16. Allocate minimum number of pages

Hard Accuracy: 42.77% Submissions: 8841 Points: 8

You are given **N** number of books. Every i^{th} book has A_i number of pages.

You have to allocate books to **M** number of students. There can be many ways or permutations to do so. In each permutation, one of the **M** students will be allocated the maximum number of pages. Out of all these permutations, the task is to find that particular permutation in which the maximum number of pages allocated to a student is minimum of those in all the other permutations and print this minimum value.

Each book will be allocated to exactly one student. Each student has to be allocated at least one book.

Note: Return **-1** if a valid assignment is not possible, and **allotment should be in contiguous order (see the explanation for better understanding).**

Example 1:

```
Input:
N = 4
A[] = {12,34,67,90}
M = 2
Output:
113
Explanation:
Allocation can be done in following ways:
{12} and {34, 67, 90} Maximum Pages = 191
{12, 34} and {67, 90} Maximum Pages = 157
{12, 34, 67} and {90}  Maximum Pages =113
Therefore, the minimum of these cases is
113, which is selected as the output.
```

Expected Time Complexity: O(NlogN)
Expected Auxilliary Space: O(1)

Given an array corresponding books for every index and no. of pages for arr[i].

We are given 'k' students, and we need to assign the books such that the max. pages read by any student is min. of all the permutations possible.

Also continuous books could only be allocated for a student.

① Naive Recursive sol^n — exponential

```
int minPages(int arr[], int n, int k)
{
    if (k == 1)
        return sum(arr, 0, n-1);
    if (n == 1)
        return arr[0];

    int res = INF;

    for (int i = 1; i < n; i++)
        res = min(res, max(minPages(arr, i, k-1),
                           sum(arr, i, n-1)
                          ));

    return res;
}
int sum(int arr[], int b, int e)
{   int s = 0;
    for(int i = b; i <= e; i++)
        s += arr[i];
    return s;
}
```


Base cases

Only one student

Just one book.

Calculate the sum for a given part.

```
int res = INF;
for (int i=1; i<n; i++)
    res = min(res, max(minPages(arr, i, k-1),
                       sum(arr, i, n-1)
                   ));
return res;
```


Rec call.

sum of partition.

Calculating the sum of all the arrays that could be formed.

$\{\, 1, 2, 3, 4, 5, 6, 7 \,\}$

allocated these to our first selection.

And we'll loop to check for all the sub partitions and every check decreasing the element by 1.

Allocate Minimum Pages (Binary Search)

$[10 \quad 20 \quad 10, \quad 30] \quad K = 2$

Sum of all Pages = $10+20+10+30 = 70$

Answer will be in range $[30, 70]$

$x = \frac{30+70}{2} = 50$, res $= 50$

$x = \frac{30+49}{2} = 39$

Binary Search Based Solution :

70 → all the books are just read by one single student

30 → however can't have a value less than this coz max(arr) is that one book with most no. of pages and that has to be read and would be a minimum no matter what.

we then calculate feasible solution, as by applying binary search for $x = (30+70)/2$

→ we will calculate how many students are required

① After finding out a feasible solution i.e., no. of students required $<= K$,

② Therefore if this $l+h/2$ is a feasible solution, the right side would also be one, thus this side could be left now.

③ Pruning the search till we get to the range where it contains just one element.

converged and then went to left of 50

$[\underbrace{10 \quad 20}_{1} \quad \underbrace{10}_{2}, \quad \underbrace{30}_{3}] \qquad K = 2$

Sum of all Pages = $10+20+10+30 = 70$

Answer will be in range $[30, 70]$

$x = \frac{30+70}{2} = 50, \quad res = 50$

$x = \frac{30+49}{2} = 39$

for this in the given array no. of students req

$[\underbrace{10, 20,}_{1} \ \underbrace{10}_{2}, \underbrace{30}_{3}]$

3

∴ can't be covered

here $3 > K$, go to right side

Allocate Minimum Pages (Binary Search)

[10 20 10, 30]   K = 2

Sum of all Pages = 10 + 20 + 10 + 30 = 70

Answer will be in range [30, 70]

$x = \frac{30+70}{2} = 50$, res = 50

$x = \frac{30+49}{2} = 39$, ✗

$x = \frac{40+49}{2} = 44$, res = 44

$\Rightarrow$

$x = \frac{40+43}{2} = 41$, res = 41

$x = \frac{40+40}{2} = 40$, res = 40

We keep on updating the res and recalculating the range till we get to the point that lower half and upper cross each other.

```
int minPages(int arr[], int n, int k)
{   int sum = 0, mx = 0;
    for (int i = 0; i < n; i++)
    {   sum += arr[i];
        mx = max(mx, arr[i]);
    }
    int low = mx, high = sum, res = 0;
    While (low <= high)
    {
        int mid = (low + high)/2;
        if (isFeasible(arr, n, k, mid))
        {   res = mid;       // If feasible, go to
            high = mid-1;    // the left half
        }
        else
            low = mid+1;     // Else go the right half
    }
    return res;
}
```

① Calculating sum of arr element and also max present.

② Our range thus → [max, sum]

③ We apply binary search over this.

Simple implementation of binary search, deciding to go to the left or right half,

If it is feasible hence it could be checked for optimization, and if not then it need be increased.

Is feasible fn.

```
bool isFeasible(int arr[], int n,
                       int k, int ans)
{  int req = 1, sum = 0;
   for(int i=0; i<n; i++)
   {  if(sum + arr[i] > ans)
      {
         req++;
         sum = arr[i];
      }
      else
         sum += arr[i];
   }
   return (req <= k);
}
```

If inclusion of next book would disqualify the given condition.

Then we'd require a new student.

New student's data.

Check condition for feasibility.