Description | Editorial | Solutions (376) | Submissions

2670. Find the Distinct Difference Array

Hint

Easy | 190 | 13

Companies

You are given a **0-indexed** array `nums` of length `n`.

The **distinct difference** array of `nums` is an array `diff` of length `n` such that `diff[i]` is equal to the number of distinct elements in the suffix `nums[i + 1, ..., n - 1]` **subtracted from** the number of distinct elements in the prefix `nums[0, ..., i]`.

Return *the **distinct difference** array of* `nums`.

Note that `nums[i, ..., j]` denotes the subarray of `nums` starting at index `i` and ending at index `j` inclusive. Particularly, if `i > j` then `nums[i, ..., j]` denotes an empty subarray.

Example 1:

```
Input: nums = [1,2,3,4,5]
Output: [-3,-1,1,3,5]
Explanation: For index i = 0, there is 1 element in the prefix and
4 distinct elements in the suffix. Thus, diff[0] = 1 - 4 = -3.
For index i = 1, there are 2 distinct elements in the prefix and 3
distinct elements in the suffix. Thus, diff[1] = 2 - 3 = -1.
For index i = 2, there are 3 distinct elements in the prefix and 2
distinct elements in the suffix. Thus, diff[2] = 3 - 2 = 1.
For index i = 3, there are 4 distinct elements in the prefix and 1
distinct element in the suffix. Thus, diff[3] = 4 - 1 = 3.
For index i = 4, there are 5 distinct elements in the prefix and
no elements in the suffix. Thus, diff[4] = 5 - 0 = 5.
```

Example 2:

```
Input: nums = [3,2,3,4,2]
Output: [-2,-1,0,2,3]
Explanation: For index i = 0, there is 1 element in the prefix and
```

Java | Auto

```java
class Solution {
    public int[] distinctDifferenceArray(int[] nums) {
        int a[]=new int[51];
        int len=nums.length;
        int res[]=new int[len];
        for(int i:nums)
        {
            a[i]++;
        }
        int b[]=new int[51];
        for(int i=0;i<len;i++)
        {
            b[nums[i]]++;
            a[nums[i]]--;
            int count=0,count1=0;
            for(int j=1;j<=50;j++)
            {
                if(a[j]>0)
                {
                    count++;
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

nums =
[1,2,3,4,5]

Output

[-3,-1,1,3,5]

Expected

Console | Run | Submit

Description | Editorial | Solutions (504) | Submissions

1601. Maximum Number of Achievable Transfer Requests

Hint

Hard | 1K | 60

Companies

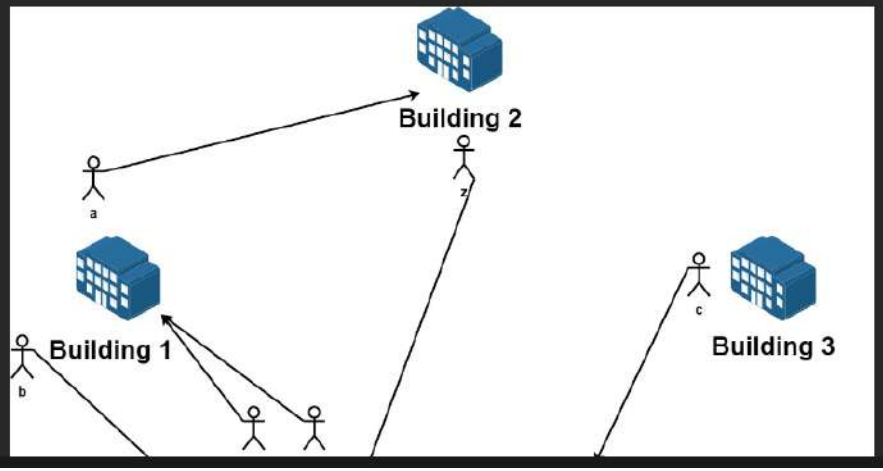
We have `n` buildings numbered from `0` to `n - 1`. Each building has a number of employees. It's transfer season, and some employees want to change the building they reside in.

You are given an array `requests` where `requests[i] = [from_i, to_i]` represents an employee's request to transfer from building `from_i` to building `to_i`.

All buildings are full, so a list of requests is achievable only if for each building, the **net change in employee transfers is zero**. This means the number of employees **leaving** is **equal** to the number of employees **moving in**. For example if `n = 3` and two employees are leaving building `0`, one is leaving building `1`, and one is leaving building `2`, there should be two employees moving to building `0`, one employee moving to building `1`, and one employee moving to building `2`.

Return *the maximum number of achievable requests.*

Example 1:



Java | Auto

```java

class Solution {
    public int maximumRequests(int n, int[][] req) {
        for (int k = req.length; k > 0; k--) {
            List<List<Integer>> combinations = generateCombinations(req.length, k);
            for (List<Integer> c : combinations) {
                int[] degree = new int[n];
                for (int i : c) {
                    degree[req[i][0]]--;
                    degree[req[i][1]]++;
                }
                if (allZeroes(degree)) {
                    return k;
                }
            }
        }
        return 0;
    }

    private List<List<Integer>> generateCombinations(int n, int k) {
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

n =
5

requests =
[[0,1],[1,0],[0,1],[1,2],[2,0],[3,4]]

Output

Console | Run | Submit