ICPC Amritapuri Regionals 2025-26

B - Fair Removals

As part of the National Quantum Mission, researchers in Bengaluru are developing India's first indigenous superconducting quantum processor. During the initialization phase, qubits are represented as binary strings that must be purified to reach a ground state. The hardware allows for a specific decoherence operation where any three-qubit subsequence containing both '0' and '1' states can be collapsed and removed from the chain. To achieve maximum quantum coherence, the researchers need to reduce these strings to their shortest possible length. By calculating the minimum final length, you are helping to optimize the error-correction protocols for the next generation of Indian supercomputers.

You are given a **binary** string s of length n.
In one operation, you can choose a subsequence[†] of s of length 3 such that it contains at least one occurrence of **both** the characters `0` and `1`, and delete it from the string. You cannot perform the operation if the length of the string is smaller than 3. The remaining characters are concatenated without changing their relative order.
Find the **minimum** possible final length of the string s after performing this operation any number of times.

[†] A subsequence is a sequence that can be derived from the given sequence by deleting zero or more elements without changing the order of the remaining elements. For example, `100` and `11` are subsequences of `1001`, but `010` and `111` are not.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains a single integer n — the length of the binary string.
  - The second line contains the binary string s.

Output Format

- For each test case, output a single integer — the minimum possible final length of the string.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq n \leq 2 \cdot 10^5$
- $s_i \in \{\texttt{0}, \texttt{1}\}$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
4
2
01
3
010
```

```
3
000
6
001000
```

Sample Output 1

```
2
0
3
3
```

Explanation

- **Test case** 1**:** The string's length is less than three, so no operation can be performed. The answer is thus the length of the string, i.e. 2.
- **Test case** 2**:** We can choose the entire string in one operation, since it has length 3 and contains both 0's and 1's. This deletes the entire string, so the answer is 0.
- **Test case** 3**:** All the characters of s are equal, so no operation can be performed.

C - Path Pair Maximization

As part of the Gati Shakti master plan, the Ministry of Railways is modernizing a regional cargo distribution network. The network is modeled as a tree with n logistics hubs, where each edge represents an indigenous freight corridor. To enhance the monitoring of high-value "Make in India" exports, the government has decided to upgrade exactly two of these corridors with advanced IoT-enabled tracking sensors. These two upgraded corridors will be designated as "Black" routes, while the remaining corridors remain standard "White" routes. A pair of hubs (u, v) is considered "security-monitored" if the unique freight path between them passes through at least one tracking-enabled Black corridor. To justify the investment in this new technology, the planners want to strategically choose which two corridors to upgrade so that the total number of security-monitored hub pairs is as high as possible. Your goal is to find this maximum number of monitored pairs for a given network layout.

You are given a tree with n vertices numbered from 1 to n.
You must color **exactly two edges** of the tree black, and the rest of the edges will be colored white.
Find the **maximum** possible number of pairs of vertices (u, v) such that $1 \leq u < v \leq n$ and the shortest path from u to v contains at least one black edge.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains a single integer n — the number of vertices in the tree.
  - The next $n-1$ lines describe the edges. The i-th of these $n-1$ lines contains two space-separated integers u_i and v_i, denoting an edge between u_i and v_i in the tree.

Output Format

- For each test case, output a single integer — the maximum possible number of pairs of vertices (u, v) such that $1 \leq u < v \leq n$ and the shortest path from u to v contains at least one black edge.

Constraints

- $1 \leq t \leq 10^4$
- $3 \leq n \leq 2 \cdot 10^5$
- $1 \leq u_i, v_i \leq n$
- The input edges describe a tree on the vertex set $\{1, 2, \ldots, n\}$.
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
2
3
1 2
2 3
```

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

Sample Output 1

```
3
5
```

Explanation

- **Test case 1:** There are only two edges, so we must color both black. Every pair of vertices will now have a black edge on the path between them, so the answer is 3.
- **Test case 2:** One solution is to color the edges $(1, 2)$ and $(2, 4)$ black. Then, every path except the one from 2 to 3 will have a black edge on it, making the answer $6 - 1 = 5$.

D - Point Mirror

Under the Make in India initiative, a team of engineers is calibrating an automated assembly line. The system consists of n high-precision robotic grippers positioned along a linear track. Initially, each gripper is located at a specific coordinate. To fine-tune the assembly process, the control software can execute two types of maneuvers: it can swap the positions of any two grippers to reorganize the sequence, or it can mirror one gripper across another to reach new calibration points. The engineering team has a target configuration where each gripper has a designated final position. Your task is to determine if the robotic system can reach this exact target state using any number of these two fundamental maneuvers.

There are n points on a number line. Initially, the i-th point is at coordinate a_i.
You can perform the following operations any number of times, in any order:

- **Operation** 1: Choose two distinct points i, j and swap them.
- **Operation** 2: Choose two distinct points i, j and mirror point i across point j. Formally, set $a_i := 2a_j - a_i$

Is it possible to have the i-th point be at b_i after all operations for all i ($1 \leq i \leq n$)?

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains a single integer n — the number of points.
  - The next n lines describe the points. The i-th of these n lines contains two space-separated integers a_i and b_i, denoting the initial coordinate and desired final coordinate of point i.

Output Format

- For each test case, output the answer — `YES` if it is possible to reach the desired final state after all operations, and `NO` otherwise.
- You can output the answer in any case (upper or lower). For example, the strings `Yes`, `yes`, `yEs`, and `YES` will all be recognized as valid responses for cases where reaching the final configuration is possible.

Constraints

- $1 \leq t \leq 10^4$
- $1 \leq n \leq 2 \cdot 10^5$
- $-10^9 \leq a_i, b_i \leq 10^9$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
6
1
1 1
1
0 2
2
0 3
1 2
2
1 5
2 7
3
-3 5
9 13
7 5
3
-3 11
9 25
7 9
```

Sample Output 1

```
YES
NO
YES
NO
NO
YES
```

Explanation

- **Test case** 1**:** The only point is already at the desired coordinate.
- **Test case** 3**:** The following sequence of operations achieves the desired coordinates.
  - The initial coordinates are $[0, 1]$.
  - Mirror point 1 across point 2. The coordinates are now $[2, 1]$.
  - Swap point 1 with point 2. The coordinates are now $[1, 2]$.
  - Mirror point 1 across point 2. The coordinates are now $[3, 2]$. These are the desired coordinates.
- **Test case** 6**:** The following sequence of operations achieves the desired coordinates.
  - The initial coordinates are $[-3, 9, 7]$.
  - Swap point 2 with point 3. The coordinates are now $[-3, 7, 9]$.
  - Mirror point 2 across point 3. The coordinates are now $[-3, 11, 9]$.
  - Mirror point 1 across point 2. The coordinates are now $[25, 11, 9]$.
  - Swap point 1 with point 2. The coordinates are now $[11, 25, 9]$. These are the desired coordinates.

- For test cases $2, 4, 5$ it can be proven that it is impossible to have the points at the desired coordinates.

E - Preventative Measures

As part of the Smart Cities Mission, engineers are designing a highly efficient municipal utility grid. The network is currently structured as a tree with $2n$ connection hubs. To maintain load balance, a specialist named Bob is tasked with pairing these hubs into n distinct links such that the total length of all connections equals exactly k kilometers. However, to ensure the grid is resilient against specific synchronization failures, a lead architect named Alice must modify the network layout. She is authorized to decommission one existing transmission line and replace it with a new one, ensuring the network remains a single connected tree. Her objective is to reconfigure the grid so that it becomes mathematically impossible for Bob to achieve a total connection length of exactly k, thereby forcing a more robust distribution of the network load.

Alice and Bob have a tree T consisting of $2n$ nodes numbered from 1 to $2n$.
Bob has an integer k, and wants to find n pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ such that each integer from 1 to $2n$ appears in exactly one pair and the sum of distances $\sum_{i=1}^{n} \text{dist}(x_i, y_i) = k$, where $\text{dist}(x_i, y_i)$ denotes the distance between nodes x_i and y_i in the tree, i.e. the number of edges on the (unique) path between x_i and y_i.
Alice doesn't want Bob to succeed. She can perform the following operation:

- Remove an edge (u, v) from T.
- Add a new edge (a, b) such that T remains a tree.

Note that it is allowed to choose the same edge to both delete and insert, which will just result in the original tree.
Help Alice find an edge (u, v) that should be deleted and an edge (a, b) that should be added. If there are multiple solutions, you may find any of them.
It can be proven that under the constraints of this problem, it is always possible to delete an edge and add an edge such that there do not exist n pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where each integer from 1 to $2n$ appears in exactly one pair and the sum of distances equals k.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains two space-separated integers n and k.
  - The next $2n - 1$ lines describe the edges. The i-th of these $2n - 1$ lines contains two space-separated integers u_i and v_i, denoting an edge between u_i and v_i in the tree.

Output Format

- For each test case, output two lines:
  - The first line should contain two integers u and v — meaning that edge (u, v) is to be deleted.
  - The second line should contain two integers a and b — meaning that edge (a, b) is to be added.
- If there are multiple solutions, print any of them.

Constraints

- $1 \leq t \leq 10^4$
- $2 \leq n \leq 2 \cdot 10^5$
- $1 \leq k \leq 10^{12}$
- The input edges describe a tree on the vertex set $\{1, 2, \ldots, 2n\}$.
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
2
2 20
1 2
2 3
3 4
2 4
1 2
2 3
3 4
```

Sample Output 1

```
1 2
1 2
1 2
1 3
```

Explanation

- **Test case 1:** The initial tree already has no way for Bob to obtain a score of $k = 20$, so Alice can simply remove and insert any existing edge.
- **Test case 2:** The initial tree has a way for Bob to obtain a score of $k = 4$, for example by choosing pairs $(1, 4)$ and $(2, 3)$. Alice deletes edge $(1, 2)$ and inserts $(1, 3)$. In this new tree, Bob cannot obtain a score of 4.

F - Red Green Game

As India establishes its first major semiconductor fabrication plant, engineers are fine-tuning the automated logic gates used in 28nm chip production. Two control algorithms, Alice and Bob, take turns processing a stream of logic signals represented by Red (Low-energy) and Green (High-energy) pulses. Both algorithms begin with an internal state of "Low" (Red). Processing a Green pulse flips the algorithm's state between Low and High, while a Red pulse keeps the current state locked. The efficiency of the final circuit depends on how many algorithms end in the "High" state. Alice's logic is designed to maximize this high-energy output, while Bob's error-correction logic seeks to minimize it to prevent overheating.

Alice and Bob play a game. They initially have x red balls and y green balls.
Alice and Bob each have a color that is initially *red*. They take turns alternately, with Alice going first. On each turn, a player does the following:

- Pick a ball and remove it.
- If the picked ball is *red*, the player's color remains unchanged.
- If the picked ball is *green*, the player's color flips: if it was *red*, it becomes *green*, and if it was *green*, it becomes *red*.

The game ends when there are no balls left.
The score of the game is the total number of *green* colors at the end (i.e., count how many players have *green* at the end). Alice wants to maximize the score, while Bob wants to minimize it.
Determine the score if both players play optimally.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of a single line of input, containing two space-separated integers x and y — the number of red and green balls, respectively.

Output Format

- For each test case, output a single integer — the score of the game if both players play optimally.

Constraints

- $1 \leq t \leq 10^5$
- $0 \leq x, y \leq 10^9$
- $x + y \geq 1$

Sample Input 1

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

Sample Output 1

```
0
1
1
2
```

Explanation

- **Test case** 1**:** Alice's only move is to choose a red ball, which doesn't change her color. The game then ends with both players having red. The answer is 0.
- **Test case** 2**:** Alice's only move is to choose a green ball, which changes her color from red to green. The game then ends with Alice having green and Bob having red. The answer is 1, since one player has green.

G - Score Queries

As part of the National Green Hydrogen Mission, a distribution hub manages n hydrogen storage tanks, where a_i represents the current pressure level of the i-th tank. To safely transport the hydrogen to industrial units, engineers use specialized extraction valves. In a single extraction cycle, a technician can select a group of tanks to depressurize by 1 unit each, provided that every selected tank currently has a distinct pressure level to prevent resonance in the manifold system. The efficiency of a configuration is measured by its score: the minimum number of extraction cycles required to completely empty all tanks. As the hub receives real-time supply updates, the pressure in tank p is frequently adjusted to a new value x. Your task is to calculate the efficiency score of the entire array after each update to ensure the facility maintains optimal throughput for India's burgeoning green energy grid.

Suppose you are given an array b consisting of m positive integers.
You can modify b by performing the following operation:

- Select a subsequence s of indices from $\{1, 2, \ldots, m\}$ such that $b_i \neq b_j$ for all pairs (i, j) where $i < j$ and both i and j are in s (i.e., all selected indices must have distinct values in b).
- Subtract 1 from b_i for all i in s.

Define score(b) to be the minimum number of operations needed to make all m elements of b equal to 0.

You are given an array a consisting of n positive integers.
There will be q updates. In each update, you will be given two integers p and x, and you need to update $a_p := x$. After each update, print score(a).
Updates to the array are permanent.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains two space-separated integers n and q — the length of the array and the number of updates to it.
  - The second line contains n space-separated integers $a_1, \ldots, a_n$
  - The next q lines describe the updates. The i-th of these q lines contains two space-separated integers p_i and x_i, denoting a point update of the form $a_{p_i} := x_i$.

Output Format

- For each test case, output q lines. The i-th line should contain a single integer — the score of the array a after the i-th update.

Constraints

- $1 \leq t \leq 10^4$
- $1 \leq n, q \leq 2 \cdot 10^5$
- $1 \leq a_i \leq n$

- $1 \leq p_i \leq n$
- $1 \leq x_i \leq 10^9$
- The sum of n and the sum of q over all test cases each won't exceed $2 \cdot 10^5$.

Sample Input 1

```
1
3 3
1 1 1
3 1
2 2
1 343
```

Sample Output 1

```
3
3
343
```

Explanation

- **Test case 1:** The array is initially $[1, 1, 1]$.
  - The first update functionally doesn't change the array, so it remains $[1, 1, 1]$. We can't pick duplicate elements, so we're forced to perform one operation on each index. The score of the array is 3.
  - The second update sets $a_2 = 2$, so now the array is $[1, 2, 1]$. The score of the array remains 3, since we can do the following:
    - Move 1: pick the subsequence $[2, 1]$ and subtract 1 from each element. Now, $a = [1, 1, 0]$.
    - Moves 2 and 3: pick $[1]$.
  - The third update sets $a_1 = 343$, so that the array is now $[343, 2, 1]$. It can be shown that the score of this array is 343.

H - Segment Elimination

As part of the Drone Shakti initiative, the Directorate General of Civil Aviation (DGCA) is managing a high-density "Green Zone" for autonomous delivery drones. On a specific aerial corridor, n drones are scheduled to pass through, each occupying a specific time segment $[l_i, r_i]$. To prevent mid-air collisions, two drones are considered to be in conflict if their time segments intersect. To streamline the traffic, a controller must clear the corridor by canceling at least $n-2$ drone flights. According to the safety protocol, she can clear a set of drones in a single operation only if no two drones in that set have intersecting time segments. Alisa needs to find the minimum number of operations, m, required to remove the necessary flights; as well as the number of subsets of at least $n-2$ drone flights that can be cleared in m moves.

There are n segments on a number axis. The i-th segment is $[l_i, r_i]$. Two segments are considered intersecting if there exists a point that is inside both segments (i.e. their intersection is non-empty). Now, Alisa wants to remove at least $n-2$ segments by doing the following operation multiple times:

- Select a non-empty set of segments S such that for any two different segments in S they do not intersect, then remove all segments in the set from the number axis.

Please help Alisa find out the minimum number of operations needed to remove at least $n-2$ segments from the number axis. Let m denote this minimum number of operations. You also need to find the number of subsets s $(0 \leq |s| \leq 2)$ of intervals so that excluding s, all the remaining intervals can be removed in m operations.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains a single integer n — the number of segments.
  - The next n lines describe the segments. The i-th of these n lines contains two space-separated integers l_i and r_i, denoting the segment $[l_i, r_i]$.

Output Format

- For each test case, output two space-separated integers: the minimum number of operations needed and the number of subsets s $(0 \leq |s| \leq 2)$ of intervals such that excluding s, all the remaining intervals can be removed in m operations.

Constraints

- $1 \leq t \leq 10^5$
- $2 \leq n \leq 2 \cdot 10^5$
- $1 \leq l_i \leq r_i \leq 2 \cdot n$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
6
4
1 6
1 2
6 8
2 2
2
1 2
3 4
5
2 6
5 9
2 7
6 9
8 9
2
1 2
2 4
3
3 4
3 4
3 4
4
1 2
3 4
5 6
7 8
```

Sample Output 1

```
1 2
0 1
2 5
0 1
1 3
1 11
```

Explanation

- **Test case 1:** The minimum number of operations is 1, because we can for example remove $[1, 2]$ and $[6, 8]$ with one move, thus leaving two segments. There are also two possible subsets of intervals of size ≤ 2 that can remain after a single move:
  - $\{[1, 6], [2, 2]\}$, which will remain after using the aforementioned removal of $[1, 2]$ and $[6, 8]$.
  - $\{[1, 6], [1, 2]\}$, which will remain after removing $[2, 2]$ and $[6, 8]$.

I - Sequence Domination

As part of the National Smart Grid Mission, engineers at the Power Grid Corporation of India are designing an automated load-shedding protocol for n high-capacity industrial zones. Each zone i is assigned two priority coefficients, a_i and b_i, ranging from 1 to m. These coefficients represent the indigenous "Efficiency Ratings" of different equipment manufacturers. The power demand in the grid is modeled by a super decreasing sequence v, where each zone's load is so significant that it outweighs the combined demand of all subsequent zones in the hierarchy. To ensure that the "Type-A" manufacturing sector always contributes more to the national grid stability than "Type-B," the ratings must be chosen such that the weighted contribution of a is greater than or equal to b for every possible super decreasing load profile. Your task is to calculate the total number of valid pairs of rating sequences (a, b) that satisfy this strategic requirement, helping the mission control ensure a stable and prioritized energy distribution across the country.

A sequence $v_1, v_2, \ldots v_n$ is called **super decreasing** if $v_n \geq 0$, and $v_i \geq v_{i+1} + v_{i+2} + \ldots + v_n$ for all $1 \leq i < n$.
Given positive integers n and m, find the number of pairs of integer sequences a and b of length n, such that:

- $1 \leq a_i, b_i \leq m$ for all $1 \leq i \leq n$.
- For all *super decreasing* sequences v of length n, $\sum_{i=1}^{n} a_i v_i \geq \sum_{i=1}^{n} b_i v_i$.

Since the number of pairs of such sequences can be very large, print the number modulo 998 244 353.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists a single line of input, containing two space-separated integers n and m.

Output Format

- For each test case, print the number of pairs of integer sequences of length n that satisfy the required conditions, modulo 998 244 353.

Constraints

- $1 \leq t \leq 100$
- $1 \leq n, m \leq 100$
- The sum of n over all test cases won't exceed 100.

Sample Input 1

```
4
1 1
2 2
2 1
34 43
```

Sample Output 1

```
1
10
1
39531442
```

Explanation

- **Test case 1:** The only possible pair of sequences is $([1], [1])$, and it is valid.
- **Test case 2:** The following are valid pairs of sequences:
  - For $b = [1, 1]$, any choice of a is valid. There are four possible choices.
  - For $b = [1, 2]$, valid choices of a are $[1, 2], [2, 1], [2, 2]$.
  - For $b = [2, 1]$, valid choices of a are $[2, 1], [2, 2]$.
  - For $b = [2, 2]$, the only valid choice of a is $[2, 2]$.

  So, the number of valid pairs of sequences equals $4 + 3 + 2 + 1 = 10$.

J - Varied Subsequences

With the success of the NavIC navigation system, ISRO is working on a new constellation of small satellites launched via the SSLV (Small Satellite Launch Vehicle). To prevent signal collisions, each satellite in a cluster must operate at a distinct relative power level compared to a central ground station frequency x. If the absolute power offsets are not unique, the signals will suffer from destructive interference. During a telemetry scan, the ground station receives an array of power levels. You are tasked with identifying a subsequence of these levels that cannot be made distinct through any reference frequency x, signaling a potential communication bottleneck in the satellite network.

An array b of length m is said to be *varied* if there exists an integer x satisfying the following:

- Let c be an array of length m such that $c_i = |b_i - x|$.
- Then, it must hold that $c_i \neq c_j$ for $1 \leq i < j \leq m$, i.e. all the elements of c are distinct.

In particular, an array of length 1 is always *varied*.
Here, $|y|$ denotes the absolute value of y. For example, $|1| = 1, |-2| = 2, |0| = 0$.

You're given an array a of length n. Your task is to find any non-empty subsequence[†] of a that's **not** *varied*.
If multiple such subsequences exist, you may find any of them. If no such subsequence exists, print -1.
[†] A subsequence is a sequence that can be derived from the given sequence by deleting zero or more elements without changing the order of the remaining elements. For example, $[1, 2, 2]$ and $[3, 2]$ are subsequences of $[1, 2, 3, 2]$, but $[3, 1]$ and $[1, 1, 2]$ are not.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains a single integer n — the length of the array.
  - The second line contains n space-separated integers $a_1, \ldots, a_n$

Output Format

- For each test case,
  - If there does not exist a subsequence of the given that's **not** *varied*, print one line containing the integer -1.
  - Otherwise, print two lines.
    - The first line should contain an integer k ($1 \leq k \leq n$), denoting the size of the subsequence you found.
    - The second line should contain k space-separated integers denoting the elements of the subsequence, in order from left to right.
- If there are multiple possible subsequences that are **not** *varied*, you may print any of them.

Constraints

- $1 \leq t \leq 1000$
- $1 \leq n \leq 2000$
- $1 \leq a_i \leq 10^9$
- The sum of n over all test cases won't exceed 2000.

Sample Input 1

```
2
4
8 9 5 8
2
10 20
```

Sample Output 1

```
3
8 9 8
-1
```

Explanation

- **Test case 1:** $[8, 9, 8]$ is a subsequence of $a = [8, 9, 5, 8]$, and it can be proved that it is not *varied*.
- **Test case 2:** The three subsequences of the array are $[10], [20], [10, 20]$. It can be proved that they're all *varied*, so no solution exists.

K - XOR Product MST

Under the BharatNet initiative, the Department of Telecommunications is deploying an indigenous fiber-optic backbone to bridge the digital divide in rural India. In a newly integrated district, n villages, indexed from 1 to n, are slated for high-speed connectivity. To optimize the deployment of underground cables, engineers have developed a specialized cost model for linking any two villages. The expenditure for laying a connection between village x and village y is determined by the product of their indices and a "signal-interference coefficient", which is calculated as the bitwise XOR-sum of all village indices in the range from x to y. To ensure every village is connected to the national grid while keeping the total infrastructure budget as low as possible, the project planners must determine the total weight of the Minimum Spanning Tree for this network. This calculation is essential for delivering e-governance, digital healthcare, and education to the furthest corners of the country.

You are given an integer n.
Consider a graph G consisting of n nodes numbered $1, 2, 3, \ldots, n$.
For any two nodes x and y where $1 \leq x < y \leq n$, there is an undirected edge between them with weight

$$x \cdot y \cdot (x \oplus (x+1) \oplus \cdots \oplus (y-1) \oplus y)$$

where $\oplus$ denotes the bitwise XOR operation.
Find the sum of the weights of the minimum spanning tree of G.
Since this value might be large, you only need to find it modulo 998 244 353.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of a single line of input, containing one integer n.

Output Format

- For each test case, output a single integer — the sum of the weights of the minimum spanning tree of G, modulo 998 244 353.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq n \leq 10^6$
- There is **no constraint** on the sum of n across all tests.

Sample Input 1

```
4
1
2
3
34343
```

Sample Output 1

```
0
6
6
181162958
```

Explanation

- **Test case 1:** There's a single vertex, the MST has weight 0.
- **Test case 2:** There's a single edge with weight $1 \cdot 2 \cdot (1 \oplus 2) = 2 \cdot 3 = 6$, and the MST consists of this edge.
- **Test case 3:** There are three edges:
  - $(1, 2)$ with weight 6
  - $(1, 3)$ with weight $1 \cdot 3 \cdot (1 \oplus 2 \oplus 3) = 3 \cdot 0 = 0$
  - $(2, 3)$ with weight $2 \cdot 3 \cdot (2 \oplus 3) = 6 \cdot 1 = 6$.

  The MST is obtained by taking the edge $(1, 3)$ along with any one of the other edges, for a cost of $0 + 6 = 6$.

L - X To Y

As part of the National Supercomputing Mission, researchers are testing a high-precision arithmetic unit for an indigenous supercomputer. This unit operates on an extremely large bit-width of 10^{100} bits to handle complex scientific simulations. To verify the efficiency of the data bus, engineers must transform an initial bit-string x into a target configuration y using the processor's two core hardware instructions: an incremental step (adding 1) and a global bit-shuffle (reordering the bit-stream). Because these operations consume clock cycles, the goal is to determine the most efficient path between these two massive numerical states to optimize the supercomputer's throughput.

You are given two integers x and y, in a 10^{100} bits system. For example, $x = 5$ is represented as $\underbrace{00\ldots00}_{(10^{100}-3)\text{ zeros}}$ 101.

In each turn, you can perform one of the following two operations on x:

- Add 1 to x. If x overflows, it becomes 0.
- Reshuffle the bits of x. That is, rearrange the bits of x in any order you want. For example, if x has binary representation $\underbrace{00\ldots00}_{(10^{100}-3)\text{ zeros}}$ 101, you can rearrange it to $\underbrace{00\ldots00}_{(10^{100}-3)\text{ zeros}}$ 110 or 1 $\underbrace{00\ldots00}_{(10^{100}-2)\text{ zeros}}$ 1, etc.

Find the minimum number of turns needed to make x equal to y.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of a single line of input, containing two space-separated integers x and y.

Output Format

- For each test case, output a single integer — the minimum number of turns needed to change x to y.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq x, y \leq 10^{18}$

Sample Input 1

```
4
1 2
3 1
3 4
7 7
```

Sample Output 1

```
1
2
1
0
```

Explanation

- **Test case** 1**:** A single addition is enough to turn $x = 1$ into $y = 2$.
- **Test case** 2**:** An optimal solution: Add 1 to x (now 4), then shuffle bits to get $00\ldots00001 = y$. The answer is 2. It can be shown that one move is not enough.
- **Test case** 4**:** x is already equal to y.