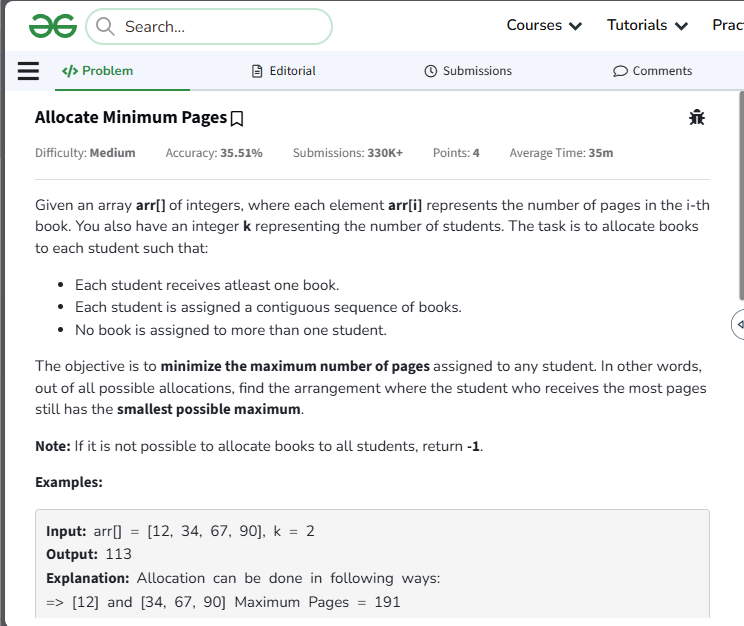
**Binary Search Book Allocation Problem**



#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

using namespace std;

class Solution {

public:

// Helper function to check if allocation is possible with given max pages

bool isPossible(vector<int>& arr, int mid, int k) {

int pages = 0, students = 1;

for (int x : arr) {

if (pages + x <= mid) {

pages += x;

} else {

pages = x;

students++;

}

}

return students <= k;

}

// Main function to find the minimum possible maximum pages

int findPages(vector<int>& arr, int k) {

if (arr.size() < k) return -1;

int low = \*max\_element(arr.begin(), arr.end()); // max single book

int high = accumulate(arr.begin(), arr.end(), 0); // sum of all books

int mid;

while (low <= high) {

mid = low + (high - low) / 2;

if (isPossible(arr, mid, k)) {

high = mid - 1;

} else {

low = mid + 1;

}

}

return low; // minimum of the maximum pages

}

};

// Example usage

int main() {

Solution sol;

vector<int> books = {12, 34, 67, 90};

int students = 2;

int result = sol.findPages(books, students);

cout << "Minimum maximum pages: " << result << endl;

return 0;

}

**📝 Notes for Documentation**

* **Problem**: Allocate books to k students such that each student gets a contiguous block and the maximum pages assigned to any student is minimized.
* **Approach**: Binary search on the answer space.
* **Time Complexity**: ( O(n \log \text{sum}) ), where n is the number of books and sum is the total pages.
* **Edge Case**: If k > n, return -1.