347. Top K Frequent Elements

link

返回k个最大的集合res.stream().mapToInt(Integer::valueOf).toArray();把arrayList转换成int数组

PriorityQueue<Map.Entry<Integer, Integer>> queue = new PriorityQueue<>((e1, e2) ->(e1.getValue() - e2.getValue()));这个必须得有一个<> 这个是上下文推断，否则会报e1是obj没有getValue()方法

```
    public int[] topKFrequent(int[] nums, int k) {
        Map<Integer, Integer> map = new HashMap();
        for(int num : nums){
            int count = map.getOrDefault(num, 0);
            map.put(num, ++count);
        }
        List<Integer> res = new ArrayList();
        PriorityQueue<Map.Entry<Integer, Integer>> queue = new PriorityQueue<>((e1,
e2) ->(e1.getValue() - e2.getValue()));
        // PriorityQueue<Map.Entry<Integer, Integer>> queue = new PriorityQueue((e1,
e2) ->(e2.getValue() - e1.getValue()));
        for(Map.Entry<Integer, Integer> entry : map.entrySet()){
            queue.offer(entry);
            if(queue.size() > k)
                queue.poll();
        }
        while(res.size() < k){
            Map.Entry<Integer, Integer> entry = queue.poll();
            res.add(entry.getKey());
        }
        return res.stream().mapToInt(Integer::valueOf).toArray();
    }
```

还可以使用bucketSort和treeMap 默认是从小到大 (a, b)->(a-b)从大到小就是return (b - a)

```
    public int[] topKFrequent(int[] nums, int k) {
        Map<Integer, Integer> map = new HashMap();
        for(int num : nums){
            int count = map.getOrDefault(num, 0);
            map.put(num, ++count);
        }
        List<Integer> res = new ArrayList();
        TreeMap<Integer, List<Integer>> treeMap = new TreeMap<>((e1, e2)->{
            return Integer.compare(e2, e1);
        });
        for(Map.Entry<Integer, Integer> entry : map.entrySet()){
            int val = entry.getValue();
            if(treeMap.containsKey(val)){
                treeMap.get(val).add(entry.getKey());
            }else{
                List tmp = new ArrayList();
                tmp.add(entry.getKey());
                treeMap.put(val, tmp);
            }

        }
        for(Map.Entry<Integer, List<Integer>> entry : treeMap.entrySet()){
            List<Integer> l = entry.getValue();
            for(int i : l){
                res.add(i);
            }
            if(res.size() == k)
                break;
        }
        return res.stream().mapToInt(Integer::valueOf).toArray();
    }
```

bucket 注意bucket的长度必须是length + 1 因为如果你是 `[1,1], 1` map里实际上是(1,2)值, 存到bucket的实际是(2,1)刚好等于nums.length + 1

```
     public int[] topKFrequent(int[] nums, int k) {
        Map<Integer, Integer> map = new HashMap();
        for(int num : nums){
            int count = map.getOrDefault(num, 0);
            map.put(num, ++count);
        }
        List<Integer> res = new ArrayList();
//这里得加1
        List<Integer>[] bucket = new ArrayList[nums.length + 1];
        for(Map.Entry<Integer, Integer> entry : map.entrySet()){
            int val = entry.getValue();
            if(bucket[val] != null){
                bucket[val].add(entry.getKey());
            }else{
                bucket[val] = new ArrayList<>();
                bucket[val].add(entry.getKey());
            }
        }
        for(int i = bucket.length - 1; i >= 0; --i){
            if(bucket[i] != null){
                for(int j : bucket[i]){
                    res.add(j);
                }
                if(res.size() == k)
                    break;
            }
        }
        return res.stream().mapToInt(Integer::valueOf).toArray();
```

```
    }
```

451. Sort Characters By Frequency

link一样的思路.

```
    public String frequencySort(String s) {
        int len = s.length();
        ArrayList<Character>[] buckets = new ArrayList[len + 1];
        Map<Character, Integer> map = new HashMap<>();
        for(int i = 0; i < len; ++i){
            int count = map.getOrDefault(s.charAt(i), 0);
            map.put(s.charAt(i), count + 1);
        }
        for(Map.Entry<Character, Integer> entry : map.entrySet()){
            if(buckets[entry.getValue()] == null){
                buckets[entry.getValue()] = new ArrayList();
            }
            buckets[entry.getValue()].add(entry.getKey());
        }
        StringBuilder sb = new StringBuilder();
        for(int i = buckets.length - 1; i > 0; --i ){
            List<Character> list = buckets[i];
            if(list != null){
                for(int j = 0; j < list.size(); ++j){
                    for(int count = 0; count < i; count++)
                        sb.append(list.get(j));
                }
            }
        }
        return sb.toString();
    }
```

如果希望输出的顺序和原字符串输入顺序一样怎么办