### **Time Complexity** Assignment

### Problem 1: Quicksort

def quicksort(arr):

if len(arr) <= 1:

return arr

pivot = arr[len(arr) // 2]

left = [x for x in arr if x < pivot]

middle = [x for x in arr if x == pivot]

right = [x for x in arr if x > pivot]

return quicksort(left) + middle + quicksort(right)

**Time Complexity:** The average time complexity of Quicksort is $$O(n \log n)$$, but in the worst case (when the pivot is the smallest or largest element), it can be $$O(n^2)$$.

### Problem 2: Nested Loop Example

def nested\_loop\_example(matrix):

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

total = 0

for i in range(rows):

for j in range(cols):

total += matrix[i][j]

return total

**Time Complexity:** The time complexity is $$O(rows \times cols)$$, which is $$O(n^2)$$ if the matrix is square.

### Problem 3: Example Function

def example\_function(arr):

result = 0

for element in arr:

result += element

return result

**Time Complexity:** The time complexity is $$O(n)$$, where $$n$$ is the length of the array.

### Problem 4: Longest Increasing Subsequence

def longest\_increasing\_subsequence(nums):

n = len(nums)

lis = [1] \* n

for i in range(1, n):

for j in range(0, i):

if nums[i] > nums[j] and lis[i] < lis[j] + 1:

lis[i] = lis[j] + 1

return max(lis)

**Time Complexity:** The time complexity is $$O(n^2)$$ due to the nested loops.

### Problem 5: Mysterious Function

def mysterious\_function(arr):

n = len(arr)

result = 0

for i in range(n):

for j in range(i, n):

result += arr[i] \* arr[j]

return result

# Time Complexity: O(n^2)

### Problem 6: Sum of Digits

def sum\_of\_digits(n):

if n == 0:

return 0

else:

return n % 10 + sum\_of\_digits(n // 10)

# Example usage:

print(sum\_of\_digits(123)) # Output: 6

### Problem 7: Fibonacci Series

def fibonacci\_series(n):

if n <= 0:

return []

elif n == 1:

return [0]

elif n == 2:

return [0, 1]

else:

series = fibonacci\_series(n - 1)

series.append(series[-1] + series[-2])

return series

# Example usage:

print(fibonacci\_series(6)) # Output: [0, 1, 1, 2, 3, 5]

### Problem 8: Subset Sum

def subset\_sum(nums, target):

if target == 0:

return True

if not nums:

return False

return subset\_sum(nums[1:], target) or subset\_sum(nums[1:], target - nums[0])

# Example usage:

print(subset\_sum([3, 34, 4, 12, 5, 2], 9)) # Output: True

### Problem 9: Word Break

def word\_break(s, word\_dict):

if not s:

return True

for word in word\_dict:

if s.startswith(word):

if word\_break(s[len(word):], word\_dict):

return True

return False

# Example usage:

print(word\_break("leetcode", ["leet", "code"])) # Output: True

### Problem 10: N-Queens

def solve\_n\_queens(n):

def is\_safe(board, row, col):

for i in range(row):

if board[i] == col or \

board[i] - i == col - row or \

board[i] + i == col + row:

return False

return True

def solve(board, row):

if row == n:

result.append(["." \* i + "Q" + "." \* (n - i - 1) for i in board])

return

for col in range(n):

if is\_safe(board, row, col):

board[row] = col

solve(board, row + 1)

result = []

solve([-1] \* n, 0)

return result

# Example usage:

print(solve\_n\_queens(4))

# Output:

# [

# [".Q..",

# "...Q",

# "Q...",

# "..Q."],

# ["..Q.",

# "Q...",

# "...Q",

# ".Q.."]

# ]