc++.h>
#define ll long long
#define pll pair<ll,ll>
#define plll pair<ll,pll>
#define mp make_pair
#define pb push_back
#define ff first
#define ss second
#define all(x) (x).begin(),(x).end()
#define foreach(v,it) for(__typeof((v).begin())it=(v).begin();it!=(v).end();it++)
#define N 100005
using namespace std;
const ll MAX_DIST=1e9+5;
vector<pll> adj[N];
bool visited[N];
ll parent[N];
ll dist[N];
ll nodes,edges;
void initialize(ll n)
{
for(int i = 0;i <= n;++i)
{
dist[i] = MAX_DIST;
parent[i]=-1;
}
}
ll bellmanFord(ll src)
{
initialize(nodes);
```

```
dist[src]=0;
parent[src]=0;
for(ll k=1;k<nodes;k++)
{//O(|V-1|)
for(ll i=1;i<=nodes;i++)//O(|E|)
{
foreach(adj[i],it)
{
ll a,b,weight;
b=(*it).ss;
weight=(*it).ff;
if(dist[i]==MAX_DIST) continue;
if( dist[i]+weight<dist[b])//checking for relaxation
{
dist[b]=dist[i]+weight;
parent[b]=i;
}
}
}
}
//step 3: check is there any negative cycle
for(ll i=1;i<=nodes;i++)
{
foreach(adj[i],it)
{
ll a,b,weight;
b=(*it).ss;
weight=(*it).ff;
if( dist[i]+weight<dist[b])//checking for relaxation
{
return i;
}
}
}
return -1;
}
void printPath(int s, int d){
if(d==-1)
{
cout<<"unreachable vertex ";
return;
}
if(s==d)
cout<<d<<" ";
else
{
printPath(s,parent[d]);
cout<<d<<" ";
}
}
int main()
{
cin>>nodes>>edges;
ll a,b,weight;
ll times=edges;
while(times--)
{
cin>>a>>b>>weight;
adj[a].pb(mp(weight,b));
}
```

```
ll start;
cout<<"Enter start vertex "<<endl;
cin>>start;
ll response=bellmanFord(start);
if(response==-1)
{
cout<<"No negative cycle "<<endl;
for(ll i=2;i<=nodes;i++)
{
cout<<"path : ";
printPath(start,i);
cout<<endl;
cout<<"distance from "<<start<<" to "<<i<<" is : "<<dist[i]<<endl;
}
}
else
{
cout<<"Negative cycle exist print cycle "<<endl;
printPath(response,parent[response]);
}
return 0;
}
/*
*Input
8 11
1 8 8
8 7 1
7 6 -1
5 6 -1
4 5 3
3 4 1
3 2 1
1 2 10
7 2 -4
2 6 2
6 3 -2
*/
```

0 votes ● Reply ● Message ● Permalink

Saurabh Sharma a year ago

My solution:

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<ll, ll> ii;
typedef vector<ll> vi;
int main(){
ll N, M;cin>>N>>M;
vector<ii>G[N+1];
for(ll i = 0; i < M; i++){
ll a, b, w; cin>>a>>b>>w;
G[a].emplace_back(b, w);
}
vi D(N+1, 987654321);
// start vertex
set<ii> Q;
D[1] = 0;
Q.insert(ii(0, 1));
while(!Q.empty()) {
// again, fetch the closest to start element
// from "queue" organized via set
ii top = *Q.begin();
Q.erase(Q.begin());
ll v = top.second, d = top.first;

// here we do not need to check whether the distance
// is perfect, because new vertices will always
// add up in proper way in this implementation
for(auto it: G[v]) {
ll v2 = it.first, cost = it.second;
if(D[v2] > D[v] + cost) {
// this operation can not be done with priority_queue,
// because it does not support DECREASE_KEY
if(D[v2] != 987654321) {
Q.erase(Q.find(ii(D[v2],v2)));
}
D[v2] = D[v] + cost;
Q.insert(ii(D[v2], v2));
}
}
}
for(ll i = 2; i <= N; i++)cout<<D[i]<<" ";
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Saurabh Sharma a year ago

Another solution using priority_queue of stl:

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<ll, ll> ii;
typedef vector<ll> vi;
int main(){
ll N, M;cin>>N>>M;
vector<ii>G[N+1];
for(ll i = 0; i < M; i++){
ll a, b, w; cin>>a>>b>>w;
G[a].emplace_back(b, w);
}
vi D(N+1, 987654321);
// distance from start vertex to each vertex
priority_queue<ii,vector<ii>, greater<ii> > Q;
// priority_queue with reverse comparison operator,
// so top() will return the least distance
// initialize the start vertex, suppose it's zero
D[1] = 0;
Q.push(ii(0,1));
// iterate while queue is not empty
while(!Q.empty()) {
// fetch the nearest element
ii top = Q.top();
Q.pop();

// v is vertex index, d is the distance
ll v = top.second, d = top.first;
// this check is very important
// we analyze each vertex only once
// the other occurrences of it on queue (added earlier)
// will have greater distance
if(d <= D[v]) {
// iterate through all outcoming edges from v
for(auto it: G[v]) {
ll v2 = it.first, cost = it.second;
if(D[v2] > D[v] + cost) {
// update distance if possible
D[v2] = D[v] + cost;
// add the vertex to queue
Q.push(ii(D[v2], v2));
}
}
}
}
for(ll i = 2; i <= N; i++)cout<<D[i]<<" ";
return 0;
}
```

0 votes ● Reply ● Message ● Permalink

Dinesh Raj Edited a year ago

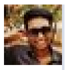
Can anyone point out whats the mistake in my code. Please I badly need help. I am stuck on this code from the last 2 days. I going crazy. please help.

Most of the output is correct, but for some vertices its incorrect.
It works on the given test case. But not on the submission test case. Gives wrong answer.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Map;
import java.util.PriorityQueue;
public class dijkstra {
public static void main(String[] args) throws IOException
{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n, edges;
boolean visited[];

// reading data
String[] arr = br.readLine().split(" ");
n = Integer.parseInt(arr[0]);
edges = Integer.parseInt(arr[1]);
visited = new boolean[n+1];
int dist[] = new int[n+1];

int graph[][] = new int[n+1][n+1];
for(int i=0;i<graph.length;i++)
Arrays.fill(graph[i], -1);


// Initialize dist
for(int i=0;i<dist.length;i++)
dist[i] = Integer.MAX_VALUE;
```

```
// Initialize visited
for (int i=0;i<visited.length;i++)
visited[i] = false;

String s;
while(true)
{
s = br.readLine();
if(s == null || s.equals(""))
break;
arr = s.split(" ");
int x = Integer.parseInt(arr[0]);
int y = Integer.parseInt(arr[1]);
int weight = Integer.parseInt(arr[2]);

graph[x][y] = weight;
//graph[y][x] = weight;
}

br.close();

dijikstra(graph, dist, visited);


for(int i=2;i<n+1;i++)
{
System.out.println(dist[i]+" ");
}
}

public static void dijikstra(int graph[][], int dist[], boolean visited[])
{
PriorityQueue<Pair> q = new PriorityQueue<Pair>(new Comparator<Pair>(){
public int compare(Pair a, Pair b) {
if(a.first < b.first)
return -1;
else if (a.first > b.first)
return 1;
else
return 0;
}

});


q.add(new Pair(0,1));
dist[1] = 0;

while(q.isEmpty() == false)
{
Pair p = q.peek();
q.poll();

if(visited[p.second] == true)
continue;

visited[p.second] = true;

for(int i=1;i<graph[p.second].length;i++)
{
if (graph[p.second][i] > -1)
{
int vertex = i;
int weight = graph[p.second][i];
int currDist = dist[vertex];
int calDist = dist[p.second] + weight;

if (calDist < currDist)
{
dist[vertex] = calDist;
q.add(new Pair(dist[vertex], vertex));
}
}
}
}
}


static class Pair{
public int first;
public int second;

public Pair(int first, int second){
this.first = first;
this.second = second;
}
@Override
public String toString() {
return "Pair [first=" + first + ", second=" + second + "]";
}
}
}
```

0 votes ● Reply ● Message ● Permalink


Mopati bharath a year ago

iam using priority queue instead of multi set why ami i getting wrong answer?

```
#include <iostream>
#include<queue>
#include<utility>
#include<vector>
using namespace std;
typedef pair<long long ,int> pl;
void prim(int x , vector <pl> adj[] ,bool v[],long long d[])
{

priority_queue<pl,vector<pl>,greater<pl> > pq;
pq.push(make_pair(0,x));
pl p;
d[x]=0;

while(!pq.empty())
{p=pq.top();
int m=p.second;

pq.pop();

if(v[m])
continue;

v[m]=true;
for(int i=0;i<adj[m].size();i++)
{int v=adj[m][i].second;
int weight=adj[m][i].first;
if(d[v]>d[m]+weight)
{
d[v]=d[m]+weight;
pq.push(adj[m][i]);}
}
}



}
int main()
{
int n,m;
cin>>n>>m;
vector <pl> adj[n+1];
bool v[n+1];
for(int i=0;i<n+1;i++)
{
v[i]=false;
}
for(int i=0;i<m;i++)
{
int a,b;
long long int c;
cin>>a>>b>>c;
adj[a].push_back(make_pair(c,b));


}
long long d[n+1];
for(int i=0;i<n+1;i++)
{
d[i]=1000000000;
}
prim(1, adj,v,d);
for(int i=2;i<n+1;i++)
{
cout<<d[i]<<" ";
}
}
```

0 votes • Reply • Message • Permalink

KUNWAR DESH DEEPAK SINGH a year ago

```
#include<bits/stdc++.h>
//DONE USING PRIORITY QUEUE AND ITS ACCEPETED YOU CAN HAVE A LOOK AT IT
using namespace std;
vector<pair<int,int> > v[10002];
int vis[10002];
int dis[10002];
int n,m,a,b,start,weight;
void dijkastra(int start){

priority_queue<pair<int,int>,vector<pair<int,int> >,greater<pair<int,int> > > Q;

Q.push(make_pair(0,start));
```

```
while(!Q.empty()){

pair<int,int> temp = Q.top();
Q.pop();

int y = temp.second;
if(vis[y]==0){

vis[y]=1;

for(int i=0;i<v[y].size();i++){
pair<int,int> t=v[y][i];
int z = t.second;
int w = t.first;

if(dis[z]>dis[y]+w){
dis[z]=dis[y]+w;
}
if(vis[z]==0){


Q.push(make_pair(dis[z],z));
}
}

}


}
}
int main() {

cin>>n>>m;
for(int i=0;i<=n;i++){
vis[i]=0;
dis[i]=100000000;
}

for(int i=0;i<m;i++){
cin>>a>>b>>weight;
v[a].push_back(make_pair(weight,b));
// v[b].push_back(make_pair(weight,a));
}
start=1;
dis[start]=0;
dijkastra(start);

for(int i=1;i<=n;i++){
if(i!=start){
if(dis[i]==100000000){
cout<<-1<<" ";
}
else{
cout<<dis[i]<<" ";
}
}
}
cout<<"\n";
for(int i=0;i<=n;i++){
v[i].clear();
}
```

```
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

KUNWAR DESH DEEPAK SINGH a year ago

IN THIS EXAMPLE GRAPH IS DIRECTED SO TAKE CARE OF THAT .....USUALLY PEOPLE CONSIDER UNDIRECTED GRAPHS...

▲ 0 votes ● Reply ● Message ● Permalink

mohammadmahdi abdollahpoor a year ago

I get Time Limit Exceeded Error , Im using dijkstra algorithm, can some one help me?

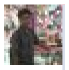
```
import java.util.*;
/**
* Created by ASUS on 22/09/2017.
*/
public class DikjestraShortestPath {
static class Edge {
int destination;
int weight;
Edge(int destination, int weight) {
this.destination = destination;
this.weight = weight;
}
}
static class Node {
int distance;
int source;
Node(int distance, int source) {
this.distance = distance;
this.source = source;
}
}
public static void main(String[] args) {
Scanner in = new Scanner(System.in);
int n = in.nextInt();
int m = in.nextInt();
ArrayList<Edge>[] graph = new ArrayList[n];
for (int i = 0; i < n; i++) {
graph[i] = new ArrayList<>();
}
for (int i = 0; i < m; i++) {
int a = in.nextInt();
int b = in.nextInt();
int w = in.nextInt();
graph[a - 1].add(new Edge(b - 1, w));
}
int[] distances = dijkstra(graph, n);
for (int i = 1; i < n; i++) {
if (distances[i] == -1) {
System.out.print("1000000000");
} else {
System.out.print(distances[i]);
}
System.out.print(" ");
}
}
public static int[] dijkstra(ArrayList<Edge>[] graph, int n) {
int[] distances = new int[n];
distances[0] = 0;
for (int i = 1; i < n; i++) {
distances[i] = -1;
}
Queue<Node> q = new PriorityQueue((o1, o2) -> {
Node n1 = (Node) o1;
Node n2 = (Node) o2;
return new Integer(n2.distance).compareTo(new Integer(n1.distance));
});
q.add(new Node(0, 0));
while (!q.isEmpty()) {
// System.out.println("q size " + q.size());
Node node = q.poll();//retrive and remove
for (int i = 0; i < graph[node.source].size(); i++) {
Edge edge = graph[node.source].get(i);
if (distances[edge.destination] == -1 || distances[edge.destination] > edge.weight + node.distance) {
q.add(new Node(node.distance + edge.weight, edge.destination));
distances[edge.destination] = node.distance + edge.weight;
}
}
}
return distances;
}
}
```

0 votes ● Reply ● Message ● Permalink

Siddhartha Narayana a year ago

I have an issue to raise for Bellman Ford Algorithm. The Data structure used for edges representation will not work in case of undirected graph. This is because if we do not place the 'from' and 'next' nodes correctly, then we may get erroneous results as 'from' and 'next' node are meaningless in an undirected graph. Kindly see and comply if I am correct.

0 votes ● Reply ● Message ● Permalink


Priyanshu Sharan Edited a year ago

getting time limit exceed for Bellman Ford's Algorithm: Any solution

```
#include <iostream>
#include<vector>
#include<limits.h>
using namespace std;
void bfa( vector< int > graph[], int v,int e)
{
int dist[v];
for(int i=0;i<v;i++)
{
dist[i]=INT_MAX;
}

dist[0]=0;
int a=v-1;
while(a--)
{
for(int i=0;i<e;i++)
{

if(dist[graph[i][0]]!=INT_MAX && dist[graph[i][0]] + graph[i][2] < dist[graph[i][1]] )
{
dist[graph[i][1]] = dist[graph[i][0]] + graph[i][2];
```

?

```
}

}
}

for(int i=1;i<v;i++)
{
cout<<dist[i]<<" ";
}
cout<<endl;
}
int main()
{
int v,e;
cin>>v>>e;
vector<int> graph[e];

for(int i=0;i<e;i++)
{
int a,b,c;
cin>>a>>b>>c;
graph[i].push_back(a-1); graph[i].push_back(b-1); graph[i].push_back(c);
}
bfa(graph,v,e);
return 0;
}
```

0 votes • Reply • Message • Permalink

Rachit Yadav a year ago

The time complexity of bellman ford is of the order of O(E3) in the worst case, here 'E - no of edges' very big, so use Dijkstra's algo.. and you will get ac

0 votes • Reply • Message • Permalink

Rachit Yadav a year ago

guys...
Firstly, its a directed graph
BTW here is the solution using min priority queue likewise as you used the concept in prims algorithm

```
#include <bits/stdc++.h>
#define INF 1e9
using namespace std;
long long int dist[10007];
bool visited[10007];
void dijkstra(vector<pair <int,int> > v[],int x)
{
priority_queue< pair<int,int> ,vector< pair<int,int> > , greater< pair<int,int> > > q;

pair <int,int> p;

q.push(make_pair(0,x));
dist[x] = 0;
while(!q.empty())
{
p = q.top();
q.pop();
x = p.second;

if(visited[x])
continue;

visited[x]=true;
for(int i=0;i<v[x].size();++i)
{
int e = v[x][i].second;
int w = v[x][i].first;

if(dist[e]>dist[x]+w)
{
dist[e] = dist[x]+w;
q.push(make_pair(dist[e],e));
}

}

}


}
int main()
{

vector< pair<int,int> > v[10007];
int n,m;
cin>>n>>m;
for(int i=0;i<10007;++i)
{
dist[i] = INF;
}
memset(visited,false,10007);

for(int i=0;i<m;++i)
{
int x,y,w;
cin>>x>>y>>w;

v[x].push_back(make_pair(w,y));

}

dijkstra(v,1);

for(int i=2;i<=n;++i)
{
cout<<dist[i]<<" ";
}
return 0;
}
```

0 votes • Reply • Message • Permalink

Vivek Kumar a year ago

Is there any case in which a node is not reachable from the source vertex?

0 votes • Reply • Message • Permalink

Abhinav Tripathi a year ago

What a worthless effort! Whoever wrote input#1 was probably dozing off. The number of lines in input is not equal to M.
As someone else on comments pointed "m is 1000000, but there are actually 999200 edges given....".
Now I want to ask those who say they have Accepted answers, how???? HOW????

0 votes • Reply • Message • Permalink

Sushant Sankpal a year ago

which was the correct algorithm for finding minimum distance in graph?

0 votes ● Reply ● Message ● Permalink

Attyuttam Saha 10 months ago

just a question, if there are a large number of vertices and i have to find the shortest distance between every pair of vertices then it is not possible to use the floyd warshall's algorithm due to space issues.
Then why is floyd warshall necessary ?
Also as the question of space complexity arises, if we apply dijkstra on all the vertices one by one wont the time complexity be O(V^3) ?

0 votes ● Reply ● Message ● Permalink

Sumbul Fatima 10 months ago

do the time limit exceeds for bellman ford algorithm?

0 votes ● Reply ● Message ● Permalink

borra subba jaya prakash 10 months ago

```
class MyPair2 implements Comparable<MyPair2>
{
private final int key;
private final int value;
public MyPair2(int aKey, int aValue)
{
key = aKey;
value = aValue;
}
public int key() { return key; }
public int value() { return value; }

public int compareTo(MyPair2 b) {
if(value>=b.value()){
return 1;
}else if(value<b.value()){
return -1;
}else{
return 0;
}
}
}
public class Dijkstras {

public Map<Integer,List<MyPair2>> krus = new TreeMap<Integer,List<MyPair2>>();
public Map<Integer,Integer> dis = new HashMap<Integer,Integer>();
public HashMap<Integer,Boolean> visited = new HashMap<Integer,Boolean>();
public void dij(MyPair2 x)
{
PriorityQueue<MyPair2> queue = new PriorityQueue<MyPair2>();


queue.add(x);



dis.put(x.key(), 0);

while(! queue.isEmpty())
{
MyPair2 ele = queue.remove();
```

```
if( !visited.get(ele.key()))
{

visited.put(ele.key(), true);





if(krus.containsKey(ele.key()))
{
for(MyPair2 temp : krus.get(ele.key()))
{

if( dis.get(ele.key()) + temp.value() < dis.get(temp.key()))
{
dis.put(temp.key(), temp.value()+dis.get(ele.key()));
queue.add(temp);
}


}
}
}

}

}

public static void main(String[] args) {
Dijkstras obj = new Dijkstras();

Scanner sc = new Scanner(System.in);
int nodes = sc.nextInt();
int edges = sc.nextInt();
MyPair2 pair = null;

for(int i=0;i<nodes;i++)
{
obj.visited.put(i+1, false);
obj.dis.put(i+1, 1000000000);
}
for(int i = 0 ; i < edges ;i++)
{
int x = sc.nextInt();
int y = sc.nextInt();
int cost = sc.nextInt();


pair = new MyPair2(y,cost);

if(obj.krus.containsKey(x))
{
obj.krus.get(x).add(pair);
}
else
{
List<MyPair2> list = new ArrayList<MyPair2>();
list.add(pair);
obj.krus.put(x, list);
}
}
obj.dij(new MyPair2(1,0));
obj.dis.remove(1);
for(int t : obj.dis.keySet())
{
System.out.print(obj.dis.get(t));
System.out.print(" ");
}





}
}
```

I am getting time limit exceeded with this code . Can any one help with the same

0 votes ● Reply ● Message ● Permalink

Chavva Venkata Rama Syam Phanindra 10 months ago

see it is directed graph while doing the problem

0 votes ● Reply ● Message ● Permalink

Akshive Pandey 8 months ago

```
#include<iostream>
#include<vector>
#include<bits/stdc++.h>
using namespace std;
#define MAX 1000000000
int main(){
int N, M;
cin>>N>>M;

vector<pair<int, int>> adj[N+1];
int x1, y, w;
for(int i = 0; i < M; i++){
cin>>x1>>y>>w;
adj[x1].push_back(make_pair(y, w));
}
int dis[N+1];
for(int i = 0; i <= N; i++){dis[i] = MAX;}
dis[1] = 0;
queue<int> q;
/*for(int i = 0; i <= N; i++){
if(dis[i] != MAX){cout<<dis[i];}
else{cout<<"1000000000";}
cout<<endl;
}*/
q.push(1);
while(!q.empty()){
int x = q.front(); q.pop();
for(int i = 0; i < adj[x].size(); i++){
if(dis[adj[x][i].first] > dis[x] + adj[x][i].second){
dis[adj[x][i].first] = dis[x] + adj[x][i].second;
q.push(adj[x][i].first);
}
}
}
for(int i = 2; i <= N; i++){
if(dis[i] != INT_MAX){
cout<<dis[i]<<" ";
}
else{
cout<<"1000000000 ";
}
}
cout<<endl;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Xiaohua Yan 8 months ago

C++ solution:

```
/*
// Sample code to perform I/O:
cin >> name; // Reading input from STDIN
cout << "Hi, " << name << ".\n"; // Writing output to STDOUT
// Warning: Printing unwanted or ill-formatted data to output will cause the test cases to fail
*/
// Write your code here
#include <iostream>
#include <limits>
#include <queue>
#include <vector>
using namespace std;
int main() {
int M, N;
cin >> N >> M;

vector<vector<pair<int, int>>> adj(N);
for (int i = 0; i < M; ++i) {
int src, dest, weight;
cin >> src >> dest >> weight;
adj[src - 1].emplace_back(dest - 1, weight);
}

// compute shortest paths for all dest vertices
vector<bool> visited(N, false);
vector<int> distances(N, numeric_limits<int>::max());
priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> Q;

distances[0] = 0;
Q.push(make_pair(0, 0));

while (!Q.empty()) {
auto v = Q.top();
auto vIndex = v.second;
auto dist = v.first;
Q.pop();
if (visited[vIndex]) {
continue;
}
visited[vIndex] = true;

for (auto neighbor : adj[vIndex]) {
auto weight = neighbor.second;
auto nIndex = neighbor.first;
if (dist + weight < distances[nIndex]) {
```

```
distances[nIndex] = dist + weight;
Q.push(make_pair(distances[nIndex], nIndex));
}
}
}

for (size_t i = 1; i < distances.size(); ++i) {
cout << distances[i] << " ";
}

}
```

▲ 0 votes ● Reply ● Message ● Permalink

영진 김 8 months ago

is it wrong if uses floyd warshall algorithm??? now i uses but got the Runtime error

▲ 0 votes ● Reply ● Message ● Permalink

skand rajmeet 7 months ago

Can somebody help why my code is giving runtime error(SIGSEGV) (Bellman ford algo)

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
int n,m;
cin>>n>>m;
vector <int> v[1000000];
for(int i=0;i<m;i++){
int x,y,z;
cin>>x>>y>>z;
v[i].push_back(x);
v[i].push_back(y);
v[i].push_back(z);
}
int d[n];
for(int i=0;i<n;i++){
d[i]=1e5;
}
d[0]=0;
for(int i=0;i<n-1;i++){
int j=0;
while(v[j].size()!=0){
if(d[v[j][0]-1]+v[j][2]<d[v[j][1]-1])
d[v[j][1]-1]=d[v[j][0]-1]+v[j][2];
j++;
}
}
for(int i=1;i<n;i++)
cout<<d[i]<<" ";
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Shubham Mahawar 6 months ago

```
for(int i = 0; i < m + 2; i++){
v[i].clear();
dis[i] = 2e9;
}
```

why in the bellman ford's algo here loop runs upto m+2?

▲ 0 votes ● Reply ● Message ● Permalink

Shubham Mahawar 6 months ago

what is p.s and p.f means in Dijkstra's Algorithm?

0 votes ● Reply ● Message ● Permalink

ky lee 6 months ago

why the data have 10000 vertices and 1000000 edges
But when I download the data - there is only 999200 edges ?

0 votes ● Reply ● Message ● Permalink

ky lee 6 months ago

My code in python does not work again
Anyone could help me ??? :<

```
# Write your code here
import Queue
def shortest_path(edge_dict, n):
distance = [-1 for i in range(0, n)]
visit = [False for i in range(0,n)]

distance[0] = 0
q = Queue.PriorityQueue()
q.put((0, 0))
while q.qsize() > 0:
(w, v) = q.get()

if not visit[v]:

for i in range(0, n):
if v in edge_dict:
if i in edge_dict[v]:
if distance[i] > (distance[v] + edge_dict[v][i]) or distance[i] == -1:
distance[i] = distance[v] + edge_dict[v][i]
q.put((distance[i], i))
visit[v] = True

print ' '.join(map(str, distance[1:]))

edge_dict = {}
[n,m] = [int(x) for x in raw_input().split()]
check = True
while check:
try:

[a,b,w] = [long(x) for x in raw_input().split()]
a -= 1
b -= 1
if a not in edge_dict:
edge_dict[a] = {b:w}
else:
edge_dict[a][b] = w

# if b not in edge_dict:
# edge_dict[b] = {a:w}
# else:
# edge_dict[b][a] = w
except:
check = False
#print edge_dict
#print len(edge_dict)
shortest_path(edge_dict, n)
```

0 votes • Reply • Message • Permalink

Ravi Kumar Tahlan 6 months ago

Working C++ code
===============

```
#include<bits/stdc++.h>
#define pii pair< int, int >
using namespace std;
int main(){
int n,m,a,b,w;
cin>>n>>m;

bool vis[n+1];
int dis[n+1];

for(int i = 1;i <= n; i++ ){
dis[i] = pow(10,9);
vis[i] = false;
}
dis[1]=0;
vector< pii > adj[n+1];
for(int i = 1; i <= m;i++){
cin>>a>>b>>w;
adj[a].push_back(make_pair(w,b));
//adj[b].push_back(make_pair(w,a));
}


priority_queue< pii, vector<pii>, greater<pii> > Q;

pii p;

Q.push(make_pair(0,1));

while(!Q.empty()){
p = Q.top();
Q.pop();
int x = p.second;

if(vis[x])
continue;

vis[x] = true;

for(int i = 0;i < adj[x].size();i++){
int u = adj[x][i].second;
int we = adj[x][i].first;
if(!vis[u]){
if(dis[u] > (dis[x]+we)){
dis[u] = dis[x]+we;
adj[x][i].first = dis[u];
Q.push(adj[x][i]);
}
}
}

}

for(int i = 2;i <= n; i++){
cout<<dis[i]<<" ";
}


return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

ak389 6 months ago

simple solution

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
#define boost ios_base::sync_with_stdio(0)
#define endl "\n"
vector<pair<ll,ll> >adj[1000001];
ll c=1000000000;
bool visited[1000001];
ll dist[1000001];
void in(ll n)
{
for(ll i=0;i<=n;i++)
{
visited[i]=0;
dist[i]=c;
}
}
void djk()
{
dist[1]=0;
multiset<pair<ll,ll> >s;
s.insert({0,1});
while(!s.empty())
{
ll a,b,w;
pair<ll,ll> p = *s.begin();
s.erase(s.begin());
a = p.second;
if(visited[a])
continue;
visited[a]=1;
for(ll i=0;i<adj[a].size();i++)
{
w=adj[a][i].first;
b=adj[a][i].second;
if(dist[a]+w<dist[b])
{
dist[b]=dist[a]+w;
s.insert({dist[b],b});
}
}
}
}
int main()
{
boost;
ll n,e,i;
cin>>n>>e;
in(n);
for(i=0;i<e;i++)
{
ll a,b,w;
cin>>a>>b>>w;
adj[a].push_back({w,b});
}
djk();
for(i=2;i<=n;i++)
{
cout<<dist[i]<<" ";
}
cout<<endl;

return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Shashank Yadav 6 months ago

is it even possible to do in python TLE

▲ 0 votes ● Reply ● Message ● Permalink

Shashank Yadav 6 months ago

yes using min heap we can avoid TLE
my python code uses it but its giving wrong answer see if you can correct it

▲ 0 votes ● Reply ● Message ● Permalink

Sanjay Kumar 6 months ago

With Java, even if I comment dijkstra code, i.e. reading input only, gives TLE. Also Fast I/O is not working (throwing Runtime Error - NZEC). Also input data set is correct although it is showing less no of edges, since C++ program just works fine.

▲ 0 votes ● Reply ● Message ● Permalink

Shashank Yadav 6 months ago

can someone point out the error in this python code it gives wrong answer

```
#uses min-heap
from sys import stdin,stdout
n,e = map(int,stdin.readline().rstrip().split())
e =999200
a = [[(0,0)] for i in range(0,n+100)]
visited = [0]*(n+1)
dis = [1000000000]*(n+1)
dis[1] = 0
minHeap = []
p,q,r = -1,-1,-1
for i in range(0,e):
p,q,r = map(int, stdin.readline().rstrip().split())
a[p].append((q,r)) #add the tuple[from]=>[(to,w1)],(to,w2)] eg: [1] => [(2,10)(4,50)] (to,weight)
#input complete
def minHeapify(a,i,l): #(minHeap,i)
#if no child exist
if((i*2+1>l) and (i*2+2>l)):
return
#if only left child exist
if(i*2+1<=l and i*2+2>l): #our counting is from 0
if(a[i*2+1][1]<a[i][1]):
t = a[i*2+1]
a[i*2+1] = a[i]
a[i] = t
minHeapify(a,i*2+1,l)
#if both child exist and left child is samller tahn both parent and right child
```

```
elif(a[i*2+1][1]<a[i*2+2][1] and a[i*2+1][1]<a[i][1]):
t = a[i*2+1]
a[i*2+1] = a[i]
a[i] = t
minHeapify(a,i*2+1,l)
#if both child exist and left child is samller
elif(a[i*2+2][1]<a[i*2+1][1] and a[i*2+2][1]<a[i][1]):
t = a[i*2+2]
a[i*2+2] = a[i]
a[i] = t
minHeapify(a,i*2+2,l)
def getMin(a):
x = []
if(len(a)>0):
x = a[0]
g = a.pop() #last element poped
if(len(a)>0):
a[0] = g
minHeapify(a,0,len(a)-1)
return x
def insert(a,z):
a.append(z)
l = len(a)-1
while(l>0):
pIndex = int((l-1)/2)
if(a[pIndex][1]>a[l][1]):
#swap
t = a[pIndex]
a[pIndex] = a[l]
a[l] = t
l=int((l-1)/2)
else:
break
#insert 1st (1,0) in min heap
insert(minHeap,[1,0])
while(len(minHeap)!=0):
t = getMin(minHeap)
#check if already visited
if(visited[t[0]] == 1):
continue
#mark it
visited[t[0]] = 1
x = t[0]
y = t[1]
for i in range(1,len(a[x])):
p = a[x][i][0]
q = a[x][i][1]
if(dis[x] + q < dis[p]):
dis[p] = dis[x] + q
insert(minHeap,[p,dis[p]])
ans = " "
for i in range(2,n+1):
ans = ans +" "+ str(dis[i])
print(ans.strip())
```

▲ 0 votes ● Reply ● Message ● Permalink

Anuj Kumar Singh 6 months ago

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

```
void shortestpath(int n,vector<pair<int,int> > v[],int src)
{
int INF=1000000000;
int dis[n+1];
for(int i=1;i<=n;i++)
dis[i]=INF;
set<pair<int,int>> st;
st.insert(make_pair(0,src));
dis[src]=0;
while(!st.empty())
{
pair<int,int> temp=*(st.begin());
st.erase(st.begin());
for(int i=0;i<v[temp.second].size();i++)
{
if(dis[v[temp.second][i].first]>dis[temp.second]+v[temp.second][i].second)
{
if(dis[v[temp.second][i].first]!=INF)
st.erase(st.find(make_pair(dis[v[temp.second][i].first],v[temp.second][i].first)));
dis[v[temp.second][i].first]=dis[temp.second]+v[temp.second][i].second;
st.insert(make_pair(dis[v[temp.second][i].first],v[temp.second][i].first));
}
}
}
for(int i=2;i<=n;i++)
printf("%d ",dis[i]);
}
int main()
{
int n,m,a,b,w;
scanf("%d%d",&n,&m);
vector <pair<int,int> > v[n+1];
for(int i=0;i<m;i++)
{
scanf("%d%d%d",&a,&b,&w);
v[a].push_back(make_pair(b,w));
}
shortestpath(n,v,1);
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Darren Mistry 5 months ago

c++ code using priority queue:

```
#include<iostream>
#include<utility>
#include<climits>
#include<vector>
#include<queue>
using namespace std;
int n,m;
const int MAX = 1e6 + 5;
vector<pair<int,int>> *adj;
int dist[MAX];
int djks(int s){
dist[s]=0;
priority_queue<pair<int,int>> q;
q.push(make_pair(s,dist[s]));
while(!q.empty()){
pair<int,int> p = q.top();
q.pop();
vector<pair<int,int>>::iterator i;
for(i=adj[p.first].begin();i!=adj[p.first].end();i++){
if(dist[(*i).first]>dist[p.first]+(*i).second){
dist[(*i).first]=dist[p.first]+(*i).second;
q.push(make_pair((*i).first,dist[(*i).first]));
}
}
}
}
int main(){
cin>>n>>m;
adj = new vector<pair<int,int>>[n+1];
for(int i=0;i<m;i++){
int x,y,w;
cin>>x>>y>>w;
adj[x].push_back(make_pair(y,w));
}
for(int i=0;i<MAX;i++){
dist[i]=INT_MAX;
}
djks(1);
for(int i=2;i<=n;i++)
cout<<dist[i]<<" ";
return 0;
}
```

0 votes • Reply • Message • Permalink

Vignesh 5 months ago

bfs works for weights with any value. Then why this tutorial has mentioned that bfs could solve only for weight 1.

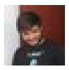
0 votes • Reply • Message • Permalink

Ravi Kumar 5 months ago

Can anyone please explain why, the time limit is exceeding in this code... thanks

```
#include<bits/stdc++.h>
using namespace std;
#define fast() ios_base::sync_with_stdio(0);cin.tie(NULL);cout.tie(NULL)
vector<int> adj[100005];
map < pair <int, int>, int> mp;
int dist[100005] = {1000005};
void shortath(int s)
{
for(int i = 0; i < 100005; i++)
dist[i] = 1000005;
dist[s] = 0;
queue <int > q;
q.push(s);
while(!q.empty())
{
//cout<< " see";
int p = q.front();
q.pop();
for(unsigned int i = 0;i < adj[p].size(); i++)
{
if(dist[adj[p][i]] > dist[p] + mp[ make_pair(p,adj[p][i]) ] )
dist[adj[p][i]] = dist[p] + mp[make_pair( p,adj[p][i]) ];
q.push(adj[p][i]);
}
}
}


int main()
{
fast();
int M, N, i, x, y, w;
cin >> N>> M;
for(i = 0;i < M; i++)
{
cin >> x>> y >>w;
adj[x].push_back(y);
//adj[y].push_back(x);
mp[make_pair(x, y)]= w;
}
shortath(1);
for(i = 2;i <= N;i++)
cout << dist[i]<<" ";
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink



Rishabh Agrawal 3 months ago

what is wrong with my code?Can anyone explain.
Thanks in advance.

```
#include <bits/stdc++.h>
using namespace std;
typedef long l;
typedef long long ll;
#define fi(i,a,b) for(int i=a;i<b;i++)
#define fl(i,a,b) for(ll i=a;i<b;i++)
#define si(c) scanf("%d",&c)
#define sl(c) scanf("%lld",&c)
#define pi(m) printf("%d ",m)
#define pl(m) printf("%lld\n",m)
#define pb push_back
#define mp make_pair
#define MOD 1000000007
#define boost ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL)
typedef pair<ll,ll> PL;
typedef pair<int,int> PI;
int n;
int dist[1000001];
void dijkstra(vector<PI> adj[])
{


int vis[n+1];
for(int i=1;i<=n;i++)
{
dist[i]=INT_MAX;
vis[i]=0;
}

multiset<PI> ms;

dist[1]=0;
ms.insert(mp(0,1));
while(!ms.empty())
{
PI p=*ms.begin();
ms.erase(ms.begin());
int u=p.second;
if(!vis[u])
{
//cout<<"u="<<u<<endl;
vis[u]=1;
for(int i=0;i<(adj[u].size());i++)
{
int wt=adj[u][i].first;
int v=adj[u][i].second;
//cout<<"v="<<v<<endl;
if(dist[u]!=INT_MAX&&!vis[v]&&dist[u]+wt<dist[v])
{

dist[v]=wt+dist[u];
ms.insert(adj[u][i]);
}
}

}
}




}
int main()
{
//boost;
int m;

si(n);si(m);

vector<PI> adj[n+1];
fi(i,0,m)
{
int a,b,wtg;
si(a);si(b);si(wtg);
adj[a].pb(mp(wtg,b));
}

dijkstra(adj);
fi(i,2,n+1)
{
if(dist[i]==INT_MAX)
printf("1000000000 ");
else
pi(dist[i]);
}




return 0;
}
```

0 votes • Reply • Message • Permalink

Tejas Duseja 3 months ago

Anyone whose submission in java getting accepted. Please help i am getting TLE error even though with O(V+E) approach

▲ 0 votes ● Reply ● Message ● Permalink

SJ 3 months ago

Here is my C++ code using priority_queue

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define pll pair<ll,ll>
constexpr ll M=pow(10,9),N=pow(10,5)+1;
bool comp(const pll &a,const pll &b)
{
return a.first>b.first;
}
vector<pll>adj[N];
vector<ll> dist(N+1,M);
priority_queue<pll,vector<pll>,function<bool(pll,pll)> >pq(comp);
bool vis[N+1];
void dijkstra()
{
pq.push(make_pair(0,1));//First Term Distance,Second Term Vertex
while(!pq.empty())
{
pll p=pq.top();
pq.pop();
ll d=p.first,v=p.second;
if(vis[v]) continue;
vis[v]=true;
if(d<=dist[v])
{
dist[v]=d;
for(int i=0;i<(int)adj[v].size();i++)
{
pll q=adj[v][i];
if(dist[q.first]>=d+q.second){
dist[q.first]=d+q.second;
pq.push(make_pair(d+q.second,q.first));
}
}
}
}
}
int main()
{
int n,e,i,e1,e2,val;
cin>>n>>e;
for(i=0;i<e;i++)
{
cin>>e1>>e2>>val;
adj[e1].push_back(make_pair(e2,val));
}
dijkstra();
for(i=2;i<=n;i++)
cout << dist[i] << ' ';
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink