A Til the Cows Come Home

Bessie is out in the field and wants to get back to the barn to get as much sleep as possible before Farmer John wakes her for the morning milking. Bessie needs her beauty sleep, so she wants to get back as quickly as possible.

Farmer John's field has N (2 <= N <= 1,000) landmarks in it, uniquely numbered 1..N. Landmark 1 is the barn; the apple tree grove in which Bessie stands all day is landmark N. Cows travel in the field using T (1 <= T <= 2,000) bidirectional cow-trails of various lengths between the landmarks. Bessie is not confident of her navigation ability, so she always stays on a trail from its start to its end once she starts it.

Given the trails between the landmarks, determine the minimum distance Bessie must walk to get back to the barn. It is guaranteed that some such route exists.

INPUT FORMAT:

* Line 1: Two integers: T and N.

* Lines 2..T+1: Each line describes a trail as three space-separated integers. The first two integers are the landmarks between which the trail travels. The third integer is the length of the trail, range 1..100.

SAMPLE INPUT:

```
5 5
1 2 20
2 3 30
3 4 20
4 5 20
1 5 100
```

OUTPUT FORMAT:

* Line 1: A single integer, the minimum distance that Bessie must travel to get from landmark N to landmark 1.

SAMPLE OUTPUT:

```
90
```

OUTPUT DETAILS:

Bessie can get home by following trails 4, 3, 2, and 1.

B Roadblock

Every morning, FJ wakes up and walks across the farm from his house to the barn. The farm is a collection of N fields (1 <= N <= 100) connected by M bidirectional pathways (1 <= M <= 10,000), each with an associated length. FJ's house is in field 1, and the barn is in field N. No pair of fields is joined by multiple redundant pathways, and it is possible to travel between any pair of fields in the farm by walking along an appropriate sequence of pathways. When traveling from one field to another, FJ always selects a route consisting of a sequence of pathways having minimum total length.

Farmer John's cows, up to no good as always, have decided to interfere with FJ's morning routine. They plan to build a pile of hay bales on exactly one of the M pathways on the farm, doubling its length. The cows wish to select a pathway to block so that they maximize the increase in FJ's distance from the house to the barn. Please help the cows determine by how much they can lengthen FJ's route.

INPUT FORMAT:

* Line 1: Two space-separated integers, N (1 <= N <= 100) and M (1 <=

  M <= 10,000).

* Lines 2..1+M: Line j+1 describes the jth bidirectional pathway in

  terms of three space-separated integers: A_j B_j L_j, where

  A_j and B_j are indices in the range 1..N indicating the

  fields joined by the pathway, and L_j is the length of the

  pathway (in the range 1...1,000,000).

SAMPLE INPUT:

5 7

2 1 5

1 3 1

3 2 8

3 5 7

3 4 3

2 4 7

4 5 2

INPUT DETAILS:

There are 5 fields and 7 pathways. Currently, the shortest path from the house (field 1) to the barn (field 5) is 1-3-4-5 of total length 1+3+2=6.

OUTPUT FORMAT:

* Line 1: The maximum possible increase in the total length of FJ's

    shortest route made possible by doubling the length of a

    single pathway.

SAMPLE OUTPUT:

2

OUTPUT DETAILS:

If the cows double the length of the pathway from field 3 to field 4 (increasing its length from 3 to 6), then FJ's shortest route is now 1-3-5, of total length 1+7=8, larger by two than the previous shortest route length.

Problem C
Horror List

It was time for the 7th *Nordic Cinema Popcorn Convention*, and this year the manager Ian had a brilliant idea. In addition to the traditional film program, there would be a surprise room where a small group of people could stream a random movie from a large collection, while enjoying popcorn and martinis.

However, it turned out that some people were extremely disappointed, because they got to see movies like *Ghosts of Mars*, which instead caused them to tear out their hair in despair and horror.

To avoid this problem for the next convention, Ian has come up with a solution, but he needs your help to implement it. When the group enters the surprise room, they will type in a list of movies in a computer. This is the so-called *horror list*, which consists of bad movies that no one in the group would ever like to see. Of course, this list varies from group to group.

You also have access to the database *Awesome Comparison of Movies* which tells you which movies are directly similar to which. You can assume that movies that are similar to bad movies will be almost as bad. More specificly, we define the *Horror index* as follows:

Photo by Janet Hudson

$$HI = \begin{cases} 0 & \text{if movie is on horror list. This overrides the other definitions.} \\ Q+1 & \text{if the worst directly similar movie has HI = Q} \\ +\infty & \text{if not similar at all to a horrible movie} \end{cases}$$

Input

The first line of input contains three positive integers N, H, L ($1 \leq H < N \leq 1000, 0 \leq L \leq 10000$), where N is the number of movies (represented by IDs, ranging from 0 to $N-1$), H is the number of movies on the horror list and L is the number of similarities in the database.

The second line contains H unique space-separated integers x_i ($0 \leq x_i < N$) denoting the ID of the movies on the horror list.

The following L lines contains two space-separated integers a_i, b_i ($0 \leq a_i < b_i < N$), denoting that movie with ID a_i is similar to movie with ID b_i (and vice verca).

Output

Output the ID of the best movie in the collection (highest Horror Index). In case of a tie, output the movie with the lowest ID.

Sample Input 1	Sample Output 1
6 3 5 0 5 2 0 1 1 2 4 5 3 5 0 2	1

Sample Input 2	Sample Output 2
6 2 3 5 2 0 5 0 1 3 4	3

D Treasures and Vikings

You have a treasure map that is arranged into a $N \times M$ grid. A grid square may be either sea or part of an island. In addition, the map shows the treasure and an enemy Viking ship that occupies one (sea) square. Finally, for convenience you have also drawn your own position.

Now you must set up a fixed route to get the treasure. The route must start at your position, end at the treasure, and consist of a sequence of moves. In each move, you can go only to an (horizontally or vertically) adjacent square that is not part of an island. But beware: The Viking ship might follow you, using the same kind of moves! After each of your moves according to your route, the Viking ship may move or not. Your move and the Vikings' reaction together is called a *round*.

After every round, the following checks are made:

- If you are in line with the Viking ship (you are in the same vertical or horizontal line as the Viking ship with only sea between the Viking ship and you), you are dead.
- If you aren't dead and at the treasure-spot, you get the treasure.

Write a program that decides whether it is possible to set up a **fixed** route **in advance** such that you can get the treasure by following this route and will not get killed by the Vikings – no matter how the Viking ship moves.

Input

The first line of input contains two integers N and M, the dimensions of the map. Each of the following N lines contain M characters. Each character describes a square in the map, and is either `.` (sea), `I` (part of an island), `V` (the Viking ship), `Y` (your position), or `T` (the treasure). Each of `V`, `Y`, and `T` will occur exactly once.

Output

The only line of the output must contain the string `YES`, if it is possible to set up a route to get the treasure, or `NO` otherwise.

Constraints

$1 \leq N, M \leq 700$.
In test cases worth 50 points, $1 \leq N, M \leq 200$.

Example

Input	Output
5 7 Y.....V ..I.... ..IIIII ....T...	YES
5 7 Y....V. ..I.... ..IIIII ....T...	NO
2 3 .YT VII	NO

In the first example, the following route will let you get the treasure:
Down, Down, Down, Right, Right, Right, Down.
In the second and third examples, there is no route to the treasure that you will survive.

Problem E Hie with the Pie

The Pizazz Pizzeria prides itself in delivering pizzas to its customers as fast as possible. Unfortunately, due to cutbacks, they can afford to hire only one driver to do the deliveries. He will wait for 1 or more (up to 10) orders to be processed before he starts any deliveries. Needless to say, he would like to take the shortest route in delivering these goodies and returning to the pizzeria, even if it means passing the same location(s) or the pizzeria more than once on the way. He has commissioned you to write a program to help him.

Input

Input will consist of multiple test cases. The first line will contain a single integer n indicating the number of orders to deliver, where $1 \leq n \leq 10$. After this will be $n+1$ lines each containing $n+1$ integers indicating the times to travel between the pizzeria (numbered 0) and the n locations (numbers 1 to n). The j^{th} value on the i^{th} line indicates the time to go directly from location i to location j without visiting any other locations along the way. Note that there may be quicker ways to go from i to j via other locations, due to different speed limits, traffic lights, etc. Also, the time values may not be symmetric, i.e., the time to go directly from location i to j may not be the same as the time to go directly from location j to i. An input value of $n = 0$ will terminate input.

Output

For each test case, you should output a single number indicating the minimum time to deliver all of the pizzas and return to the pizzeria.

Sample Input

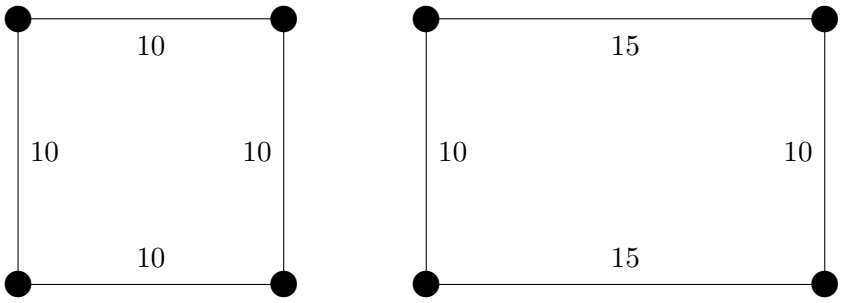
```
3
0 1 10 10
1 0 1 2
10 1 0 10
10 2 10 0
0
```

Sample Output

```
8
```

Problem F Inspectors

Yuri is a rich business owner. His company has factories in many cities across Russia, all of which need regular, monthly inspections. Now Yuri not only has a large portfolio, but a large family as well and he would like to see some of his children start to learn how the business is run. He figures the best way to do this is to have them perform the factory inspections, but he has two concerns. First, he wants each child to be assigned at least two different factories – that way they will be able to compare different management styles at the factories and get a better idea of what works and what doesn't work. This of course requires them to travel between the factories, which brings up Yuri's second problem: while he won't be paying his children anything (their weekly allowance is more than adequate), he's worried about the total of all the travel costs, and would like it to be as small as possible. Each child is expected to inspect each of their assigned factories once a month, always ending back at the factory they started at, so to minimize cost he figures he need only assign the factories to minimize the total distance of these factory tours. The more he thinks about this, the more important saving money becomes. He's willing to use as few as only one of his children if that's the cheapest way to inspect all the factories. For example, if Yuri had four factories placed as in the figure on the left below, he could employ just one child, who could visit each factory for a monthly distance of 40 (assume here that the unlisted distances between factories are all > 10). However, if his factories were located as shown on the right, he would need to employ two children to obtain the monthly minimal distance of 40 (note: he could use two children to achieve the minimal distance in the first figure as well).



Since the number of factories changes over time, Yuri would like a program to determine the minimum distance he can expect for any layout of factories.

Input Time Limit: 3 secs, No. of Test Cases: 101, Input File Size 716K

The input file begins with a line containing a single integer m indicating the number of test cases. Each test case starts with an integer n indicating the number of factories, where $2 \leq n \leq 100$. The next $n-1$ lines contain the positive integer distances between the factories: line 1 contains $n-1$ values specifying the distances between factory 1 and factories $2, 3, 4, \ldots, n$; line 2 contains $n-2$ values specifying the distances between factory 2 and factories $3, 4, \ldots, n$; and so on. The maximum distance between any two factories is 1000. All distances between any three cities will satisfy the triangle inequality.

Output

For each test case display the minimum factory tour distance. You should always assume that Yuri has at least $n/2$ kids for each set of n factories.

Sample Input

```
2
4
10 20 10
10 20
10
4
15 20 10
10 20
15
```

Sample Output

```
Case 1: 40
Case 2: 40
```

G Can of Worms

There is an old adage about opening a can of worms. A lesser known adage is one about shooting a can of exploding worms with a BB gun.

Imagine we place some cans of exploding worms on a long, straight fence. When a can is shot, all of the worms inside will explode. Different types of worms have different blast radii. Each can contains only one kind of worm.

When a can explodes, if another can is in the blast radius, then that can will also explode, possibly creating a chain reaction. Each can explodes only once. This process continues until all explosions stop.

For each can, suppose that it is the only can shot. How many cans in total will explode?

Input

There will be several test cases in the input. Each test case will begin with a line with a single integer n ($1 \le n \le 100{,}000$) representing the number of cans on that fence. Each of the next n lines will have two integers x ($-10^9 \le x \le 10^9$) and r ($1 \le r \le 10^9$), where x is the location of the can on the fence and r is the blast radius. No two cans will occupy the same location. The input will end with a line with a single 0.

Output

For each fence, print n integers on a single line separated by single spaces. The i^{th} integer represents the number of cans that will explode if the i^{th} can is the one that is shot. Output no extra spaces, and do not print any blank lines between outputs.

Sample Input	Sample Output
3 4 3 -10 9 -2 3 12 2 2 7 7 10 1 19 3 23 12 29 8 33 1 35 17 39 2 40 1 46 11 52 3 0	1 2 1 1 3 1 1 9 9 1 9 2 2 9 1