LC 744: Find Smallest Letter Greater Than Target

```
class Solution(object):
    def nextGreatestLetter(self, letters, target):
        """
        :type letters: List[str]
        :type target: str
        :rtype: str
        """
        l, r = 0, len(letters)
        while l < r:
            mid = l + (r - l) // 2
            if letters[mid] <= target:
                l = mid + 1
            else:
                r = mid
        return letters[l%len(letters)]
```

Success Details ›

Runtime: 76 ms, faster than 96.16% of Python online submissions for Find Smallest Letter Greater Than Target.

Memory Usage: 13.7 MB, less than 33.33% of Python online submissions for Find Smallest Letter Greater Than Target.

LC 475: Heaters

```python
class Solution(object):
    def findRadius(self, houses, heaters):
        # If heaters are given in unsorted style,
        # we may need to sort it first
        houses.sort()
        heaters.sort()
        heaters.append(float('inf'))
        res = 0
        he_id = 0
        for h in houses:
            while he_id < len(heaters) and (h > heaters[he_id] + (heaters[he_id + 1] - heaters[he_id])//2):
                he_id += 1
            res = max(res, abs(heaters[he_id] - h))
        return res
```

Success Details ›

Runtime: 248 ms, faster than 73.06% of Python online submissions for Heaters.

Memory Usage: 14.5 MB, less than 66.67% of Python online submissions for Heaters.

LC 74: Search a 2D Matrix

```python
class Solution(object):
    def searchMatrix(self, matrix, target):
        if not matrix or not matrix[0] or target is None:
            return False

        m = len(matrix)
        n = len(matrix[0])
        if target < matrix[0][0] or target > matrix[-1][-1]:
            return False

        l = 0
        r = m*n - 1
        while l <= r:
            mid = (l + r)/2
            num = matrix[mid/n][mid%n]
            if num == target:
                return True
            elif num < target:
                l = mid + 1
            else:
                r = mid - 1
        return False
```

Success Details ›

Runtime: 48 ms, faster than 82.27% of Python online submissions for Search a 2D Matrix.

Memory Usage: 13.5 MB, less than 71.43% of Python online submissions for Search a 2D Matrix.

LC 34: Find First and Last Position of Element in Sorted Array

```python
class Solution(object):
    def search_left(self, nums, target, left):
        low = 0
        high = len(nums)

        while low < high:
            mid = low + (high - low)/2
            if nums[mid] > target or (left and target == nums[mid]):
                high = mid
            else:
                low = mid + 1
        return low

    def searchRange(self, nums, target):
        left_idx = self.search_left(nums, target, True)

        if left_idx == len(nums) or nums[left_idx] != target:
            return [-1, -1]

        right_idx = self.search_left(nums, target, False) - 1
        return [left_idx, right_idx]
```

Success Details ›

Runtime: 60 ms, faster than 98.70% of Python online submissions for Find First and Last Position of Element in Sorted Array.

Memory Usage: 13.2 MB, less than 23.53% of Python online submissions for Find First and Last Position of Element in Sorted Array.

LC 719: Find K-th Smallest Pair Distance

```
class Solution(object):
    def smallestDistancePair(self, nums, k):
        def countPairs(a, target):
            n = len(a)
            res, j = 0, 0
            for i in range(n):
                while j < n and a[j] - a[i] <= target:
                    j += 1
                res += j - i - 1
            return res

        n = len(nums)
        nums.sort()
        # Temp Minimum absolute difference
        low = nums[1] - nums[0]
        for i in range(1, n - 1):
            low = min(low, nums[i + 1] - nums[i])
        # Maximum absolute difference
        high = nums[n - 1] - nums[0]

        #Do binary search for k-th absolute difference
        while low < high:
            mid = low + (high - low)/2
            cnt = countPairs(nums, mid)
            if cnt < k:
                low = mid + 1
            else:
                high = mid

        return low
```

Success Details ›

Runtime: 104 ms, faster than 62.79% of Python online submissions for Find K-th Smallest Pair Distance.

Memory Usage: 12.2 MB, less than 50.00% of Python online submissions for Find K-th Smallest Pair Distance.

google 面经题

有一条公路,长度是m, 中间有k个加油站,
由此我们可以得到一个相邻加油站之间的最大距离,然后 给你一个数t,这个数代表增加的加油站的数量(往里面插入),求使得相 邻加油站之间最大距离变得最小的值,返回这个最小的最大距离。

```python
def minMaxDist(nums, k):
    l, r = 0, nums[0]
    while l < r:
        needed = 0
        mid = l + (r - l)//2
        i = 0
        while i < len(nums) and nums[i] > mid:
            needed += int(nums[i] // mid) + (nums[i] % mid > 0) - 1
            i += 1
        if needed <= k:
            r  = mid
        else:
            l = mid + 1
    return l
m = 125
gas_loc = [5, 8, 10, 25, 28, 31, 72, 80, 85, 100]
t = 8
dist = [gas_loc[i] - gas_loc[i-1] for i in range(1, len(gas_loc))]
if gas_loc[0] > 0:
    dist.insert(0, dist[0] - 0)
if gas_loc[-1] < m:
    dist.append(m - gas_loc[-1])
dist.sort(reverse = True)
res = minMaxDist(dist, t)
print('Road Length m:', m)
print('Gas Locations:', gas_loc)
print('Distances:', dist)
print('Extra gas stations t:', t)
print('Min Max Dist:', res)
```

```
Road Length m: 125
Gas Locations: [5, 8, 10, 25, 28, 31, 72, 80, 85, 100]
Distances: [41, 25, 15, 15, 8, 5, 3, 3, 3, 3, 2]
Extra gas stations t: 8
Min Max Dist: 9
```