Painter's Partition Problem CodeStudio

You are given an array/list of length 'N', where each element represents the length of a board. You also have 'K' painters available to paint these boards. Each unit of a board takes 1 unit of time to paint. Your task is to determine the minimum time required to paint all the 'N' boards, considering the constraint that each painter can only paint continuous sections of boards.

Input: boards = [5, 5, 5, 5], painters = 2

Explanation: There are four boards with lengths 5, 5, 5, and 5. We have 2 painters available. The optimal way to paint these boards is to assign one painter to paint the first two boards and the other painter to paint the remaining two boards. Each painter will take 5 units of time to paint their assigned boards. So, the minimum time required is 5.

Input: boards = [10, 20, 30, 40], painters = 2

Explanation: There are four boards with lengths 10, 20, 30, and 40. We have 2 painters available. The optimal way to paint these boards is to assign one painter to paint the first three boards and the other painter to paint the last board. Each painter will take 60 units of time (10 + 20 + 30 and 40) to paint their assigned boards. So, the minimum time required is 60.

Input: boards = [48, 90], painters = 2

Explanation: There are two boards with lengths 48 and 90. We have 2 painters available. Each painter can paint one board, and each board will take the same amount of time, which is equal to its length. So, the minimum time required is the length of the longer board, which is 90.

Approach 1: Using Binary Search Algorithm to find the minimum number of boards allocated to the painter with the maximum allocation.

The isPossibleSolution function is used to check if a given mid value is a possible solution. It iterates through the boards and keeps track of the total area painted by a painter. If the total area exceeds mid, it increments the painter count and checks if it violates the number of available painters or if the board length itself is greater than mid.

The paintLargestMinArea function calculates the total area of all boards and initializes the low and high variables. It performs a binary search between low and high to find the minimum time required.

In each iteration of the binary search, it calculates the mid value and checks if it is a possible solution by calling the isPossibleSolution function. If it is a valid solution, it updates the ans variable with the current mid value and adjusts the search range by updating high to mid - 1.

If the mid value is not a valid solution, it adjusts the search range by updating low to mid + 1.

After the binary search is complete, the ans variable will hold the minimum time required to paint all the boards, which is then returned.

Time Complexity:

The binary search algorithm has a time complexity of O(log N), where N is the sum of the board lengths. The isPossibleSolution function has a time complexity of O(N), as it iterates through the boards. Therefore, the overall time complexity of the code is O(N log N).

Space Complexity:

The code has a space complexity of O(1) as it uses a constant amount of extra space regardless of the input size.