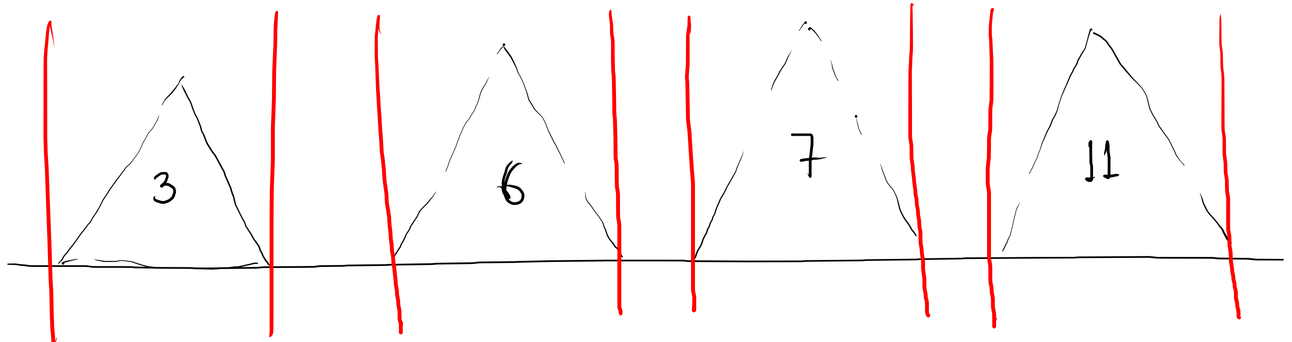
The banana challenge

h = 8

arr =

0	1	2	3
3	6	7	11



Imp points :-

- we have 'h' hours only
- n group of banana's with arr[i] banana's
- find speed of eating banana's
- Imp within 1 hour, we can choose only 1 pile.

$$arr = \begin{array}{|c|c|c|c|} \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} \\ \hline 3 & 6 & 7 & 11 \\ \hline \end{array}, \quad \underline{\underline{h = 8}}$$

si = min possible speed = 1

ei = max possible speed = 11 $\Rightarrow$ max(arr)

$$K = 1, 2, 3, \underset{\substack{\uparrow \\ si \\ \uparrow \\ mid}}{4}, \underset{\substack{\uparrow \\ ei}}{5}, 6, 7, 8, 9, 10, 11$$

mid = 6 , time = 1 + 1 + 2 + 2 = 6

mid = 3 , time = 1 + 2 + 3 + 4 = 10

mid = 4 , time = 1 + 2 + 2 + 3 = 8

→ check time with speed 'mid'

	0	1	2	3
arr =	3	6	7	11

mid = 3

3	6	7	11
$3/3 = \boxed{1}$	$6/3 = \boxed{2}$	$7/3 = \boxed{2}$	$11/3 = \boxed{3}$
3 % 3 != 0	6 % 3 != 0	7 % 3 != 0	11 % 3 != 0
		then $\boxed{t++;}$	then $\boxed{t++}$;

formula)

```
loop from start to end
    time = arr[i]/mid;
    if ( arr[i] % mid != 0) {
        time ++;
    }
```

(Saperate example)

Ex :- [10, 12, 14, 21, 3, 20]

$\boxed{mid = 10}$

↓

randomly

10/10 = 1
(10%10 != 0)
time = 1

12/10 = 1
(12%10 != 0)
time = 2

14/10 = 1
(14%10 != 0)
time = 2

21/10 = 2
(21%10 != 0)
time = 3

3/10 = 0
(3%10 != 0)
time = 1

20/10 = 2
(20%10 != 0)
time = 2

time = 1 + 2 + 2 + 3 + 1 + 2

code

input

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int hours = scn.nextInt();
    System.out.println(bananaChallenge(arr, n, hours));
}
```

B.S template

```java
public static int bananaChallenge(int[] arr, int n, int hours) {
    int si = 1;
    int ei = max(arr);
    while ( si <= ei ) {
        int mid = (si + ei) / 2; // speed of eating bananas
        if ( checkTime(arr, mid, hours) == true ) { // check if able to eat all banana in h hours
            ei = mid - 1;
        } else {
            si = mid + 1;
        }
    }
    return si;
}
```

Imp

```java
public static boolean checkTime(int[] arr, int speed, int totalTime) {
    int time = 0;
    for (int i = 0; i < arr.length; i++) {
        time += arr[i] / speed;
        if ( arr[i] % speed != 0 ) {
            time++;
        }
    }

    if (time <= totalTime) {
        return true;
    } else {
        return false;
    }
}
```

$$T.C = O(n + n\log(n))$$

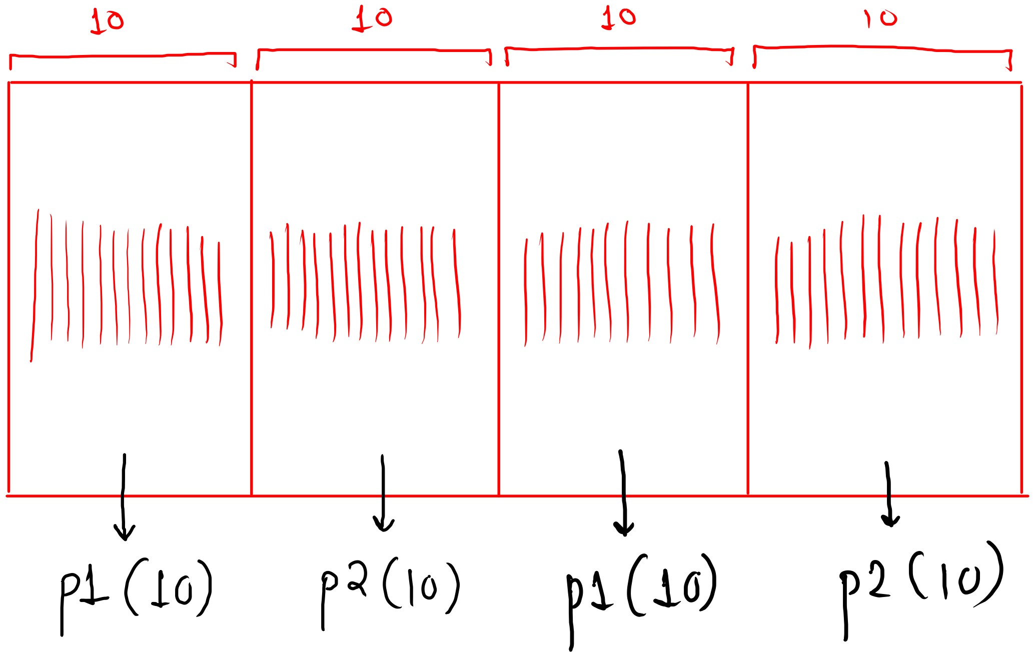
$$T.C = O(n\log(n))$$

find max

```java
public static int max(int[] arr) {
    int ans = 0;
    for (int i = 0; i < arr.length; i++) {
        ans = Math.max(ans, arr[i]);
    }
    return ans;
}
```

The painter

arr = [10 , 10 , 10 , 10] , K = painters available = 2



Total time = 20

Note:- only 1 painter can paint 1 series of board decide range

arr = [10, 20, 30, 40] , p = 4

p1 p2 p3 p4

time = 40

si = max(arr);

---

arr = [10, 20, 30, 40] , p = 1

p1 p1 p1 p1

time = 100

ei = sum(arr)

check function

$$[\underbrace{10,\ 10}_{p1},\ \underbrace{10,\ 10}_{p2}],\quad p = 2$$

time

10, 14 15 19 20 , 40

$\uparrow$ mid (14), $\uparrow$ si (15), $\uparrow$ ei (19)

mid = 20 , painter = 2

mid = 14 , painters = 4

1)
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int painters = scn.nextInt();
    System.out.println(thePainters(arr, n, painters));
}
```
2)
```java
public static int thePainters(int[] arr, int n, int painters) {
    int si = max(arr);
    int ei = sum(arr);
    while ( si <= ei ) {
        int mid = (si + ei) / 2; // time
        if ( check(arr, mid) > painters ) { // how many painter required in mid time
            si = mid + 1;
        } else {
            ei = mid - 1;
        }
    }
    return si;
}
```

3)
```java
public static int check(int[] arr, int time) {
    int painters = 1;
    int sum = 0;
    for (int i = 0; i < arr.length; i++) {
        sum += arr[i];
        if ( sum > time ) {
            painters++;
            sum = arr[i];
        }
    }
    return painters;
}
```

4)
```java
public static int max(int[] arr) {
    int ans = 0;
    for (int i = 0; i < arr.length; i++) {
        ans = Math.max(ans, arr[i]);
    }
    return ans;
}
```

5)
```java
public static int sum(int[] arr) {
    int ans = 0;
    for (int i = 0; i < arr.length; i++) {
        ans = ans + arr[i];
    }
    return ans;
}
```