

Question )

Problem List

Run

Submit

🕒

📄

Description

Editorial

Solutions

Submissions

1514. Path with Maximum Probability

Sc

Medium

Topics

Companies

Hint

You are given an undirected weighted graph of `n` nodes (0-indexed), represented by an edge list where `edges[i] = [a, b]` is an undirected edge connecting the nodes `a` and `b` with probability of success of traversing that edge `succProb[i]`.

Given two nodes `start` and `end`, find the path with the maximum probability of success to go from `start` to `end` and return its success probability.

If there is no path from `start` to `end`, return 0. Your answer will be accepted if it differs from the correct answer by at most  $1e-5$ .

**Example 1:**

```
graph TD; 0((0)) ---|0.5| 1((1)); 0 ---|0.2| 2((2)); 1 ---|0.5| 2
```

**Input:** `n = 3, edges = [[0,1],[1,2],[0,2]], succProb = [0.5,0.5,0.2], start = 0, end = 2`

**Output:** `0.25000`

**Explanation:** There are two paths from start to end, one having a probability of success = 0.2 and the other has  $0.5 * 0.5 = 0.25$ .

**Example 2:**

👍 3K

👎

💬 39

☆

🔗

?

Problem List

Run

Submit

🕒

📄

Description

Editorial

Solutions

Submissions

Example 2:

```
graph TD; 0((0)) ---|0.5| 1((1)); 0 ---|0.3| 2((2)); 1 ---|0.5| 2
```

**Input:** `n = 3, edges = [[0,1],[1,2],[0,2]], succProb = [0.5,0.5,0.3], start = 0, end = 2`

**Output:** `0.30000`

**Example 3:**

```
graph TD; 0((0)) ---|0.5| 1((1)); 2((2))
```

**Input:** `n = 3, edges = [[0,1]], succProb = [0.5], start = 0, end = 2`

**Output:** `0.00000`

**Explanation:** There is no path between 0 and 2.

👍 3K

👎

💬 39

☆

🔗

?

Code :

```
1  class Solution {
2  public:
3      double maxProbability(int n, vector<vector<int>>& edges, vector<double>& succProb, int start_node, int end_node) {
4          vector<vector<pair<int,double>>>graph(n);
5          int i=0;
6          for(auto e:edges){
7              graph[e[0]].push_back({e[1],succProb[i]});
8              graph[e[1]].push_back({e[0],succProb[i]});
9              i++;
10         }
11
12         priority_queue<pair<double,int>>q;
13         vector<bool>visited(n,false);
14         q.push({1.0,start_node});
15         vector<double>dist(n,0);
16         dist[start_node]=1.0;
17         while(!q.empty()){
18             auto top = q.top();
19             q.pop();
20             int node = top.second;
21             double proba = top.first;
22             if(visited[node])continue;
23             visited[node]=true;
24
25             for(auto conn:graph[top.second]){
26                 if(dist[conn.first] < conn.second*proba){
27                     dist[conn.first] = conn.second*proba;
28                     q.push({dist[conn.first],conn.first});
29                 }
30             }
31         }
32         return dist[end_node];
33     }
34 };
```

Output:

ist < > ↺

🔥 Run

📁 Submit

🔍

📄

🔧 0

Editorial | 📖 Solutions | ⌚ Submissions | ⏱ Accepted ×

👤 Aswin submitted at Nov 28, 2023 19:37 🔗

✅ Accepted

📖 Editorial

+ Solution

⌚ Runtime

178 ms

Beats 25.69% of users with C++

💾 Memory

71.26 MB

Beats 16.44% of users with C++

C++

```
class Solution {
public:
    double maxProbability(int n, vector<vector<int>>& edges, vector<double>& succProb, int start_node, int end_node) {
        vector<vector<pair<int,double>>>graph(n);
        int i=0;
        for(auto e:edges){
            graph[e[0]].push_back({e[1],succProb[i]});
            graph[e[1]].push_back({e[0],succProb[i]});
            i++;
        }

        priority_queue<pair<double,int>>q;
        vector<bool>visited(n,false);
        q.push({1.0,start_node});
        vector<double>dist(n,0);
        dist[start_node]=1.0;
        while(!q.empty()){
            auto top = q.top();
            q.pop();
            int node = top.second;
            double proba = top.first;
            if(visited[node])continue;
            visited[node]=true;

            for(auto conn:graph[top.second]){
                if(dist[conn.first] < conn.second*proba){
                    dist[conn.first] = conn.second*proba;
                    q.push({dist[conn.first],conn.first});
                }
            }
        }
        return dist[end_node];
    }
};
```