15-295 Spring 2019 #3 Shortest Paths

A. Chicago

5 seconds, 256 megabytes

In the movie "Blues Brothers", the orphanage where Elwood and Jake were raised may be sold to the Board of Education if they do not pay 5000 dollars in taxes at the Cook County Assessor's Office in Chicago. After playing a gig in the Palace Hotel ballroom to earn these 5000 dollars, they have to find a way to Chicago. However, this is not so easy as it sounds, since they are chased by the Police, a country band and a group of Nazis. Moreover, it is 106 miles to Chicago, it is dark and they are wearing sunglasses. As they are on a mission from God, you should help them find the safest way to Chicago. In this problem, the safest way is considered to be the route which maximizes the probability that they are not caught.

Input

The input file contains several test cases. Each test case starts with two integers n and m ($2 \le n \le 100$, $1 \le m \le n*(n-1)/2$). n is the number of intersections, m is the number of streets to be considered. The next m lines contain the description of the streets. Each street is described by a line containing 3 integers a, b and p ($1 \le a, b \le n$, $a \ne b$, $1 \le p \le 100$): a and b are the two end points of the street and p is the probability in percent that the Blues Brothers will manage to use this street without being caught. Each street can be used in both directions. You may assume that there is at most one street between two end points. The last test case is followed by a zero.

Output

For each test case, calculate the probability of the safest path from intersection 1 (the Palace Hotel) to intersection n (the Honorable Richard J. Daley Plaza in Chicago). You can assume that there is at least one path between intersection 1 and n. Print the probability as a percentage with exactly 6 digits after the decimal point. The percentage value is considered correct if it differs by at most 10^{-6} from the judge output. Adhere to the format shown below and print one line for each test case.

input

```
5 7
5 2 100
3 5 80
2 3 70
2 1 50
3 4 90
4 1 85
3 1 70
0
```

output

```
61.200000
```

B. Short in Average

5 seconds, 256 megabytes

There are a lot of problems related to the shortest paths. Nevertheless, there are not much problems, related to the shortest paths in average.

Consider a directed graph G, consisting of N nodes and M edges. Consider a walk from the node A to the node B in this graph. The average length of this walk will be total sum of weight of its' edges divided by number of edges. Every edge counts as many times as it appears in this path.

Now, your problem is quite simple. For the given graph and two given nodes, find out the shortest average length of the walk between these nodes. Please note, that the length of the walk need not to be finite, but average walk length will be.

Input

The first line of the input contains an integer T denoting the number of test cases. The description of T test cases follows.

The first line of each test case contains a pair of space-separated integers N and M denoting the number of nodes and the number of edges in the graph.

Each of the following M lines contains a triple of space-separated integers Xi Yi Zi, denoting the arc, connecting the node Xi to the node Yi (but not vice-versa!) having the weight of Zi.

The next line contains a pair of space separated integers A and B, denoting the first and the last node of the path.

- 1 ≤ N ≤ 500
- 1 ≤ M ≤ 1000
- A is not equal to B
- 1 ≤ A, B, Xi, Yi ≤ N
- 1 ≤ Zi ≤ 100
- There are no self-loops and multiple edges in the graph.
- 1 ≤ sum of N over all test cases ≤ 10000
- 1 ≤ sum of M over all test cases ≤ 20000

Output

For each test case, output a single line containing the length of the shortest path in average.

If there is no path at all, output just -1 on the corresponding line of the output.

Your answer will be considered correct in case it has an absolute or relative error of no more than 10^{-6}.

input

```
2
3 3
1 2 1
2 3 2
3 2 3
1 3
3 3
1 2 10
2 3 1
3 2 1
1 3
```

output

```
1.5
1.0
```

C. Mike and Shortcuts

3 seconds, 256 megabytes

Recently, Mike was very busy with studying for exams and contests. Now he is going to chill a bit by doing some sight seeing in the city.

City consists of n intersections numbered from 1 to n. Mike starts walking from his house located at the intersection number 1 and goes along some sequence of intersections. Walking from intersection number i to intersection j requires $|i - j|$ units of energy. The *total energy* spent by Mike to visit a sequence of intersections $p_1 = 1, p_2, ..., p_k$ is equal to $\sum_{i=1}^{k-1} |p_i - p_{i+1}|$ units of energy.

Of course, walking would be boring if there were no shortcuts. A *shortcut* is a special path that allows Mike walking from one intersection to another requiring only 1 unit of energy. There are exactly n shortcuts in Mike's city, the i^{th} of them allows walking from intersection i to intersection a_i ($i \le a_i \le a_{i+1}$) (but not in the opposite direction), thus there is exactly one shortcut starting at each intersection. Formally, if Mike chooses a sequence $p_1 = 1, p_2, ..., p_k$ then for each $1 \le i < k$ satisfying $p_{i+1} = a_{p_i}$ and $a_{p_i} \ne p_i$ Mike will spend **only 1 unit of energy** instead of $|p_i - p_{i+1}|$ walking from the intersection p_i to intersection p_{i+1}. For example, if Mike chooses a sequence $p_1 = 1, p_2 = a_{p_1}, p_3 = a_{p_2}, ..., p_k = a_{p_{k-1}}$, he spends exactly $k - 1$ units of total energy walking around them.

Before going on his adventure, Mike asks you to find the minimum amount of energy required to reach each of the intersections from his home. Formally, for each $1 \le i \le n$ Mike is interested in finding minimum possible total energy of some sequence $p_1 = 1, p_2, ..., p_k = i$.

Input

The first line contains an integer n $(1 \le n \le 200\ 000)$ — the number of Mike's city intersection.

The second line contains n integers $a_1, a_2, ..., a_n$ $(i \le a_i \le n\ ,\ a_i \le a_{i+1} \forall i < n)$, describing shortcuts of Mike's city, allowing to walk from intersection i to intersection a_i using only 1 unit of energy. Please note that the shortcuts don't allow walking in opposite directions (from a_i to i).

Output

In the only line print n integers $m_1, m_2, ..., m_n$, where m_i denotes the least amount of total energy required to walk from intersection 1 to intersection i.

input
3 2 2 3
output
0 1 2

input
5 1 2 3 4 5
output
0 1 2 3 4

input
7 4 4 4 4 7 7 7
output
0 1 2 1 2 3 3

In the first sample case desired sequences are:

$1: 1; m_1 = 0;$

$2: 1, 2; m_2 = 1;$

$3: 1, 3; m_3 = |3 - 1| = 2.$

In the second sample case the sequence for any intersection $1 < i$ is always $1, i$ and $m_i = |1 - i|$.

In the third sample case — consider the following intersection sequences:

$1: 1; m_1 = 0;$

$2: 1, 2; m_2 = |2 - 1| = 1;$

$3: 1, 4, 3; m_3 = 1 + |4 - 3| = 2;$

$4: 1, 4; m_4 = 1;$

$5: 1, 4, 5; m_5 = 1 + |4 - 5| = 2;$

$6: 1, 4, 6; m_6 = 1 + |4 - 6| = 3;$

$7: 1, 4, 5, 7; m_7 = 1 + |4 - 5| + 1 = 3.$

Problem D: Subway

Johny is going to visit his friend Michelle. His dad allowed him to go there on his own by subway. Johny loves traveling by subway and would gladly use this opportunity to spend half a day underground, but his dad obliged him to make as few line changes as possible. There are a lot of stations in the city, and several subway lines connecting them. All trains are perfectly synchronized – the travel between two consecutive stations on every line takes exactly one minute, and changing lines at any station takes no time at all.

Given the subway map, help Johny to plan his trip so that he can travel for as long as possible, while still following his dad's order.

Input

First line of input contains the number of test cases T. The descriptions of the test cases follow:

The description of each test case starts with an empty line. The next two lines begin with the strings **Stops:** and **Lines:**, and contain the names (separated by a comma and a space) of all subway stops and lines, respectively. A single line for each subway line follows (in no particular order), beginning with **<line-name> route:** and listing the names of the stops along this particular line. The final two lines specify the names of the (different) stations nearby Johny's and Michelle's homes.

In each test case, there are at most 300 000 stations and 100 000 lines, whose total length does not exceed 1 000 000. The names of lines and stations are between 1 and 50 characters long and can contain letters, digits, hyphens (**-**), apostrophes (**'**) and "and" signs (**&**). All lines are bidirectional (although changing the direction of travel counts as a line change) and there are no self-crossings.

Output

Print the answers to the test cases in the order in which they appear in the input. For each test case, print a single line summarizing the optimal route Johny can take (see example output for exact format). You may assume that such a route always exists.

Example

Some lines in the example test data below were too long and had to be wrapped. You can access full sample tests at your workstation.

For an example input

```
3

Stops: OxfordCircus, PiccadillyCircus, HydeParkCorner, King'sCross,
GreenPark, Arsenal, Victoria, Highbury&Islington, LeicesterSquare
Lines: Blue, Cyan
Cyan route: Highbury&Islington, King'sCross, OxfordCircus,
GreenPark, Victoria
Blue route: HydeParkCorner, GreenPark, PiccadillyCircus,
LeicesterSquare, King'sCross, Arsenal
Johny lives at King'sCross
Michelle lives at GreenPark

Stops: OxfordCircus, PiccadillyCircus, HydeParkCorner, King'sCross,
GreenPark, Arsenal, Victoria, Highbury&Islington, LeicesterSquare
Lines: Blue, Cyan
Cyan route: Highbury&Islington, King'sCross, OxfordCircus,
GreenPark, Victoria
Blue route: HydeParkCorner, GreenPark, PiccadillyCircus,
LeicesterSquare, King'sCross, Arsenal
Johny lives at PiccadillyCircus
Michelle lives at LeicesterSquare

Stops: OxfordCircus, PiccadillyCircus, HydeParkCorner, King'sCross,
GreenPark, Arsenal, Victoria, Highbury&Islington, LeicesterSquare
Lines: Blue, Cyan
Cyan route: Highbury&Islington, King'sCross, OxfordCircus,
GreenPark, Victoria
Blue route: HydeParkCorner, GreenPark, PiccadillyCircus,
LeicesterSquare, King'sCross, Arsenal
Johny lives at Victoria
Michelle lives at HydeParkCorner
```

the correct answer is:

```
optimal travel from King'sCross to GreenPark: 1 line, 3 minutes
optimal travel from PiccadillyCircus to LeicesterSquare: 1 line, 1 minute
optimal travel from Victoria to HydeParkCorner: 2 lines, 7 minutes
```

E - Admiral

Michiel Adriaenszoon de Ruyter is the most famous admiral in Dutch history and is well known for his role in the Anglo-Dutch Wars of the 17th century. De Ruyter personally commanded a flagship and issued commands to allied warships during naval battles.

In De Ruyter's time, graph theory had just been invented and the admiral used it to his great advantage in planning his naval battles. Waypoints at sea are represented by vertices, and possible passages from one waypoint to another are represented as directed edges. Given any two waypoints W_1 and W_2, there is at most one passage $W_1 \to W_2$. Each directed edge is marked with the number of cannonballs that need to be fired in order to safely move a ship along that edge, sinking the enemy ships encountered along the way.

One of De Ruyter's most successful tactics was the *De Ruyter Manoeuvre*. Here, two warships start at the same waypoint, and split up and fight their way through the enemy fleet, joining up again at a destination waypoint. The manoeuvre prescribes that the two warships take disjunct routes, meaning that they must not visit the same waypoint (other than the start and end-points), or use the same passage during the battle.

Being Dutch, Admiral De Ruyter did not like to waste money; in 17th century naval warfare, this meant firing as few expensive cannonballs as possible.

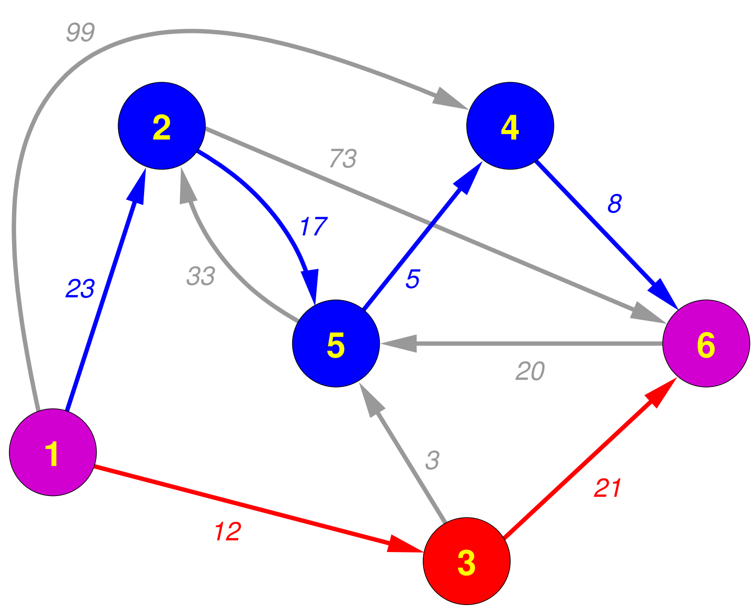

Figure 1: A particular instance of De Ruyter's tactic, visualised as a graph. Two ships ('red' and 'blue') move from a shared starting point (1) to a shared endpoint (6). The red ship's route is $1 \to 3 \to 6$ (firing 33 canonballs along the way); the blue ship's route is $1 \to 2 \to 5 \to 4 \to 6$ (firing 53 canonballs along the way). In total, 86 canonballs are fired during the manoeuvre. Except for the start- and end-point, no vertices or edges are visited by both ships.

Input

For each test case, the input consists of:

- A line containing two integers v $(3 \leq v \leq 1000)$ and e $(3 \leq e \leq 10000)$, the number of waypoints and passages, respectively.
- Then, e lines follow: for each passage, a line containing three integers:
  1. a_i $(1 \leq a_i \leq v)$, the starting-point of a passage, which is represented by a waypoint;
  2. b_i $(1 \leq b_i \leq v)$ and $(a_i \neq b_i)$, the end-point of a passage, which is represented by a waypoint. All passages are directed passages;
  3. c_i $(1 \leq c_i \leq 100)$, the number of cannonballs that are fired when travelling along this passage.

The starting waypoint is 1 and the destination waypoint is v. There are always at least two disjunct routes from waypoint 1 to waypoint v.

Output

For each test case, the output consists of a single positive integer: the smallest possible sum of cannonballs fired by both ships when reaching the destination waypoint.

Example

input	output
6 11 1 2 23 1 3 12 1 4 99 2 5 17 2 6 73 3 5 3 3 6 21 4 6 8 5 2 33 5 4 5 6 5 20	86