1 Longest Common Subsequence

Task: You are given two sequences (a_i) and (b_i) of the same the length n, where $a_i, b_i \in [1 : 2^{31} - 1]$. Compute the length of a longest common subsequence.

Input: The first line contains the number of test cases. A test case is specified as follows: The first line of a test case contains the length n of the sequences. The second line contains the numbers of the sequence (a_i) and the third line contains the numbers of the sequence (b_i).

Output: In the i-th line print the result of the i-th test case.

Sample Input:

```
2
5
2 3 5 4 4
1 3 1 1 5
8
1 5 3 3 5 4 2 3
5 4 5 2 1 3 5 5
```

Sample Output:

```
2
4
```

2 Magic necklace

Task: You are given a string and some magic pearls $(a_1, \ldots, a_n)$ and you want to make a necklace for a wizard. We identify the pearls with their diameters, so $a_i \in \mathbb{N}$ denotes the diameter of the i-th pearl. You can assume that all pearls have a different diameter.

The string has a left and a right end. You want to arrange the pearls on the string such that their diameters are decreasing from left to right. You treat the pearls one after another according to their indices. For each pearl you can decide whether you string it on the left or on the right side of the string. You can also discard pearls, but because they are magic, you can never touch them again.

What is the maximum number of pearls the necklace could consist of?

Input: The first line contains the number of test cases. The test cases will be concatenated. A test case is specified as follows: The first line of a test case contains the number n of pearls. The following n lines contain the diameters a_1 to a_n of the pearls.

Output: In the i-th line print the result of the i-th test case.

Sample Input:

1
3
1
2
3

Sample Output:

3

3 Sustainable Lifestyle

Task: Every time Alice visits a new city, she rents a car to drive around and see life in the city. Her goal is always to see all streets (i.e., to drive on every road). After doing this, she returns the car at her starting point. But, Alice is very ecology-minded. So she wants to drive on each road only one time to reduce her CO_2-output. Is it possible to find a way from her beginning in the starting point and ending in the starting point, that uses each street exactly one time? (Crossroads does not matter!)

Some additional information about the cities Alice visits:

- There are two types of cities: In the first type, you can drive at every street in both direction. In the second type all streets are one-way streets.
- It is not guaranteed that she can reach every road from her starting point, due to construction works and so on.
- It is possible that two crossroads are connected by more than one street.

Input: There is only a single test case specified: The first line contains a 1, if all streets are one-way streets and 0 otherwise. The second line contains two integers n and m, the number of crossroads and roads. In the following m lines the roads are described, i.e., in each line is given a pair of two integers a and b. (Crossroads are numbered from 0 to $n-1$.) In case of one-way streets, this means that there is a road from a to b.

You can assume that Alice starts at crossroad 0.

Output: If it is possible to find a way, which satisfies the requirements output "yes", otherwise "no".

Sample Input 1:

0
4 5
1 2
1 3
2 3
3 4
3 4

Sample Output 1:

yes

Sample Input 2:

```
1
4 4
0 1
1 0
2 3
3 2
```

Sample Output 2:

no

4 Friendship

Task: You are at home and want to go to the cinema because you need a little distraction, but there is a problem: Your (former) best friend pinched your girlfriend a few days ago. At this time, he is usually heading from work to the gym. Of course you don't know the exact time he is leaving his work. You know that he always uses a fastest way from work to gym, but you don't know which fastest way he will take today.

You want to go to the cinema, but avoid meeting him on the street or at crossings of streets by all means. How long do you need?

Input: The input is given as a directed graph, in which each edge represents a street. The first line of a test case the number n of vertices and m of edges. It holds $0 \leq n \leq 1000$ and $0 \leq m \leq 10000$. A test case with $n = m = 0$ terminates the input. The second line contains the locations of your home and the cinema. The third line contains your friend's workplace and the gym location. The following m lines encode edges by giving the start vertex, the end vertex, and the edge length, which is a non-negative integer.

Output: For every test case, print a line containing the length of a shortest feasible path from your home to the cinema. If none exists, output -1.

Sample Input:

```
10 5
0 9
1 2
0 1 1
1 2 1
2 9 1
0 5 3
5 9 4
0 0
```

Sample Output:

7

5 Time travel

Task: We are in the year X in a fictional world, which consists of n islands. The islands are numbered from 0 to $n-1$. Long ago in the past, there was a nuclear war – nobody knows when, only that it is more than 2000000 years ago. You can travel between the islands using a technology, which uses portals. Other ways of travelling are not possible due to contamination. On every island there is a portal, and it is possible to travel from one island to another using a connection between the corresponding portals. But, if you use the portals, you automatically travel in time. E.g., there is portal connection from island 1 to island 2, and if you use it you travel 5 years to the future. But if you travel from island 2 to island 3, then you go back 2 years to the past.

You can assume, that it there is a sequence of portal connections, such that you can reach arbitrary islands, independent of your current location.

Is it possible to travel back in time, such that you can prevent the nuclear war?

Input: The first line of the input contains the number of test cases. The second line contains at first the number of island n and the number of portal connections m. In the following m lines the portal connections are specified: Each line contains three integers a, b, t. a is island, where the connection starts, and b where it ends. t denotes the movement in time using this connection.

The number of islands is in $[1 : 1000]$, the number of connections is in $[0 : 2000]$, and $|t| \leq 1000$.

Output: In the i-th line print "possible", if it is possible in the i-th test case to travel back in time long enough, and otherwise print "not possible".

Sample Input:

```
2
3 4
0 1 900
1 0 1
1 2 13
2 1 -22
3 3
0 1 1
1 2 1
2 0 1
```

Sample Output:

```
possible
not possible
```

6 Express Delivery

Task: Your join a company that makes deliveries using smart drones. In each station, you have a different type of drone with different speed and battery capacity. Your company does deliveries from one station to another and your task is to find the fastest possible delivery paths. After a drone leaves a station, it can drive until its capacity is depleted. At this point, it has to be inside another station to hand of the delivery to another drone.

You know the distances between stations, the speeds and capacities as well as a list of possible delivery requests. For each possible request, compute the fastest possible delivery time.

Input: The input has the following form:

- The first line contains two integers N, the number of stations, and Q, the number of deliveries.
- The next N lines contain two integers each. C_i is the capacity in hours and S_i is the speed in kilometers per hour of drones at station i.
- The next N lines contain N integers each. The j-th integer on the i-th line gives the distance $D_{i,j}$ from station i to station j in kilometers. The distance -1 indicates that station j cannot be reached directly from station i.
- The next Q lines contain two integers each. U_k, the start, and V_k, the end station of delivery request k.

Output: Compute the time y_i in hours required for each request i rounded to two places. The output is a single line $y_1\ y_2\ \ldots\ y_Q$ given as floating point numbers. In cases where the delivery time is integral or rounded to an integer, the output is still expected to be a floating point number.

Sample Input 1:

```
3 1
2 3
2 4
4 4
-1 1 -1
-1 -1 1
-1 -1 -1
1 3
```

Sample Output 1:

0.58

Sample Input 2:

```
4 3
30 60
10 1000
12 5
20 1
-1 10 -1 31
10 -1 10 -1
-1 -1 -1 10
15 6 -1 -1
2 4
3 1
3 2
```

Sample Output 2:

```
0.51 8.01 8.0
```

7 Computing a maximum flow

Task: You are given a directed graph $G = (V, E)$, and a capacity function $u : E \to \mathbb{R}_{\geq 0}$. Furthermore, let $a, b \in \mathbb{N}$ and you are given several sources $S = \{s_1, \ldots, s_a\}$ and sinks $T = \{t_1, \ldots, t_b\}$ with $S \cap T = \emptyset$. You want to compute the value of a maximum flow from the sources to the sinks.

Input: The first line gives you the number of test cases. A test case is specified as follows: The first line of a test case contains the number of nodes n and the number of edges m (both values are integers). The nodes are numbered from 0 to $n - 1$. The next line describes the number of sources a. In the next line, there are a integers, denoting the indices of the sources. Then, the next line describes the number of sinks b, and in the following line the indices of the sinks are given. Afterwards, there are m lines, which specify the edges. The first two numbers are integers, and contain the start and end node of an edge. The third value gives the capacity of the edge, and is given by a floating point number.

Output: In the i-th line the result of the i-th test case is printed. The value of the maximum flow should be rounded to two decimal places.

Sample Input:

```
2
2 1
1
0
1
```

```
1
0 1 3.14
6 7
1
0
1
5
0 1 1.0
0 2 1.0
0 3 1.0
1 4 1.0
2 4 1.0
3 5 2.5
4 5 0.5
```

Sample Output:

```
3.14
1.50
```

8 Numerical Integration

Task: An *ellipse* is defined as follows: Let $p_i = (x_i, y_i)$ (with $i \in \{1, 2\}$) and $r > \|p_1 - p_2\|_2$ be given. Then, we define

$$E(p_1, p_2, r) = \left\{ p \in \mathbb{R}^2 \,:\, \|p - p_1\|_2 + \|p - p_2\|_2 = r \right\}.$$

For an ellipse E, let $A[E]$ denote the set of all points enclosed by E, including all points in E.

Integration is difficult. Assume you are given two ellipses $E_1 = E(p_1, p_2, r_1)$ and $E_2 = E(p_3, p_4, r_2)$. Now you are interested in the volume, i.e., in the size of the area, that is in the intersection of $A := A[E_1] \cap A[E_2] \cap [0, 1]^2$. It is hard to calculate A exactly.

One way of approximating A is using a Monte-Carlo technique: We generate $n \in \mathbb{N}$ random points $U_1, \ldots, U_n$, that are distributed uniformly in $[0, 1]^2$. Then, we can approximate A by

$$\tilde{A} := \frac{1}{n} \sum_{i=1}^{n} I_{\{U_i \in A\}}.$$

Here I denotes the indicator function, i.e., it is equal to one if $U_i \in A$, and zero otherwise. Your task is to calculate $\tilde{A}$.

Input: There is only a single test case specified. At first ellipse E_1 is specified. The first line contains two doubles specifying the x- and y-value of p_1. The next line contains two doubles specifying the x- and y-value of p_2. The third line

contains a double r_1. Afterwards, a second ellipse is described in the same way. Then, a single line contains an integer n that is the number of random points U_i. In the following lines the U_i are described. You can assume, that all doubles have exactly three decimal places.

Output: Compute $\tilde{A}$ rounded to three decimal places.

Sample Input:

```
0.000 0.000
1.000 0.000
2.000
1.000 0.000
1.000 1.000
2.000
5
0.496 0.848
0.727 0.060
0.644 0.253
0.120 0.885
0.882 0.980
```

Sample Output:

```
0.600
```