hw5_Hui_Duan

October 12, 2019

1 167. Two Sum II - Input array is sorted python

```
In [ ]: class Solution:
            def twoSum(self, numbers, target):
                """
                :type numbers: List[int]
                :type target: int
                :rtype: List[int]
                """
                if not numbers or len(numbers)<2:
                    return Null

                start = 0
                end = len(numbers)-1

                while end>start:
                    sum = numbers[start]+numbers[end]
                    if sum == target:
                        return [start+1,end+1]
                    elif sum > target: end -=1
                    else: start += 1
```

2 441. Arranging Coins

```
In [ ]: class Solution:
            def arrangeCoins(self, n: int) -> int:
                low,high = 0,n+1
                while low<high:
                    mid = low + int((high-low)/2)
                    if mid*(mid+1)/2 <=n:
                        low = mid+1
                    else:
                        high = mid
                return low-1
```

3 973. K Closest Points to Origin

```
In [ ]: class Solution:
            def kClosest(self, points: List[List[int]], K: int) -> List[List[int]]:
                points.sort(key = lambda x: x[0]**2+x[1]**2)
                return points[:K]
```

4 932. Beautiful Array

```
In [ ]: class Solution:
            def beautifulArray(self, N: int) -> List[int]:
                beautifularray = [1]
                while len(beautifularray) < N:
                    beautifularray = [2*i-1 for i in beautifularray] + [2*i for i in beautifular
                return [i for i in beautifularray if i<=N]
```

5 327. Count of Range Sum

```
In [ ]: class Solution(object):
            def countRangeSum(self, nums, lower, upper):
                """
                :type nums: List[int]
                :type lower: int
                :type upper: int
                :rtype: int
                """

                def mergeSort(lo, hi):
                    if lo == hi:
                        return 0
                    mi = lo + (hi - lo) / 2
                    cnt = mergeSort(lo, mi) + mergeSort(mi + 1, hi)
                    x = y = lo
                    for i in range(mi + 1, hi + 1):
                        while x <= mi and sums[i] - sums[x] >= lower:
                            x += 1
                        while y <= mi and sums[i] - sums[y] > upper:
                            y += 1
                        cnt += x - y
                    part = sums[lo : hi + 1]

                    l, h = lo, mi + 1
                    for i in range(lo, hi + 1):
                        x = part[l - lo] if l <= mi else max(sums)
                        y = part[h - lo] if h <= hi else max(sums)
                        if x < y:
                            l += 1
                        else:
                            h += 1
                        sums[i] = min(x, y)
                    return cnt

                sums = [0] * (len(nums) + 1)
                for x in range(1, len(nums) + 1):
                    sums[x] += nums[x - 1] + sums[x - 1]
                return mergeSort(0, len(nums))
```

6 315. Count of Smaller Numbers After Self

http://bookshadow.com/weblog/2015/12/06/leetcode-count-of-smaller-numbers-after-self/

```
In [ ]: class Solution(object):
            def countSmaller(self, nums: List[int]) -> List[int]:
                def mergeSort(num):
                    mid = len(num) / 2
                    if mid:
                        left, right = sort(num[:mid]), sort(num[mid:])
                        m, n = len(left), len(right)
                        i = j = 0
                        while i < m or j < n:
                            if j == n or i < m and left[i][1] <= right[j][1]:
                                num[i+j] = left[i]
                                smaller[left[i][0]] += j
                                i += 1
                            else:
                                num[i+j] = right[j]
                                j += 1
                    return enum
                smaller = [0] * len(nums)
                sort(list(enumerate(nums)))
                return smaller
```