Day 8 programs

1. def minimumLength(nums):

n = len(nums)

removable\_pairs = 0

while i < n - 1:

if nums[i] < nums[i + 1]:

removable\_pairs += 1

i += 2

else:

i += 1

return n - 2

nums = [1, 2, 3, 4]

print(minimumLength(nums))

2. class TreeNode:

def \_\_init\_\_(self, val=0, left=None, right=None):

self.val = val

self.left = left

self.right = right

def sortedArrayToBST(nums):

if not nums:

return None

mid = len(nums) // 2

root = TreeNode(nums[mid])

root.left = sortedArrayToBST(nums[:mid])

root.right = sortedArrayToBST(nums[mid+1:])

return root

def inorderTraversal(root):

if root:

inorderTraversal(root.left)

print(root.val, end=' ')

inorderTraversal(root.right)

nums = [-10, -3, 0, 5, 9]

root = sortedArrayToBST(nums)

inorderTraversal(root)

3. def substringWords(words):

result = []

n = len(words)

for i in range(n):

for j in range(n):

if i != j and words[i] in words[j]:

result.append(words[i])

break

return result

words = ["mass", "as", "hero", "superhero"]

print(substringWords(words))

4. def wiggleSort(nums):

nums.sort()

half = len(nums[::2])

nums[::2], nums[1::2] = nums[:half][::-1], nums[half:][::-1]

nums1 = [1, 5, 1, 1, 6, 4]

wiggleSort(nums1)

print(nums1)

nums2 = [1, 3, 2, 2, 3, 1]

wiggleSort(nums2)

print(nums2)

5. from collections import deque

def updateMatrix(mat):

rows, cols = len(mat), len(mat[0])

queue = deque()

for i in range(rows):

for j in range(cols):

if mat[i][j] == 0:

queue.append((i, j))

else:

mat[i][j] = float('inf')

directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]

while queue:

cell = queue.popleft()

for d in directions:

new\_i, new\_j = cell[0] + d[0], cell[1] + d[1]

if 0 <= new\_i < rows and 0 <= new\_j < cols and mat[new\_i][new\_j] > mat[cell[0]][cell[1]] + 1:

mat[new\_i][new\_j] = mat[cell[0]][cell[1]] + 1

queue.append((new\_i, new\_j))

return mat

mat1 = [[0, 0, 0], [0, 1, 0], [0, 0, 0]]

mat2 = [[0, 0, 0], [0, 1, 0], [1, 1, 1]]

print(updateMatrix(mat1))

print(updateMatrix­­­­­­­­­­­­­­­­­­­­­­­­(mat2))