Minimum no of days to make M bouquets

bloomDay = [7 7 7 7 13 11 12 7]

↓ Flowers bloom on 7th day

m = 2 → no of bouquets

k = 3 → adjacent flowers required

Here we need to find min no of days to make m bouquets for that

Condition is 1 bouquet contains K flowers and they must be adjacent

take 13 days

[7 7 7 7 13 11 12 7]

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

1 1

↳ will get 2 bouquets but we want min

we take max element in array as high

[7 7 7 7 13 11 12 7]

✓ ✓ ✓ ✓ ✗ ✓

4/3 1 1/3 0

1+0 = 1

1 ≠ 2

we take min element in array as low

```
public boolean solve(int[] a, int mid, int m, int n)
{
    int c=0;
    int noofboy=0;
    for(int i=0;i<n;i++)
    {
        if(a[i]<=mid){
            c=c+1
        }else{
            noofboy+=c/k;
            c=0
        }
    }
    noofboy+=c/k;
    if(noofboy>=m) return true;
    return false;
}
```

low = 7, mid = 10, high = 13

10 not possible

[7 7 7 7 13 11 12 7]

11 = 2x

low = mid+1

low = 11, mid = 12, high = 13

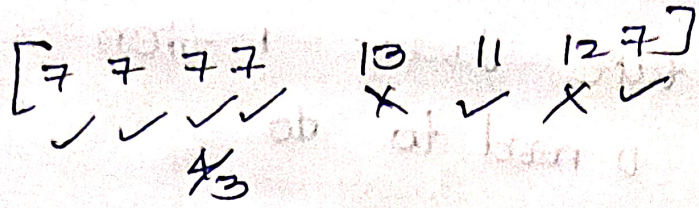
[7 7 7 7 13 11 12 7]

4/3 3/3 1+1=2

we need min right so we go left side

high = mid-1

low = 11 mid=11 high=11



1

so we go right side

low= mid+1

low=12 high=11

crosses ends

answer is in Low

```
public int Solve(int[] a, int m, int k){
    int low = min(a);
    int high = max(a);
    while(low<=high){
        int mid= low+(high-low)/2;
        if(solve(a, mid, m, k))
        {
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return low;
}
```

Find the smallest divisor given a threshold

arr[] = [1 2 5 9] threshold = 7

Here take divisor 1 then :

u need to do

$\frac{1}{1} + \frac{2}{1} + \frac{5}{1} + \frac{9}{1}$

17 <= 6 wrong

take max value or

Like 100

$\frac{1}{100} + \frac{2}{100} + \frac{5}{100} + \frac{9}{100}$

Here u will take ciel for each fractions

1+1+1+1

=4

Suppose take 9 max ele in array

it will also give 4

so take high as max element in array

low as 1

low mid high

1 5 9

[1 2 5 9]

$\frac{1}{5}$ $\frac{2}{5}$ $\frac{5}{5}$ $\frac{9}{5}$

1 1 1 2 = 5

5 <= 7 true

but we need smallest divisor so we go left side

low=1 mid=2 high=4

[1 2 5 9]

$\frac{1}{2}$ $\frac{2}{2}$ $\frac{5}{2}$ $\frac{9}{2}$

1 1 3 5

10<=7 false

So we move right side

low=mid+1

low=3 mid=3 high=4

[1 2 5 9]

$\frac{1}{3}$ $\frac{2}{3}$ $\frac{5}{3}$ $\frac{9}{3}$

1 1 2 3

7<=7 ~~false~~ true

we move right side

low=mid+1)x high=mid-1

high=2

low=4 mid=4 high=4

[1 2 5 9]

$\frac{1}{4}$ $\frac{2}{4}$ $\frac{5}{4}$ $\frac{9}{4}$

1 1 2 3

7<=6

low=3 high=2

we return low

bcuz high is always possible from starting so it ends with not possible

```
public boolean possible (int[] a, int mid, int threshold)
{
    int sum=0;
    for(int i=0; i<a.length; i++)
    {
        sum += Math.ceil (a[i]/mid)
    }
    if(sum<= threshold) return true;
    return false;
}
public int solve (int[] a, int n, int threshold)
{
    int low=1;
    int high= max(a);
    while (low<= high){
        int mid= low+ (high-low)/2;
        if ( possible (a, mid, threshold))
        {
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return low;
}
```

Find the least capacity to ship the package within D days

Given weights = [1,2,3,4,5,6,7,8,9,10] D = 5 days

to pack into ship
minimum capacity is max ele in array

we take max ele as Low

sum of all weights is taken as high

low = 10 high = sum(weights) 55

mid = 32

↓

it is possible but we need smallest capacity within 5 days

So we move left

high = mid - 1
= 31

low = 10 mid = 20 high = 31

$\frac{41}{2}$ 20

1st day 1,2,3,4,5 15
2nd day 6,7 15
3rd day 8
4th day 9
5th day 10

for 20 also possible

still need small capacity

we go left side

high = mid - 1

low = 10 high = 19

mid = 14

1st day → 1, 2, 3, 4

2nd day → 5, 6

3rd day → 7

4th day → 8

5th day → 9

6th day → 10

it exceeded 5 days

so we increase capacity

low = mid + 1

low = 15 high = 19 $\frac{34}{2}$

mid = 17

for 17 possible

high = mid - 1

low = 15 high = 16

mid = 15

possible

we go left side to check is there any small capacity but not these

```
public int findDays(int[] a, int mid)
{
    int load=0;
    int noofdays=0;
    for(int i=0;i<a.length;i++)
    {
        if(load+a[i]>mid){
            noofdays+=1;
            load=a[i];
        }
        else{
            load+=a[i];
        }
    }
    return noofdays;
}
public int solve(int[] w, int d)
{
    int low= max(w);
    int high= sum(w);
    while(low<=high){
        int mid= low+(high-low)/2
        int noofdays = findDays(w,mid);
        if(noofdays<=d){
            high=mid-1;
        }else{
            low=mid+1;
        }
    }
    return low;
}
```

Here we are calculating days for each capacity and we are taking only if it is less than D days. but they asked minimum so we need to store

Similarly in koko eating bananas you are calculating timehrs for no of bananas if totaltime is less than the given t hrs then we will store we need to find minimum.