# Problem A. Pair Pressure

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 3 seconds |
| Memory limit: | 1024 megabytes |

An array $b$ of length $2k$ is called *good* if $b_{2i-1} = b_{2i}$ is satisfied over all $1 \le i \le k$. In other words, $b$ consists of $k$ consecutive pairs of equal elements.

For an array $a$, define its *score* as the maximum integer $k$ such that there exists a *good* subsequence* of $a$ of length $2k$.

You are given two integers $n$ and $M$, where $M$ is a prime number. Find the sum, modulo $M$, of the scores over all arrays of length $2n$ where each integer from 1 to $n$ appears exactly twice.

* A subsequence of an array is a sequence that can be derived by deleting zero or more elements without changing the order of the remaining elements.

## Input

The first line contains a single integer $t$ ($1 \le t \le 20$) — the number of test cases.

Each of the next $t$ lines contains two space-separated integers $n$ ($1 \le n \le 400$) and $M$ ($10^8 \le M \le 10^9$). It is guaranteed that $M$ is a prime number.

It is guaranteed that the sum of $n$ over all test cases does not exceed 400.

## Output

For each test case, print the answer modulo $M$.

## Example

| standard input | standard output |
|---|---|
| 3<br>2 998244353<br>3 973733479<br>4 998244353 | 8<br>150<br>4944 |

## Note

In the first test case, for $n = 2$, there are 6 possible arrays where each integer from 1 to 2 appears exactly twice. The arrays and their respective scores are:

- $[1, 1, 2, 2]$: score 2 (using the subsequence $[1, 1, 2, 2]$).
- $[1, 2, 1, 2]$: score 1 (using the subsequence $[1, 1]$ or $[2, 2]$).
- $[1, 2, 2, 1]$: score 1 (using the subsequence $[1, 1]$ or $[2, 2]$).
- $[2, 1, 1, 2]$: score 1 (using the subsequence $[1, 1]$ or $[2, 2]$).
- $[2, 1, 2, 1]$: score 1 (using the subsequence $[1, 1]$ or $[2, 2]$).
- $[2, 2, 1, 1]$: score 2 (using the subsequence $[2, 2, 1, 1]$).

The sum of all scores is $2 + 1 + 1 + 1 + 1 + 2 = 8$.

# Problem B. The Last Bit of Us

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

Ellie has a tree with $n$ nodes. Each node $i$ has a binary value $a_i$ which is either 0 or 1.

In a post-apocalyptic digital world, where every bit counts for survival, Ellie must eliminate all traces of digital presence by performing the following operation any number of times:

- Select an edge $(u, v)$ in the tree. Then, simultaneously flip the values of $a_u$ and $a_v$ (i.e. change 0 to 1 and 1 to 0).

Ellie wants to convert all values in the tree to 0 to escape detection. Your task is to determine if this is possible, and if so, find the minimum number of operations required.

## Input

The first line contains a single integer $t$ ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains an integer $n$ ($2 \le n \le 3 \cdot 10^5$) — the number of nodes in the tree.

The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($a_i \in \{0, 1\}$) — the initial values of the nodes.

The next $n-1$ lines describe the edges of the tree. Each line contains two integers $u$ and $v$ ($1 \le u, v \le n$), denoting an undirected edge between nodes $u$ and $v$. It is guaranteed that the given input forms a tree.

It is guaranteed that the sum of $n$ over all test cases does not exceed $3 \cdot 10^5$.

## Output

For each test case, if it is impossible to convert all values to 0, output $-1$. Otherwise, output the minimum number of operations required.

## Example

| standard input | standard output |
|---|---|
| 2<br>8<br>0 0 1 1 0 1 0 0<br>2 8<br>7 5<br>1 8<br>5 1<br>6 7<br>3 5<br>4 3<br>6<br>0 0 0 1 0 1<br>3 6<br>4 2<br>2 5<br>6 1<br>1 2 | -1<br>3 |

## Note

In the first test case, it is impossible to convert all values to 0, so the output is $-1$.

In the second test case, one possible sequence of operations with the minimum number of operations is:

- Select the edge $(1, 2)$, so both $a_1$ and $a_2$ are flipped to 1.
- Select the edge $(4, 2)$, so both $a_4$ and $a_2$ are flipped to 0.
- Select the edge $(6, 1)$, so both $a_6$ and $a_1$ are flipped to 0.

It can be shown that it is not possible to convert all values to 0 in fewer than 3 operations.

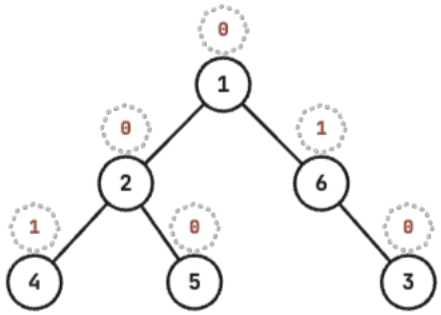


Figure: tree from the second test case.

# Problem C. Triangle Trap

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

You are given $n$ distinct points $p_1, p_2, \ldots, p_n$ on a 2D plane. No three points are collinear.

Please find four distinct integers $i$, $j$, $k$, and $l$ such that $p_i$ lies **strictly** inside the triangle formed by points $p_j$, $p_k$, and $p_l$.

## Input

The first line contains an integer $t$ ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains an integer $n$ ($4 \le n \le 4000$) — the number of points.

Each of the next $n$ lines contains two integers $x_i$ and $y_i$ ($1 \le x_i, y_i \le 5 \cdot 10^5$) — the coordinates of the $i$-th point. It is guaranteed that all points are distinct and no three points are collinear.

It is guaranteed that the sum of $n$ over all test cases does not exceed 4000.

## Output

For each test case, output four distinct integers $i$, $j$, $k$, and $l$ ($1 \le i, j, k, l \le n$) that satisfy the above condition.

If multiple solutions exist, **output the one with the smallest index** $i$. If there are still multiple solutions with the same smallest index $i$, output any of them. If no such four points exist, output $-1$.

## Example

| standard input | standard output |
|---|---|
| 2<br>4<br>1 1<br>1 5<br>5 1<br>5 5<br>7<br>1 2<br>1 7<br>2 1<br>2 8<br>3 4<br>5 2<br>7 1 | -1<br>5 1 7 4 |

## Note

In the first test case, there are no four distinct indices $i$, $j$, $k$, and $l$ such that $p_i$ lies strictly inside the triangle formed by $p_j$, $p_k$, and $p_l$. Therefore, the output is $-1$.

In the second test case, one valid combination with the smallest index $i$ is $i = 5$, $j = 1$, $k = 7$, and $l = 4$ as point $p_5 = (3, 4)$ lies strictly inside the triangle formed by points $p_1 = (1, 2)$, $p_7 = (7, 1)$, and $p_4 = (2, 8)$.

Another possible combination, but with a larger index $i$, is $i = 6$, $j = 3$, $k = 7$, and $l = 5$ as point $p_6 = (5, 2)$ lies strictly inside the triangle formed by points $p_3 = (2, 1)$, $p_7 = (7, 1)$, and $p_5 = (3, 4)$.

However, the problem requires us to output the solution with the smallest index $i$. Since $i = 5$ is smaller than $i = 6$, we must choose the first combination.

If there are multiple triangles where $p_5$ is inside (that is, multiple valid combinations with $i = 5$ but different $j, k, l$ values), any of these combinations would be acceptable.

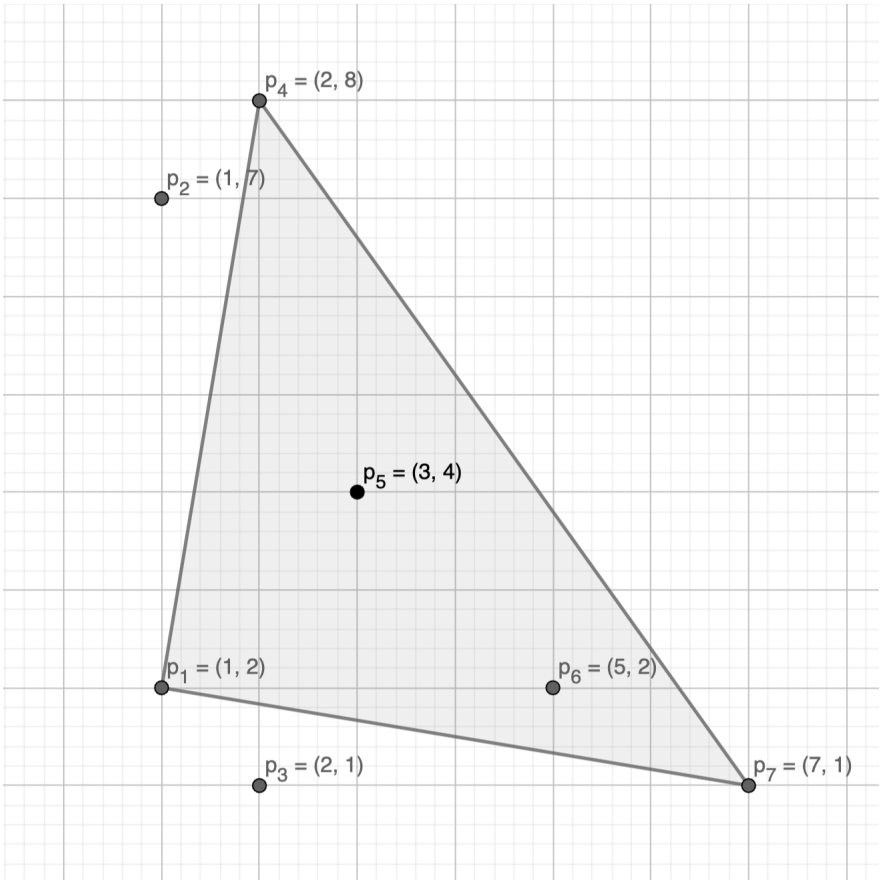


Figure: explanation for second test case.

# Problem D. An Interesting Problem

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

You are given a string $s$. For a string $t$, let $|t|$ be the length of $t$ and $f(t)$ be the number of times $t$ appears in $s$ as a substring*. For example, if $s = \texttt{abababc}$, then $f(\texttt{ab}) = 3$, $f(\texttt{abc}) = 1$ and $f(\texttt{aba}) = 2$.

We define string $s$ as *interesting* if it satisfies a special property: for every non-empty substring $t$ of the string, the length of $t$ must be divisible by the number of occurrences of $t$ in the original string $s$. Formally, $|t| \bmod f(t) = 0$ for all non-empty substrings $t$ of $s$.

Your task is to determine whether the given string $s$ is *interesting*.

* A string $t$ is a substring of a string $s$ if $t$ can be obtained from $s$ by deleting several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

## Input

The first line contains an integer $t$ ($1 \le t \le 10^5$) — the number of test cases.

The first line of each test case contains an integer $n$ ($1 \le n \le 3 \cdot 10^5$) — the length of the string.

The second line contains a string $s$ of length $n$, consisting of lowercase English letters.

It is guaranteed that the sum of $n$ over all test cases does not exceed $3 \cdot 10^6$.

## Output

For each test case, print `YES` if the string is *interesting*, and `NO` otherwise.

## Example

| standard input | standard output |
|---|---|
| 2<br>2<br>ab<br>7<br>abababc | YES<br>NO |

## Note

In the first test case, the string is *interesting* because:

- For $t = \texttt{a}$, $|t| = 1$ and $f(t) = 1$, so $|t| \bmod f(t) = 1 \bmod 1 = 0$.
- For $t = \texttt{b}$, $|t| = 1$ and $f(t) = 1$, so $|t| \bmod f(t) = 1 \bmod 1 = 0$.
- For $t = \texttt{ab}$, $|t| = 2$ and $f(t) = 1$, so $|t| \bmod f(t) = 2 \bmod 1 = 0$.

As the condition is satisfied for all non-empty substrings, the string is *interesting*.

In the second test case, the string is not *interesting*, because, for example, for the substring $t = \texttt{bab}$, $|t| = 3$ and $f(t) = 2$, so $|t| \bmod f(t) = 3 \bmod 2 = 1 \neq 0$.

# Problem E. Polynomial K Paths

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 3 seconds |
| Memory limit: | 256 megabytes |

You're given a directed graph $G = (V, E)$ with $n$ vertices and $m$ edges, where $V$ is the set of vertices and $E$ is the set of edges. The vertices are numbered from 1 to $n$. You're also given the coefficients $c_0, c_1, \ldots, c_d$ of a $d$ degree polynomial $f$ described by $f(X) = \sum_{i=0}^{d} c_i X^i$.

Your task is to find $k$ (not necessarily distinct) simple paths from vertex 1 to vertex $n$ such that the following cost is minimized:

- Suppose the edge $e$ is used $X_e$ times across the $k$ paths. Then the cost incurred is $\sum_{e \in E} X_e \cdot f(X_e)$.

Output the minimum possible cost if the paths are chosen optimally. It is guaranteed that there exists at least one path from 1 to $n$.

## Input

The first line contains a single integer $t$ ($1 \le t \le 20$) — the number of test cases.

The first line of each test case contains four integers $n$ ($1 \le n \le 100$), $m$ ($0 \le m \le 200$), $k$ ($1 \le k \le 50$), and $d$ ($1 \le d \le 6$).

The next line contains $d + 1$ non-negative integers $c_0, c_1, \ldots, c_d$ ($0 \le c_i \le 50$).

The following $m$ lines contain two integers each: $u$ and $v$ ($1 \le u, v \le n$, $u \ne v$), denoting there is a directed edge from vertex $u$ to vertex $v$.

It is guaranteed that there are no self-loops and multiple edges in the graph and there is at least one path from vertex 1 to vertex $n$.

## Output

For each test case, print the minimum cost if the $k$ paths are chosen optimally.

## Example

| standard input | standard output |
|---|---|
| 1<br>6 7 4 3<br>0 1 0 2<br>1 2<br>1 3<br>1 4<br>1 5<br>2 6<br>3 6<br>4 6 | 84 |

## Note

Here, the optimal solution is to choose the following 4 paths: $1 \to 2 \to 6$, $1 \to 3 \to 6$, $1 \to 4 \to 6$, and $1 \to 4 \to 6$. Edges $(1, 2), (1, 3), (2, 6), (3, 6)$ are used 1 time each, contributing $1 \cdot f(1) + 1 \cdot f(1) + 1 \cdot f(1) + 1 \cdot f(1) = 12$ to the total cost. Edges $(1, 4)$ and $(4, 6)$ are used 2 times each, contributing $2 \cdot f(2) + 2 \cdot f(2) = 72$ to the total cost. So the total cost is $12 + 72 = 84$. It can be proven that you can not achieve a lower cost than 84.

# Problem F. Distinct of Distincts

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

There is a grid with $n$ rows and $m$ columns. Consider the following process:

- For each row $i$ $(1 \le i \le n)$, count the number of distinct integers in that row. Let this count be $r_i$.
- For each column $j$ $(1 \le j \le m)$, count the number of distinct integers in that column. Let this count be $c_j$.
- Form a set $S = \{r_1, r_2, \ldots, r_n, c_1, c_2, \ldots, c_m\}$ containing all these $n + m$ counts.

Your task is to fill the grid with integers from 1 to $10^9$ in such a way that the number of distinct elements in the set $S$ is maximized.

## Input

The first line contains a single integer $t$ $(1 \le t \le 10^4)$ — the number of test cases.

The only line of each test case contains two integers $n$ and $m$ $(1 \le n \cdot m \le 3 \cdot 10^5)$ — the number of rows and columns of the grid.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $3 \cdot 10^5$.

## Output

For each test case, the first line should contain a single integer — the maximum possible number of distinct elements in the set $S$.

Then, output $n$ lines, each containing $m$ space-separated integers. The $j$-th integer in the $i$-th line should be the element in the $i$-th row and $j$-th column of the grid. All integers in the grid must be from 1 to $10^9$.

If there are multiple possible grids, output any of them.

## Example

| standard input | standard output |
|---|---|
| 2<br>2 2<br>1 4 | 2<br>2 5<br>5 5<br>2<br>6 2 3 6 |

## Note

In the first test case, for the output grid:

- The first row has elements $\{2, 5\}$, so $r_1 = 2$ as there are 2 distinct elements.
- The second row has elements $\{5, 5\}$, so $r_2 = 1$.
- The first column has elements $\{2, 5\}$, so $c_1 = 2$.
- The second column has elements $\{5, 5\}$, so $c_2 = 1$.

So, the set $S = \{r_1, r_2, c_1, c_2\} = \{2, 1, 2, 1\}$ has 2 distinct elements. It can be shown that no other grid has more than 2 distinct elements in $S$.

In the second test case, for the output grid:

- The first row has elements $\{6, 2, 3, 6\}$, so $r_1 = 3$.
- The first column has elements $\{6\}$, so $c_1 = 1$.
- The second column has elements $\{2\}$, so $c_2 = 1$.
- The third column has elements $\{3\}$, so $c_3 = 1$.
- The fourth column has elements $\{6\}$, so $c_4 = 1$.

So, the set $S = \{r_1, c_1, c_2, c_3, c_4\} = \{3, 1, 1, 1, 1\}$ has 2 distinct elements and this is the maximum possible.

# Problem G. To Infinity and Beyond

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

You are given two arrays $a$ and $b$ of length $n$, satisfying $a_1 = -1$, $b_1 = -1$, $1 \le a_i < i$, $1 \le b_i < i$ (for all $2 \le i \le n$).

For any integer $x \ge 1$, let $\text{count}_a(x)$ be the number of integer arrays $A$ of length $n$ where $1 \le A_i \le x$ for all $i$, and $A_i \le A_{a_i}$ is satisfied for all $i > 1$.

Similarly, let $\text{count}_b(x)$ be the number of integer arrays $B$ of length $n$ where $1 \le B_i \le x$ for all $i$, and $B_i \le B_{b_i}$ is satisfied for all $i > 1$.

It can be shown that, as $x$ approaches infinity, the ratio $\frac{\text{count}_a(x)}{\text{count}_b(x)}$ approaches a rational number $r$. That is,

$$r = \lim_{x \to \infty} \frac{\text{count}_a(x)}{\text{count}_b(x)}$$

exists and is a rational number. So $r$ can be represented as $\frac{p}{q}$ where $p$ and $q$ are coprime integers. Compute $p \cdot q^{-1} \mod 998\,244\,353$, where $q^{-1}$ denotes the multiplicative inverse of $q$ modulo $998\,244\,353$. It is guaranteed that $q^{-1}$ exists modulo $998\,244\,353$ under the given constraints.

## Input

The first line contains an integer $t$ ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains a single integer $n$ ($2 \le n \le 3 \cdot 10^5$).

The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($a_1 = -1$, and $1 \le a_i < i$ for all $i > 1$).

The third line contains $n$ integers $b_1, b_2, \ldots, b_n$ ($b_1 = -1$, and $1 \le b_i < i$ for all $i > 1$).

It is guaranteed that the sum of $n$ over all test cases does not exceed $3 \cdot 10^5$.

## Output

For each test case, output an integer — the answer modulo $998\,244\,353$.

## Example

| standard input | standard output |
|---|---|
| 2<br>3<br>-1 1 1<br>-1 1 2<br>4<br>-1 1 1 2<br>-1 1 2 1 | 2<br>1 |

## Note

Consider the first test case. For array $a$, the constraints are $A_2 \le A_1$ and $A_3 \le A_1$. For each value of $A_1 = k$ (where $1 \le k \le x$), both $A_2$ and $A_3$ can be any value from 1 to $k$, giving $k^2$ possibilities for each $k$. Summing over all possible values of $k$ gives $\text{count}_a(x) = \frac{x(x+1)(2x+1)}{6}$.

For array $b$, the constraints are $B_2 \le B_1$ and $B_3 \le B_2$, forming a descending chain. This gives $\text{count}_b(x) = \frac{x(x+1)(x+2)}{6}$.

As $x$ tends to infinity, their ratio $\frac{\text{count}_a(x)}{\text{count}_b(x)} = \frac{2x+1}{x+2}$ approaches 2.

# Problem H. Litmus Test

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

You are given a list of $n$ aqueous chemicals, each known to be either an acid, a base, or a salt. You have to prove the type of each chemical using litmus papers.

There are two types of litmus papers:

- Red litmus paper: turns blue when exposed to a base; remains red in acid or salt.
- Blue litmus paper: turns red when exposed to an acid; remains blue in base or salt.

To prove a chemical's type, you must test it in a way that rules out all other possibilities. In other words, the result must be enough to uniquely confirm that the chemical is acid, base, or salt.

You may reuse a single litmus paper multiple times (**even on the same chemical**). When a litmus paper changes color (e.g., red to blue), you can continue using it in its new color state until it changes color again. For example, if a red litmus paper turns blue after testing a base, you can use this blue paper to test other chemicals until it turns red again, and then, similarly, continue reusing it if needed. You may safely assume that the effect of contamination from earlier tests on the same paper is negligible.

Your task is to compute the minimum **total** number of litmus papers required to verify all the given chemicals. It doesn't matter how many times you use each individual litmus paper; you only want to minimize the total count of litmus papers used.

## Input

The first line contains an integer $t$ ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains an integer $n$ ($1 \le n \le 10^5$) — the number of chemicals.

The second line of each test case contains a string $s$ of length $n$ consisting only of the characters `A`, `B`, and `S`, where the $i$-th character of $s$ represents the type of the $i$-th chemical — `A` represents an acid, `B` represents a base, and `S` represents a salt.

It is guaranteed that the sum of $n$ over all test cases does not exceed $3 \cdot 10^5$.

## Output

For each test case, output a single integer — the minimum total number of litmus papers required.

## Example

| standard input | standard output |
|---|---|
| 6<br>5<br>AAAAA<br>2<br>AB<br>1<br>S<br>4<br>SSBB<br>4<br>BASS<br>5<br>AAAAB | 5<br>1<br>2<br>2<br>1<br>1 |

## Note

In the first test case, we use one fresh blue litmus paper for each chemical. Each paper turns red, confirming that all chemicals are acid.

In the second test case, we use a red litmus paper on the second chemical. It turns blue, confirming the chemical is base. We then use the same litmus paper, now blue, on the first chemical and it turns red, confirming the chemical is acid.

In the third test case, we use a red and a blue litmus paper on the chemical. Neither changes color, confirming that the chemical is salt.

In the fourth test case, we use two red litmus papers on the first two chemicals. Both papers remain red, indicating the chemicals are either acid or salt. We then use the same papers on the next two chemicals. Both turn blue, confirming that those chemicals are base. Finally, we reuse the now-blue litmus papers on the first two chemicals. Both remain blue, confirming that the first two chemicals are salt.

# Problem I. XOR This OR That

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

You are given an array $a$ of $n$ integers. You need to split it into two non-empty subsequences* $s_1$ and $s_2$, such that every element from the array belongs to exactly one subsequence.

Let $x$ be the bitwise XOR of all elements in $s_1$, and $y$ be the bitwise OR of all elements in $s_2$. Your task is to minimize the value of $x \times y$.

* A subsequence of an array is a sequence that can be obtained by deleting some (possibly zero) elements without changing the order of the remaining elements.

## Input

The first line contains a single integer $t$ ($1 \le t \le 10$) — the number of test cases.

The first line of each test case contains a single integer $n$ ($2 \le n \le 10^5$) — the number of elements in the array.

The second line of each test case contains $n$ integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — the elements of the array.

## Output

For each test case, output the answer in a single line.

## Example

| standard input | standard output |
|---|---|
| 3<br>3<br>12 23 11<br>5<br>8 17 9 12 19<br>2<br>3 2 | 161<br>26<br>6 |

## Note

In the first test case, choose $s_1 = \{12, 11\}$ and $s_2 = \{23\}$. The bitwise XOR of $s_1$ is $12 \oplus 11 = 7$. The bitwise OR of $s_2$ is 23. So, the value is $7 \times 23 = 161$.

In the second test case, choose $s_1 = \{17, 19\}$ and $s_2 = \{8, 9, 12\}$. The bitwise XOR of $s_1$ is $17 \oplus 19 = 2$. The bitwise OR of $s_2$ is 13. So, the value is $2 \times 13 = 26$.

In the third test case, choose $s_1 = \{3\}$ and $s_2 = \{2\}$. The bitwise XOR of $s_1$ is 3. The bitwise OR of $s_2$ is 2. So, the value is $3 \times 2 = 6$.

# Problem J. LCM Factorization

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

For an integer $x$, define $f(x)$ as the sum of all distinct prime factors of $x$. For example, $f(60) = f(2^2 \cdot 3 \cdot 5) = 2 + 3 + 5 = 10$. In particular, $f(1) = 0$.

You are given a sequence of $n$ integers $a_1, a_2, \ldots, a_n$ and an integer $k$. Find the sum, modulo 998 244 353, of $f(\text{LCM}(a_{i_1}, a_{i_2}, \ldots, a_{i_k}))$ over all possible subsequences $1 \le i_1 < i_2 < \cdots < i_k \le n$ of length $k$.

Here, $\text{LCM}(a_{i_1}, a_{i_2}, \ldots, a_{i_k})$ denotes the least common multiple of the integers $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$. For example, $\text{LCM}(10, 20, 30) = 60$.

## Input

The first line contains a single integer $t$ ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains two integers $n$ and $k$ ($1 \le k \le n \le 3 \cdot 10^5$) — the length of the sequence and the length of subsequences, respectively.

The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the sequence.

It is guaranteed that the sum of $n$ over all test cases does not exceed $3 \cdot 10^5$.

## Output

For each test case, print a single integer — the answer modulo 998 244 353.

## Example

| standard input | standard output |
|---|---|
| 3<br>4 2<br>2 1 3 4<br>3 3<br>2 2 2<br>1 1<br>1 | 19<br>2<br>0 |

## Note

In the first test case,

- $f(\text{LCM}(a_1, a_2)) = f(\text{LCM}(2, 1)) = f(2) = 2$,
- $f(\text{LCM}(a_1, a_3)) = f(\text{LCM}(2, 3)) = f(6) = 2 + 3 = 5$,
- $f(\text{LCM}(a_1, a_4)) = f(\text{LCM}(2, 4)) = f(4) = 2$,
- $f(\text{LCM}(a_2, a_3)) = f(\text{LCM}(1, 3)) = f(3) = 3$,
- $f(\text{LCM}(a_2, a_4)) = f(\text{LCM}(1, 4)) = f(4) = 2$,
- $f(\text{LCM}(a_3, a_4)) = f(\text{LCM}(3, 4)) = f(12) = 2 + 3 = 5$.

The sum of these values is $2 + 5 + 2 + 3 + 2 + 5 = 19$.