



Shortest paths

locked

Problem

Submissions

Leaderboard

Discussions

Имате даден ориентиран претеглен граф с N върха и M ребра.

Намерете дължината на минималния път от връх с номер 1 до връх с номер N и броя различни начини, по които може да бъде постигнат.

Два пътя се считат за различни ако ползват поне едно различно ребро.

Input Format

На първия ред на стандартния вход ще бъдат зададени целите числа N и M – броят върхове и ребра в графа.

На следващите M реда ще има по една тройка числа u_i, v_i, c_i , задаващи по едно насочено ребро с неговото тегло.

Възможно е да има повече от едно ребро между двойка върхове (u, v) , както е и възможно да има ребро от връх до себе си.

Constraints

$$2 \leq N \leq 10^5$$

$$1 \leq M \leq 5 \times 10^5$$

$$1 \leq c_i \leq 10^9$$

$$1 \leq u_i, v_i \leq N$$

Output Format

На стандартния изход изведете разделени с интервал две цели числа – дължината на минималния път и броя различни начини, по които може да бъде постигнат.

Тъй като броят пътища може да е много голям, изведете само неговия остатък по модул $1000000007 = 10^9 + 7$.

В случай, че няма път между 1 и N , като цена изведете " -1 ", а като брой пътища - " 0 ".

Sample Input 0

```
5 6
1 2 10
1 4 29
2 3 50
4 3 13
4 5 24
3 5 11
```

Sample Output 0

```
53 2
```

Sample Input 1

```
3 1
1 2 13
```

Sample Output 1

-1 0

[f](#) [t](#) [in](#)



Submissions: 91

Max Score: 100

Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

[More](#)Current Buffer (saved locally, editable)  

C++14



```
1 #include <cmath>
2 #include <cstdio>
3 #include <vector>
4 #include <iostream>
5 #include <algorithm>
6 #include <list>
7 #include <set>
8 #include <unordered_map>
9 #include <unordered_set>
10 #include <climits>
11 using namespace std;
12
13 unordered_map<long long, vector<pair<long long, long long>>> adj;
14
15 void shortestPathDijkstra(int n) {
16     vector<long long> distances(n + 1, LONG_LONG_MAX);
17     vector<long long> same(n + 1, 0);
18
19     set < pair<long long, long long>> notVisited;
20
21     notVisited.insert({ 0,1 });
22     distances[1] = 0;
23     same[1] = 1;
24
25     while (!notVisited.empty()) {
26         auto node = notVisited.begin();
27
28         for (long long i = 0; i < adj[node->second].size(); i++) {
29
30             long long newNode = adj[node->second][i].first;
31             long long newDist = node->first + adj[node->second][i].second;
32
33             if (newDist == distances[newNode]) {
34                 same[newNode] += same[node->second];
35                 same[newNode]=same[newNode]%1000000007;
36             }
37             if (newDist < distances[newNode]) {
38                 notVisited.erase({ distances[newNode], newNode });
39                 notVisited.insert({ newDist, newNode });
40                 distances[newNode] = newDist;
41                 same[newNode] = same[node->second];
42                 same[newNode]=same[newNode]%1000000007;
43             }
44         }
45     }
46
47     notVisited.erase(node);
48
49
50
51 }
52 if (distances[n] != LONG_LONG_MAX) {
53     cout << distances[n] << ' ' << same[n];
54 }
55 else {
```

```
56     cout << -1 << ' ' << 0;
57 }
58 }
59
60 int main() {
61     ios_base::sync_with_stdio(false);
62     cin.tie(NULL);
63     long long n, m, xi, yi, wi;
64     cin >> n >> m;
65
66     for (int i = 0; i < m; i++) {
67         cin >> xi >> yi >> wi;
68         adj[xi].push_back({ yi, wi });
69     }
70
71     //print
72     /*for (int i = 0; i < n; i++) {
73         for (long unsigned int j = 0; j < adj[i].size(); j++) {
74             cout << i << " -> " << adj[i][j].first << ", weight: " << adj[i][j].second << endl;
75         }
76     }*/
77
78
79
80
81     shortestPathDijkstra(n);
82     return 0;
83 }
```

Line: 1 Col: 1

[Upload Code as File](#) ☐ Test against custom input

Run Code

Submit Code