A. Math Division

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

Ecrade has an integer x. He will show you this number in the form of a binary number of length n.

There are two kinds of operations.

1. Replace x with $\left\lfloor \frac{x}{2} \right\rfloor$, where $\left\lfloor \frac{x}{2} \right\rfloor$ is the greatest integer $\leq \frac{x}{2}$.
2. Replace x with $\left\lceil \frac{x}{2} \right\rceil$, where $\left\lceil \frac{x}{2} \right\rceil$ is the smallest integer $\geq \frac{x}{2}$.

Ecrade will perform several operations until x becomes 1. Each time, he will independently choose to perform either the first operation or the second operation with probability $\frac{1}{2}$.

Ecrade wants to know the expected number of operations he will perform to make x equal to 1, modulo $10^9 + 7$. However, it seems a little difficult, so please help him!

Input

The first line contains a single integer t ($1 \leq t \leq 10^5$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \leq n \leq 10^5$) — the length of x in binary representation.

The second line of each test case contains a binary string of length n: the number x in the binary representation, presented from the most significant bit to the least significant bit. It is guaranteed that the most significant bit of x is 1.

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

Output

For each test case, print a single integer representing the expected number of operations Ecrade will perform to make x equal to 1, modulo $10^9 + 7$.

Formally, let $M = 10^9 + 7$. It can be shown that the exact answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \bmod M$. In other words, output such an integer x that $0 \leq x < M$ and $x \cdot q \equiv p \pmod{M}$.

Standard Input	Standard Output
3 3 110 3 100 10 1101001011	500000006 2 193359386

Note

For simplicity, we call the first operation $\mathrm{OPER}\ 1$ and the second operation $\mathrm{OPER}\ 2$.

In the first test case, $x = 6$, and there are six possible series of operations:

- $6 \xrightarrow{\mathrm{OPER}\ 1} 3 \xrightarrow{\mathrm{OPER}\ 1} 1$, the probability is $\frac{1}{4}$.
- $6 \xrightarrow{\mathrm{OPER}\ 1} 3 \xrightarrow{\mathrm{OPER}\ 2} 2 \xrightarrow{\mathrm{OPER}\ 1} 1$, the probability is $\frac{1}{8}$.
- $6 \xrightarrow{\mathrm{OPER}\ 1} 3 \xrightarrow{\mathrm{OPER}\ 2} 2 \xrightarrow{\mathrm{OPER}\ 2} 1$, the probability is $\frac{1}{8}$.
- $6 \xrightarrow{\mathrm{OPER}\ 2} 3 \xrightarrow{\mathrm{OPER}\ 1} 1$, the probability is $\frac{1}{4}$.
- $6 \xrightarrow{\mathrm{OPER}\ 2} 3 \xrightarrow{\mathrm{OPER}\ 2} 2 \xrightarrow{\mathrm{OPER}\ 1} 1$, the probability is $\frac{1}{8}$.
- $6 \xrightarrow{\mathrm{OPER}\ 2} 3 \xrightarrow{\mathrm{OPER}\ 2} 2 \xrightarrow{\mathrm{OPER}\ 2} 1$, the probability is $\frac{1}{8}$.

Thus, the expected number of operations is
$2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} = \frac{5}{2} \equiv 500\,000\,006 \pmod{10^9 + 7}$.

,

B. Balancing

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

Ecrade has an integer array $a_1, a_2, \ldots, a_n$. It's guaranteed that for each $1 \leq i < n$, $a_i \neq a_{i+1}$.

Ecrade can perform several operations on the array to make it strictly increasing.

In each operation, he can choose two integers l and r ($1 \leq l \leq r \leq n$) and replace $a_l, a_{l+1}, \ldots, a_r$ with any $r - l + 1$ integers $a'_l, a'_{l+1}, \ldots, a'_r$ satisfying the following constraint:

- For each $l \leq i < r$, the comparison between a'_i and a'_{i+1} is the same as that between a_i and a_{i+1}, i.e., if $a_i < a_{i+1}$, then $a'_i < a'_{i+1}$; otherwise, if $a_i > a_{i+1}$, then $a'_i > a'_{i+1}$; otherwise, if $a_i = a_{i+1}$, then $a'_i = a'_{i+1}$.

Ecrade wants to know the minimum number of operations to make the array strictly increasing. However, it seems a little difficult, so please help him!

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \leq n \leq 2 \cdot 10^5$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$).

It is guaranteed that for each $1 \leq i < n$, $a_i \neq a_{i+1}$.

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

Output

For each test case, output a single integer representing the minimum number of operations to make the array strictly increasing.

Standard Input	Standard Output
4 3 3 2 1 3 3 1 2 4 -2 -5 5 2 7 1 9 1 9 8 1 0	2 1 1 3

Note

In the first test case, a possible way to obtain the minimum number of operations:

- In the first operation, choose $l = 2, r = 2$ and $a'_2 = 4$, then $a = [3, 4, 1]$;

- In the second operation, choose $l = 1, r = 2$ and $a'_1 = -1, a'_2 = 0$, then $a = [-1, 0, 1]$.

In the second test case, a possible way to obtain the minimum number of operations:

- In the first operation, choose $l = 2, r = 3$ and $a'_2 = 4, a'_3 = 5$, then $a = [3, 4, 5]$.

In the third test case, a possible way to obtain the minimum number of operations:

- In the first operation, choose $l = 2, r = 3$ and $a'_2 = -1, a'_3 = 1$, then $a = [-2, -1, 1, 2]$.

,

C. Quaternary Matrix

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

A matrix is called quaternary if all its elements are 0, 1, 2, or 3.

Ecrade calls a quaternary matrix A *good* if the following two properties hold.

- The bitwise XOR of all numbers in each row of matrix A is equal to 0.
- The bitwise XOR of all numbers in each column of matrix A is equal to 0.

Ecrade has a quaternary matrix of size $n \times m$. He is interested in the minimum number of elements that need to be changed for the matrix to become *good*, and he also wants to find one of the possible resulting matrices.

However, it seems a little difficult, so please help him!

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 2 \cdot 10^5$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 10^3$).

This is followed by n lines, each containing exactly m characters and consisting only of 0, 1, 2, and 3, describing the elements of Ecrade's matrix.

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

Output

For each test case, print the minimum number of elements that need to be changed for the matrix to become *good* on the first line, then print n lines, each containing exactly m characters and consisting only of 0, 1, 2, and 3, describing one of the possible resulting matrices.

If there are multiple possible resulting matrices, output any of them.

Standard Input	Standard Output
5 3 3 313 121 313 3 3 000 000 000 4 4 0123 1230 2301 3012	3 213 101 312 0 000 000 000 0 0123 1230 2301 3012 6

4 4 1232 2110 3122 1311 4 4 1232 2110 3122 1312	0132 2310 3131 1313 5 0132 2310 3120 1302

,

D. MST in Modulo Graph

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   512 megabytes
```

You are given a complete graph with n vertices, where the i-th vertex has a weight p_i. The weight of the edge connecting vertex x and vertex y is equal to $\max(p_x, p_y) \bmod \min(p_x, p_y)$.

Find the smallest total weight of a set of $n - 1$ edges that connect all n vertices in this graph.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test contains an integer n ($1 \le n \le 5 \cdot 10^5$).

The next line of each test contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 5 \cdot 10^5$).

The sum of n over all test cases does not exceed $5 \cdot 10^5$.

The sum of $\max(p_1, p_2, \ldots, p_n)$ over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output a single integer — the weight of the minimum spanning tree.

Standard Input	Standard Output
4 5 4 3 3 4 4 10 2 10 3 2 9 9 4 6 4 6 12 33 56 48 41 89 73 99 150 55 100 111 130 7 11 45 14 19 19 8 10	1 0 44 10

Note

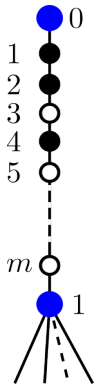
In the first test case, one of the possible ways to connect the edges is to draw the edges $(1, 2)$, $(1, 4)$, $(1, 5)$, $(2, 3)$. The weight of the first edge is $\max(p_1, p_2) \bmod \min(p_1, p_2) = 4 \bmod 3 = 1$, and the weights of all other edges are 0.

,

E. Quantifier

Input file:	standard input
Output file:	standard output
Time limit:	2.5 seconds
Memory limit:	1024 megabytes

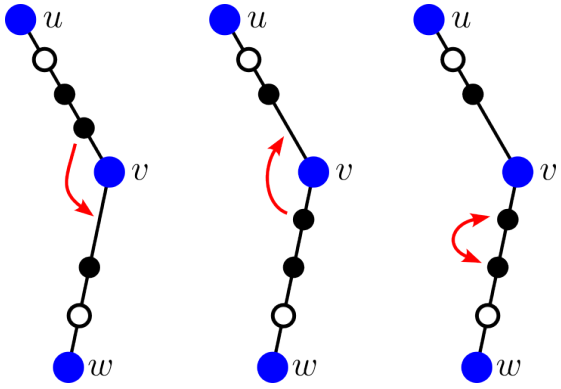
Given a rooted tree with $n+1$ nodes labeled from 0 to n, where the root is node 0, and **its only child is node** 1. There are m **distinct** chips labeled from 1 to m, each colored either black or white. Initially, they are arranged on edge $(0, 1)$ from top to bottom in ascending order of labels.



The initial position of the chips. Tree nodes are shown in blue.

You can perform the following operations any number of times (possibly zero) in any order:

- Select two edges (u, v) and (v, w) such that u is the parent of v and v is the parent of w, where edge (u, v) contains at least one chip. Move the **bottommost** chip on edge (u, v) to the **topmost** place on edge (v, w), i. e., above all existing chips on (v, w).
- Select two edges (u, v) and (v, w) such that u is the parent of v and v is the parent of w, where edge (v, w) contains at least one chip. Move the **topmost** chip on edge (v, w) to the **bottommost** place on edge (u, v), i. e., below all existing chips on (u, v).
- Select two **adjacent** chips of the same color on the same edge, and swap their positions.



Permitted operations.

Each chip i has a movement range, defined as all edges on the simple path from the root to node d_i. During operations, you must ensure that no chip is moved to an edge outside its movement range.

Finally, you must move all chips back to edge $(0, 1)$. It can be found that the order of the chips may change. Compute the number of possible permutations of chips for the final arrangement on the edge $(0, 1)$ modulo $998\,244\,353$.

A permutation of chips is defined as a sequence of length m consisting of the **labels** of the chips from top to bottom.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5000$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 5000$) — the size of the tree minus one (i. e., the tree has $n + 1$ nodes) and the number of chips.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i < i$) — the parent nodes for nodes from 1 to n. It is guaranteed that $p_i = 0$ if and only if $i = 1$ (the root's only child is node 1).

The third line contains m integers $c_1, c_2, \ldots, c_m$ ($c_i \in \{0, 1\}$) — the color of each chip (0 for black, 1 for white).

The fourth line contains m integers $d_1, d_2, \ldots, d_m$ ($1 \le d_i \le n$) — the movement range of each chip.

It is guaranteed that the sum of n does not exceed 5000 and the sum of m does not exceed 5000 across all test cases.

Output

For each test case, output a single integer — the number of possible permutations of chips modulo $998\,244\,353$.

Standard Input	Standard Output
4 3 2 0 1 1 0 1 2 3 4 4 0 1 1 2 0 0 1 1 1 2 3 3 6 6 0 1 1 1 4 5 0 0 0 0 1 1 5 6 1 2 4 3 16 15 0 1 1 3 1 3 4 3 3 7 1 6 11 5 8 10 1 0 1 1 0 1 1 1 1 0 1 1 0 0 0 12 14 13 10 9 16 11 14 13 15 16 10 2 2 5	2 8 108 328459046

Note

In the first test case, we can reach follow 2 permutations: $(1, 2)$ and $(2, 1)$.

In the second test case, we can reach follow 8 permutations: $(1, 2, 3, 4)$, $(1, 2, 4, 3)$, $(1, 3, 2, 4)$, $(1, 3, 4, 2)$, $(1, 4, 2, 3)$, $(1, 4, 3, 2)$, $(2, 1, 3, 4)$, and $(2, 1, 4, 3)$.

,

F. Hot Matrix

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 512 megabytes

Piggy Zhou loves matrices, especially those that make him get excited, called *hot matrix*.

A hot matrix of size $n \times n$ can be defined as follows. Let $a_{i,j}$ denote the element in the i-th row, j-th column ($1 \leq i, j \leq n$).

1. Each column and row of the matrix is a permutation of all numbers from 0 to $n-1$.
2. For each pair of indices i, j, such that $1 \leq i, j \leq n$, $a_{i,j} + a_{i,n-j+1} = n-1$.
3. For each pair of indices i, j, such that $1 \leq i, j \leq n$, $a_{i,j} + a_{n-i+1,j} = n-1$.
4. All ordered pairs $(a_{i,j}, a_{i,j+1})$, where $1 \leq i \leq n$, $1 \leq j < n$, are distinct.
5. All ordered pairs $(a_{i,j}, a_{i+1,j})$, where $1 \leq i < n$, $1 \leq j \leq n$, are distinct.

Now, Piggy Zhou gives you a number n, and you need to provide him with a hot matrix if the hot matrix exists for the given n, or inform him that he will never get excited if the hot matrix does not exist for the given n.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 1000$). The description of the test cases follows.

The only line of each test case contains one integer n ($1 \leq n \leq 3000$).

It is guaranteed that the sum of n across all test cases does not exceed 3000.

Output

For each test case, print "NO" (without quotes) if the hot matrix does not exist for the given n.

Otherwise, print "YES" (without quotes) on the first line. Then output n rows, with n numbers in each row, representing the hot matrix of size $n \times n$ that meets the requirements of the problem.

If there are multiple solutions, print any of them.

Standard Input	Standard Output
4 1 2 3 4	YES 0 YES 0 1 1 0 NO YES 0 1 2 3 1 3 0 2 2 0 3 1 3 2 1 0

Note

In the first, second and fourth test case, it can be verified that the matrix provided in the example meets all the conditions stated in the problem.

In the third test case, by enumerating all possible matrices, it can be proven that there is no hot matrix that satisfies the conditions of the problem.

,

G1. Hard Formula

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 1024 megabytes

This is the easy version of the problem. The difference between the versions is that in this version, the limits on n and the *time limit* are smaller. You can hack only if you solved all versions of this problem.

You are given an integer n, and you need to compute $(\sum_{k=1}^{n} k \bmod \varphi(k)) \bmod 2^{32}$, where $\varphi(k)$ equals the number of positive integers no greater than k that are coprime with k.

Input

The only line contains a single integer n ($1 \leq n \leq 10^{10}$).

Output

Print a single integer, representing $(\sum_{k=1}^{n} k \bmod \varphi(k)) \bmod 2^{32}$.

Standard Input	Standard Output
5	2
10000000	2316623097
10000000000	282084447

,

G2. Hard Formula (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     10 seconds
Memory limit:   1024 megabytes
```

This is the hard version of the problem. The difference between the versions is that in this version, the limit on n and the *time limit* are higher. You can hack only if you solved all versions of this problem.

You are given an integer n, and you need to compute $(\sum_{k=1}^{n} k \bmod \varphi(k)) \bmod 2^{32}$, where $\varphi(k)$ equals the number of positive integers no greater than k that are coprime with k.

Input

The only line contains a single integer n ($1 \leq n \leq 10^{12}$).

Output

Print a single integer, representing $(\sum_{k=1}^{n} k \bmod \varphi(k)) \bmod 2^{32}$.

Standard Input	Standard Output
5	2
10000000	2316623097
10000000000	282084447