LC 167: Two Sum II

```python
class Solution(object):
    def twoSum(self, numbers, target):
        """
        :type numbers: List[int]
        :type target: int
        :rtype: List[int]
        """
        residual_dic = {}
        for i in range(len(numbers)):
            num = numbers[i]
            res = target - num
            if num in residual_dic:
                return [residual_dic[num], i + 1]
            else:
                residual_dic[res] = i + 1
```

Success Details ›

Runtime: 40 ms, faster than 97.98% of Python online submissions for Two Sum II - Input array is sorted.

Memory Usage: 12.4 MB, less than 14.54% of Python online submissions for Two Sum II - Input array is sorted.

LC 441: Arranging Coins

```python
class Solution(object):
    def arrangeCoins(self, n):
        """
        :type n: int
        :rtype: int
        """
        if not n:
            return 0
        # Can be solved in O(n), O(log(n)), O(1)
        # Here shows O(log(n))
        l, r = 1, n
        while (l < r):
            mid = l + (r - l + 1)/2
            if mid*(mid + 1)/2.0 <= n:
                l = mid
            else:
                r = mid - 1
        return l
```

Success Details ›

Runtime: 12 ms, faster than 98.61% of Python online submissions for Arranging Coins.

Memory Usage: 11.8 MB, less than 14.29% of Python online submissions for Arranging Coins.

LC 973: Medium K Closest Points to Origin

```python
class Solution(object):
    def kClosest(self, points, K):
        """
        :type points: List[List[int]]
        :type K: int
        :rtype: List[List[int]]
        """
        dist = [p[0]**2 + p[1]**2 for p in points]

        low = 0
        high = len(dist) - 1
        res = []

        while low <= high:
            idx = self.quickSort(dist, points, low, high)
            if idx - low == K - 1:
                return res + points[low : low + K]
            elif  idx - low > K - 1:
                high = idx - 1
            else:
                K -= (idx - low + 1)
                res.extend(points[low : idx + 1])
                low = idx + 1

    def quickSort(self, dist, points, low, high):
        pivot = dist[high]
        i = low
        for j in range(low, high):
            if dist[j] < pivot:
                dist[i], dist[j] = dist[j], dist[i]
                points[i], points[j] = points[j], points[i]
                i += 1

        dist[high], dist[i] =  dist[i], dist[high]
        points[high], points[i] =  points[i], points[high]
        return i
```

Success Details ›

Runtime: 560 ms, faster than 87.98% of Python online submissions for K Closest Points to Origin.

Memory Usage: 17.4 MB, less than 81.13% of Python online submissions for K Closest Points to Origin.

LC 932: Beautiful Array

```python
class Solution(object):
    def beautifulArray(self, N):
        """
        :type N: int
        :rtype: List[int]
        """
        tmp_map = {1:[1]}
        def find(n):
            if n not in tmp_map:
                left = find((n + 1)//2)
                right = find(n//2)
                tmp_map[n] = [2*i - 1 for i in left] + [2*j for j in right]
            return tmp_map[n]

        return find(N)
```

Success Details ›

Runtime: 20 ms, faster than 88.94% of Python online submissions for Beautiful Array.

Memory Usage: 12.1 MB, less than 100.00% of Python online submissions for Beautiful Array.

LC 327: Count of Range Sum

```python
class Solution(object):
    def countRangeSum(self, nums, lower, upper):
        """
        :type nums: List[int]
        :type lower: int
        :type upper: int
        :rtype: int
        """
        tmp_sum = [0]
        for num in nums:
            tmp_sum.append(tmp_sum[-1] + num)

        def solution(low, high):
            mid = low + (high - low) / 2
            if low == mid:
                return 0
            tmp_cnt = solution(low, mid) + solution(mid, high)
            i, j = mid, mid
            for num in tmp_sum[low:mid]:
                while i < high and tmp_sum[i] - num < lower:
                    i += 1
                while j < high and tmp_sum[j] - num <= upper:
                    j += 1
                tmp_cnt += j - i
            tmp_sum[low:high] = sorted(tmp_sum[low:high])
            return tmp_cnt

        return solution(0, len(tmp_sum))
```

Runtime: 176 ms, faster than 78.22% of Python online submissions for Count of Range Sum.

Memory Usage: 13.5 MB, less than 100.00% of Python online submissions for Count of Range Sum.

LC 315: Count of Smaller Numbers After Self

```python
class Solution(object):
    def countSmaller(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """
        ### Idea is to use mergeSort and keep track of how many numbers moved\
        ### from right to left of a number

        enum = list(enumerate(nums))
        res = [0]*len(nums)
        self.sort(enum, res)
        return res

    def sort(self, enum, res):
        half = len(enum)/2
        if half:
            left, right = self.sort(enum[:half], res), self.sort(enum[half:], res)
            for i in range(len(enum))[::-1]:
                if not right or left and left[-1][1] > right[-1][1]:
                    res[left[-1][0]] += len(right)
                    enum[i] = left.pop()
                else:
                    enum[i] = right.pop()
        return enum
```

Success Details ›

Runtime: 152 ms, faster than 47.87% of Python online submissions for Count of Smaller Numbers After Self.

Memory Usage: 17.2 MB, less than 12.50% of Python online submissions for Count of Smaller Numbers After Self.