HOMEWORK DAY: 13.2

01 October 2023 14:02

HW 05: Book Allocation Problem

Problem Statement

*You are given an array **'pages'** of integer numbers.*
*In this array, the **'pages[i]'** represents the number of pages in the **'i-th'** book.*
*There are **'m'** number of students, and the task is to allocate all the books to the students.*

Allocate books in a way such that:

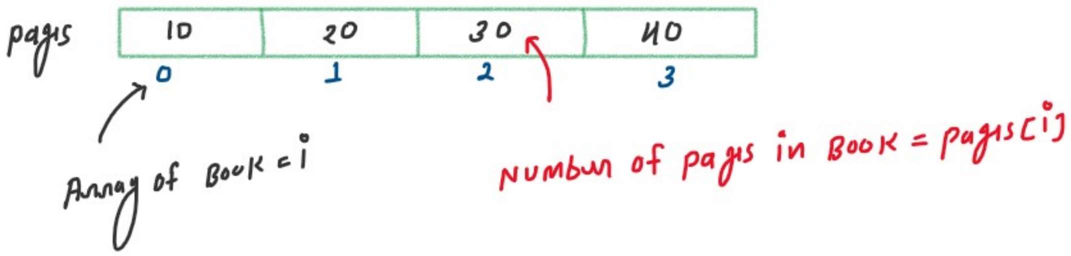
- *Each student gets at least one book.*
- *Each book should be allocated to a student.*
- *Book allocation should be in a contiguous manner.* → Monotonic ORDER 1 2 3 ---- 10 ---∞

*You have to allocate the books to **'m' students** such that the **maximum number of pages assigned to a student is minimum.***

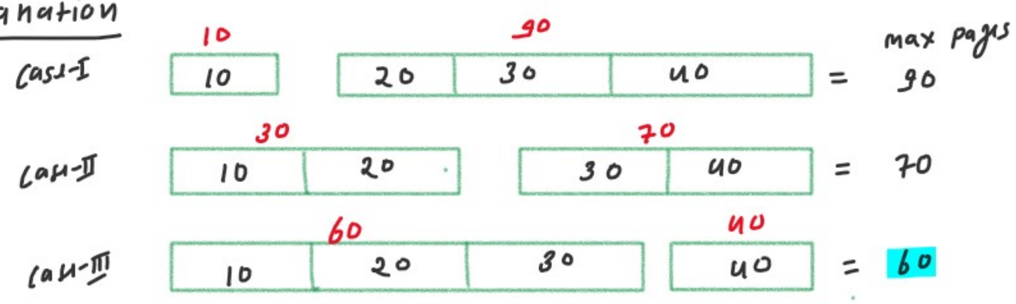
Example 01:

Input: pages[] = { 10,20,30,40}, n=4 and m = 2
Output: 60

Number of books(n) = 4
Number of students(m) = 2
Number of pages in a book(i) = pages[i]
The minimum of the maximum number of pages assigned = min{90,70,60} = 60

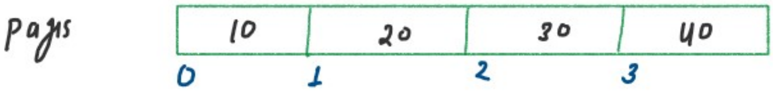


Explanation



Output { Minimum pages of maximum pages = 60 }

OPTIMAL APPROACH: *Define search space and predicate function* Highest
Step 01: *find total sum of array to create search space's end point (Maximum number of pages)*
Step 02: *apply binary search on search space BinarySearch()*
Step 03: *create predicate function isPossibbleSolution()*

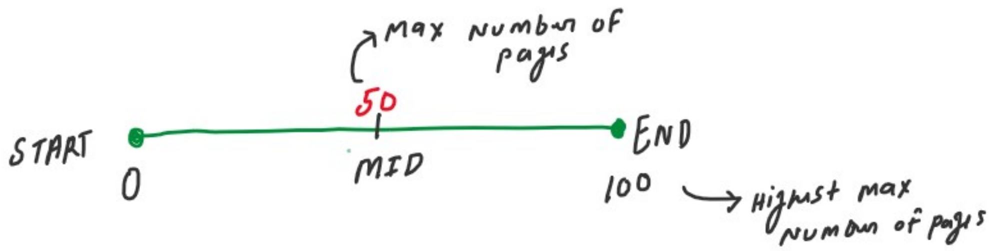


STEP:01 SUM = $\boxed{100}$ 10+20+30+40 = 100

↗Max Number of pages

10+20+30+...

STEP:02



Iter:01
Sum = 100
Start = 0
End = 100
mid = 50
M = 2
Ans = -1

$\text{mid} = \frac{0+100}{2}$

$= 50$

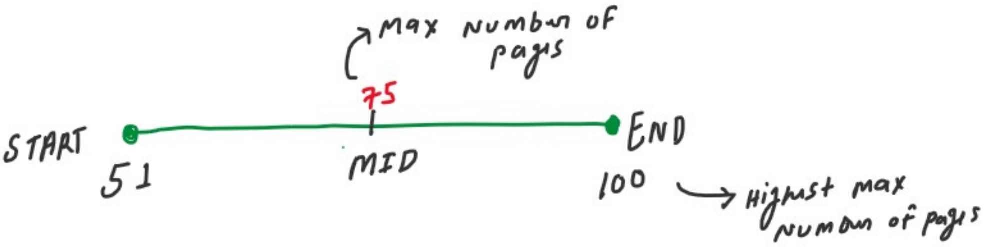
① mid <= sum
→ Find minimum numbers of pages of max Number of pages till Case <= M

STEP:3

Case	Psum <= mid	
I →	10 + 20 + 30	= 30 ✓
II →	30 + 40	= 30 ✓
III →	Ruk ja... case <= M ✗	

NO possible
→ Start = mid + 1

STEP:02



Iter:02
Sum = 100
Start = 51
End = 100
mid = 75
M = 2
Ans = -1

$\text{mid} = \frac{51+100}{2}$

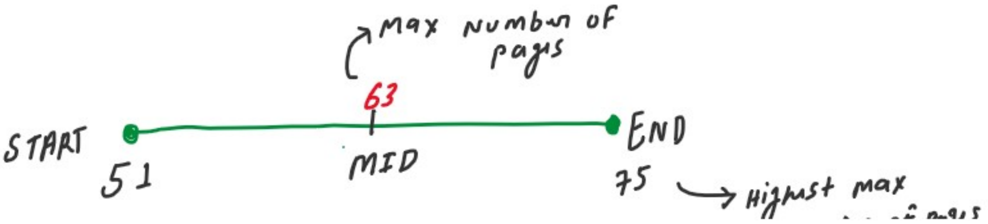
$= 75$

① mid <= sum
→ Find minimum numbers of pages of max Number of pages till Case <= M

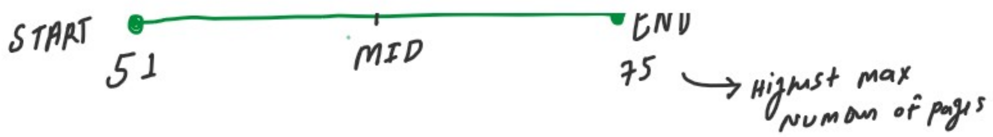
Case	Psum <= mid	
I →	10 + 20 + 30 + 40	= 60
II →	40	= 40

Possible sol.n
→ ANS = mid
= 75
End = mid - 1

STEP:02



Iter:03



Itern:03

Sum = 100

Start = 51

End = 74

mid = 63

M = 2

Ans = 75

$$\text{Mid} = \frac{51+75}{2} = 63$$

① mid <= sum

→ Find minimum Numbers of pages of max Number of pages till Case <= M

Case	Psum <= mid	
I →	$10+20+30+\boxed{40}$	= 60
II →	40	= 40

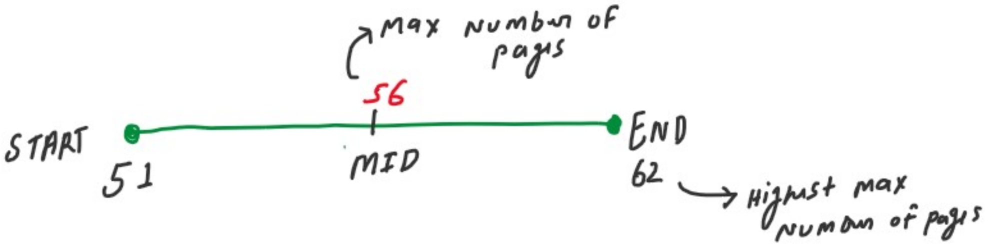
Possible sol.n

→ ANS = mid = 63

End = mid-1

:

STEP:02



Itern:04

Sum = 100

Start = 51

End = 62

mid = 56

M = 2

Ans = 63

$$\text{Mid} = \frac{51+62}{2} = 56$$

① mid <= sum

→ Find minimum Numbers of pages of max Number of pages till Case <= M

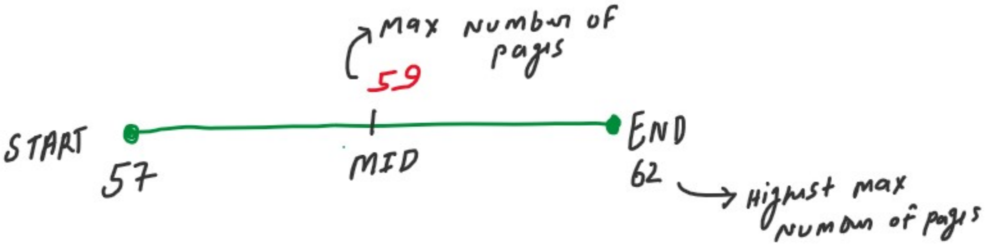
Case	Psum <= mid	
I →	$10+20+\boxed{30}$	= 30
II →	$30+\boxed{40}$	= 30
III →	END	

NO Possible sol.n

→ start = mid+1

:

STEP:02



Itern:05

Sum = 100

Start = 57

End = 62

mid = 59

$$\text{Mid} = \frac{57+62}{2}$$

59

① mid <= sum

minimum numbers of pages of

End = 62
mid = 59
M = 2
Ans = 63

$\text{mid} = \frac{\quad}{2}$

$= 59$

① mid <= sum

↳ Find minimum numbers of pages of max number of pages till Case <= M

Case	Psum <= mid	
I →	10 + 20 + 30	= 30
II →	30 + 40	= 30
III →	END	

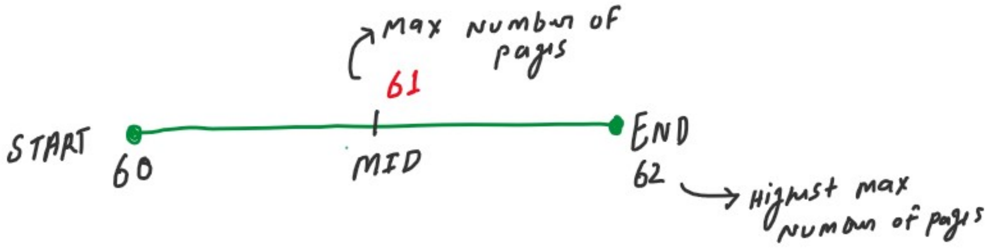
NO Possible sol.n

↳ Start = mid + 1

.

:

STEP: 02



Iter: 6
Sum = 100
Start = 60
End = 62
mid = 61
M = 2
Ans = 63

$\text{mid} = \frac{60+62}{2}$

$= 61$

① mid <= sum

↳ Find minimum numbers of pages of max number of pages till Case <= M

Case	Psum <= mid	
I →	10 + 20 + 30	= 60
II →	40	= 40

Possible sol.n

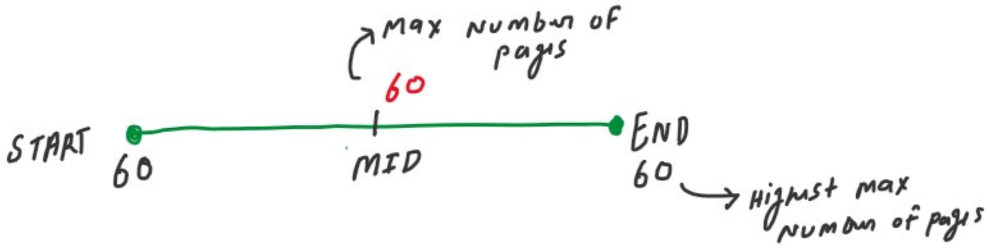
↳ Ans = mid
= 61

End = mid - 1

.

:

STEP: 02



Iter: 7
Sum = 100
Start = 60
End = 60
mid = 60
M = 2
Ans = 61

$\text{mid} = \frac{60+60}{2}$

$= 60$

① mid <= sum

↳ Find minimum numbers of pages of max number of pages till Case <= M

M = 2

Ans = 61

max Number of pages till case <= M

Case	Psum <= mid	
I →	10 + 20 + 30	= 60
II →	40	= 40

Possible sol.ⁿ

↳ ANS = mid
= 61

End = mid - 1

START
60
END
59

Iteration 8

Start = 60

End = 59

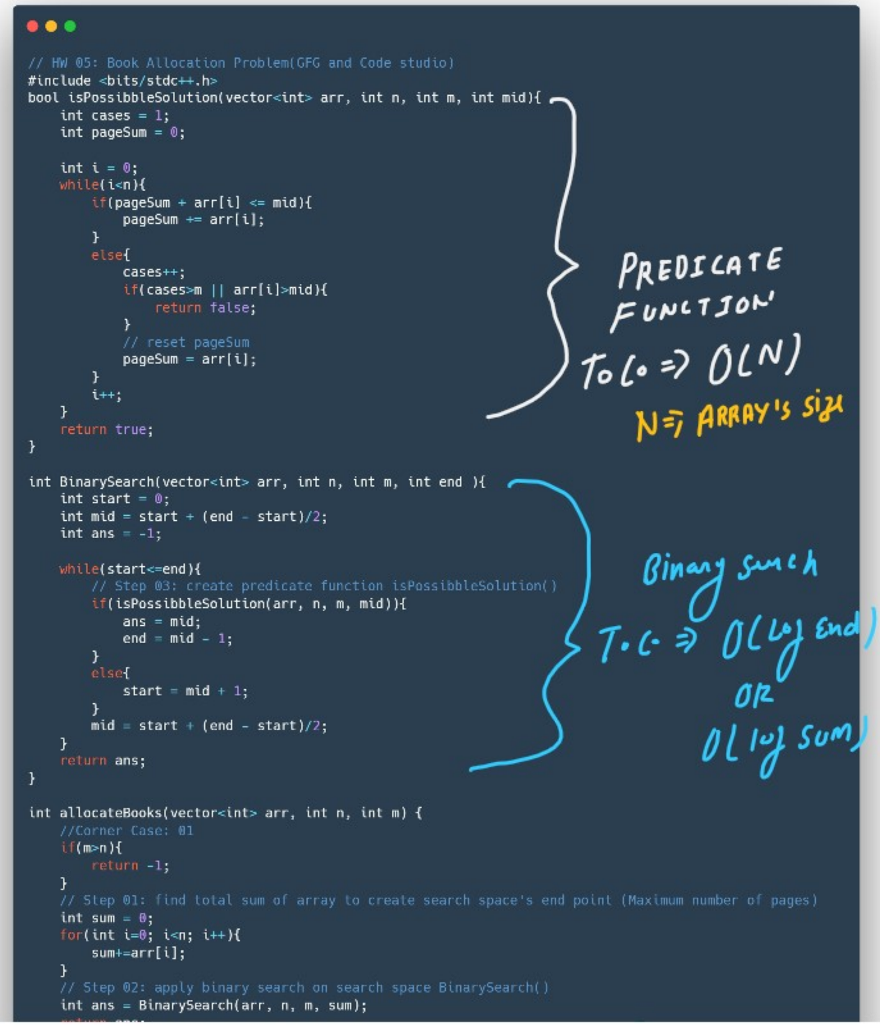
ANS = 60

Ruk joo.... (Start > END)

Final Output

Minimum Number of maximum number of pages

⇒ 60



```
// HW 05: Book Allocation Problem(GFG and Code studio)
#include <bits/stdc++.h>
bool isPossibbleSolution(vector<int> arr, int n, int m, int mid){
    int cases = 1;
    int pageSum = 0;

    int i = 0;
    while(i<n){
        if(pageSum + arr[i] <= mid){
            pageSum += arr[i];
        }
        else{
            cases++;
            if(cases>m || arr[i]>mid){
                return false;
            }
            // reset pageSum
            pageSum = arr[i];
        }
        i++;
    }
    return true;
}

int BinarySearch(vector<int> arr, int n, int m, int end ){
    int start = 0;
    int mid = start + (end - start)/2;
    int ans = -1;

    while(start<=end){
        // Step 03: create predicate function isPossibbleSolution()
        if(isPossibbleSolution(arr, n, m, mid)){
            ans = mid;
            end = mid - 1;
        }
        else{
            start = mid + 1;
        }
        mid = start + (end - start)/2;
    }
    return ans;
}

int allocateBooks(vector<int> arr, int n, int m) {
    //Corner Case: 01
    if(m>n){
        return -1;
    }
    // Step 01: find total sum of array to create search space's end point (Maximum number of pages)
    int sum = 0;
    for(int i=0; i<n; i++){
        sum+=arr[i];
    }
    // Step 02: apply binary search on search space BinarySearch()
    int ans = BinarySearch(arr, n, m, sum);
```

PREDICATE FUNCTION

T.C. ⇒ O(N)

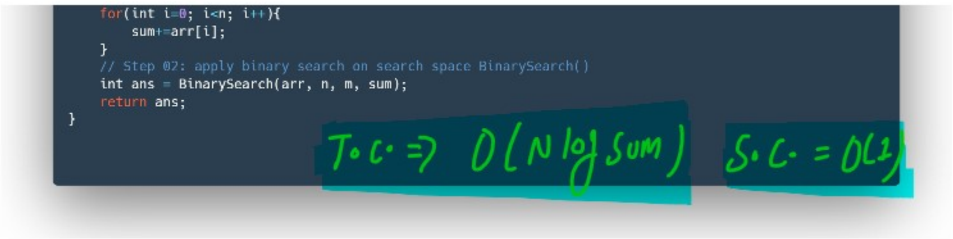
N ⇒ ARRAY's size

Binary search

T.C. ⇒ O(log End)

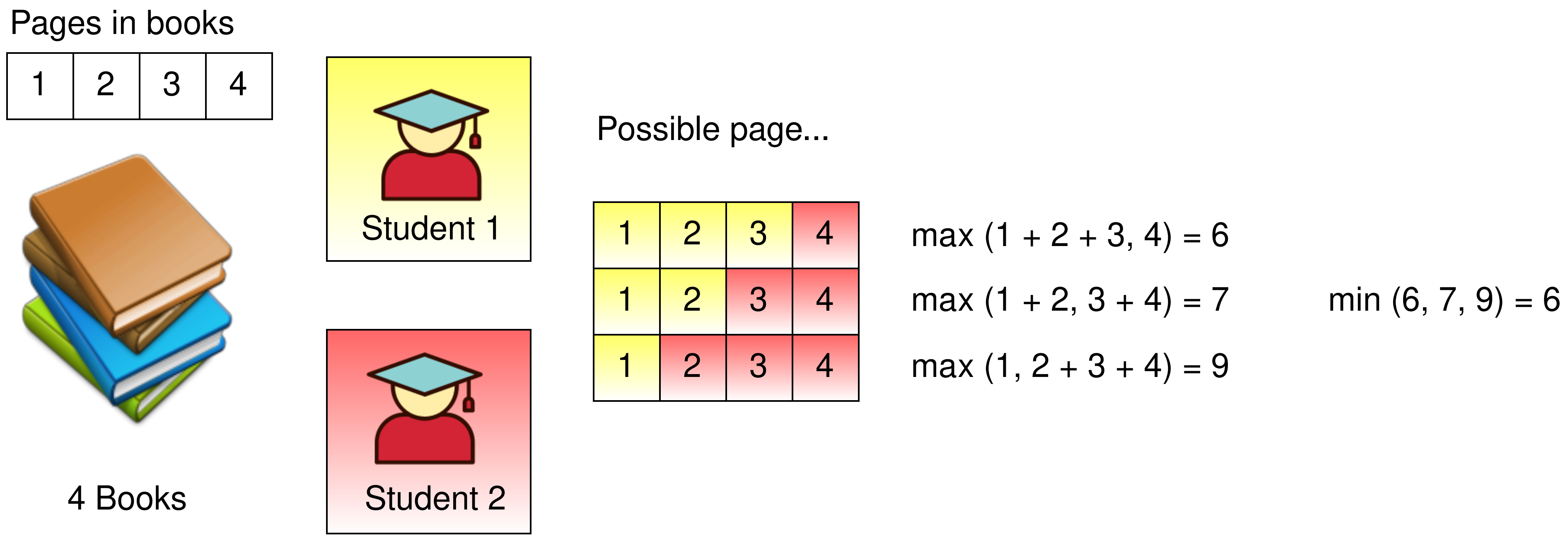
OR

O(log sum)



```
    for(int i=0; i<n; i++){
        sum+=arr[i];
    }
    // Step 02: apply binary search on search space BinarySearch()
    int ans = BinarySearch(arr, n, m, sum);
    return ans;
}
```

T.C. => O(N log sum)   S.C. = O(1)



Pages in books

1	2	3	4

4 Books

Student 1

Student 2

Possible page...

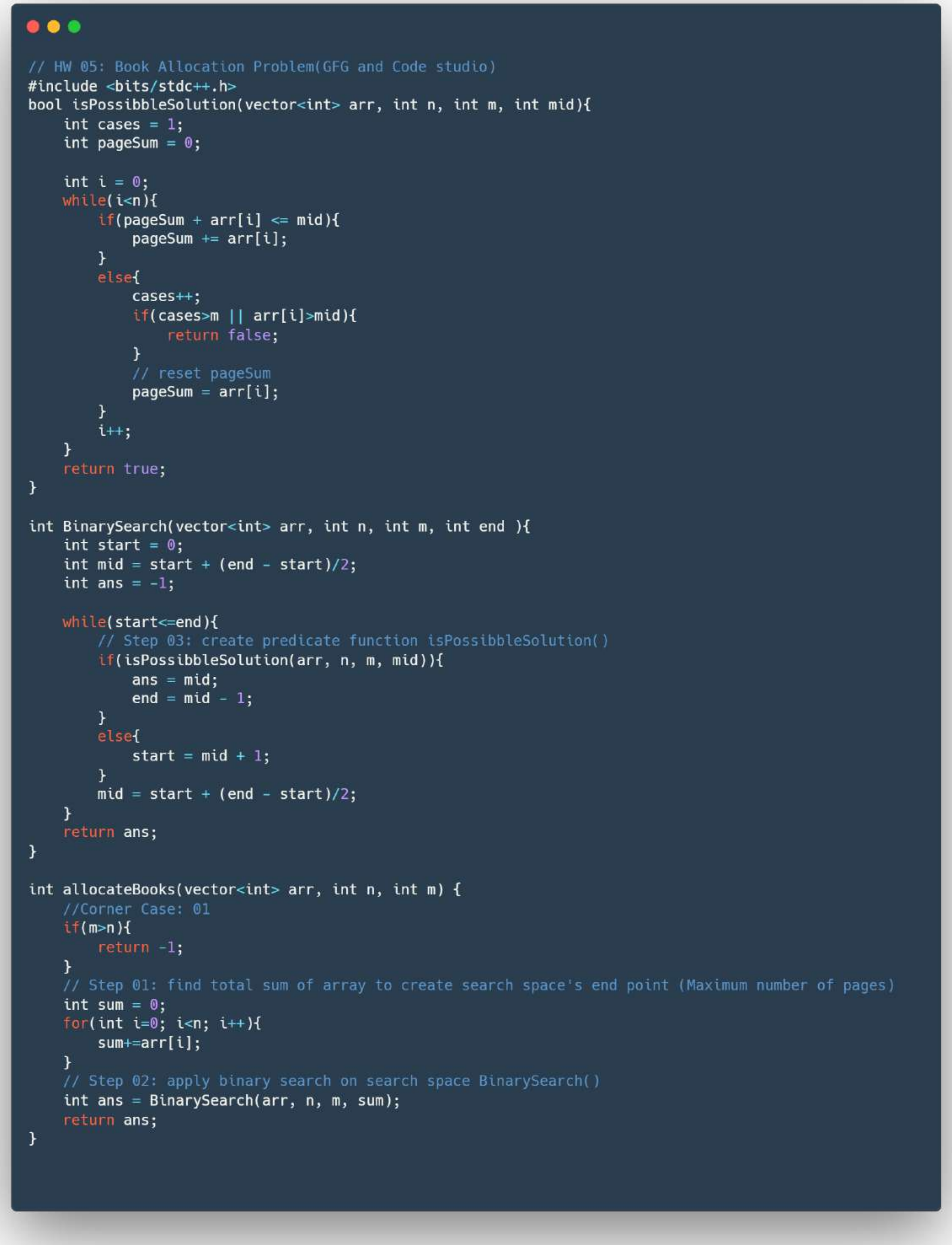
1	2	3	4
1	2	3	4
1	2	3	4

max (1 + 2 + 3, 4) = 6

max (1 + 2, 3 + 4) = 7

max (1, 2 + 3 + 4) = 9

min (6, 7, 9) = 6



```cpp
// HW 05: Book Allocation Problem(GFG and Code studio)
#include <bits/stdc++.h>
bool isPossibbleSolution(vector<int> arr, int n, int m, int mid){
    int cases = 1;
    int pageSum = 0;

    int i = 0;
    while(i<n){
        if(pageSum + arr[i] <= mid){
            pageSum += arr[i];
        }
        else{
            cases++;
            if(cases>m || arr[i]>mid){
                return false;
            }
            // reset pageSum
            pageSum = arr[i];
        }
        i++;
    }
    return true;
}

int BinarySearch(vector<int> arr, int n, int m, int end ){
    int start = 0;
    int mid = start + (end - start)/2;
    int ans = -1;

    while(start<=end){
        // Step 03: create predicate function isPossibbleSolution()
        if(isPossibbleSolution(arr, n, m, mid)){
            ans = mid;
            end = mid - 1;
        }
        else{
            start = mid + 1;
        }
        mid = start + (end - start)/2;
    }
    return ans;
}

int allocateBooks(vector<int> arr, int n, int m) {
    //Corner Case: 01
    if(m>n){
        return -1;
    }
    // Step 01: find total sum of array to create search space's end point (Maximum number of pages)
    int sum = 0;
    for(int i=0; i<n; i++){
        sum+=arr[i];
    }
    // Step 02: apply binary search on search space BinarySearch()
    int ans = BinarySearch(arr, n, m, sum);
    return ans;
}
```