```cpp
#include <bits/stdc++.h>
bool isPossible(vector<int> arr, int n, int m, int mid){
    int pageSum = 0;
    int studentCount = 1;

    for(int i = 0; i<n; i++){
        if(pageSum + arr[i] <= mid){
            pageSum += arr[i];
        }else{
            studentCount++;
            if(studentCount > m|| arr[i] > mid){
                return false;
            }
            pageSum = arr[i];
        }
    }
    return true;
}
int allocateBooks(vector<int> arr, int n, int m) {
    int s = 0;
    int sum = 0;
    int ans;

    for(int i = 0; i <n; i++){
        sum += arr[i];
    }
    int e = sum;
    int mid = s + ((e-s)>>2);

    while(s<=e){
        if(isPossible(arr, n, m, mid)){
            ans = mid;
            e = mid - 1;
        }else{
            s = mid + 1;
        }
        mid = s + ((e-s)>>2);
    }
    return ans;
}
```

Decoding Book Allocation Problem

Created	@January 12, 2023 11:01 PM
Class	
Type	
Materials	https://www.codingninjas.com/codestudio/problems/allocate-books_1090540?leftPanelTab=0
Reviewed	☐

This problem is best solved by binary search : O(N*log(N))

Objective:

A book will be allocated to exactly one student.

Each student has to be allocated at least one book.

Allotment should be in contiguous order, for example, A student cannot be allocated book 1 and book 3, skipping book 2.

Calculate and return the **minimum possible number**. Return -1 if a valid assignment is not possible.

Why Binary Search?

The game of SEARCH SPACE

Let's take an example: {`12 34 67 90`} n(no. of books)= 4; m(number of students) = 2

```
there are three ways I can divide the given books
               STD 1   STD 2    MAX  MIN
12 | 34 67 90  12       191     191
```

```
12 34 | 67 90  46       157     157
12 34 67 | 90  113       90     113  113(ANS)
```

Binary search operates in a search space, so we have to establish a search space to search for the required number of books

Defining the initial search space.

Here the search space will be between the lowest possible value you can assign to the student (the book with the lowest number) and the maximum value you can assign is the sum of all the pages in the books(i.e you allocate all the books to a student).

How to use binary search and the search space to find the answer?

let’s go back to the example:

```
12 34 67 90

The search space here is:
low                                 high
12----------------------------------203
12----------------107-----------------203 mid = 107
now, 12 34 67 90
student 1 = 12 + 34
student 2 = 67
student 3 (not possible) = 90
```

so what happened here?

I found a middle element and let’s assume its my answer here 107:

now let’s start allocating books accordingly.

1. I allocated student 1 a 12-page book.
    a. 12 pages < 107 pages so,
    b. allocated a second book of 34 pages to student 1.

    c. still 12 + 34 = 46 < 107 so,
    d. allocated the third book of 67 pages to student 1 here we encounter a problem
    e. as 12 + 34 + 67 > 107
2. allocating student 2 with a 67-page book now
    a. as 67 < 107 so,
    b. allocating 90-page book to 67 and again 67 + 90 > 107
3. allocating student 3 with 90 pages book but WAIT we had only two students right?

Now we know that 107 and any number lesser than 107 can't be the answer as 107 is too

low a BARRIER. so we will now shorten our SEARCH SPACE from 107 + 1 to 203

```
The new search space is;
low                                  high
108 ---------------------------------203
108 --------------155-----------------203
```

Now repeating the same process:

```
12 34 67 90
student 1: 12 + 34 + 67 < 155
student 2: 90  < 153

so 155 can be an answer so does 156. but there is a problem
```

The problem:

The problem here is the answer 155 is not the minimum most value that can return us the answer because even 140 or 130 can also be a probable answer here as 12 + 34 + 67 < 140 or 130 as well

So to find the minimum most value we have to shorten the search space even more:

so we will now bring the search space from max value to 153 - 1;

```
108 -------------------------155

Repeating this step a few more times:
108-------------131-------------155
12 34 67 90
student 1: 12 + 34 + 67 < 131
student 2: 90  < 131

108--------119---------130
12 34 67 90
student 1: 12 + 34 + 67 < 119
student 2: 90  < 119

108------113-------118
12 34 67 90
student 1: 12 + 34 + 67 = 113
student 2: 90  < 113

108-----110-----112
12 34 67 90
student 1: 12 + 34
student 2: 67
student 3: 90 (NOT Possible)

110------111-------112(not possible)

111------111---------112(not possible)
112 -----112---------112(not possible)

so now the high will be 112 and low will be 113
113 ---------------112

this is not possible
113 is the answer
```

so 113 was the last value where we could properly divide the books. hence the minimum number of pages we can allot maximum to a student is 113.(Ans)

Explaining the code:

```
#include <bits/stdc++.h>
bool isPossible(vector<int> arr, int n, int m, int mid){
    int pageSum = 0;
```

```
    int studentCount = 1;
    for(int i = 0; i<n; i++){
        if(pageSum + arr[i] <= mid){
            pageSum += arr[i];
        }else{
            studentCount++;
            if(studentCount > m|| arr[i] > mid){
                return false;
            }
            pageSum = arr[i];
        }
    }
    return true;
}
int allocateBooks(vector<int> arr, int n, int m) {
    int low = 0;
    int sum = 0;
    int ans;

    for(int i = 0; i <n; i++){
        sum += arr[i];
    }
    int high = sum;
    int mid = low + ((high-low)>>2);

    while(low<=high){
        if(isPossible(arr, n, m, mid)){
            ans = mid;
            high = mid - 1;
        }else{
            low = mid + 1;
        }
        mid = low + ((high-low)>>2);
    }
    return ans;
}
```

Using binary search to implement the logic;

1. initializing low as 0;
2. finding sum of all elements in the array and assigning it to high;
3. finding mid using the formulae
4. making a isPossible function to see whether its possible to allot the books or not
5. if possible assigning ans as mid and bringing shortening the search space by lowering the high value to mid - 1;
6. else bringing the low value to mid + 1;

7. returning the ans.

IsPossible function.

This boolean functions job is to determine whether it is possible to allot the books to all the

students and not exceeding the values of number of books or students

whenever we call the function with a new mid it essentially performs this

```
108 -------------------------155

Repeating this step a few more times:
mid 1 108-------------131-------------155
      12 34 67 90
      student 1: 12 + 34 + 67 < 131
      student 2: 90  < 131

mid 2 108--------119---------130
      12 34 67 90
      student 1: 12 + 34 + 67 < 119
      student 2: 90  < 119

mid 3 108------113-------118
      12 34 67 90
      student 1: 12 + 34 + 67 = 113
      student 2: 90  < 113

mid 4 108-----110-----112
      12 34 67 90
      student 1: 12 + 34
      student 2: 67
      student 3: 90 (NOT Possible)
```

and returns a bool value.