## 1570. Dot Product of Two Sparse Vectors

Medium 500 63 Add to List Share

Given two sparse vectors, compute their dot product.

Implement class `SparseVector`:

- `SparseVector(nums)` Initializes the object with the vector `nums`
- `dotProduct(vec)` Compute the dot product between the instance of *SparseVector* and `vec`

A **sparse vector** is a vector that has mostly zero values, you should store the sparse vector **efficiently** and compute the dot product between two *SparseVector*.

**Follow up:** What if only one of the vectors is sparse?

**Example 1:**

```
Input: nums1 = [1,0,0,2,3], nums2 = [0,3,0,4,0]
Output: 8
Explanation: v1 = SparseVector(nums1) , v2 = SparseVector(nums2)
v1.dotProduct(v2) = 1*0 + 0*3 + 0*0 + 2*4 + 3*0 = 8
```

**Example 2:**

```
Input: nums1 = [0,1,0,0,0], nums2 = [0,0,0,0,2]
Output: 0
Explanation: v1 = SparseVector(nums1) , v2 = SparseVector(nums2)
v1.dotProduct(v2) = 0*0 + 1*0 + 0*0 + 0*0 + 0*2 = 0
```

**Example 3:**

```
Input: nums1 = [0,1,0,0,2,0,0], nums2 = [1,0,0,0,3,0,4]
Output: 6
```

**Constraints:**

- `n == nums1.length == nums2.length`
- `1 <= n <= 10^5`
- `0 <= nums1[i], nums2[i] <= 100`

Accepted 87,993 | Submissions 96,965

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook | 84 Google | 2 Goldman Sachs | 2

Related Topics

Array Hash Table Two Pointers Design

Hide Hint 1

Because the vector is sparse, use a data structure that stores the index and value where the element is nonzero.

Java Autocomplete

```java
class SparseVector {

    List<int[]> pairs;

    SparseVector(int[] nums) {
        pairs = new ArrayList<>();
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] != 0) {
                pairs.add(new int[]{i, nums[i]});
            }
        }
    }

    // Return the dotProduct of two sparse vectors
    public int dotProduct(SparseVector vec) {
        int result = 0, p = 0, q = 0;
        while (p < pairs.size() && q < vec.pairs.size()) {
            if (pairs.get(p)[0] == vec.pairs.get(q)[0]) {
                result += pairs.get(p)[1] * vec.pairs.get(q)[1];
                p++;
                q++;
            }
            else if (pairs.get(p)[0] > vec.pairs.get(q)[0]) {
                q++;
            }
            else {
                p++;
            }
        }
        return result;
    }
}
```