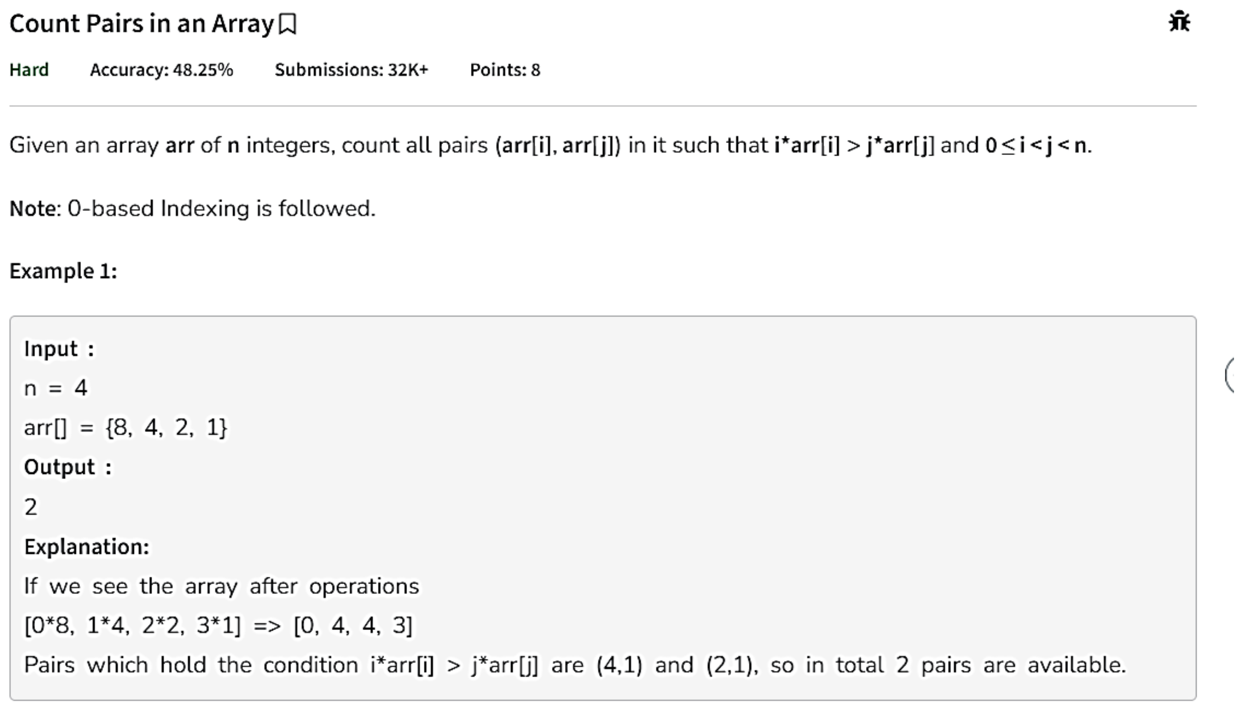
PROBLEM



Count Pairs in an Array

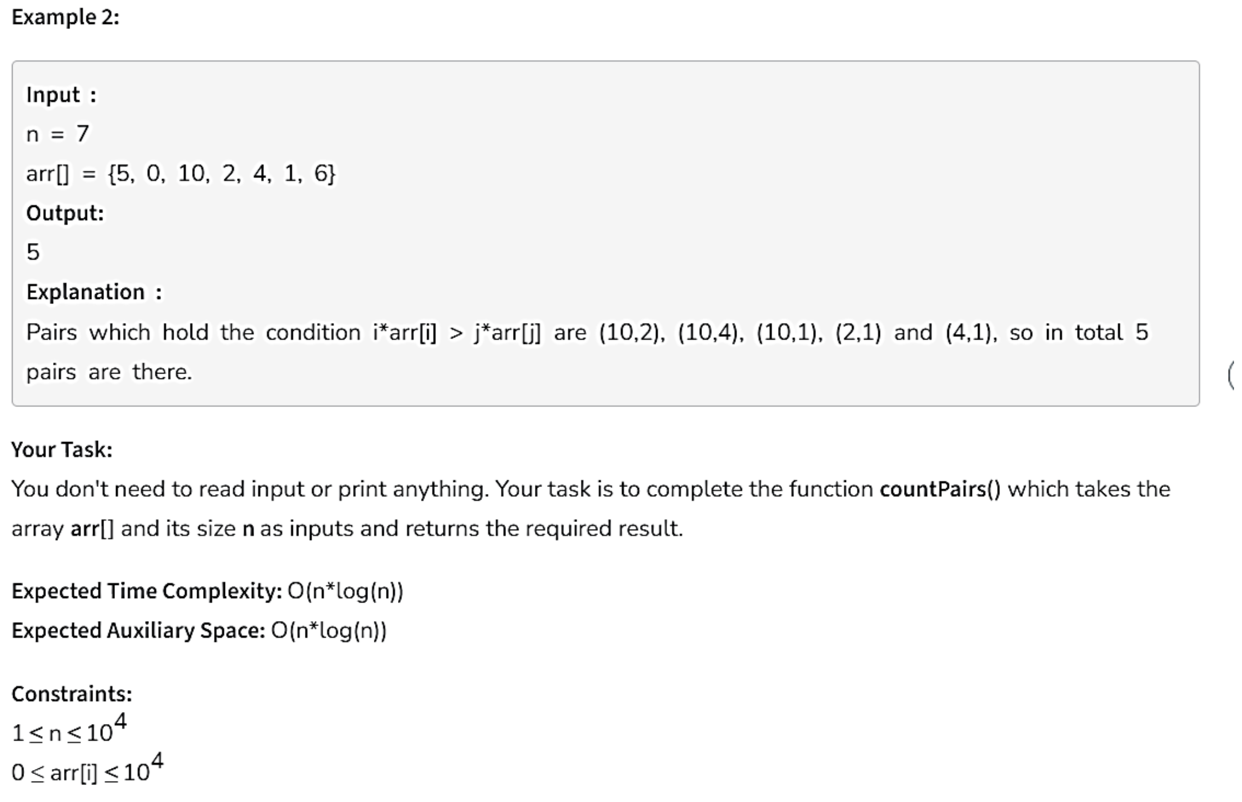
Hard Accuracy: 48.25% Submissions: 32K+ Points: 8

Given an array **arr** of **n** integers, count all pairs (**arr[i]**, **arr[j]**) in it such that **i*arr[i]** > **j*arr[j]** and **0 ≤ i < j < n**.

Note: 0-based Indexing is followed.

Example 1:

```
Input :
n = 4
arr[] = {8, 4, 2, 1}
Output :
2
Explanation:
If we see the array after operations
[0*8, 1*4, 2*2, 3*1] => [0, 4, 4, 3]
Pairs which hold the condition i*arr[i] > j*arr[j] are (4,1) and (2,1), so in total 2 pairs are available.
```



Example 2:

```
Input :
n = 7
arr[] = {5, 0, 10, 2, 4, 1, 6}
Output:
5
Explanation :
Pairs which hold the condition i*arr[i] > j*arr[j] are (10,2), (10,4), (10,1), (2,1) and (4,1), so in total 5 pairs are there.
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **countPairs()** which takes the array **arr**[] and its size **n** as inputs and returns the required result.

Expected Time Complexity: O(n*log(n))
Expected Auxiliary Space: O(n*log(n))

Constraints:
$1 \leq n \leq 10^4$
$0 \leq arr[i] \leq 10^4$

CODE

```
#User function Template for python3

class Solution:
    def countPairs(self,arr, n):
        # Your code goes here

        def merge(arr, low, mid, high):
            left = low
            right = mid + 1
            count = 0
            ans = []

            while left <= mid and right <= high:
                if arr[left] <= arr[right]:
                    ans.append(arr[left])
                    left += 1
                else:
                    count += (mid - left + 1)
                    ans.append(arr[right])
                    right += 1

            while left <= mid:
                ans.append(arr[left])
                left += 1

            while right <= high:
                ans.append(arr[right])
```

```
          right += 1


    for i in range(low, high+1):

        arr[i] = ans[i - low]


    return count




def merge_sort(arr, left, right):

    mid = (left + right) // 2

    count = 0

    if left < right:

        count += merge_sort(arr, left, mid)

        count += merge_sort(arr, mid+1, right)

        count += merge(arr, left, mid, right)

    return count


for i in range(n):

    arr[i] *= i


return merge_sort(arr, 0, n-1)
```