16h 12m left

ALL

1

2

3

4

5

1. Sort the Summary

Given an array of integers, create a 2-dimensional array where the first element is a distinct *value* from the array and the second element is that value's *frequency* within the array. Sort the resulting array descending by frequency. If multiple values have the same frequency, they should be sorted ascending.

Example

arr = [3, 3, 1, 2, 1]

- There are two values, 3 and 1, each with a frequency of 2, and one value 2 with a frequency of 1: [[3, 2], [1, 2], [2, 1]]
- Sort the 2-dimensional array descending by frequency: [[3,2], [1, 2], [2, 1]]
- Sort the 2-dimensional array ascending by value for values with matching frequencies: [[1,2], [3, 2], [2, 1]]

Function Description

Complete the function *groupSort* in the editor below.

groupSort has the following parameter(s):
int arr[n]: an array of integers
Returns:
int[n][2]: a 2-d array of integers sorted as described

Constraints

- $1 \le n \le 10^5$
- $1 \le arr[i] \le 10^5$

▸ **Input Format Format for Custom Testing**

▾ **Sample Case 0**

Sample Input

```
STDIN       Function
-----       --------
4       → arr[] size n = 4
2       → arr = [2, 1, 2, 2]
1
2
2
```

Sample Output

```
2 3
1 1
```

Explanation

- The value 2 occurs 3 times and 1 occurs 1 time: [[2, 3], [1, 1]]
- Sort the 2-dimensional array descending by frequency: [[2, 3], [1, 1]]
- Sort the 2-dimensional array ascending by value for values with matching frequencies: [[2, 3], [1, 1]]

▸ **Sample Case 1**

Python 3 ● Autocomplete Ready

```
1  > #!/bin/python···
10   import collections
11
12
13   def groupSort(arr):
14
15       frequency = {}
16       size = len(arr)
17       for i in range (0, size):
18           frequency[arr[i]] = frequency.get(arr[i], 0) + 1
19       a = []
20       arr_size = len(arr)
21       for i in frequency:
22           output=(i, frequency[i])
```

Line: 10 Col: 1

Test Results | **Custom Input** | **Run** | **Submit Code**

Compiled successfully. All available test cases passed

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5

Input (stdin) — Run as Custom Input | Download

```
4
2
1
2
2
```

Your Output (stdout)

```
2 3
1 1
```