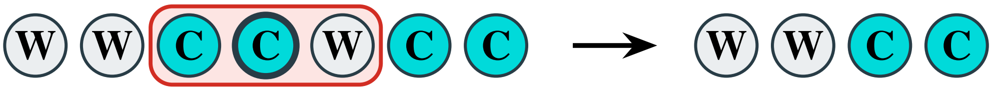
Problem A
White Night 2
Time limit: 2 seconds

Little Cyan Fish has a row of n balls. Each ball is either cyan or white. The row is described by a string s of length n, where `C` denotes a cyan ball and `W` denotes a white ball.

In one operation, he chooses a cyan ball in the current row. Suppose this ball is at position i in the current row. Then all balls whose positions differ from i by at most k are deleted, including the chosen cyan ball itself. After the deletion, the remaining left and right parts are concatenated.

For example, when $k = 1$, one possible operation is



where the ball with the thick border is chosen and the balls inside the red box are deleted.

Find the minimum number of operations needed to delete the whole row. If it is impossible, report that.

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \leq T$), indicating the number of test cases.

For each test case, the first line contains two integers n and k ($1 \leq k \leq n \leq 10^6$). The second line contains a string s of length n, consisting only of characters `C` and `W`.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, if it is impossible to delete the whole row, output a single line with a single integer -1. Otherwise, output a single line containing a single integer, indicating the minimum number of operations needed to delete the whole row.

Sample Input 1

```
3
1 1
C
5 2
WWCWW
4 1
WWWW
```

Sample Output 1

```
1
1
-1
```

In the first sample test case, Little Cyan Fish chooses the only ball.

In the second sample test case, he chooses the middle cyan ball; because $k = 2$, the whole row is deleted in one operation.

In the third sample test case, there is no cyan ball, so no operation can be performed.

This page is intentionally left blank.

Problem B
Alternative Accounts 2
Time limit: 3 seconds

Everybody knows that zh0ukangyang = orzdevinwang.

In the year 6202, when nobody wants to do Universal Cup anymore, Little Cyan Fish decided to host the International Olympiad in Fishing. There are n people interested in joining this event, but... well, they are not following the single-account rule: each contestant has registered two different accounts!

This is terrible, but fortunately they are all very nice people and would never use multiple accounts to cheat. Each contestant follows this strategy: at any moment, they will always participate using the worse-ranked one of their two accounts. That is, if a contestant has two accounts ranked x and y, they will use the account ranked $\max(x, y)$ in the next contest.

Let p_i denote the account currently ranked i, and label the accounts from 1 to $2n$ according to their initial ranks (so $p_i = i$ at the beginning). Then, in each of the next m contests, exactly one contestant participates. Specifically, in contest i ($1 \le i \le m$), the account currently ranked k_i ($k_i \ge 2$) participates and moves up by exactly one rank (that is, p_{k_i} and p_{k_i-1} are swapped after the contest).

Little Cyan Fish wants to figure out which accounts belong to the same player. He notices that the contestants' strategy reveals some information about the account holders. Help him compute the number of possible ways to pair the $2n$ accounts into n contestants, consistent with the contest history. Since the answer can be huge, output it modulo $998\,244\,353$.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 5\,000$), indicating the number of players and the number of events.

The next m lines each contain an integer k_i ($2 \le k_i \le 2n$), meaning that in the i-th event, the account currently ranked k_i played a game.

Output

Output a single line containing a single integer, indicating the number of possible account matchings modulo $998\,244\,353$.

Sample Input 1

```
1 1
2
```

Sample Output 1

```
1
```

Sample Input 2

```
2 2
2
3
```

Sample Output 2

```
0
```

This page is intentionally left blank.

Problem C
Ring of Beads
Time limit: 2 seconds

Little Cyan Fish has n beads arranged in a ring, labeled from 1 to n in clockwise order. Each bead can be colored either cyan or white. He wants to use $S(l, r)$ to denote the number of cyan beads in the *interval* $I(l, r)$, where the *interval* $I(l, r)$ is defined as follows:

- If $l \le r$, it includes the beads with labels $l, l+1, \dots, r$.
- If $l > r$, it includes the beads with labels $l, l+1, \dots, n, 1, 2, \dots, r$ (a circular interval that wraps around from n to 1).

He has not yet decided which beads to color cyan, but he has written down m constraints he wants the final coloring to satisfy. There are a total of m constraints of three different types, each denoted by a tuple (t, l, r, v), where $t \in \{1, 2, 3\}$. The meanings of these three types of constraints are as follows:

- Type 1 ($t = 1$): $S(l, r)$ must be at least v ($S(l, r) \ge v$).
- Type 2 ($t = 2$): $S(l, r)$ must be at most v ($S(l, r) \le v$).
- Type 3 ($t = 3$): $S(l, r)$ must be exactly v ($S(l, r) = v$).

Little Cyan Fish wants to know if a coloring satisfying all these constraints exists. If at least one valid coloring exists, he also wants to know how many intervals $I(l, r)$ ($1 \le l, r \le n$) have the same value of $S(l, r)$ in every valid coloring. Two intervals $I(l_1, r_1)$ and $I(l_2, r_2)$ are considered different if and only if $l_1 \ne l_2$ or $r_1 \ne r_2$.

Input

The first line of the input contains two integers n ($2 \le n \le 10^9$) and m ($1 \le m \le 1\,000$), indicating the number of beads and the number of constraints, respectively.

The next m lines describe all the constraints. Each of these lines contains four integers t, l, r, v ($t \in \{1, 2, 3\}$, $1 \le l, r \le n$, $0 \le v \le L$, where $L = |I(l, r)|$ is the total number of beads in the interval $I(l, r)$), describing a constraint of type t.

Output

If no valid coloring exists, output a single line with a single integer -1. Otherwise, output a single line containing a single integer, denoting the number of intervals $I(l, r)$ whose value $S(l, r)$ is the same in every valid coloring.

Sample Input 1	Sample Output 1
16 8 1 1 3 1 1 5 7 1 1 9 11 1 1 13 15 1 2 2 6 1 2 6 10 1 2 10 14 1 2 14 2 1	128

Sample Input 2

```
1000000000 1
2 114514 114513 0
```

Sample Output 2

```
1000000000000000000
```

Sample Input 3

```
100 3
3 1 10 1
3 91 100 1
3 81 20 2
```

Sample Output 3

```
253
```

Sample Input 4

```
6 3
1 1 3 2
1 4 6 2
2 2 5 1
```

Sample Output 4

```
-1
```

Explanation of Sample 1: There are only two possible colorings (`W` = white, `C` = cyan):

- `CWWWCWWWCWWWCWWW`
- `WWCWWWCWWWCWWWCW`

Exactly half of the 16^2 intervals, namely 128 of them, contain the same number of cyan beads in both colorings. Note that although the intervals $I(1,16), I(2,1), I(3,2), \ldots, I(16,15)$ all include all 16 beads, they are counted as distinct intervals when computing the answer.

In the second example, the only valid coloring is the one in which all beads are white.

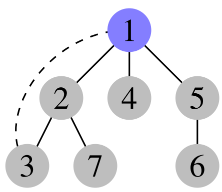
Problem D
DFS Order 6
Time limit: 1 second

Little Cyan Fish loves DFS orders. Today he is studying them once more, but on undirected simple graphs rather than rooted trees.

Fix a connected undirected simple graph G on n vertices labeled 1 to n, and a starting vertex s. The *DFS order* of G from s is the sequence in which the vertices are first visited by the depth-first search shown below; ties are broken by always descending into the unvisited neighbor with the smallest label, so the DFS order is unique.

Algorithm 1 The DFS order used in this problem

```
procedure DFS(vertex x)
    Mark x as visited
    Append x to the end of dfs_order
    for each vertex y adjacent to x in G, in ascending order of label do
        if y is not yet visited then
            DFS(y)
        end if
    end for
end procedure
procedure GENERATE(G, vertex s)
    Mark all vertices as unvisited
    Let dfs_order be an empty list
    DFS(s)
    return dfs_order
end procedure
```



A graph with 7 vertices and 7 edges. The DFS order from vertex 1 is $[1, 2, 3, 7, 4, 5, 6]$.

Little Cyan Fish has prepared n permutations $a_1, a_2, \ldots, a_n$ of 1 to n. Each $a_i = [a_{i,1}, a_{i,2}, \ldots, a_{i,n}]$ is what he claims would be the DFS order from vertex i.

Reconstruct any connected undirected simple graph G on the vertices $1, 2, \ldots, n$ such that the DFS order from every starting vertex i equals a_i, or determine that no such graph exists.

Input

There are multiple test cases. The first line of the input contains a single integer T $(1 \leq T)$, indicating the number of test cases.

For each test case, the first line contains a single integer n $(1 \leq n \leq 200)$. Each of the next n lines contains n integers; the i-th of these lines contains $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ $(1 \leq a_{i,j} \leq n)$ — the DFS order Little Cyan Fish claims is produced when the search starts from vertex i. It is guaranteed that $a_{i,1} = i$, and each row a_i is a permutation of $1, 2, \ldots, n$.

It is guaranteed that the sum of n^2 over all test cases does not exceed 4×10^4.

Output

For each test case, if no suitable graph exists, output a single line containing "`No`".

Otherwise, output "`Yes`" on the first line. On the next line, output a single integer m ($n - 1 \le m \le \frac{n(n-1)}{2}$) — the number of edges in your graph.

Each of the following m lines must contain two integers u and v ($1 \le u, v \le n$, $u \ne v$), denoting an undirected edge between vertices u and v. The resulting graph must be simple and connected, and its DFS order from each vertex i must equal a_i.

If multiple graphs satisfy the requirements, output any of them.

Sample Input 1	Sample Output 1
2 3 1 2 3 2 1 3 3 2 1 3 1 2 3 2 3 1 3 1 2	Yes 2 1 2 2 3 No

In the first sample test case, the path $1 - 2 - 3$ is a valid graph: its DFS orders starting from vertices 1, 2, and 3 are $[1, 2, 3]$, $[2, 1, 3]$, and $[3, 2, 1]$ respectively.

In the second sample test case, no suitable graph exists.

Problem E
Back Edges
Time limit: 4 seconds

Little Cyan Fish has a connected simple undirected graph with n vertices and m edges. The vertices are labeled from 1 to n.

Little Cyan Fish wants to choose a spanning tree of this graph and root it at the vertex 1. After the tree is rooted, every vertex has an ancestor-descendant relation with some other vertices.

The following pseudocode describes how the rooted tree is obtained from the chosen spanning tree:

Algorithm 2 Rooting the chosen spanning tree

```
procedure DFS(vertex x, vertex p)
    Mark x as visited
    Set the parent of x to p
    for each vertex y adjacent to x in the chosen spanning tree do
        if y ≠ p then
            DFS(y, x)
        end if
    end for
    Finish vertex x
end procedure
procedure ROOT-TREE
    Mark all vertices as unvisited
    DFS(1, 0)
end procedure
```

For two vertices u and v, vertex u is an ancestor of vertex v if u appears on the path from the root 1 to v in the chosen spanning tree. A vertex is also considered an ancestor of itself.

An edge of the original graph that is not included in the spanning tree is called a *back edge* if one of its endpoints is an ancestor of the other endpoint in the rooted spanning tree.

Little Cyan Fish wants the number of back edges to be exactly c. Construct such a spanning tree.

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \le T$), indicating the number of test cases.

For each test case, the first line of the input contains three integers n, m, c ($1 \le n \le 10^6$, $n-1 \le m \le \frac{n(n-1)}{2}$, $0 \le c \le m-n+1$). Each of the next m lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$), denoting an undirected edge between vertices u and v. The i-th edge in the input has index i. The graph in each test case is simple and connected, and it is guaranteed that at least one valid spanning tree exists.

It is guaranteed that the sum of n over all test cases does not exceed 10^6 and the sum of m over all test cases does not exceed 2×10^6.

Output

For each test case, output $n-1$ distinct integers on one line: the indices of the edges chosen into the spanning tree.

The selected edges must form a spanning tree rooted at vertex 1 whose number of back edges is exactly c.

If there are multiple valid answers, print any of them.

Sample Input 1

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

Sample Output 1

```
1 3 6 8
1 2 3 10
```

Problem F
Sticks 2
Time limit: 6 seconds

Little Cyan Fish has n sticks, where the length of the i-th stick is 2^{a_i}.

He wants to place all sticks consecutively on a number line from left to right, starting no earlier than position m. Formally, given a sequence $a_1, a_2, \ldots, a_n$ and a starting bound m, let l_i and r_i denote the left and right endpoints of the i-th placed stick, respectively. A placement is *valid* if all of the following conditions hold:

- $l_1 \geq m$;
- $r_i = l_i + 2^{a_i}$ for every $1 \leq i \leq n$;
- l_i is divisible by 2^{a_i} for every $1 \leq i \leq n$;
- $r_i \leq l_{i+1}$ for every $1 \leq i < n$.

For a fixed ordering $a_1, a_2, \ldots, a_n$, define $f(a)$ as the minimum possible value of r_n over all valid placements.

Before placing the sticks, Little Cyan Fish may rearrange $a_1, a_2, \ldots, a_n$ in any order he wishes. Find the maximum possible value of $f(a)$ over all such rearrangements.

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \leq T$), indicating the number of test cases.

For each test case, the first line contains two integers n and m ($1 \leq n \leq 2 \times 10^5$, $0 \leq m < 2^{100}$). The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i < 100$).

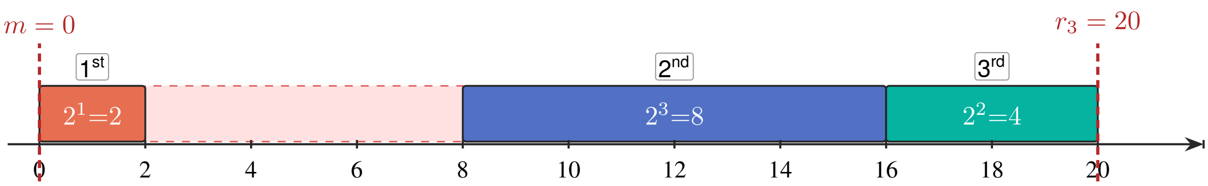
It is guaranteed that the sum of n over all test cases does not exceed 2×10^5.

Output

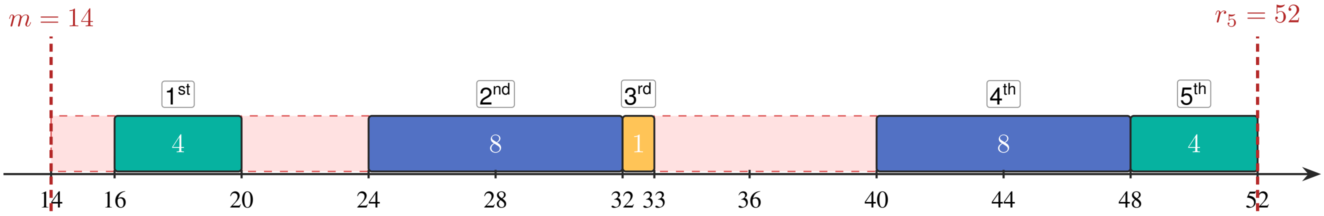
For each test case, output a single integer, indicating the maximum possible value of $f(a)$ after rearranging $a_1, a_2, \ldots, a_n$.

Sample Input 1	Sample Output 1
2 3 0 1 2 3 5 14 0 2 2 3 3	20 52

In the first sample, Little Cyan Fish can rearrange the sticks to have lengths $2^1, 2^3, 2^2$. Placing them at intervals $[0, 2]$, $[8, 16]$, and $[16, 20]$ is valid, so this rearrangement has $f(a) = 20$. This is the maximum possible value.



In the second sample, one optimal rearrangement has stick lengths $4, 8, 1, 8, 4$. The figure shows a valid placement ending at 52, so the answer is 52.



Problem G
Art of Data Structures
Time limit: 5 seconds

Little Cyan Fish is teaching the Data Structure Master Class at the University of Cup. In traditional data structure problems, you would now be handed a pile of queries and asked to evaluate some convoluted expression on a fixed data structure. Oh, come on... who wants to do that in 2026? Little Cyan Fish wants to do something different. He asks you to invent the data structure on your own.

Your task is to construct a rooted binary tree $\mathcal{T}$:

- Every *internal vertex*[1] of $\mathcal{T}$ has exactly two children.
- $\mathcal{T}$ has exactly m leaves, labeled from 1 to m.

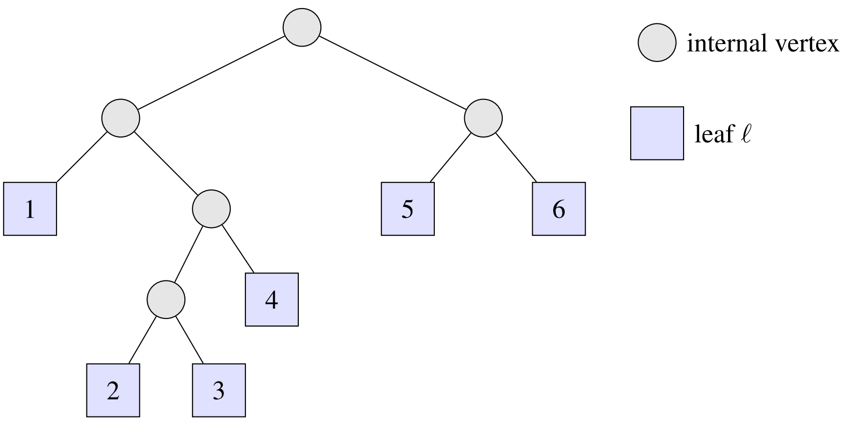


Figure 1: A valid $\mathcal{T}$ for $m = 6$. Every internal vertex has exactly two children, and the leaves carry the labels $1, \ldots, m$ in some order. Here the depth is 5.

For any set S of leaf labels, define its cost on $\mathcal{T}$ as the number of internal vertices v of $\mathcal{T}$ such that the subtree of v contains both:

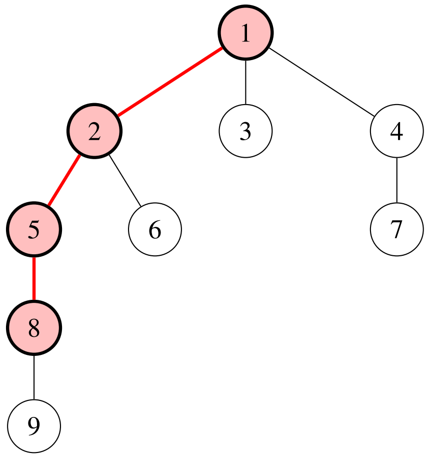
- At least one leaf whose label belongs to S.
- At least one leaf whose label does not belong to S.

Little Cyan Fish gives you two rooted trees T_1 and T_2. Both trees have vertices labeled from 1 to n, and vertex 1 is the root of each tree. He also gives you m ordered pairs (x_i, y_i), where x_i is a vertex of T_1 and y_i is a vertex of T_2. The leaf labeled ℓ in $\mathcal{T}$ is associated with the values x_ℓ and y_ℓ.

For a rooted tree T and a vertex x, let $\text{path}(T, x)$ be the set of vertices on the path from x to the root of T, including both endpoints.

Little Cyan Fish wants you to know that, for each vertex u of T_1, define $Q_1(u) = \{\ell \mid x_\ell \in \text{path}(T_1, u)\}$. Similarly, for each vertex u of T_2, define $Q_2(u) = \{\ell \mid y_\ell \in \text{path}(T_2, u)\}$. Each $Q_i(u)$ is a set of leaf labels of $\mathcal{T}$.

[1] A *leaf* is a vertex with no children, and an *internal vertex* is a vertex that is not a *leaf*.



With $x = 8$: $\text{path}(T_i, 8) = \{8,\ 5,\ 2,\ 1\}$ (highlighted in red).

Figure 2: The set $\text{path}(T_i, x)$ contains every vertex on the unique path from x up to the root, including both endpoints.

The sets that Little Cyan Fish checks are $Q_1(u)$ and $Q_2(u)$ for every $1 \le u \le n$. Little Cyan Fish accepts your data structure if it satisfies both requirements:

- the depth of every vertex in $\mathcal{T}$ is at most 100, where the root has depth 1;
- among all $2n$ checked sets, the maximum cost is at most 16 666.

Show Little Cyan Fish that you are the real master of data structures!

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 10^6$).

The next line of the input contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), describing the tree T_1. The integer p_i is the parent of vertex i in T_1.

The next line of the input contains $n - 1$ integers $p'_2, p'_3, \ldots, p'_n$ ($1 \le p'_i < i$), describing the tree T_2. The integer p'_i is the parent of vertex i in T_2.

The next m lines describe the ordered pairs. The i-th of these lines contains two integers x_i and y_i ($1 \le x_i, y_i \le n$).

Output

Output a single line containing a sequence of integers that describes the binary tree $\mathcal{T}$ you construct.

- A leaf labeled i is described by the integer i.
- An internal vertex is described by the integer 0, followed by the description of its left subtree, then by the description of its right subtree.

Under this encoding, every integer from 1 to m must appear exactly once, and each occurrence of 0 represents one internal vertex.

For example, the sequence 0 1 0 2 3 describes a tree whose root has leaf 1 as its left child and an internal vertex as its right child; that internal vertex has leaves 2 and 3 as its children.

Sample Input 1	Sample Output 1
1 1 1 1	1

Sample Input 2	Sample Output 2
3 3 1 1 1 2 1 1 2 2 3 3	0 1 0 2 3

Sample Input 3	Sample Output 3
5 8 1 2 3 4 1 1 1 1 1 1 2 1 3 2 4 2 5 3 5 5 1 5 3 4	0 0 1 0 0 3 8 0 2 7 0 4 0 5 6

Explanation of Sample 1: The binary tree has a single leaf labeled 1. Its depth is 1, and every possible query has cost 0.

This page is intentionally left blank.

Problem H
Guess the Permutation
Time limit: 3 seconds

This is an interactive problem. Remember to flush the output buffer after every print. To flush your output, you can use:

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python;
- See the documentation for other languages.

Little Cyan Fish has a secret permutation $p_0, p_1, \ldots, p_n$ of length $n+1$, consisting of integers from 0 to n. He wants you to guess the secret permutation. To do that, Little Cyan Fish allows you to perform no more than $1\,024$ queries in the following form:

- `?` i j: Get the index k ($0 \le k \le n$) such that $p_k = |p_i - p_j|$.

Of course, to perform a valid query, you must guarantee that i and j are two integers within $[0, n]$. For example, if $p_0 = 2, p_1 = 0, p_2 = 1$, then the query "`? 0 2`" will give you the index 2 since $|p_0 - p_2| = |2 - 1| = 1 = p_2$.

Little Cyan Fish wants you to come up with a strategy to find the permutation. Of course, Little Cyan Fish does not want you to win the game too easily. Therefore, he will **adaptively** change the permutation p to trap you (see the "Interaction Protocol" for more details).

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \le T \le 100$), indicating the number of test cases.

For each test case, the first line of the input contains a single integer n ($1 \le n \le 1\,000$). Then, the interaction begins.

Interaction Protocol

To make a query, print a line in the following format:

$$?\ i\ j$$

where i and j are two integers from 0 to n. You must ensure that the number of queries you make does not exceed $1\,024$. As a response to the query, you will get an integer k such that $p_k = |p_i - p_j|$.

To report the answer, output a line in the following format:

$$!\ p_0\ p_1\ \ldots\ p_n$$

where p is the permutation you wish to report.

After outputting a query or reporting the answer, do not forget to output a newline character and flush the output stream.

In this problem, the interactor is **adaptive**. This means that the permutation p might change depending on your queries, as long as it does not contradict previous queries you asked.

Read **Sample Interaction 1** **Write**

Read	Write
1 3	
	? 2 2
1	
	? 0 3
3	
	! 2 0 3 1

Testing Tool: A testing tool is provided to help contestants develop and test their solutions. You can download this tool from the attachments. Executing the tool with a "`-h`" option should describe how to use the tool. The testing tool will only implement some test scenarios and only some functionality of the real judge program.

Problem I
Defense Line
Time limit: 2 seconds


Prepared to deploy a turret.

Little Cyan Fish has been hired to test yet another tower-defense game. The studio insists that the game is almost ready to ship, although the boss still walks along a perfectly straight line, the turrets can only be placed at integer coordinates, and the boss's shield has developed a highly specific grudge against the rightmost turret on the field.

The battlefield is represented by the number line. There are n turrets available for deployment. The i-th turret has a_i units of energy and can be deployed at most once. Deploying a turret means choosing an empty integer coordinate and placing a turret there. A turret with positive energy is active; once its energy becomes 0, it disappears from the battlefield. This is called automatic cleanup in the manual.

The game is played in rounds. During each round, the following actions happen, in order:

1. Little Cyan Fish chooses one of two options: deploy one previously undeployed turret at an empty integer coordinate on the number line, or do nothing.
2. Each deployed turret whose energy is positive deals 1 damage to the boss.
3. Among all deployed turrets whose energy is positive, the one at the greatest coordinate is cursed by the boss and loses 1 energy. If that turret's energy drops to 0, the turret disappears from the defense line.

The simulator is configured to run for the first 9^{9^9} rounds, because someone typed that number into a settings panel and nobody wants to touch production. Help Little Cyan Fish find the maximum total damage that can be dealt to the boss.

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \le T$), indicating the number of test cases.

For each test case, the first line contains an integer n ($1 \le n \le 2 \times 10^5$). The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed 2×10^5.

Output

For each test case, output a single line containing a single integer: the maximum total damage.

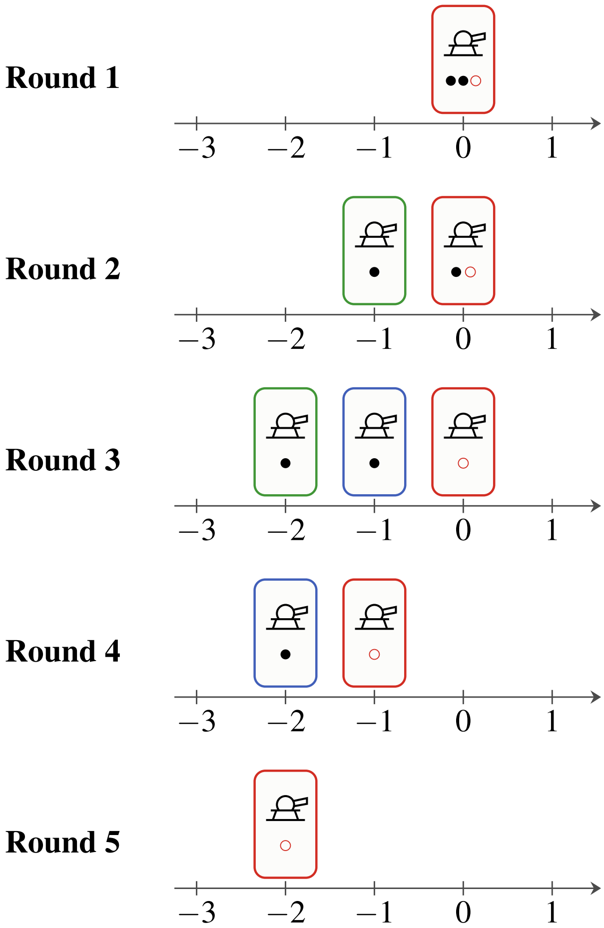
Sample Input 1

```
4
3
3 1 1
3
2 2 2
4
4 5 6 7
3
123456789 987654321 998244353
```

Sample Output 1

```
9
12
60
5093498490
```

Explanation of Sample 1: Place a turret with energy 3 at 0, then place turrets with energy 1 at $-1, -2$ successively.



Problem J
Emergency Pipeline Repair
Time limit: 5 seconds

A severe water leak has been detected in the city's underground utility network! Little Cyan Fish has retrieved the layout of the main water pipelines, which reveals that there are n aging primary water pipes running beneath the city. In the 2D schematic map, each pipe is modeled as an infinite straight line in the plane.

To quickly seal the leaks and restore the water supply, the city has deployed a new emergency repair system: a retractable robotic repair arm that can be inserted from the surface and extended underground as a straight line segment.

The arm is equipped with smart connectors and can automatically dock and perform on-site sealing and repair, as long as the segment intersects a water pipe, i.e., shares at least one common point with it. However, the cost and mechanical complexity of the repair arm are directly proportional to its length. Moreover, geological constraints limit how far the arm can extend laterally underground.

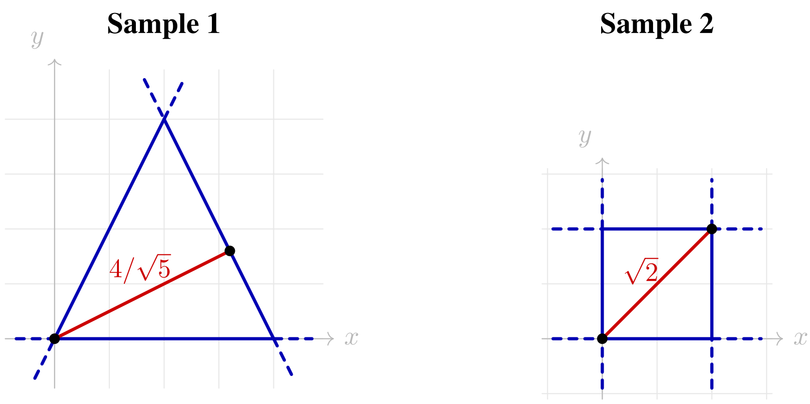


Illustration of the two sample answers.

Therefore, Little Cyan Fish must determine the shortest possible length of such a segment that intersects all n water pipes. As the lead algorithm engineer of the city's smart infrastructure division, Little Cyan Fish is urgently tasked with solving this critical geometric optimization problem.

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \le T$), indicating the number of test cases.

For each test case, the first line contains an integer n ($1 \le n \le 10^5$), indicating the number of pipes. Each of the following n lines contains four integers x_1, y_1, x_2, and y_2 ($-10^3 \le x_1, y_1, x_2, y_2 \le 10^3$, $(x_1, y_1) \ne (x_2, y_2)$), indicating a pipe passing through the points (x_1, y_1) and (x_2, y_2).

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output a line containing a real number, indicating the shortest length of such a segment that intersects all n water pipes.

Your answer is acceptable if its absolute or relative error does not exceed 10^{-6}. Formally speaking, suppose that your output is a and the jury's answer is b, and your output is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Sample Input 1

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

Sample Output 1

```
1.788854381948
1.414213562373
0
```

Problem K
Call You With Your Name 3
Time limit: 1 second

Today, Little Cyan Fish invites you to continue studying human nicknames with him.

In Little Cyan Fish's universe, human nicknames can be represented as a string containing only lowercase Latin letters (`a` to `z`). For example, "`qingyu`" and "`xiuga`" are human nicknames, but "`Abacde`" is not a human nickname.

Little Cyan Fish considers a name s to be a *Lyndon* string if and only if for each proper suffix[2] t of s, s is lexicographically smaller than t. For example, "`abacde`" is a Lyndon string, but "`qingyu`" is not a Lyndon string (because the original string is not lexicographically smaller than the proper suffix "`ingyu`").

After the previous studies, Little Cyan Fish finds that some human nicknames can be divided into several consecutive parts, where none of the parts is Lyndon, and the parts are sorted in non-decreasing lexicographical order.

Now, Little Cyan Fish gives you a human nickname s, and you need to determine whether there exist integers k and $p_1, p_2, \ldots, p_k$ such that:

- $1 \le k \le |s|$, $1 \le p_1 < p_2 < \ldots < p_k = |s|$;
- if we define $p_0 = 0$, then for each $1 \le i \le k$, the string $s[p_{i-1}+1 \ldots p_i]$ is not a Lyndon string, and;
- for each $2 \le i \le k$, $s[p_{i-2}+1 \ldots p_{i-1}]$ is not lexicographically larger than $s[p_{i-1}+1 \ldots p_i]$.

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \le T$), indicating the number of test cases.

For each test case, the input contains a single line with a string s ($1 \le |s| \le 2 \times 10^6$), indicating the human nickname.

It is guaranteed that the sum of $|s|$ over all test cases does not exceed 2×10^6.

Output

For each test case, if no such integers exist, output a single line containing a single word "`No`".

Otherwise, output "`Yes`" on the first line. On the second line, output a single integer k ($1 \le k \le |s|$). On the third line, output k integers $p_1, p_2, \ldots, p_k$ ($1 \le p_1 < p_2 < \ldots < p_k = |s|$), indicating your division of the nickname.

If there are multiple answers, you may output any of them.

Sample Input 1

```
4
a
aaa
ababa
littlecyanfishsaidadisabacde
```

Sample Output 1

```
No
Yes
1
3
Yes
2
3 5
Yes
1
28
```

[2] A proper suffix is a non-empty suffix that is not equal to the original string. For example, "`qqoj`" has 3 proper suffixes, which are "`j`", "`oj`", and "`qoj`".

This page is intentionally left blank.

Problem L
Hanoi Tower Puzzle
Time limit: 8 seconds

Little Cyan Fish has three pegs on his desk, labeled A, B, and C. He also has n disks of distinct sizes, numbered 1 through n from small to large. Initially, all disks are stacked on peg A, with smaller disks above larger disks.

Little Cyan Fish wants to transform this initial configuration into a given target configuration. Unlike the classical Tower of Hanoi, this puzzle allows the following special type of move.

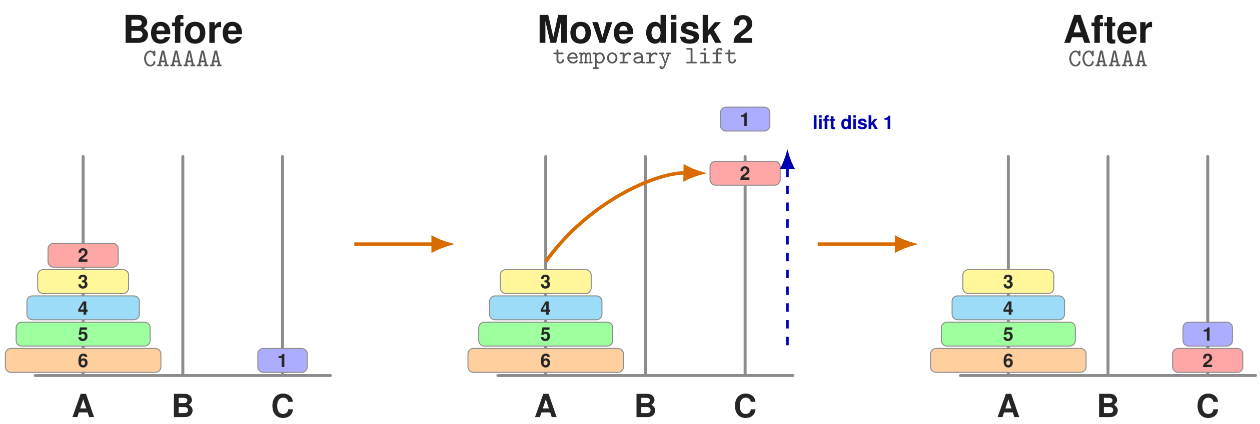
In one move, Little Cyan Fish performs the following operations:

1. Choose a non-empty source peg and remove its top disk;
2. Choose a destination peg that is different from the source peg;
3. On the destination peg, temporarily lift all disks that are smaller than the disk being moved, preserving their original top-to-bottom order;
4. Place the moved disk onto the destination peg;
5. Put the lifted disks back onto the destination peg, preserving their original top-to-bottom order.

Equivalently, a move takes the top disk from a source peg and inserts it into a different destination peg, below all smaller disks and above all larger disks on that destination peg. The relative order of the disks originally on the destination peg does not change.

A configuration can be represented by a string. For a configuration string t, the i-th character of t denotes the peg containing disk i, where disk i is the i-th smallest disk. Since disks on each peg are always ordered with smaller disks above larger disks, such a string uniquely determines the entire configuration.

For example, the following figure illustrates one move from the first sample.



One move in sample 1: CAAAAA → CCAAAA.

The target configuration is specified by a string s. The i-th character of s is the peg on which disk i must be located in the final configuration.

Help Little Cyan Fish compute the minimum number of moves required to reach the target configuration from the initial configuration.

Input

There are multiple test cases. The first line of the input contains a single integer T ($1 \le T$), indicating the number of test cases.

For each test case, the first line of the input contains a non-empty string s consisting only of characters `A`, `B`, and `C`. The length of s is the number of disks in the corresponding test case, and the i-th character of s specifies the peg on which disk i must be located in the final configuration.

It is guaranteed that the sum of $|s|$ over all test cases does not exceed 3×10^6.

Output

For each test case, output a single line containing a single integer, denoting the minimum number of moves required to reach the target configuration from the initial configuration.

Sample Input 1	Sample Output 1
4 BCABCA BCBAB AAAAAAAAABC BACBACBACBACBACBACBACBAC	7 8 20 88

Explanation of Sample 1: For the first test case, one optimal strategy is

$$\texttt{AAAAAA} \to \texttt{CAAAAA} \to \texttt{CCAAAA} \to \texttt{CCBAAA} \to \texttt{CCBBAA} \to \texttt{CCBBCA} \to \texttt{CCABCA} \to \texttt{BCABCA}.$$

For the second test case, one optimal strategy is

$$\texttt{AAAAA} \to \texttt{BAAAA} \to \texttt{BCAAA} \to \texttt{BCBAA} \to \texttt{BCBCA} \to \texttt{BCBCB} \to \texttt{BABCB} \to \texttt{BABAB} \to \texttt{BCBAB}.$$

Problem M
Minesweeper
Time limit: 2 seconds

This is a communication problem.


A dangerous mine!

Little Cyan Fish loves the game *Minesweeper* a lot, and he really wants to play this game during the Universal Cup Finals. Therefore, he gives you the opportunity to experience another Minesweeper-like game, called *CyanSweeper*, which is played on an undirected connected planar graph instead of a grid.

To play this game, he gives you a planar graph with n vertices and m edges. The vertices are labeled from 1 to n. There are k special vertices containing bombs, and the game consists of two phases.

First run. Little Cyan Fish gives you the graph and the labels of the k bomb vertices $b_1, b_2, \ldots, b_k$. You must produce an array $a_1, a_2, \ldots, a_n$ as follows:

- For every vertex v without a bomb, a_v must equal the number of bomb-containing neighbors of v (the standard *Minesweeper* count).
- For every vertex v with a bomb, a_v may be any integer in $\{1, 2, \ldots, k\}$, subject to the constraint that the values $a_{b_1}, a_{b_2}, \ldots, a_{b_k}$ at the bomb vertices form a permutation of $\{1, 2, \ldots, k\}$.

In addition to the array a, Little Cyan Fish asks you to transmit a non-empty binary string s of length at most 100 to the second run.

Second run. Little Cyan Fish gives you the same graph (with the same vertex labels), the array $a_1, \ldots, a_n$ exactly as produced in the first run, and the transmitted binary string s. However, you will not be given the labels of the bomb vertices. It is your job to output the set of bomb vertices based on the graph, the array a, and the transmitted string s.

Write a program to play *CyanSweeper*.

Input

There are multiple test cases. The first line of the input contains two integers r ($r \in \{1, 2\}$) and T ($1 \le T \le 10$), indicating the run number and the number of test cases.

For each test case:

- If $r = 1$, the first line contains three integers n, m, k ($1 \le k \le n \le 5 \times 10^5$, $n - 1 \le m \le 10^6$). The next m lines each contain two integers u and v ($1 \le u, v \le n$, $u \ne v$), describing an undirected edge. The graph is guaranteed to be connected, simple, and planar. The next line contains k distinct integers $b_1, b_2, \ldots, b_k$ ($1 \le b_i \le n$), the indices of the bomb vertices.
- If $r = 2$, the first line contains two integers n and m. The next m lines describe the edges in the same format as above. The following line contains n integers $a_1, a_2, \ldots, a_n$, exactly as the first-run solution produced them. The last line contains the binary string s that the first-run solution transmitted.

It is guaranteed that the sum of n over all test cases does not exceed 5×10^5 and the sum of m over all test cases does not exceed 10^6.

Output

If $r = 1$, for each test case output the array $a_1, a_2, \ldots, a_n$ on one line separated by spaces, followed by the binary string s on the next line. The array must satisfy the rules described above, and s must consist only of the characters `0` and `1` and have length at least 1 and at most 100.

If $r = 2$, for each test case output two lines. The first line contains a single integer k, the number of bombs. The second line should contain k distinct integers corresponding to the bomb vertex indices. You may output the bomb vertex indices in any order. The reported set must coincide exactly with the set of bombs from the corresponding first-run input.

Sample Input 1 **Pass 1** **Sample Output 1**

Sample Input 1	Sample Output 1
1 1 4 3 2 1 2 2 3 3 4 2 4	1 2 2 1 0101

Sample Input 1 **Pass 2** **Sample Output 1**

Sample Input 1	Sample Output 1
2 1 4 3 1 2 2 3 3 4 1 2 2 1 0101	2 2 4

Testing Tool: A testing tool is provided to help contestants develop and test their solutions. You can download this tool from the attachments. Executing the tool with a "`-h`" option should describe how to use the tool. The testing tool will only implement some test scenarios and only some functionality of the real judge program.