LC 34

class Solution:

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

if not nums:

return [-1,-1]

lo = 0

hi = len(nums) - 1

mid = lo + (hi-lo) // 2

while lo <= hi and nums[mid] != target:

if nums[mid] > target:

hi = mid - 1

elif nums[mid] < target:

lo = mid + 1

mid = lo + (hi-lo) // 2

if nums[mid] != target:

return [-1,-1]

lo2 = 0

hi2 = mid

while lo2 <= hi2:

mid2 = lo2 + (hi2-lo2) // 2

if nums[mid2] == target:

hi2 = mid2 - 1

else:

lo2 = mid2 + 1

lo = mid

hi = len(nums) - 1

while lo <= hi:

mid2 = lo + (hi-lo) // 2

if nums[mid2] > target:

hi = mid2 - 1

else:

lo = mid2 + 1

return [lo2,hi]

LC 1011

class Solution:

def shipWithinDays(self, weights: List[int], days: int) -> int:

lo = max(weights)

hi = sum(weights)

while lo <= hi:

mid = lo + (hi - lo) // 2

day = 1

ind = 0

cur\_sum = 0

for wt in weights:

if cur\_sum + wt > mid:

day += 1

cur\_sum = 0

if day > days:

lo = mid + 1

break

cur\_sum += wt

ind += 1

if lo < mid + 1:

hi = mid - 1

return lo

LC 33

class Solution:

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

if len(nums) == 1:

return 0 if nums[0] == target else -1

def binarySearch(nums,target):

lo = 0

hi = len(nums) - 1

while lo <= hi:

mid = lo + (hi - lo) // 2

if nums[mid] == target:

return mid

elif nums[mid] < target:

lo = mid + 1

else:

hi = mid - 1

return -1

lo = 0

hi = len(nums) - 1

if nums[lo] < nums[hi]:

return binarySearch(nums,target)

if target < nums[0] and target > nums[-1]:

return -1

while lo <= hi:

mid = lo + (hi - lo) // 2

if nums[mid] >= nums[0]:

lo = mid + 1

else:

hi = mid - 1

if target < nums[lo] or target > nums[hi]:

return -1

if target <= nums[-1]:

res = binarySearch(nums[lo:],target)

if res == -1:

return -1

return lo + res

return binarySearch(nums[:lo],target)

LC 74

class Solution:

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

m = len(matrix)

n = len(matrix[0])

lo = 0

hi = m\*n-1

while lo <= hi:

mid = (hi + lo) // 2

row = mid // n

col = mid % n

temp = matrix[row][col]

if temp == target:

return True

elif temp < target:

lo = mid + 1

else:

hi = mid - 1

return False

LC 887

Original solution:

class Solution:

def superEggDrop(self, k: int, n: int) -> int:

if k == 1:

return n

if n < 3:

return n

dp = [[1] \* n for \_ in range(k)]

for j in range(1,n):

dp[0][j] = j+1

for i in range(1,k):

dp[i][1] = 2

dp[i][2] = 2

m = 2

for j in range(3,n):

a = max(dp[i-1][m-2],dp[i][j-m])

b = max(dp[i-1][m-1],dp[i][j-m-1])

if a<b:

dp[i][j] = a+1

else:

dp[i][j] = b+1

m += 1

return dp[-1][-1]

official solution:

class Solution(object):

def superEggDrop(self, K, N):

def f(x):

ans = 0

r = 1

for i in range(1, K+1):

r \*= x-i+1

r //= i

ans += r

if ans >= N: break

return ans

lo, hi = 1, N

while lo < hi:

mi = (lo + hi) // 2

if f(mi) < N:

lo = mi + 1

else:

hi = mi

return lo