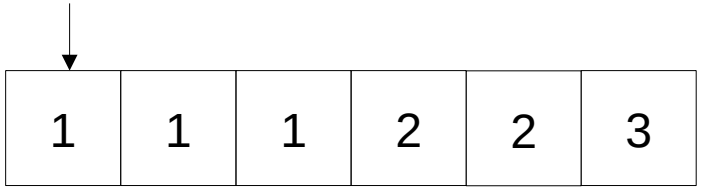
LeetCode #347

Top K Frequent Elements

Hash table count-value and value-count

To get the top k frequent elements, we first count the occurrences of all elements using a hash table. Then, we iterate through the hash table and use a heap to keep track of the k most frequent elements, popping the smallest one when the heap size exceeds k. Alternatively, we can use an array of size n + 1, where n is the size of the input array. In this approach, we store a list at each index of the array and append the i-th key to the list at the index corresponding to its frequency. Finally, we iterate through this array in reverse order to fill the result array.

Counting



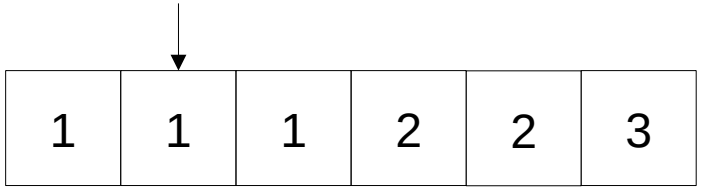
k = 2

freq

key	value
1	1

freq[nums[i]] += 1 = freq[1] = 1

Counting



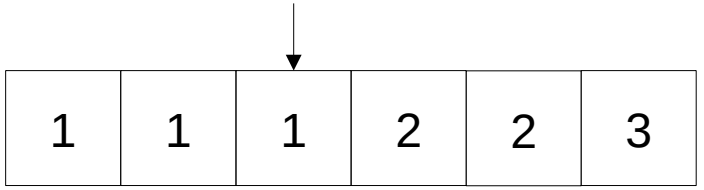
k = 2

freq

key	value
1	2

freq[nums[i]] += 1 = freq[1] = 2

Counting



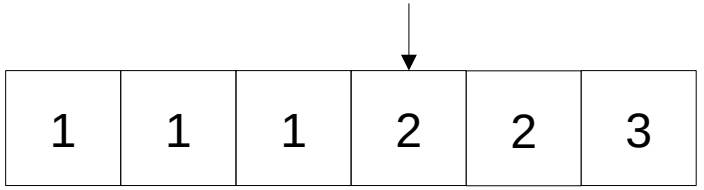
k = 2

freq

key	value
1	3

freq[nums[i]] += 1 = freq[1] = 3

Counting



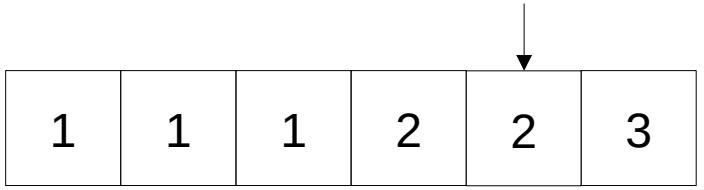
k = 2

freq

key	value
1	3
2	1

freq[nums[i]] += 1 = freq[2] = 1

Counting



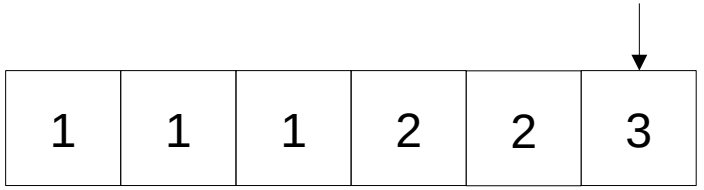
k = 2

freq

key	value
1	3
2	2

freq[nums[i]] += 1 = freq[2] = 2

Counting



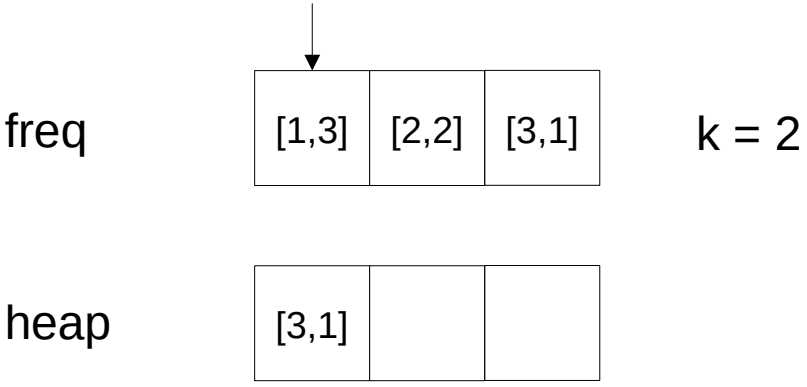
k = 2

freq

key	value
1	3
2	2
3	1

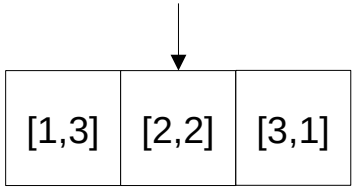
freq[nums[i]] += 1 = freq[3] = 1

Track frequency with a heap

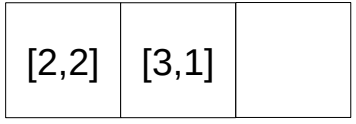


k, v = freq[i] = 1, 3
heap.push((v, k)) = heap.push((3, 1))

Track frequency with a heap

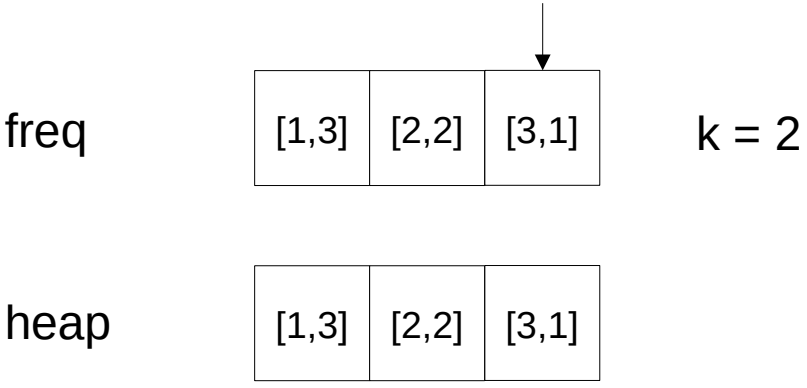


k = 2



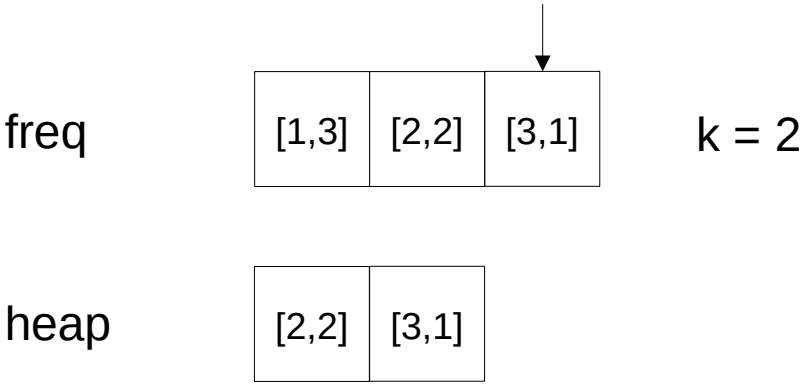
k, v = freq[i] = 2, 2
heap.push((v, k)) = heap.push((2, 2))

Track frequency with a heap



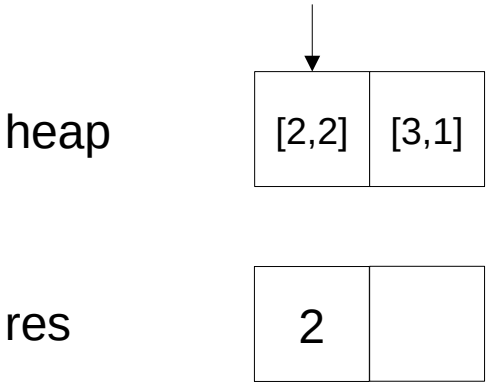
k, v = freq[i] = 3, 1
heap.push((v, k)) = heap.push((1, 3))

Track frequency with a heap



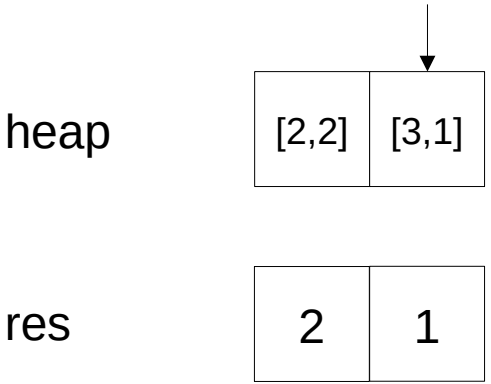
As the heap size exceeds k, then we pop the smallest element
heap.pop()

Fill the result array



res[i] = heap[i][1] = 2

Fill the result array



res[i] = heap[i][1] = 1