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
3
```

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 Collecting Eggs

Task: After playing basketball, your group of friends is hungry. Fortunately, you are at your grandparent's farm and there are dozens of chickens sweeping around and laying eggs. Because chickens are animals of habit, each chicken has only one specific nest where it lays its eggs. Of course, you know all of these nests because you and your friends exploit your grandparents way too often. So you plan to split up your group and collect the eggs such that every nest gets visited by at least one of your friends. Afterwards, you meet at the kitchen to make scrambled eggs.

Input: The first line of input contains the number of test cases. The first line of each test case contains the number $N \leq 100$ of locations (that is, the nests, the basketball court, and the kitchen), which are numbered from 0 to $N-1$. The next line contains the number E of neighborships. The following R lines each contain two distinct numbers $0 \leq v, w < N$, denoting that location v and location w are neighbored. You can travel in 1 minute between neighbored locations. The last line of each test case contains the location indices of the basketball court and the kitchen. It will always be possible to reach every location starting from the kitchen (but not necessarily in one step).

Output: For each test case, the output is a single line containing the case number followed by the minimum time (in minutes) required until you can start cooking.

Sample Input:

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

Sample Output:

```
Case 1: 4
Case 2: 1
```

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 Small-world experiment

Task: In 1967 Stanley Milgram published his famous work "The Small World Problem". In this paper we can find the following question: "Given any two people in the world, person X and person Z, how many intermediate acquaintance links are needed before X and Z are connected?" Nowadays, using social networks, it is easier to answer the question than sending letters across the world.

It is likely that the degree of connection today is even higher than in the sixties, therefore we want to investigate this phenomenon again: You are given a social network graph, i.e., there is a number of persons and between two persons there is an acquaintance link if and only if they know each other. What is the diameter of the given graph?

Input: There is only one test case specified! A test case is specified as follows: The first line contains the number of persons n ($n \leq 2000$) in the social network and the number of links m ($m \leq 500000$). Each of the following m lines describes a single link: A link is given by two strings s_1 and s_2, where s_1 and s_2 are names of persons in the social network. (You can assume that each person has a unique name!) A test case is ended by a single line containing 0 0.

Output: Print the diameter of the given social network graph. If the network is not connected, then print "INFINITY".

Sample Input 1:

```
3 3
William Harry
Harry Kate
Kate William
0 0
```

Sample Output 1:

```
1
```

Sample Input 2:

```
4 3
William Harry
Harry Kate
Kate William
0 0
```

Sample Output 2:

```
INFINITY
```

6 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
```

7 Computing a maximum flow

Task: You are given a directed graph $G = (V, E)$, and a capacity function $u : E \to \mathbb{R}_{\geq 0}$. You want to compute the value of a maximum flow from the source node s to the sink t.

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 source is given by the node with number 0 and the sink by $n-1$. 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.

Sample Input:

```
2
2 1
0 1 3.14
6 7
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 Computing the distance to a piecewise linear curve

Task: You are given a point $a \in \mathbb{Z}^2$ and a piecewise linear curve P consisting of n line segments. Compute the point $y \in P$, which minimizes the distance between a and P. In case that there are several points, which minimize the distance, choose the one, which is minimal w.r.t. lexicographic order.

Here, a piecewise linear curve is given by $n+1$ points $p_1, \ldots, p_{n+1} \in \mathbb{Z}^2$, where p_i and p_{i+1} are the endpoints of the i-th line segment.

Input: The input consists of concatenated test cases. The first two lines of a test case give the x- and y-coordinate of a (integer values). The third line denotes the number of line segments n, which specify the piecewise linear curve. Afterwards, the $n+1$ points are given. The fourth and fifth line give x- and y-coordinate of p_1, and so on.

Output: In the $(2i-1)$-th line the x-, and in the $2i$-th line the y-coordinate of the point, which minimizes the distance in the i-th test case, is printed. The coordinates are rounded to two decimal places.

Sample Input:

```
1
1
3
2
1
2
0
0
0
0
3
0
0
1
-2
1
1
-2
```

Sample Output:

```
0.00
1.00
-0.50
-0.50
```