Problem 1868: Product of Two Run-Length Encoded Arrays

Problem Information

Difficulty: Medium
Acceptance Rate: 59.53%
Paid Only: Yes
Tags: Array, Two Pointers

Problem Description

Run-length encoding is a compression algorithm that allows for an integer array `nums` with many segments of **consecutive repeated** numbers to be represented by a (generally smaller) 2D array `encoded`. Each `encoded[i] = [vali, freqi]` describes the `ith` segment of repeated numbers in `nums` where `vali` is the value that is repeated `freqi` times.

* For example, `nums = [1,1,1,2,2,2,2,2]` is represented by the **run-length encoded** array `encoded = [[1,3],[2,5]]`. Another way to read this is "three `1`'s followed by five `2`'s".

The **product** of two run-length encoded arrays `encoded1` and `encoded2` can be calculated using the following steps:

1. **Expand** both `encoded1` and `encoded2` into the full arrays `nums1` and `nums2` respectively. 2. Create a new array `prodNums` of length `nums1.length` and set `prodNums[i] = nums1[i] * nums2[i]`. 3. **Compress** `prodNums` into a run-length encoded array and return it.

You are given two **run-length encoded** arrays `encoded1` and `encoded2` representing full arrays `nums1` and `nums2` respectively. Both `nums1` and `nums2` have the **same length**. Each `encoded1[i] = [vali, freqi]` describes the `ith` segment of `nums1`, and each `encoded2[j] = [valj, freqj]` describes the `jth` segment of `nums2`.

Return _the**product** of _`encoded1` _and_`encoded2`.

Note: Compression should be done such that the run-length encoded array has the **minimum** possible length.

Example 1:

Input: encoded1 = [[1,3],[2,3]], encoded2 = [[6,3],[3,3]] **Output:** [[6,6]] **Explanation:** encoded1 expands to [1,1,1,2,2,2] and encoded2 expands to [6,6,6,3,3,3]. prodNums = [6,6,6,6,6,6], which is compressed into the run-length encoded array [[6,6]].

Example 2:

Input: encoded1 = [[1,3],[2,1],[3,2]], encoded2 = [[2,3],[3,3]] **Output:** [[2,3],[6,1],[9,2]] **Explanation:** encoded1 expands to [1,1,1,2,3,3] and encoded2 expands to [2,2,2,3,3,3]. prodNums = [2,2,2,6,9,9], which is compressed into the run-length encoded array [[2,3],[6,1],[9,2]].

Constraints:

* `1 <= encoded1.length, encoded2.length <= 105` * `encoded1[i].length == 2` * `encoded2[j].length == 2` * `1 <= vali, freqi <= 104` for each `encoded1[i]`. * `1 <= valj, freqj <= 104` for each `encoded2[j]`. * The full arrays that `encoded1` and `encoded2` represent are the same length.

Code Snippets

C++:

```
class Solution {
public:
vector<vector<int>> findRLEArray(vector<vector<int>>& encoded1,
vector<vector<int>>& encoded2) {

}
};
```

Java:

```
class Solution {
public List<List<Integer>> findRLEArray(int[][] encoded1, int[][] encoded2) {

}
}
```

Python3:

```
class Solution:
def findRLEArray(self, encoded1: List[List[int]], encoded2: List[List[int]])
-> List[List[int]]:
```