Problem 1923: Longest Common Subpath

Problem Information

Difficulty: Hard
Acceptance Rate: 28.95%
Paid Only: No
Tags: Array, Binary Search, Rolling Hash, Suffix Array, Hash Function

Problem Description

There is a country of `n` cities numbered from `0` to `n - 1`. In this country, there is a road connecting **every pair** of cities.

There are `m` friends numbered from `0` to `m - 1` who are traveling through the country. Each one of them will take a path consisting of some cities. Each path is represented by an integer array that contains the visited cities in order. The path may contain a city **more than once** , but the same city will not be listed consecutively.

Given an integer `n` and a 2D integer array `paths` where `paths[i]` is an integer array representing the path of the `ith` friend, return _the length of the**longest common subpath** that is shared by **every** friend's path, or _`0` _if there is no common subpath at all_.

A **subpath** of a path is a contiguous sequence of cities within that path.

Example 1:

Input: n = 5, paths = [[0,1,_2,3_ ,4], [_2,3_ ,4], [4,0,1,_2,3_]] **Output:** 2 **Explanation:** The longest common subpath is [2,3].

Example 2:

Input: n = 3, paths = [[0],[1],[2]] **Output:** 0 **Explanation:** There is no common subpath shared by the three paths.

Example 3:

Input: n = 5, paths = [[_0_ ,1,2,3,4], [4,3,2,1,_0_]] **Output:** 1 **Explanation:** The possible longest common subpaths are [0], [1], [2], [3], and [4]. All have a length of 1.

Constraints:

* `1 <= n <= 105` * `m == paths.length` * `2 <= m <= 105` * `sum(paths[i].length) <= 105` * `0 <= paths[i][j] < n` * The same city is not listed multiple times consecutively in `paths[i]`.

Code Snippets

C++:

```
class Solution {
public:
int longestCommonSubpath(int n, vector<vector<int>>& paths) {

}
};
```

Java:

```
class Solution {
public int longestCommonSubpath(int n, int[][] paths) {

}
}
```

Python3:

```
class Solution:
def longestCommonSubpath(self, n: int, paths: List[List[int]]) -> int:
```