Heap

#G面试准备

Kth Smallest Element in a Sorted Matrix

Kth Smallest Element in a Sorted Matrix - LeetCode

- for循环走k-1次 因为poll k-1次之后 heap顶上的数字就是第k小的数
- 进循环之前先把(0,0)放进去了
- 循环每次都会把最小值拿出来 然后找他的下面和右边两个数
  - 因为往下和往右都是sorted 但是不知道找到的两个数谁大谁小
- 为什么要用Pair的数据结构？为了存点坐标？可是好像都没用到 一个hashmap存是否遍历好像就够了呀
- 注意怎么写comparator

```
class PairComparator implements Comparator<Pair> {
// Comparator 还是abstract的 要加类型
    public int compare (Pair a, Pair b) {
        return a.val - b.val;
    }
}
```

调用：

```
PriorityQueue<Pair> pq = new PriorityQueue<>(k, new PairComparator());
```

Merge k Sorted Lists

Merge k Sorted Lists - LeetCode

- 用pq来存当前所有的链表头
- 又要写comparator了 这回是真的需要 因为都是ListNode

```
class NodeComparator implements Comparator<ListNode> {
    public int compare(ListNode a, ListNode b) {
        return a.val - b.val;
    }
}
```

- 判断null不仅要在开头判断 还要判断每个链表里是不是空node
  - 这种情况一开始没处理 就没过

- 要在遍历链表头的for循环里 加入判断链表头是否为空

Find Median from Data Stream

Find Median from Data Stream

- 两个heap 一个minheap一个maxheap
- `maxHeap = new PriorityQueue<>(1000, Collections.reverseOrder());`
- 维护左半边maxHeap大小永远比右边大1
- 加入的时候判断现在数字总共是奇数个还是偶数个
- 偶数个时加入 如果小于等于左半边maxHeap.peek()就放入左边
  - 大于的时候要注意！先把新数字放进右边minHeap 再去把minHeap.poll()加入左边maxHeap

  原来的错误写法

  ```
  int right_min = (int) minHeap.poll();
  maxHeap.offer(right_min);
  minHeap.offer(num); // 这行应该最先做!
  ```

  例子：

  左[12] 右[40] 加入16

  判断条件会去跟12比 比他大所以要加入右边

  加入之前却先把40挪到了左边

  结果会变成[12 40] [16] 但应该是[12 16] [40]

  所以挪运行顺序 先加入minHeap再去把minHeap里最小的值加入左边 还是能维持左右节点个数之差不超过1
- 奇数个时加入 如果大于就直接加入右边minHeap
  - 小于要先加入num 再挪左边的最大maxHeap.poll()到右边minHeap