



[Click on the logo to find **Problem Statement**]

## Intuition

---

This code aims to find the top k most frequently occurring elements in a given list, `nums`. It employs a dictionary, `repeated_values`, to count the frequency of each element in the list. The dictionary is then sorted in descending order based on the frequencies, and the top k elements are extracted and returned in the output list

## Approach

---

### 1. Frequency Counting:

- Initialize an empty dictionary, `repeated_values`, to keep track of the frequency of each element in the input list.
- Iterate through the elements of the input list ( `nums` ).
- For each element, update its frequency in the dictionary ( `repeated_values` ).

### 2. Sorting by Frequency:

- Create a list of tuples, `sorted_repeated_values`, from the items of the frequency dictionary.
- Sort this list in descending order based on the frequency of each element.

### 3. Extracting Top k Elements:

- Initialize an empty list, `output`, to store the top k most frequent elements.
- Iterate over the sorted list and append the first k elements to the `output` list.

### 4. Return Result:

- Return the `output` list containing the top k most frequent elements in the input list.

## Complexity

---

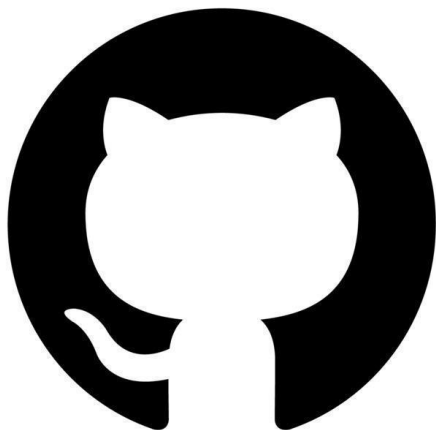
- Time complexity:  $O(n + m \log m + k)$
- Space complexity:  $O(n)$

# Code

---

```
class Solution:
    def topKFrequent(self, nums: List[int], k: int) -> List[int]:
        repeated_values = {}
        output = []
        for i in nums:
            repeated_values[i] = 1 + repeated_values.get(i, 0)
        sorted_repeated_values = sorted(repeated_values.items(), key=lambda x: x[1],
                                         reverse=True)
        for i in range(k):
            output.append(sorted_repeated_values[i][0])
        return output
```

If you want to see more solutions to coding problems, you can visit:



# GitHub