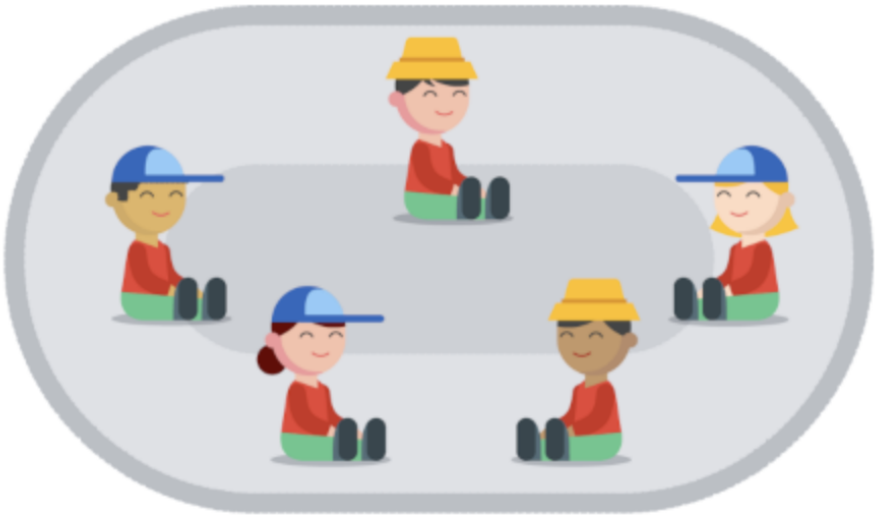
Problem

In the game "Duck, Duck, Goose", all players but one sit on the floor and form a circle. The remaining player walks around the circle calling each player "duck" until they select one sitting player and, while touching their head, call them "goose" instead. At that point, the goose chases the selecting player and our interest in the game fades.

In the new game "Duck, Duck, Geese", the walking player instead chooses a contiguous subset of at least two (but not all) sitting players to be "geese"! Furthermore, each sitting player is wearing a hat. Each hat is one of $\mathbf{C}$ possible colors, numbered 1 through $\mathbf{C}$.



For each color i, the quantity of selected geese wearing a hat of color i must be either 0 or between $\mathbf{A_i}$ and $\mathbf{B_i}$, inclusive.

Can you help count the number of choices that fulfill these requirements? Two choices are considered different if there is some player that is included in one choice but not the other.

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. Each test case starts with a line containing two integers $\mathbf{N}$ and $\mathbf{C}$: the number of sitting players and hat colors, respectively. Then, $\mathbf{C}$ lines follow. The i-th of these lines contains two integers $\mathbf{A_i}$ and $\mathbf{B_i}$, as explained above. The last line of a test case contains $\mathbf{N}$ integers $\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_N}$ representing that the j-th sitting player in clockwise order (starting from an arbitrary one) is wearing a hat of color $\mathbf{P_j}$.

Output

For each test case, output one line containing `Case #`x`:` y, where x is the test case number (starting from 1) and y is the number of sets of at least 2 and at most $\mathbf{N} - 1$ contiguously sitting players that fulfill all the color requirements.

Limits

Time limit: 20 seconds.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.
$2 \leq \mathbf{C} \leq \mathbf{N}$.
$0 \leq \mathbf{A_i} \leq \mathbf{B_i} \leq \mathbf{N}$, for all i.
$1 \leq \mathbf{P_j} \leq \mathbf{C}$, for all j.

Test Set 1 (Visible Verdict)

$3 \leq \mathbf{N} \leq 1000$.

Test Set 2 (Hidden Verdict)

$3 \leq \mathbf{N} \leq 10^5$.

Link to problem:
https://codingcompetitions.withgoogle.com/codejam/round/0000000000879b4/0000000000b45244

Analysis

1) A naive approach that works well for lower values of N is as follows:

    - A backtracking approach performing dfs (depth-first search) on the decision tree this problem represents.

    - The overarching goal is to exhaustively check all possible **contiguous** subsets. To be time-efficient, we must be able to check these subsets quickly — ideally in O(1).

    - We would iterate over all the possible starting indices and for each starting index we would try all the possible allowed sizes.

    - We can keep track and update the count of valid and invalid colors as needed.

    - When the count is equal to see, we can increment the result.

2) The time complexity here is $O(N^2)$ as there are N^2 subsets and checking for validity is in O(1).

3) For larger values of *N*, we can look into building a segment tree, which is a data structure useful in storing information about intervals.

4) Given a starting index, *s*, each color has two possible ranges of values of ending indices for potential contiguous subsets. This includes 0 number of hats of the color in consideration or one that fits in the accepted range.

5) Between [*s* + 1, *s* + **N** - 2] (because we need at most N - 1 geese) forms the range for the possible end indices. This process reduces down to how many contiguous subsets are valid for our current start index.

6) When the starting index moves (to the right), we should update our segment tree when moving the starting index to the right. An important observation is as follows:

moving our starting index to the right by one will only affect the valid end index ranges for the hat color of the child we just moved past.

7) Point 6) yields a potentially clever solution: for every position considered, we can precompute where the next occurrence of that child's hat color is, we can compute the valid ranges for this color.

8) Final time-complexity: O (*N log N)*. This was a very intuitive explanation especially if one is familiar with the tree data structure. Inserts/removals are in O(log N) and since we are considering all the start indices (0 (N)) this results in O (*N log N*). Remember, we can count the number of contiguous subsets for each index in log (N).

Code