#LeetCode 692

https://leetcode.com/problems/top-k-frequent-words/description/

Yifeng Zeng

#题目描述

692. Top K Frequent Words

#思路报告

We have an unsorted array of words, and we want to find the top K frequent words, the first thing that I can think of is to have a sort of container or data structure to hold the most frequent elements (String). And once a new String comes, we compare the least frequent element in the container to this new String, if this new String is less frequent, we do nothing, otherwise, we remove the least frequent element out of the container and add this new String into that container. To find the least frequent or the smallest number in a container in O(1) time, we can think of a minHeap. So we maintain a size K minHeap to get the top K frequent words.

During coding, we can discuss one thing that needs to be taken care of: Every time we want to peek the least frequent element in the heap, so this heap should be a min heap of frequency. But what if the frequency is the same? For example we have a X 2, b X 2 and c X 2 and k = 2, we want to keep a and b, and ditch c. So with the same frequency, we want to keep a smaller string in lexicographical order. So use str2.CompareTo(str1) when frequency is the same. Another thing that needs to be careful is that it is a minHeap so least frequent word will be polled out first, so after polling out all the strings from the min heap, we want to reverse the list so that the most frequent string is at the beginning.

代码如下：

```
class Solution {
    public List<String> topKFrequent(String[] words, int k) {
        Map<String, Integer> map = new HashMap<>();
        for (String word : words) {
            map.put(word, map.getOrDefault(word, 0) + 1);
        }
```

```
        Queue<String> pq = new PriorityQueue<>(
            (str1, str2) -> (map.get(str1) == map.get(str2) ? str2.compareTo(str1) : ma

        for (String key : map.keySet()) {
            pq.offer(key);
            if (pq.size() > k) {
                pq.poll();
            }
        }

        List<String> res = new ArrayList<>();
        while (!pq.isEmpty()) {
            res.add(pq.poll());
        }
        Collections.reverse(res);

        return res;
    }
}
```

#套路总结

- Consider using a heap when dealing with "top K" question.
- map.getOrDefault(word, 0) + 1 is a new thing that I learned.
- Lambda expression is a new thing that I learned.
- Becareful about the String order when frequency is the same.