Book Allocation Problem GFG

You are given an array A[] of N books, where each book has a specific number of pages represented by Ai. There are M students who need to be allocated a contiguous set of books, with each student receiving at least one book. The goal is to find the permutation where the student with the maximum number of pages allocated to them receives the minimum number of pages, out of all possible permutations.

Write a function that takes the number of books N, an array A[] representing the number of pages in each book, and the number of students M as inputs. The function should return the minimum number of pages allocated to the student with the maximum allocation among all possible permutations. If a valid assignment is not possible, the function should return -1.

Example:

Consider the following example:

N = 4

A[] = {12, 34, 67, 90}

M = 2

Explanation:

In this example, we have 4 books with the number of pages given as {12, 34, 67, 90}, and there are 2 students who need to be allocated books.

The possible allocations are as follows:

{12} and {34, 67, 90} with a maximum allocation of 191 pages.

{12, 34} and {67, 90} with a maximum allocation of 157 pages.

{12, 34, 67} and {90} with a maximum allocation of 113 pages.

Among these allocations, the minimum number of pages allocated to the student with the maximum allocation is 113 pages. Therefore, the function should return 113 as the output.

If a valid assignment is not possible (i.e., if the number of books is less than the number of students or if the sum of all book pages is less than the number of students), the function should return -1 to indicate that a valid allocation cannot be made.

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

considering the given input {10, 20, 30, 40} with 4 books and 2 students.

Step 1: Include the necessary libraries and define the namespace.

Step 2: Define the function isPossibleSolution to check if a given allocation is a valid solution. This function takes the book array, number of books, number of students, and the current allocation (mid) as parameters.

Step 3: In the isPossibleSolution function, initialize studentCount and pageCount variables to keep track of the current student and their allocated pages, respectively.

Step 4: Iterate through the books using a for loop.

Step 5: Check if adding the current book's pages to the pageCount is within the allocation limit (mid).

Step 6: If the allocation is valid, update the pageCount with the added book's pages.

Step 7: If the allocation exceeds the limit, move to the next student by incrementing studentCount.

Step 8: Check if the number of students exceeds the given limit or if the current book has more pages than the maximum allowed (mid). If either condition is true, return false, indicating an invalid allocation.

Step 9: If all books have been allocated with valid conditions, return true.

Step 10: Define the function allocateMinimumPages to find the minimum number of pages allocated to the student with the maximum allocation. This function takes the book array, number of books, and number of students as parameters.

Step 11: Check if the number of students is greater than the number of books. If so, return -1, as there are not enough books for each student.

Step 12: Initialize variables low, high, mid, ans, and pageSum. low is set to 0, high is set to the sum of all book pages, ans is set to -1 (to track the minimum allocation), and pageSum is calculated by summing all book pages.

Step 13: Use a binary search algorithm to find the minimum allocation. Perform binary search by updating low and high based on whether the current mid value represents a possible solution. The search continues until low becomes greater than high.

Step 14: Within each iteration, calculate the mid value as the average of low and high to search for the optimal allocation.

Step 15: Check if the current mid value represents a possible solution by calling the isPossibleSolution function.

Step 16: If the allocation is possible, update the ans with the current mid value and search for smaller allocations by updating high to mid - 1.

Step 17: If the allocation is not possible, search for larger allocations by updating low to mid + 1.

Step 18: Once the binary search is complete, return the ans, which represents the minimum number of pages allocated to the student with the maximum allocation.

Step 19: In the main function, create a vector arr containing the given book pages {10, 20, 30, 40}.

Step 20: Call the allocateMinimumPages function with the book array, number of books (4), and number of students (2).

Step 21: Print the returned value, which represents the minimum number of pages allocated to the student with the maximum allocation.

Time Complexity: **The time complexity of this solution is O(N*log(S)), where N is the number of books and S is the sum of all book pages. The binary search algorithm iterates log(S) times, and in each iteration, the isPossibleSolution function is called, which takes O(N) time to iterate through the books.**

Space Complexity: **The space complexity is O(1) since the algorithm uses a constant amount of extra space for variables.**