LC 215

class Solution:

def findKthLargest(self, nums: List[int], k: int) -> int:

temp = nums[:k]

heapq.heapify(temp)

for i in range(k,len(nums)):

if nums[i] > temp[0]:

heapq.heapreplace(temp,nums[i])

return temp[0]

LC 973

class Solution:

def kClosest(self, points: List[List[int]], k: int) -> List[List[int]]:

lst = list(map(lambda x:x[0] \*\* 2 + x[1] \*\* 2, points))

for i in range(len(points)):

points[i] = (points[i],lst[i])

lst = heapq.nlargest(k,points,key=lambda i:-i[1])

return [item[0] for item in lst[:k]]

LC 240

class Solution:

def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:

m = len(matrix)

n = len(matrix[0])

i = m-1

j = 0

while i >= 0 and j < n:

if matrix[i][j] == target: return True

elif matrix[i][j] < target: j += 1

else: i -= 1

return False

LC 493

class Solution:

def reversePairs(self, nums: List[int]) -> int:

if len(nums) == 1:

return 0

elif len(nums) == 2:

return int(nums[0] > nums[1] \* 2)

m = len(nums) // 2

a = nums[:m]

b = nums[m:]

res = self.reversePairs(a) + self.reversePairs(b)

b = sorted([item \* 2 for item in b])

a.sort()

ind1 = m-1

ind2 = len(nums)-m-1

while ind1 >= 0 and ind2 >= 0:

if a[ind1] > b[ind2]:

res += (ind2 + 1)

ind1 -= 1

else:

ind2 -= 1

return res