Allocate minimum number of pages

Approach :

The allocation can be made only in contiguous manner. We need to find the maximum of minimum pages that can be allotted for example

student 1- 12, student 2- 40 ⇒ min =12

student 1 - 10, student 2- 50 ⇒min=10

max=12

So we can make partition at each point to see what allocation is maximum.

We can use recursion but the complexity will be exponential. Using binary search for reducing complexity.

We have n students and every student needs to be allocated 1 book so we need to keep a check on this.

Initially we have to find the maximum so we allot the maximum pages book to student1 and maximum allotment can be summation of all pages of the book, so the search space will be between these two points.

We find a mid and using isPossible function check whether the pages =mid can be allocated ton students being mid as maximum, if yes then we move towards right to find a still greater number is possible or not and keep storing the last allocation in res.

isPossible counts the student when the allocation exceeds mid page a new students is added. If at end the count of student is equal to actual students then we move to right to find for maximum but if it is not possible then we move left because the students are greater than maximum pages so we need to reduce pages to accommodate   all students.

```
#include <bits/stdc++.h>
bool isPossible(vector<int> time,long long mid,int n)
{
  long long sum=0;
  int cnt=1;
  for(int i=0;i<time.size();i++)
  {
    sum+=time[i];
    if(sum>mid)
    {
      cnt++;
      sum=time[i];
    }
  }
  if(cnt<=n)
    return true;
  return false;
}
long long ayushGivesNinjatest(int n, int m, vector<int> time)
{
  // Write your code here.
  long long high=0,low=INT_MIN;
  for(int i=0;i<m;i++)
  {
    high+=time[i];
    low=max(1LL*time[i],low);
  }
  while(low<=high)
  {
    long long mid=low+(high-low)/2;
    if(isPossible(time,mid,n))
    {
      high=mid-1;
    }
    else
    {
      low=mid+1;
    }
  }
  return low;
}
```