Leetcode(python版) 排序难题

57插入区间

题目：给出一个*无重叠的 ，*按照区间起始端点排序的区间列表。

在列表中插入一个新的区间，你需要确保列表中的区间仍然有序且不重叠（如果有必要的话，可以合并区间）。

**示例 1:**

**输入:** intervals = [[1,3],[6,9]], newInterval = [2,5]

**输出:** [[1,5],[6,9]]

**示例 2:**

**输入:** intervals = [[1,2],[3,5],[6,7],[8,10],[12,16]], newInterval = [4,8]

**输出:** [[1,2],[3,10],[12,16]]

**解释:** 这是因为新的区间 [4,8] 与 [3,5],[6,7],[8,10] 重叠

思路：1）插入新的区间；2）合并重复区间，即56的做法。

代码：

1. # Definition for an interval.
2. # class Interval(object):
3. #     def \_\_init\_\_(self, s=0, e=0):
4. #         self.start = s
5. #         self.end = e
7. **class** Solution(object):
8. **def** insert(self, intervals, newInterval):
9. """
10. :type intervals: List[Interval]
11. :type newInterval: Interval
12. :rtype: List[Interval]
13. """
14. intervals.append(newInterval)
15. l=len(intervals)
16. res=[]
17. intervals=sorted(intervals,key=**lambda** intervals:intervals.start)
18. low=intervals[0].start
19. high=intervals[0].end
20. **for** i **in** range(1,l):
21. **if** intervals[i].start<=high:
22. high=max(high,intervals[i].end)
23. **else**:
24. res.append([low,high])
25. low=intervals[i].start
26. high=intervals[i].end
27. res.append([low,high])
28. **return** res

164最大间距

题目：给定一个无序的数组，找出数组在排序之后，相邻元素之间最大的差值。

如果数组元素个数小于 2，则返回 0。

**示例 1:**

**输入:** [3,6,9,1]

**输出:** 3

**解释:** 排序后的数组是 [1,3,6,9]***,*** 其中相邻元素 (3,6) 和 (6,9) 之间都存在最大差值 3。

**示例 2:**

**输入:** [10]

**输出:** 0

**解释:** 数组元素个数小于 2，因此返回 0。

思路一：思路是采用桶排序的方法。

首先，求随机数组中的最大元素 , 最小元素，为线性时间复杂度；

然后，在最大元素和最小元素之间，分配（N-1）个间隔，即为N个桶（bucket）。每个桶为一个有序实数对（None,None），将每个数依次往目标桶里塞。

其中，桶的上下界若已有元素（存在一个桶对应多个元素），则只需计算新元素与下界的较小值，同时计算其与上界的较大值，并将该最值作为新下界/上界即可；

代码：

1. **class** Solution(object):
2. **def** maximumGap(self, nums):
3. """
4. :type nums: List[int]
5. :rtype: int
6. """
7. **if** len(nums) < 2:
8. **return** 0
9. a, b = min(nums), max(nums)
10. **if** a == b:
11. **return** 0
12. ans = 0
13. gap = int(math.ceil((b-a + 0.0)/(len(nums) - 1)))
14. bucketMin = [None **for** \_ **in** xrange(0, len(nums)+1)]
15. bucketMax = [None **for** \_ **in** xrange(0, len(nums)+1)]
17. **for** num **in** nums:
18. index = (num - a)/gap
19. **if** bucketMin[index] **is** None:
20. bucketMin[index] = num
21. **else**:
22. bucketMin[index] = min(bucketMin[index], num)
23. **if** bucketMax[index] **is** None:
24. bucketMax[index] = num
25. **else**:
26. bucketMax[index] = max(bucketMax[index], num)
27. bucketMin = [b **for** b **in** bucketMin **if** b **is** **not** None]
28. bucketMax = [b **for** b **in** bucketMax **if** b **is** **not** None]
29. **for** i **in** xrange(0, len(bucketMin) - 1):
30. ans = max(ans, bucketMin[i + 1] - bucketMax[i])
31. **return** ans

思路2：比较简单使用了python中的内置排序方法sort（），剩下就是根据题目要求，依，计算差值，最后返回最大差值

代码：

1. res = 0
2. **if** len(nums) < 2:
3. **return** 0
4. nums.sort()
5. **for** i **in** range(len(nums)-1):
6. x = abs(nums[i] - nums[i+1])
7. res = max(x,res)
8. **return** res

710黑名单中的随机数

题目：给定一个包含 [0，n ) 中独特的整数的黑名单 B，写一个函数从 [ 0，n ) 中返回一个**不在** B 中的随机整数。

对它进行优化使其尽量少调用系统方法 Math.random() 。

**提示:**

1. 1 <= N <= 1000000000
2. 0 <= B.length < min(100000, N)
3. [0, N) 不包含 N，详细参见 [interval notation](https://en.wikipedia.org/wiki/Interval_(mathematics)) 。

**示例 1:**

**输入:**

["Solution","pick","pick","pick"]

[[1,[]],[],[],[]]

**输出:** [null,0,0,0]

**示例 2:**

**输入:**

["Solution","pick","pick","pick"]

[[2,[]],[],[],[]]

**输出:** [null,1,1,1]

**示例 3:**

**输入:**

["Solution","pick","pick","pick"]

[[3,[1]],[],[],[]]

**Output:** [null,0,0,2]

**示例 4:**

**输入:**

["Solution","pick","pick","pick"]

[[4,[2]],[],[],[]]

**输出:** [null,1,3,1]

**输入语法说明：**

输入是两个列表：调用成员函数名和调用的参数。Solution的构造函数有两个参数，N 和黑名单 B。pick 没有参数，输入参数是一个列表，即使参数为空，也会输入一个 [] 空列表。

思路：

代码：

1. **class** Solution(object):
3. **def** \_\_init\_\_(self, N, blacklist):
4. """
5. :type N: int
6. :type blacklist: List[int]
7. """
8. self.map = {}
10. **for** i **in** blacklist:
11. self.map[i] = 1
12. self.M = N - len(blacklist)
13. **for** i **in** blacklist:
14. **if** i < self.M:
15. N = N-1
16. **while** N **in** self.map:
17. N -= 1
18. self.map[i] = N
20. **def** pick(self):
21. """
22. :rtype: int
23. """
24. **import**  random
25. res =random.randint(0,self.M-1)
26. **return** self.map[res] **if** res **in** self.map **else** res

29. # Your Solution object will be instantiated and called as such:
30. # obj = Solution(N, blacklist)
31. # param\_1 = obj.pick()