Xinyan LIU TIN093

Problem 1

(a) Time Complexity Analysis

Given an array $C(A)$:

$$1 \quad 2 \quad \cdots \quad \cdots \quad n$$

- **Outer loop**: For i in range $C(n) \longrightarrow O(n)$
- **Middle loop**: For j in range $(i+1, n) \longrightarrow O(n)$
- **Inner operation**: Calculate the distance between i and j, which is $O(n)$ in the worst case.

Overall time complexity:

$$n \times n \times n = O(n^3)$$

(b) Create Array to Store Distance

- **Outer loop**: For i in range $(n) \longrightarrow O(n)$
- **Middle loop**: For j in range $(i+1, n) \longrightarrow O(n)$
- **Inner operation**:

$$\text{distance}(i, j) = \text{distance}(i, j-1) + \text{distance}(i, j)$$

Do not need to iterate over all the elements, like (a)]

Only two for loop. Overall time complexity:

$$n \times n = O(n^2)$$

(c)

No, it can not be improved, as we need to iterate all pairs to get the sum, The minimal number of pairs is:

$$\frac{n(n-1)}{2} = O(n^2)$$

We cannot optimize further.

Problem 2

(a)

Assume size[i] means the storage space for i-th room

Initialize left = 1 and right = n.
while left < right **do**
  if size[left] + size[right] == s **then**
    return "We find"
  else if size[left] + size[right] < s **then**
    left = left + 1
  else
    right = right − 1
  end if
end while
return "We cannot find"

The time complexity of the two-pointer algorithm is $O(n)$, where n is the size of the input array. Since it just iterates over all the n elements once without a nested loop

b

Since the array is sorted, moving the left pointer to the right decreases the sum, and moving the right pointer to the left increases the sum. The algorithm systematically explores all possible pairs by adjusting the pointers based on the current sum.
If a valid pair exists, the algorithm will eventually find it because the left and right pointers cover all possible pairs without skipping any elements. The loop terminates only when all possible pairs have been checked (i.e., when left and right meet).
If the loop ends without finding a valid pair, it means no such pair exists in the array, and the algorithm correctly returns "We cannot find."