Problem B. Quick Trip

For each stop, we will remember how many people want to get on/off. If a stop corresponds to people of only one type (boarding/exiting), they will not be able to perform their action, and their number will be added to the answer of the corresponding type.

Problem D. Valeria and Pencils

1 Calculating Probability

The probability that all pencils will be capped after T operations is given by the formula:

$$\text{Probability} = \frac{\text{Number of valid sequences}}{\text{Total number of possible sequences}}$$

1.1 Calculating the Denominator

The denominator accounts for all possible sequences of T operations:

- Total available caps: $N \times M$ (all different, even within the same color)
- Number of possible sequences: $P(NM, T) = (NM)(NM-1)\cdots(NM-T+1)$

1.2 Calculating the Numerator

We use dynamic programming to count the valid sequences where all pencils receive caps in T operations.

1.2.1 Defining the DP State

Let $\text{dp}[i][j]$ be the number of sequences where:

- After i operations
- Exactly j different pencils have received caps
- Counted modulo 998244353

1.2.2 DP Transitions

The recurrence relation is:

$$\text{dp}[i+1][j] = \underbrace{\text{dp}[i][j] \times A}_{\text{Existing color}} + \underbrace{\text{dp}[i][j-1] \times B}_{\text{New color}}$$

where:

- $A = jM - i$ (remaining caps of already capped colors)
- $B = (N - j + 1)M$ (remaining caps of uncapped colors)
- For $j = 0$, the second term is considered as 0

1.2.3 Initial Conditions

$$\text{dp}[0][0] = 1$$
$$\text{dp}[0][j] = 0 \quad \text{for } j \geq 1$$

1.2.4 Final Result

The numerator is $\text{dp}[T][N]$, representing sequences where all N pencils are capped after exactly T operations.

2 Modular Arithmetic

Given the numerator $a = \text{dp}[T][N]$ and the denominator $b = P(NM, T)$, we need to find x such that:

$$bx \equiv a \pmod{998244353}$$

2.1 Modular Inverse

Since 998244353 is prime, we use Fermat's Little Theorem:

- Find x', such that $bx' \equiv 1 \pmod{998244353}$
- Then $x = ax' \bmod 998244353$
- Compute $x' \equiv b^{998244353-2} \pmod{998244353}$ using binary exponentiation

3 Complexity Analysis

- Time complexity: $O(NT)$ for computing DP
- Space complexity: $O(NT)$ (can be optimized to $O(N)$)
- Computing the modular inverse: $O(\log MOD)$ per query

This approach efficiently solves the problem within the given constraints ($N \times T \leq 100,000$).

Problem F. Gryffindor is Gathering Alumni

The situation described in the problem can be modeled using an undirected graph, where:

- Vertices represent alumni
- Edges connect pairs of friends

Initially, the vertices corresponding to the alumni who accepted the invitation are marked. In each subsequent iteration, a new set of marked vertices is created consisting of:

- Vertices that had at least one marked neighbor in the previous step

Key Patterns in Changes of Marked Vertices

Consider two adjacent vertices, where at least one is marked:

- A marked vertex may become unmarked in the next iteration
- However, after exactly one additional iteration, it will definitely become marked again
- This vertex will regularly appear in the marked set every two iterations

For two adjacent vertices that are both marked:

- They will remain marked regardless of the number of iterations

Approach to the Solution

The solution uses breadth-first search (BFS) to:

- Track the evolution of marked vertices
- Determine the maximum possible set of marked vertices
- Identify stable configurations where the marked set no longer changes

Problem H. Additional Classes

Use a segment tree, where at each node for each $x, y \in \{0, 1\}$, store the number of ways to choose a subset of columns from the corresponding segment such that when modeling the sum on the numbers obtained from these columns, if there is a carry x in the least significant bit, then there is a carry y in the most significant bit. The segment tree supports efficient updates and queries. The answer to the problem is the answer for the root with $x = y = 0$.

The complexity is $\mathcal{O}(n \log n + q \log n)$.

Problem I. Arithmetic and Arrays

Let's call the beauty of the array the value of the sum of the k-th powers that needs to be maximized.

We solve the problem by considering the values of k:

- If $k = 0$, the value of each power will be 1, except for the value of 0^0. In this case, you need to count the number of zeros in the array and increase each of them by 1.
- If $k = 1$, the beauty of the array is equal to the sum of its elements. In this case, it does not matter which element to increase, so you can apply all x operations to any element.
- If $k = 2$ and we apply at least one operation, then all x operations should be applied to the largest element. This is true because suppose we applied a total of D operations, of which D_i to element A_i, and let the largest element have the value a. Then the final sum will be:

$$\sum_{i=1}^{n}(A_i + D_i)^2 = \sum_{i=1}^{n}(A_i^2 + 2 \cdot A_i \cdot D_i + D_i^2) = \sum_{i=1}^{n}(A_i^2 + 2 \cdot A_i \cdot D_i) + \sum_{i=1}^{n} D_i^2 \leq$$
$$\sum_{i=1}^{n}(A_i^2 + 2 \cdot a \cdot D_i) + \sum_{i=1}^{n} D_i^2 = \sum_{i=1}^{n} A_i^2 + 2 \cdot a \cdot D + D^2$$

This is exactly equal to the beauty of the array if we applied all D operations to the largest element. Thus, if we apply any operations, all of them should be applied to the largest element. However, note that in this case the beauty changes by $2 \cdot a \cdot D + D^2$, which reaches its maximum either at $D = 0$ or at $D = x$. Therefore, if we apply operations, we apply all x operations to the largest element.

Algorithm Recommendations

Note that if $k = 2$, it is sufficient to check whether adding x to the largest element increases its absolute value. If it does, then we apply all x operations to it; otherwise, we do not apply any operations. If the absolute value remains the same, we do not apply any operations, as we aim to minimize the number of operations.