Lab Assignment 2

Explanation

Name: Fathin Ishrak
ID: 20301027
Section: 13

task 1(1)

This code uses a brute-force approach to find 2 values from the list that sum to S in O(n^2) time complexity. It takes the input from a file named "input 1(1).txt" and writes the output in a file named "output 1(1).txt". Both input and output files ~~were~~ are in the same directory.

task 1(2)

This code efficiently uses a hash table to keep track of seen values and their positions in the list, allowing it to find a pair of values that sum to the target S in O(n) time complexity. It takes the input from a file named "input 1(2).txt" and writes the output in a file named "output 1(2).txt". Both input and output files are in the same directory.

task 2(1)

The merge-sort algorithm is used to merge 2 sorted lists into a single sorted list efficiently with a time complexity of O(nlog n). This code merges 2 sorted lists into a single sorted list by iteratively comparing and ~~combining~~ combining elements from both lists. It takes the input from a file named "input2(1).txt" and writes the output to the output file named "output2(1).txt". Both the input and output files are in the same directory.

task 2(2)

To merge 2 sorted lists into a single list in O(n) time complexity, we can two-pointer approach that does not require comparing elements one by one. This approach is more efficient. It takes the input from a file named "input2(2).txt" and writes the output to the output file named "output2(2).txt". Both the input and output file files are in the same directory.

task 3

This problem is solved using greedy algorithm. It sorts the tasks based on their end times and then choose tasks that do not overlap with each other, maximizing the number of tasks completed. The code reads the input from "input3.txt" file, sorts the tasks based on their end times, selects non-overlapping tasks to maximize completion, and writes the output to "output3.txt" file.

task 4

To solve the problem efficiently and achieve O(nlogn) time complexity, we use greedy algorithm along with sorting. The idea is to sort the activities by their end times and then assign them to people one by one, ensuring that the activites don't overlap. This code reads the input from "input4.txt" processes it using greedy algorithm that sorts and assign activities, and then writes the result to "output4.txt".