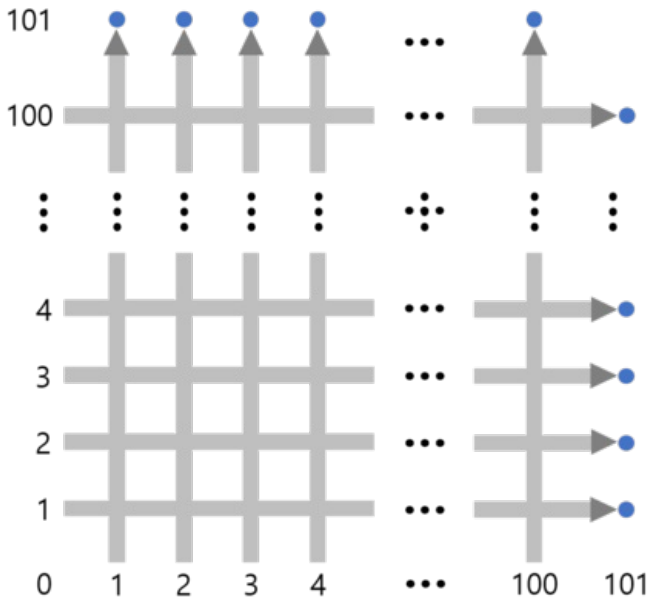
A. Cancel the Trains

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Gildong's town has a train system that has 100 trains that travel from the bottom end to the top end and 100 trains that travel from the left end to the right end. The trains starting from each side are numbered from 1 to 100, respectively, and all trains have the same speed. Let's take a look at the picture below.



The train system can be represented as coordinates on a 2D plane. The i-th train starting at the bottom end is initially at $(i, 0)$ and will be at (i, T) after T minutes, and the i-th train starting at the left end is initially at $(0, i)$ and will be at (T, i) after T minutes. All trains arrive at their destinations after 101 minutes.

However, Gildong found that some trains scheduled to depart at a specific time, simultaneously, are very dangerous. At this time, n trains are scheduled to depart from the bottom end and m trains are scheduled to depart from the left end. If two trains are both at (x, y) at the same time for some x and y, they will crash into each other. Therefore, he is asking you to find the **minimum** number of trains that should be cancelled to prevent all such crashes.

Input

Each test contains one or more test cases. The first line contains the number of test cases t ($1 \le t \le 100$).

Each test case contains three lines. The first line of each test case consists of two integers n and m ($1 \le n, m \le 100$) — the number of trains scheduled to depart from the bottom end, and the number of trains scheduled to depart from the left end, respectively.

The second line of each test case contains n integers. Each integer is a train number that is scheduled to start from the **bottom** end. The numbers are given in strictly increasing order, and are between 1 and 100, inclusive.

The third line of each test case contains m integers. Each integer is a train number that is scheduled to start from the **left** end. The numbers are given in strictly increasing order, and are between 1 and 100, inclusive.

Output

For each test case, print a single integer: the minimum number of trains that should be canceled in order to prevent all crashes.

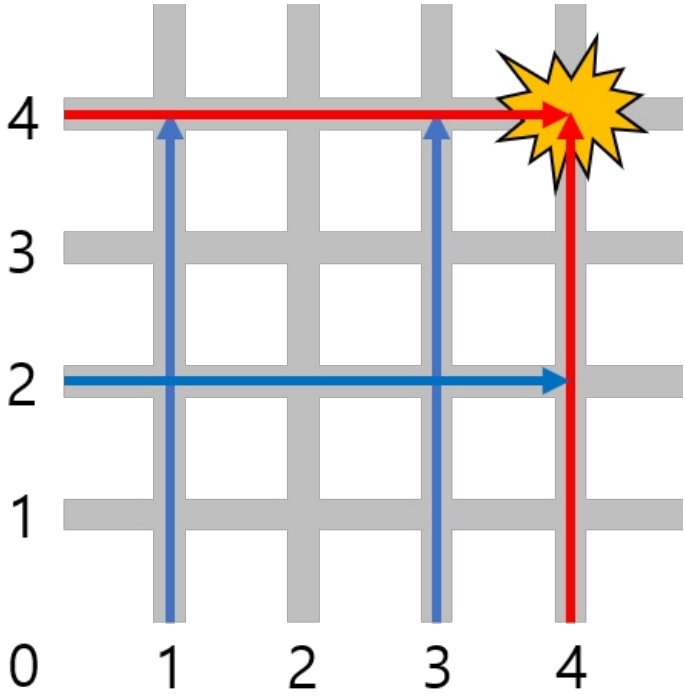
Example

input
3 1 2 1 3 4 3 2 1 3 4 2 4 9 14 2 7 16 28 33 57 59 86 99

3 9 14 19 25 26 28 35 41 59 85 87 99 100
output
0 1 3

Note
In the first case, we can show that there will be no crashes if the current schedule is followed. Therefore, the answer is zero.

In the second case, at $T = 4$, there will be a crash, as can be seen in the picture below. We can prove that after canceling one of these trains, the remaining trains will not crash. Therefore, the answer is one.



B. Suffix Operations

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Gildong has an interesting machine that has an array a with n integers. The machine supports two kinds of operations:

1. Increase all elements of a suffix of the array by 1.
2. Decrease all elements of a suffix of the array by 1.

A suffix is a subsegment (contiguous elements) of the array that contains a_n. In other words, for all i where a_i is included in the subsegment, all a_j's where $i < j \leq n$ must also be included in the subsegment.

Gildong wants to make all elements of a equal — he will always do so using the minimum number of operations necessary. To make his life even easier, before Gildong starts using the machine, you have the option of changing one of the integers in the array to any other integer. You are allowed to leave the array unchanged. You want to minimize the number of operations Gildong performs. With your help, what is the minimum number of operations Gildong will perform?

Note that even if you change one of the integers in the array, you should **not** count that as one of the operations because Gildong did not perform it.

Input
Each test contains one or more test cases. The first line contains the number of test cases t ($1 \leq t \leq 1000$).

Each test case contains two lines. The first line of each test case consists of an integer n ($2 \leq n \leq 2 \cdot 10^5$) — the number of elements of the array a.

The second line of each test case contains n integers. The i-th integer is a_i ($-5 \cdot 10^8 \leq a_i \leq 5 \cdot 10^8$).

It is guaranteed that the sum of n in all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case, print one integer — the minimum number of operations Gildong has to perform in order to make all elements of the array equal.

Example

input
7 2 1 1 3 -1 0 2 4 99 96 97 95 4 -3 -5 -2 1 6 1 4 3 2 4 1 5 5 0 0 0 5 9 -367741579 319422997 -415264583 -125558838 -300860379 420848004 294512916 -383235489 425814447
output
0 1 3 4 6 5 2847372102

Note

In the first case, all elements of the array are already equal. Therefore, we do not change any integer and Gildong will perform zero operations.

In the second case, we can set a_3 to be 0, so that the array becomes $[-1,0,0]$. Now Gildong can use the 2-nd operation once on the suffix starting at a_2, which means a_2 and a_3 are decreased by 1, making all elements of the array -1.

In the third case, we can set a_1 to 96, so that the array becomes $[96,96,97,95]$. Now Gildong needs to:

- Use the 2-nd operation on the suffix starting at a_3 once, making the array $[96,96,96,94]$.
- Use the 1-st operation on the suffix starting at a_4 2 times, making the array $[96,96,96,96]$.

In the fourth case, we can change the array into $[-3,-3,-2,1]$. Now Gildong needs to:

- Use the 2-nd operation on the suffix starting at a_4 3 times, making the array $[-3,-3,-2,-2]$.
- Use the 2-nd operation on the suffix starting at a_3 once, making the array $[-3,-3,-3,-3]$.

C. Triangles

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Gildong has a square board consisting of n rows and n columns of square cells, each consisting of a single digit (from 0 to 9). The cell at the j-th column of the i-th row can be represented as (i, j), and the length of the side of each cell is 1. Gildong likes big things, so for each digit d, he wants to find a triangle such that:

- Each vertex of the triangle is in the center of a cell.
- The digit of every vertex of the triangle is d.
- At least one side of the triangle is parallel to one of the sides of the board. You may assume that a side of length 0 is parallel to both sides of the board.
- The area of the triangle is maximized.

Of course, he can't just be happy with finding these triangles as is. Therefore, for each digit d, he's going to change the digit of exactly one cell of the board to d, then find such a triangle. He changes it back to its original digit after he is done with each digit. Find the maximum area of the triangle he can make for each digit.

Note that he can put multiple vertices of the triangle on the same cell, and the triangle can be a degenerate triangle; i.e. the area of the triangle can be 0. Also, note that he is allowed to change the digit of a cell from d to d.

Input

Each test contains one or more test cases. The first line contains the number of test cases t ($1 \le t \le 1000$).

The first line of each test case contains one integer n ($1 \le n \le 2000$) — the number of rows and columns of the board.

The next n lines of each test case each contain a string of n digits without spaces. The j-th digit of the i-th line is the digit of the cell at (i, j). Each digit is one of the characters from 0 to 9.

It is guaranteed that the sum of n^2 in all test cases doesn't exceed $4 \cdot 10^6$.

Output

For each test case, print one line with 10 integers. The i-th integer is the maximum area of triangle Gildong can make when $d = i - 1$, multiplied by 2.

Example

input
5 3 000 122 001 2 57 75 4 0123 4012 3401 2340 1 9 8 42987101 98289412 38949562 87599023 92834718 83917348 19823743 38947912
output
4 4 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 9 6 9 9 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 49 49 49 49 15 0 30 42 42

Note

In the first case, for $d = 0$, no matter which cell he chooses to use, the triangle with vertices at $(1, 1)$, $(1, 3)$, and $(3, 1)$ is the biggest triangle with area of $\frac{2 \cdot 2}{2} = 2$. Since we should print it multiplied by 2, the answer for $d = 0$ is 4.

For $d = 1$, Gildong can change the digit of the cell at $(1, 3)$ into 1, making a triangle with vertices on all three 1's that has an area of 2.

For $d = 2$, Gildong can change the digit of one of the following six cells into 2 to make a triangle with an area of $\frac{1}{2}$: $(1, 1)$, $(1, 2)$, $(1, 3)$, $(3, 1)$, $(3, 2)$, and $(3, 3)$.

For the remaining digits (from 3 to 9), the cell Gildong chooses to change will be the only cell that contains that digit. Therefore the triangle will always be a degenerate triangle with an area of 0.

In the third case, for $d = 4$, note that the triangle will be bigger than the answer if Gildong changes the digit of the cell at $(1, 4)$ and use it along with the cells at $(2, 1)$ and $(4, 3)$, but this is invalid because it violates the condition that at least one side of the triangle must be parallel to one of the sides of the board.

D. Checkpoints

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Gildong is developing a game consisting of n stages numbered from 1 to n. The player starts the game from the 1-st stage and should beat the stages in increasing order of the stage number. The player wins the game after beating the n-th stage.

There is at most one checkpoint on each stage, and there is always a checkpoint on the 1-st stage. At the beginning of the game, only the checkpoint on the 1-st stage is activated, and all other checkpoints are deactivated. When the player gets to the i-th stage that has a checkpoint, that checkpoint is activated.

For each try of a stage, the player can either beat the stage or fail the stage. If they beat the i-th stage, the player is moved to the $i + 1$-st stage. If they fail the i-th stage, the player is moved to the most recent checkpoint they activated, and they have to beat the stages after that checkpoint again.

For example, assume that $n = 4$ and the checkpoints are on the 1-st and 3-rd stages. The player starts at the 1-st stage. If they fail on the 1-st stage, they need to retry the 1-st stage because the checkpoint on the 1-st stage is the most recent checkpoint they activated. If the player beats the 1-st stage, they're moved to the 2-nd stage. If they fail it, they're sent back to the 1-st stage again. If they beat both the 1-st stage and the 2-nd stage, they get to the 3-rd stage and the checkpoint on the 3-rd stage is activated. Now whenever they fail on the 3-rd stage, or the 4-th stage after beating the 3-rd stage, they're sent back to the 3-rd stage. If they beat both the 3-rd stage and the 4-th stage, they win the game.

Gildong is going to build the stages to have equal difficulty. He wants you to find any series of stages and checkpoints using at most 2000 stages, where the expected number of tries over all stages is exactly k, for a player whose probability of beating each stage is

exactly $\frac{1}{2}$.

Input
Each test contains one or more test cases. The first line contains the number of test cases t ($1 \le t \le 50$).

Each test case contains exactly one line. The line consists of a single integer k ($1 \le k \le 10^{18}$) — the expected number of tries over all stages Gildong wants to set for a player whose probability of beating each stage is exactly $\frac{1}{2}$.

Output
For each test case, print -1 if it's impossible to construct such a series of stages and checkpoints using at most 2000 stages.

Otherwise, print two lines. The first line should contain a single integer n ($1 \le n \le 2000$) – the number of stages. The second line should contain n integers, where the i-th integer represents whether the i-th stage has a checkpoint. The i-th integer should be 0 if the i-th stage doesn't have a checkpoint, and 1 if it has a checkpoint. Note that the first integer must be 1 according to the description.

Example

input
4 1 2 8 12
output
-1 1 1 4 1 1 1 1 5 1 1 0 1 1

Note
In the first and the second case, we can see that the 'easiest' series of stages is to have 1 stage with a checkpoint. This already requires 2 tries in expectation, so it is impossible to make it to require only 1 try.

In the third case, it takes 2 tries in expectation to beat each stage, and the player can always retry that stage without falling back to one of the previous stages if they fail it. Therefore the total expected number of tries is 8. Note that there exists an answer with fewer stages, but you are not required to minimize the number of stages used.

E. Dog Snacks

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Gildong is playing with his dog, Badugi. They're at a park that has n intersections and $n-1$ bidirectional roads, each 1 meter in length and connecting two intersections with each other. The intersections are numbered from 1 to n, and for every a and b ($1 \le a, b \le n$), it is possible to get to the b-th intersection from the a-th intersection using some set of roads.

Gildong has put one snack at every intersection of the park. Now Gildong will give Badugi a mission to eat all of the snacks. Badugi starts at the 1-st intersection, and he will move by the following rules:

- Badugi looks for snacks that are as close to him as possible. Here, the distance is the length of the shortest path from Badugi's current location to the intersection with the snack. However, Badugi's sense of smell is limited to k meters, so he can only find snacks that are less than or equal to k meters away from himself. If he cannot find any such snack, he fails the mission.
- Among all the snacks that Badugi can smell from his current location, he chooses a snack that minimizes the distance he needs to travel from his current intersection. If there are multiple such snacks, Badugi will choose one arbitrarily.
- He repeats this process until he eats all n snacks. After that, he has to find the 1-st intersection again which also must be less than or equal to k meters away from the last snack he just ate. If he manages to find it, he completes the mission. Otherwise, he fails the mission.

Unfortunately, Gildong doesn't know the value of k. So, he wants you to find the minimum value of k that makes it possible for Badugi to complete his mission, if Badugi moves optimally.

Input
Each test contains one or more test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$).

The first line of each test case contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of intersections of the park.

The next $n-1$ lines contain two integers u and v ($1 \le u, v \le n$, $u \ne v$) each, which means there is a road between intersection u and v. All roads are bidirectional and distinct.

It is guaranteed that:

- For each test case, for every a and b ($1 \leq a, b \leq n$), it is possible to get to the b-th intersection from the a-th intersection.
- The sum of n in all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print one integer — the minimum possible value of k such that Badugi can complete the mission.

Example

input
3 3 1 2 1 3 4 1 2 2 3 3 4 8 1 2 2 3 3 4 1 5 5 6 6 7 5 8
output
2 3 3

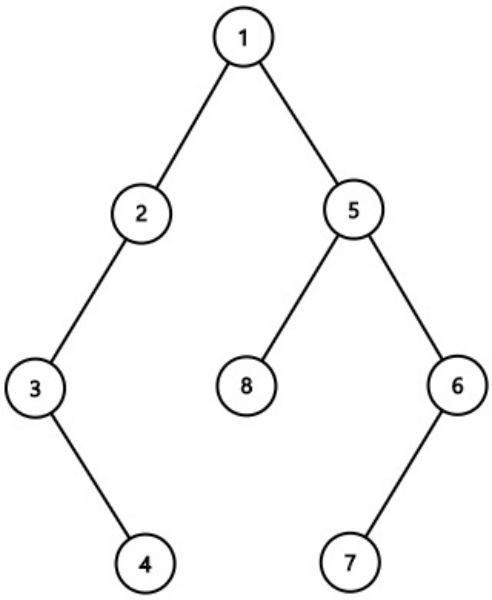
Note

In the first case, Badugi can complete his mission with $k = 2$ by moving as follows:

1. Initially, Badugi is at the 1-st intersection. The closest snack is obviously at the 1-st intersection, so he just eats it.
2. Next, he looks for the closest snack, which can be either the one at the 2-nd or the one at the 3-rd intersection. Assume that he chooses the 2-nd intersection. He moves to the 2-nd intersection, which is 1 meter away, and eats the snack.
3. Now the only remaining snack is on the 3-rd intersection, and he needs to move along 2 paths to get to it.
4. After eating the snack at the 3-rd intersection, he needs to find the 1-st intersection again, which is only 1 meter away. As he gets back to it, he completes the mission.

In the second case, the only possible sequence of moves he can make is 1 – 2 – 3 – 4 – 1. Since the distance between the 4-th intersection and the 1-st intersection is 3, k needs to be at least 3 for Badugi to complete his mission.

In the third case, Badugi can make his moves as follows: 1 – 5 – 6 – 7 – 8 – 2 – 3 – 4 – 1. It can be shown that this is the only possible sequence of moves for Badugi to complete his mission with $k = 3$.



F. Even Harder

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Gildong is now developing a puzzle game. The puzzle consists of n platforms numbered from 1 to n. The player plays the game as a character that can stand on each platform and the goal of the game is to move the character from the 1-st platform to the n-th platform.

The i-th platform is labeled with an integer a_i ($0 \le a_i \le n - i$). When the character is standing on the i-th platform, the player can move the character to any of the j-th platforms where $i + 1 \le j \le i + a_i$. If the character is on the i-th platform where $a_i = 0$ and $i \neq n$, the player loses the game.

Since Gildong thinks the current game is not hard enough, he wants to make it even harder. He wants to change some (possibly zero) labels to 0 so that there remains exactly one way to win. He wants to modify the game as little as possible, so he's asking you to find the **minimum** number of platforms that should have their labels changed. Two ways are different if and only if there exists a platform the character gets to in one way but not in the other way.

Input

Each test contains one or more test cases. The first line contains the number of test cases t ($1 \le t \le 500$).

Each test case contains two lines. The first line of each test case consists of an integer n ($2 \le n \le 3000$) — the number of platforms of the game.

The second line of each test case contains n integers. The i-th integer is a_i ($0 \le a_i \le n - i$) — the integer of the i-th platform.

It is guaranteed that:

- For each test case, there is at least one way to win initially.
- The sum of n in all test cases doesn't exceed 3000.

Output

For each test case, print one integer — the minimum number of different labels that should be changed to 0 so that there remains exactly one way to win.

Example

input
3 4 1 1 1 0 5 4 3 2 1 0 9 4 1 4 2 1 0 2 1 0
output
0 3 2

Note

In the first case, the player can only move to the next platform until they get to the 4-th platform. Since there is already only one way to win, the answer is zero.

In the second case, Gildong can change a_2, a_3, and a_4 to 0 so that the game becomes $4\ 0\ 0\ 0\ 0$. Now the only way the player can win is to move directly from the 1-st platform to the 5-th platform.

In the third case, Gildong can change a_2 and a_8 to 0, then the only way to win is to move in the following way: 1 – 3 – 7 – 9.