Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

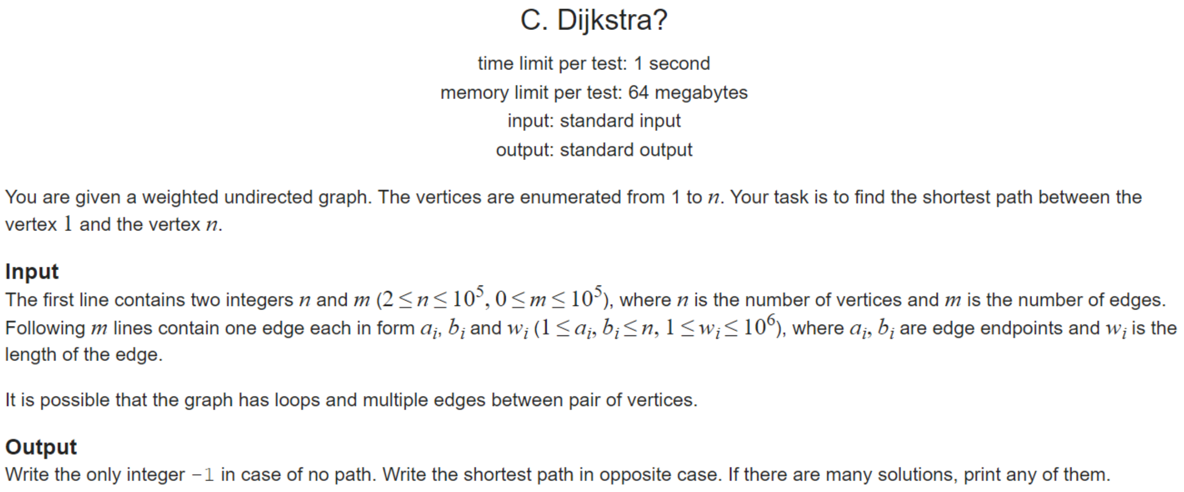
RAPORT

Lucrare de laborator nr.1

Structuri de date si algoritmi

A efectuat: st. gr.FAF-212 Lupascu Felicia

A verificat: dr. conf. Univ S. Corlat



C. Dijkstra?

time limit per test: 1 second
memory limit per test: 64 megabytes
input: standard input
output: standard output

You are given a weighted undirected graph. The vertices are enumerated from 1 to n. Your task is to find the shortest path between the vertex 1 and the vertex n.

Input

The first line contains two integers n and m ($2 \le n \le 10^5$, $0 \le m \le 10^5$), where n is the number of vertices and m is the number of edges. Following m lines contain one edge each in form a_i, b_i and w_i ($1 \le a_i, b_i \le n, 1 \le w_i \le 10^6$), where a_i, b_i are edge endpoints and w_i is the length of the edge.

It is possible that the graph has loops and multiple edges between pair of vertices.

Output

Write the only integer -1 in case of no path. Write the shortest path in opposite case. If there are many solutions, print any of them.

Dijkstra's algorithm finds the shortest path between any two graph vertices. It differs from the minimum spanning tree because the shortest distance between two vertices might not include all the vertices of the graph.

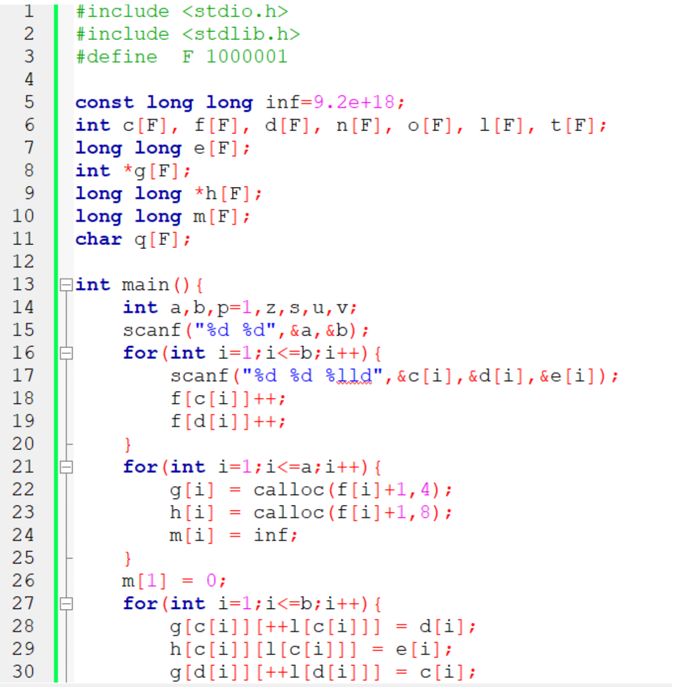
I use Dijkstra to find the shortest path value, and the shortest path should be recorded here. Here's how to record the path:
1. Using the Dijkstra algorithm to find the shortest path. The relaxation operation is actually a process of continuously selecting the shortest path. Each relaxation is a process of " selecting a path shorter than the current path ". In the end, The end of the relaxation operation means that the selection of the shortest path is determined.
2. Because the choice of the path exists in the relaxation operation, we should record the path in the relaxation operation. In the Dijkstra algorithm, each relaxation operation uses a " point with the shortest path determined " to relax other points, that is, the edge used for relaxation each time point from the point with the shortest path to the "shortest path ".

At each step:

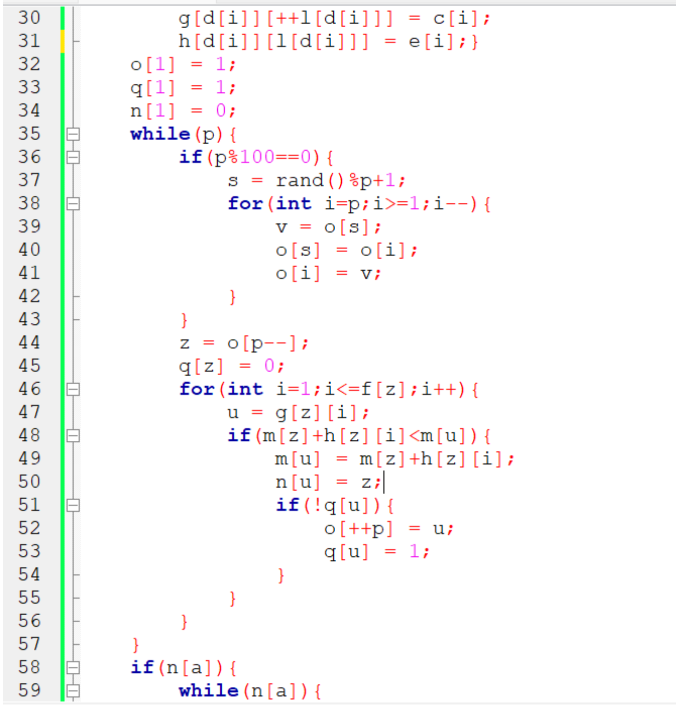
1) Pick the closest unknown vertex
2) Add it to known vertices
3) Update distances

The algorithm works by building a set of nodes that have a minimum distance from the source.

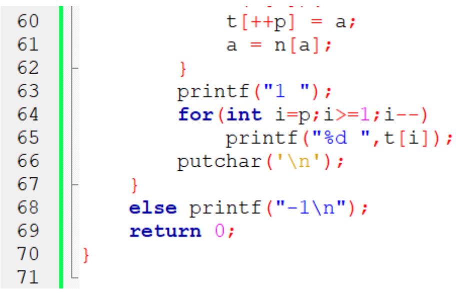
No Heaps Implementation



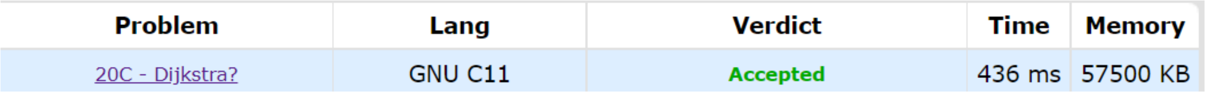
```c
#include <stdio.h>
#include <stdlib.h>
#define  F 1000001

const long long inf=9.2e+18;
int c[F], f[F], d[F], n[F], o[F], l[F], t[F];
long long e[F];
int *g[F];
long long *h[F];
long long m[F];
char q[F];

int main(){
    int a,b,p=1,z,s,u,v;
    scanf("%d %d",&a,&b);
    for(int i=1;i<=b;i++){
        scanf("%d %d %lld",&c[i],&d[i],&e[i]);
        f[c[i]]++;
        f[d[i]]++;
    }
    for(int i=1;i<=a;i++){
        g[i] = calloc(f[i]+1,4);
        h[i] = calloc(f[i]+1,8);
        m[i] = inf;
    }
    m[1] = 0;
    for(int i=1;i<=b;i++){
        g[c[i]][++l[c[i]]] = d[i];
        h[c[i]][l[c[i]]] = e[i];
        g[d[i]][++l[d[i]]] = c[i];
```



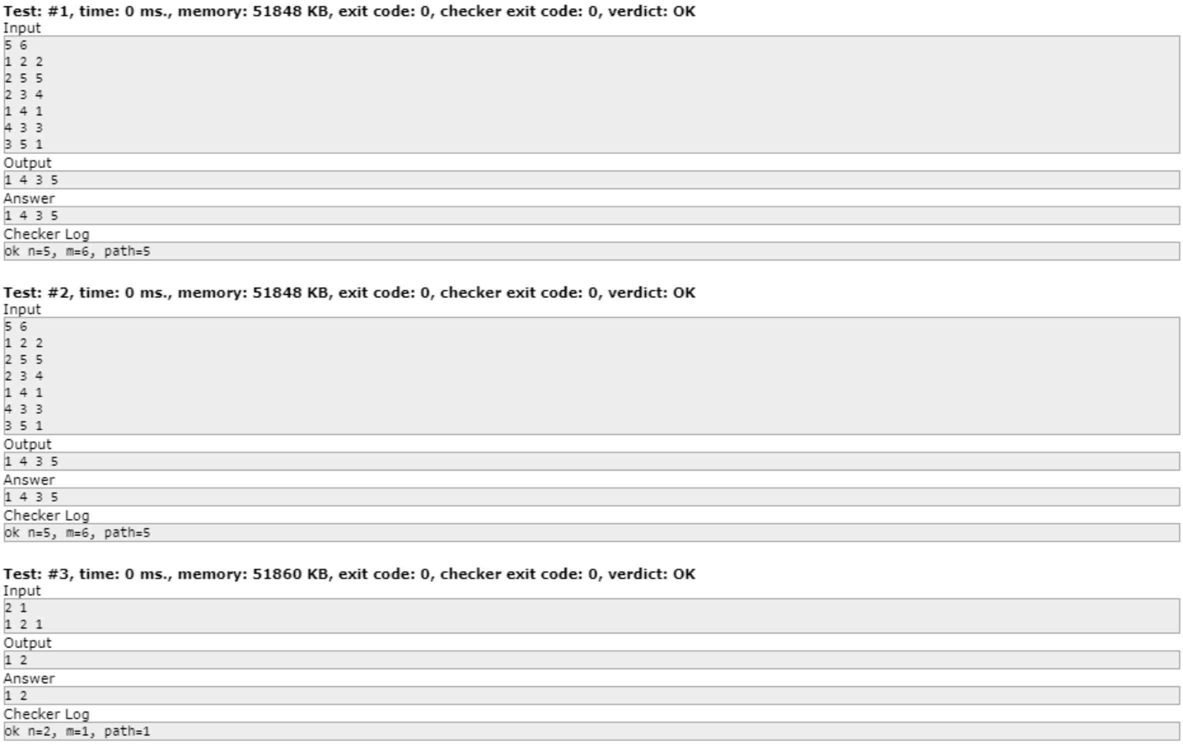
```c
        g[d[i]][++l[d[i]]] = c[i];
        h[d[i]][l[d[i]]] = e[i];}
    o[1] = 1;
    q[1] = 1;
    n[1] = 0;
    while(p){
        if(p%100==0){
            s = rand()%p+1;
            for(int i=p;i>=1;i--){
                v = o[s];
                o[s] = o[i];
                o[i] = v;
            }
        }
        z = o[p--];
        q[z] = 0;
        for(int i=1;i<=f[z];i++){
            u = g[z][i];
            if(m[z]+h[z][i]<m[u]){
                m[u] = m[z]+h[z][i];
                n[u] = z;
                if(!q[u]){
                    o[++p] = u;
                    q[u] = 1;
                }
            }
        }
    }
    if(n[a]){
        while(n[a]){
```



```
            t[++p] = a;
            a = n[a];
        }
        printf("1 ");
        for(int i=p;i>=1;i--)
            printf("%d ",t[i]);
        putchar('\n');
    }
    else printf("-1\n");
    return 0;
}
```



Problem	Lang	Verdict	Time	Memory
20C - Dijkstra?	GNU C11	Accepted	436 ms	57500 KB



Test: #1, time: 0 ms., memory: 51848 KB, exit code: 0, checker exit code: 0, verdict: OK

Input

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

Output

```
1 4 3 5
```

Answer

```
1 4 3 5
```

Checker Log

```
ok n=5, m=6, path=5
```

Test: #2, time: 0 ms., memory: 51848 KB, exit code: 0, checker exit code: 0, verdict: OK

Input

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

Output

```
1 4 3 5
```

Answer

```
1 4 3 5
```

Checker Log

```
ok n=5, m=6, path=5
```

Test: #3, time: 0 ms., memory: 51860 KB, exit code: 0, checker exit code: 0, verdict: OK

Input

```
2 1
1 2 1
```

Output

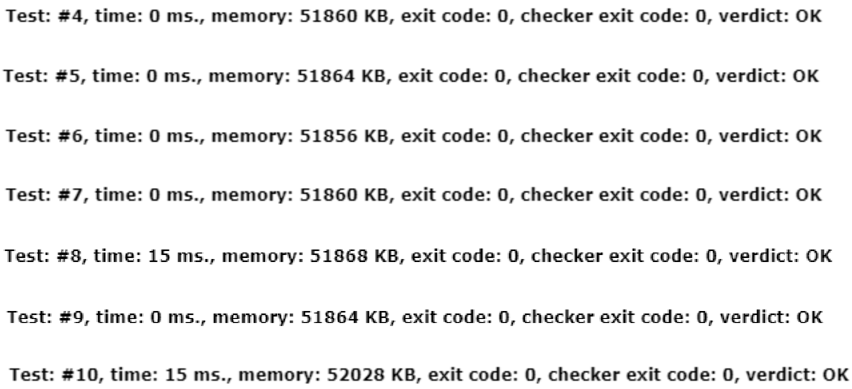
```
1 2
```

Answer

```
1 2
```

Checker Log

```
ok n=2, m=1, path=1
```



Test: #4, time: 0 ms., memory: 51860 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #5, time: 0 ms., memory: 51864 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #6, time: 0 ms., memory: 51856 KB, exit code: 0, checker exit code: 0, verdict: OK

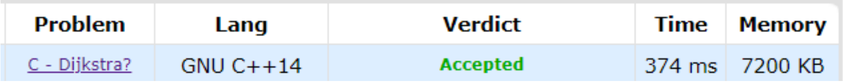
Test: #7, time: 0 ms., memory: 51860 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #8, time: 15 ms., memory: 51868 KB, exit code: 0, checker exit code: 0, verdict: OK

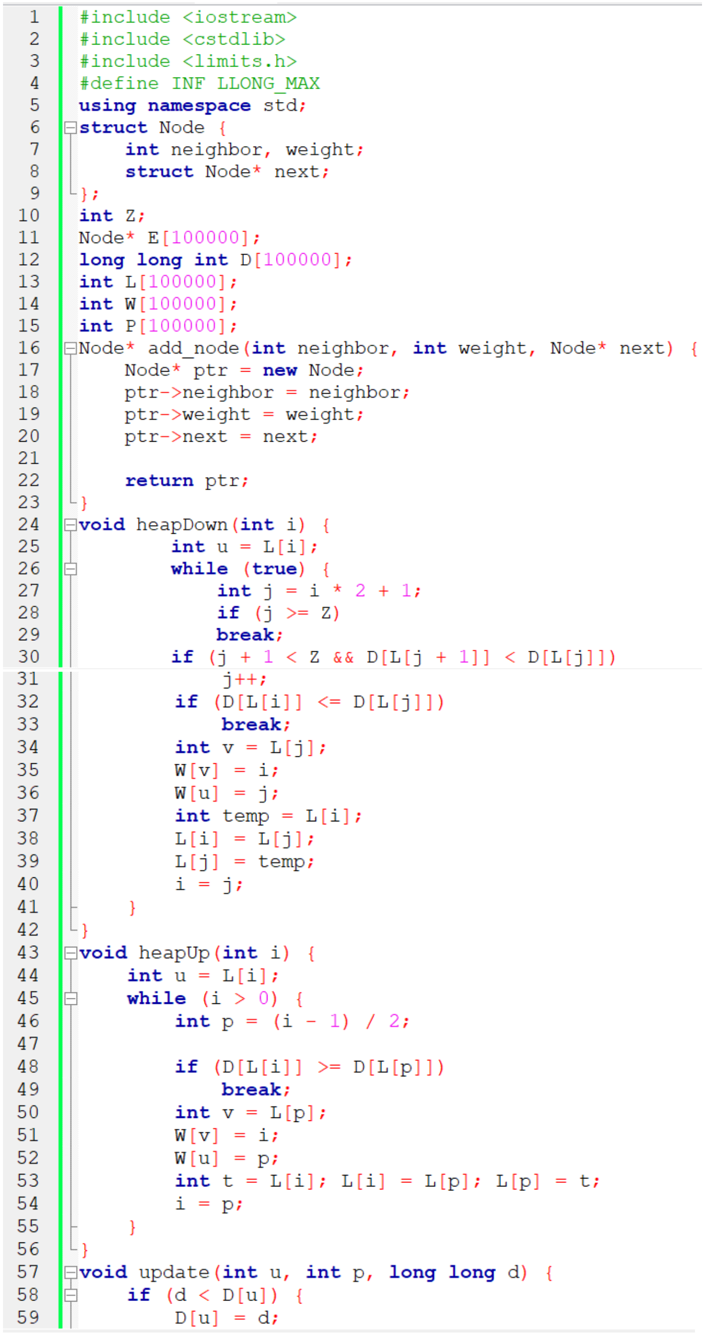
Test: #9, time: 0 ms., memory: 51864 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #10, time: 15 ms., memory: 52028 KB, exit code: 0, checker exit code: 0, verdict: OK

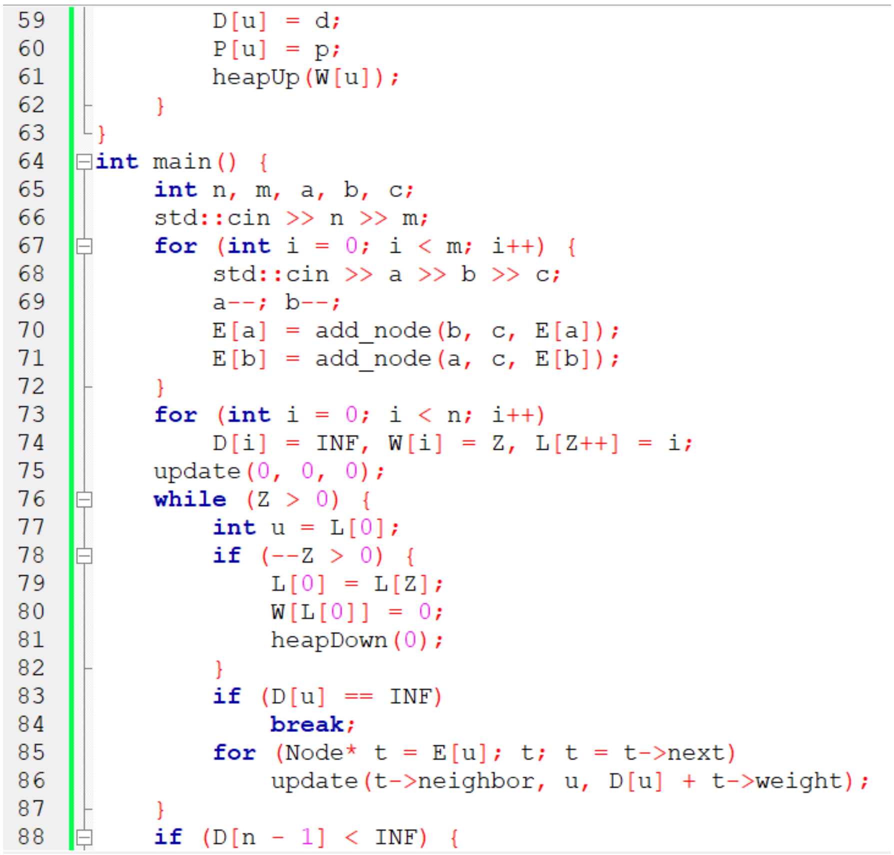
2.



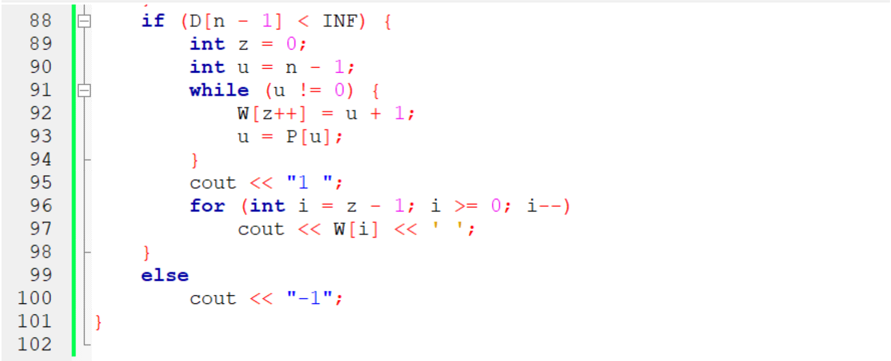
Problem	Lang	Verdict	Time	Memory
C - Dijkstra?	GNU C++14	Accepted	374 ms	7200 KB



```
#include <iostream>
#include <cstdlib>
#include <limits.h>
#define INF LLONG_MAX
using namespace std;
struct Node {
    int neighbor, weight;
    struct Node* next;
};
int Z;
Node* E[100000];
long long int D[100000];
int L[100000];
int W[100000];
int P[100000];
Node* add_node(int neighbor, int weight, Node* next) {
    Node* ptr = new Node;
    ptr->neighbor = neighbor;
    ptr->weight = weight;
    ptr->next = next;

    return ptr;
}
void heapDown(int i) {
        int u = L[i];
        while (true) {
            int j = i * 2 + 1;
            if (j >= Z)
            break;
        if (j + 1 < Z && D[L[j + 1]] < D[L[j]])
            j++;
        if (D[L[i]] <= D[L[j]])
            break;
        int v = L[j];
        W[v] = i;
        W[u] = j;
        int temp = L[i];
        L[i] = L[j];
        L[j] = temp;
        i = j;
    }
}
void heapUp(int i) {
    int u = L[i];
    while (i > 0) {
        int p = (i - 1) / 2;

        if (D[L[i]] >= D[L[p]])
            break;
        int v = L[p];
        W[v] = i;
        W[u] = p;
        int t = L[i]; L[i] = L[p]; L[p] = t;
        i = p;
    }
}
void update(int u, int p, long long d) {
    if (d < D[u]) {
        D[u] = d;
```



```cpp
        D[u] = d;
        P[u] = p;
        heapUp(W[u]);
    }
}
int main() {
    int n, m, a, b, c;
    std::cin >> n >> m;
    for (int i = 0; i < m; i++) {
        std::cin >> a >> b >> c;
        a--; b--;
        E[a] = add_node(b, c, E[a]);
        E[b] = add_node(a, c, E[b]);
    }
    for (int i = 0; i < n; i++)
        D[i] = INF, W[i] = Z, L[Z++] = i;
    update(0, 0, 0);
    while (Z > 0) {
        int u = L[0];
        if (--Z > 0) {
            L[0] = L[Z];
            W[L[0]] = 0;
            heapDown(0);
        }
        if (D[u] == INF)
            break;
        for (Node* t = E[u]; t; t = t->next)
            update(t->neighbor, u, D[u] + t->weight);
    }
    if (D[n - 1] < INF) {
```



```cpp
    if (D[n - 1] < INF) {
        int z = 0;
        int u = n - 1;
        while (u != 0) {
            W[z++] = u + 1;
            u = P[u];
        }
        cout << "1 ";
        for (int i = z - 1; i >= 0; i--)
            cout << W[i] << ' ';
    }
    else
        cout << "-1";
}
```

Test: #1, time: 0 ms., memory: 2352 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #2, time: 0 ms., memory: 2352 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #3, time: 15 ms., memory: 2352 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #4, time: 0 ms., memory: 2352 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #5, time: 0 ms., memory: 2348 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #6, time: 0 ms., memory: 2492 KB, exit code: 0, checker exit code: 0, verdict: OK

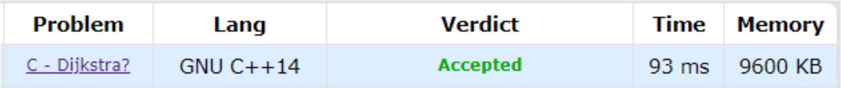
Test: #7, time: 0 ms., memory: 2488 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #8, time: 0 ms., memory: 2488 KB, exit code: 0, checker exit code: 0, verdict: OK

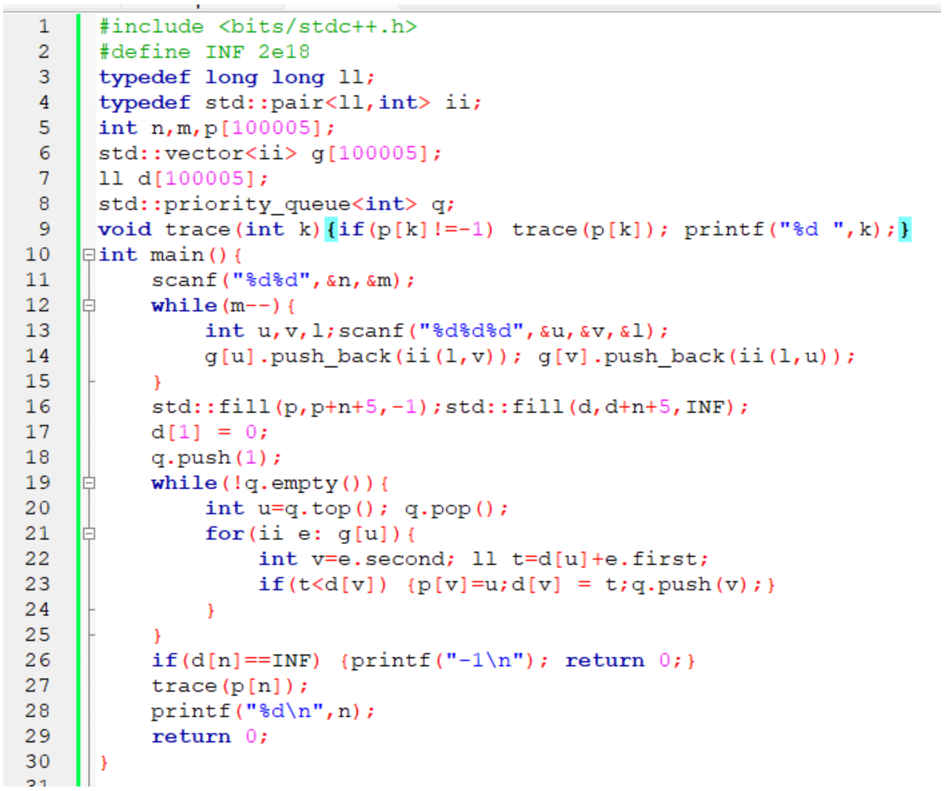
Test: #9, time: 15 ms., memory: 2492 KB, exit code: 0, checker exit code: 0, verdict: OK

Test: #10, time: 0 ms., memory: 2536 KB, exit code: 0, checker exit code: 0, verdict: OK

3.



Problem	Lang	Verdict	Time	Memory
C - Dijkstra?	GNU C++14	Accepted	93 ms	9600 KB



```cpp
#include <bits/stdc++.h>
#define INF 2e18
typedef long long ll;
typedef std::pair<ll,int> ii;
int n,m,p[100005];
std::vector<ii> g[100005];
ll d[100005];
std::priority_queue<int> q;
void trace(int k){if(p[k]!=-1) trace(p[k]); printf("%d ",k);}
int main(){
    scanf("%d%d",&n,&m);
    while(m--){
        int u,v,l;scanf("%d%d%d",&u,&v,&l);
        g[u].push_back(ii(l,v)); g[v].push_back(ii(l,u));
    }
    std::fill(p,p+n+5,-1);std::fill(d,d+n+5,INF);
    d[1] = 0;
    q.push(1);
    while(!q.empty()){
        int u=q.top(); q.pop();
        for(ii e: g[u]){
            int v=e.second; ll t=d[u]+e.first;
            if(t<d[v]) {p[v]=u;d[v] = t;q.push(v);}
        }
    }
    if(d[n]==INF) {printf("-1\n"); return 0;}
    trace(p[n]);
    printf("%d\n",n);
    return 0;
}
```

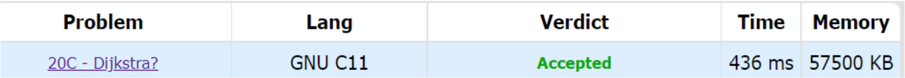
Test: #1, time: 0 ms., memory: 2344 KB, exit code: 0, checker exit code: 0, verdict: OK
Test: #2, time: 0 ms., memory: 2344 KB, exit code: 0, checker exit code: 0, verdict: OK
Test: #3, time: 0 ms., memory: 2352 KB, exit code: 0, checker exit code: 0, verdict: OK
Test: #4, time: 0 ms., memory: 2344 KB, exit code: 0, checker exit code: 0, verdict: OK
Test: #5, time: 15 ms., memory: 2348 KB, exit code: 0, checker exit code: 0, verdict: OK
Test: #6, time: 0 ms., memory: 2348 KB, exit code: 0, checker exit code: 0, verdict: OK
Test: #7, time: 0 ms., memory: 2344 KB, exit code: 0, checker exit code: 0, verdict: OK
Test: #8, time: 0 ms., memory: 2344 KB, exit code: 0, checker exit code: 0, verdict: OK
Test: #9, time: 0 ms., memory: 2352 KB, exit code: 0, checker exit code: 0, verdict: OK
Test: #10, time: 0 ms., memory: 2568 KB, exit code: 0, checker exit code: 0, verdict: OK

….

Test: #32, time: 93 ms., memory: 8764 KB, exit code: 0, checker exit code: 0, verdict: OK
Test: #33, time: 78 ms., memory: 8416 KB, exit code: 0, checker exit code: 0, verdict: OK

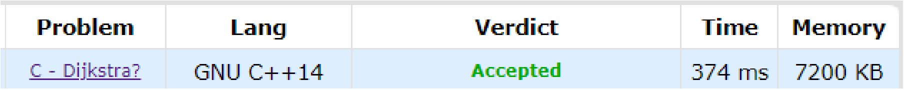
Conclusions:

For the first case:



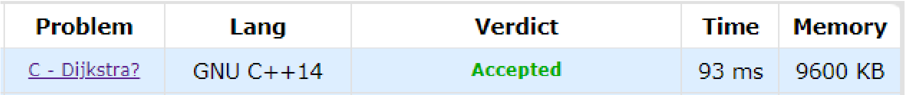
Problem	Lang	Verdict	Time	Memory
20C - Dijkstra?	GNU C11	Accepted	436 ms	57500 KB

For the second case:



Problem	Lang	Verdict	Time	Memory
C - Dijkstra?	GNU C++14	Accepted	374 ms	7200 KB

For the third case:



Problem	Lang	Verdict	Time	Memory
C - Dijkstra?	GNU C++14	Accepted	93 ms	9600 KB