1283. Find the Smallest Divisor Given a Threshold

Medium | Topics | Companies | Hint

Given an array of integers `nums` and an integer `threshold`, we will choose a positive integer `divisor`, divide all the array by it, and sum the division's result. Find the **smallest** `divisor` such that the result mentioned above is less than or equal to `threshold`.

Each result of the division is rounded to the nearest integer greater than or equal to that element. (For example: `7/3 = 3` and `10/2 = 5`).

The test cases are generated so that there will be an answer.

Same as # 1011 #875

Example 1:

```
Input: nums = [1,2,5,9], threshold = 6
Output: 5
Explanation: We can get a sum to 17 (1+2+5+9) if the divisor is 1.
If the divisor is 4 we can get a sum of 7 (1+1+2+3) and if the divisor is 5
the sum will be 5 (1+1+1+2).
```

Example 2:

```
Input: nums = [44,22,33,11,1], threshold = 5
Output: 44
```

Constraints:

- `1 <= nums.length <= 5 * 10^4`
- `1 <= nums[i] <= 10^6`
- `nums.length <= threshold <= 10^6`

Approach 1: Brute force

Start checking from d = 1 until we get our answer.

```
d = 1
while (1)
{
    S = find Sum with divisor as d
```

```
if ( S ≤ threshold)
    return d
else
    d++
}
```

TC: $O(m\,n)$ — m: max num in given array

Approach 2: do binary search instead of linear search

```
class Solution {
public:
    int check(vector<int> &nums,int d){
        int sum=0;
        for(auto i:nums)
            sum=sum+ceil((double)i/d);
        return sum;
    }
    int binarysearch(vector<int> &nums,int l,int r,int t){
        int ans=INT_MAX;

        while(l<=r){
            int m=r-(r-l)/2;
            int s=check(nums,m);
            if(s<=t){
                ans=min(ans,m);
                r=m-1;
            }
            else l=m+1;
        }
        return ans;
    }
    int smallestDivisor(vector<int>& nums, int threshold) {
        int maxnum=INT_MIN;

        for(auto i:nums) maxnum=max(maxnum,i);

        return binarysearch(nums,1,maxnum,threshold);
    }
};
```

↳ becoz the max divisor

can be the largest num

$T(n)$: $O(n \log m)$

m: $\max(nums[i])$