Search insert position

$$\begin{array}{lcccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \text{arr} = [& 1, & 2, & 5, & 8, & 9, & 10\,] \end{array}$$

, ans for just greater element is 'i'

target = 4

$$\begin{array}{lccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \text{imaginary array} = [& 1, & 2, & \boxed{4,} & 5, & 8, & 9, & 10\,] \end{array}$$

Code

```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 target = scn.nextInt();
    System.out.println(searchInsert(arr, n, target));
}
public static int searchInsert(int[] arr, int n, int target) {
    int i = 0;
    int j = n - 1;
    while ( i <= j ) {
        int mid = (i + j) / 2;
        if (target == arr[mid]) {
            return mid;
        } else if ( target < arr[mid] ) {
            j = mid - 1;
        } else {
            i = mid + 1;
        }
    }
    return i;
}
```

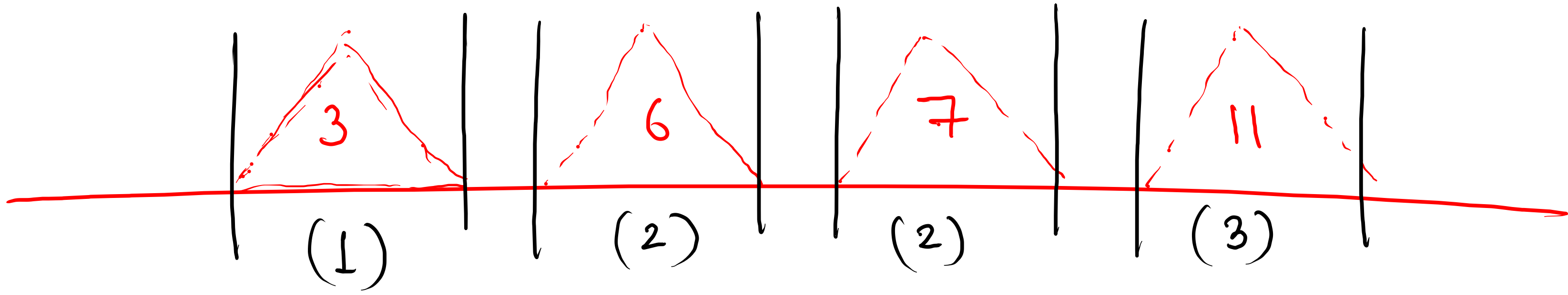
T.C = $O(\log N)$
S.C = $O(1)$

The banana challenge

n = 4

$$\text{arr} = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ 3, & 6, & 7, & 11 \end{array}$$

K = speed of eating bananas //4



totalTime = 8

guard will return in h = 8 hours

Imp point

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

psudo code

si = least possible speed = 1

ei = max possible speed = max(arr)

$$\begin{array}{c|c|c|c|c|}
 & 0 & 1 & 2 & 3 \\
\text{arr} = & 3 & 6 & 7 & 11 \\
 & (1) & (2) & (2) & (3)
\end{array}$$

h = 8

K = 1 2 3 4 5 6 7 8 9 10 11

$$\begin{array}{cc} \uparrow & \uparrow \\ j & i \\ & \uparrow \\ & \text{mid} \end{array}$$

also speed of eating banana's → mid = 6 total Time = 6

mid = 3 total Time = 10

$\boxed{\text{mid} = 4}$ total Time = 8

$\boxed{\text{ans} = 4}$

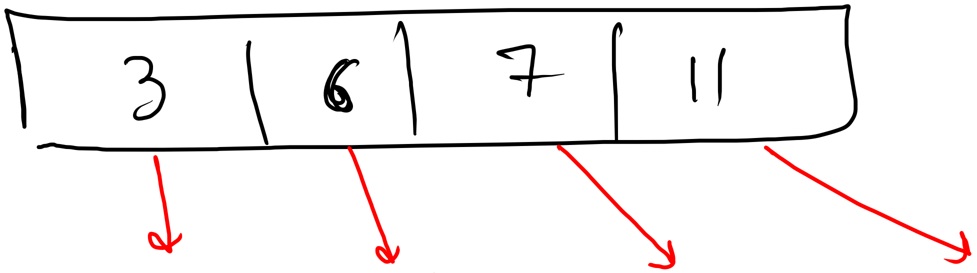
$\boxed{\text{ans} = i}$

check function

speed given = mid

time = h

find totalTime = ??



mid = 3
(speed)

$3/3=1$	$6/3=2$	$7/3=2$	$11/3=3$
$3\%3=0$	$6\%3=0$	$7\%3=1\neq 0$	$11\%3=2\neq 0$
(1)	(2)	(2+1)	(3+1)
		(3)	(4)

code

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 hours = scn.nextInt();
    System.out.println(kokoEatingBananas(arr, n, hours));
}
```

2) B.S Template
```java
public static int kokoEatingBananas(int[] arr, int n, int hours) {
    int i = 1;
    int j = max(arr);
    while ( i <= j ) {
        int mid = (i + j) / 2; // speed of eating bananas
        if ( check(mid, hours, arr) == true ) {
            j = mid - 1;
        } else {
            i = mid + 1;
        }
    }
    return i;
}
```

T.C = (N log N)

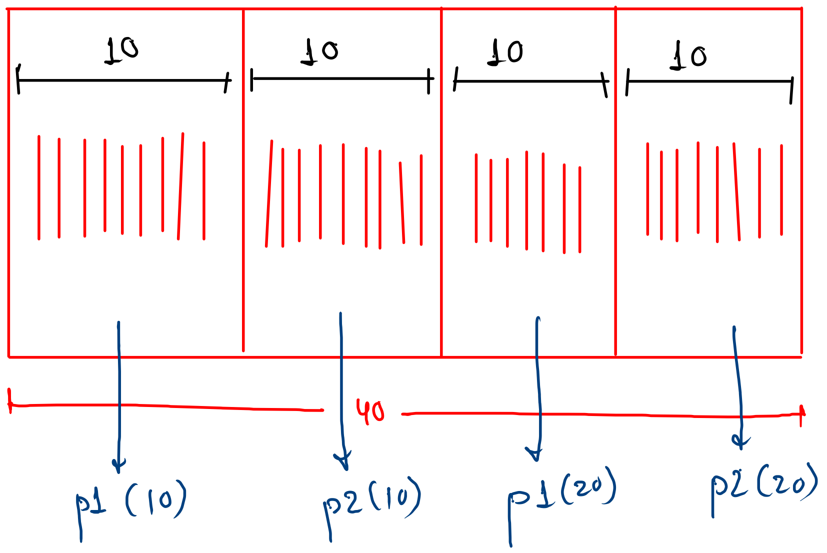
Imp
```java
public static boolean check(int speed, int time, int[] arr) {
    int totalTime = 0;
    for (int i = 0; i < arr.length; i++) {
        totalTime += arr[i] / speed;
        if ( arr[i] % speed != 0 ) {
            totalTime++;
        }
    }
    if ( totalTime > time ) {
        return false;
    } else {
        return true;
    }
}
```

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

The painter

$n = 4$

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



totalTime = 20 h

Note :- only 1 painter can paint 1 group of boards

range

$$arr\left[\underset{\substack{\uparrow\\ p1}}{\overset{\downarrow}{\boxed{10}}}, \underset{\substack{\uparrow\\ p2}}{\overset{\downarrow}{\boxed{20}}}, \underset{\substack{\uparrow\\ p3}}{\overset{\downarrow}{\boxed{30}}}, \underset{\substack{\uparrow\\ p4}}{\overset{\downarrow}{\boxed{40}}}\right], \quad \underline{\underline{p = 4}}$$

$$\underline{\underline{time = 40}}$$

$$\underline{\underline{s_i = \max(arr)}}$$

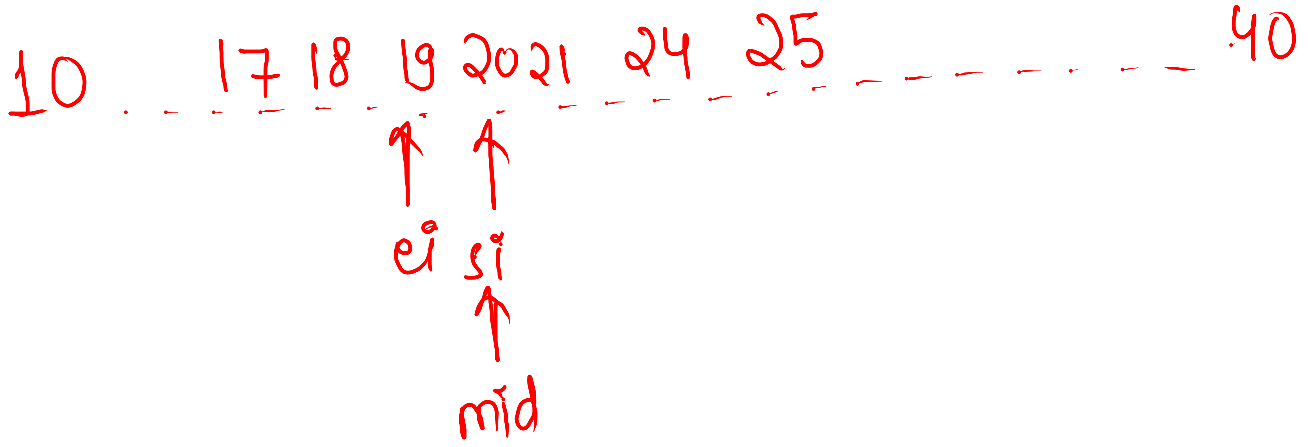
$$arr\left[\underset{\substack{\nearrow\\ 10}}{10}, \underset{\substack{\nearrow\\ 20}}{20}, \underset{\substack{\nearrow\\ 30}}{30}, \underset{\substack{\nearrow\\ 40}}{40}\right], \quad \underline{\underline{p = 1}}$$

$$\underline{\underline{time = 100}}$$

$$\underline{\underline{e_j = sum(arr)}}$$

$\text{arr}\ [\underbrace{10,\ 10}_{p1},\ \underbrace{10,\ 10}_{p2}]$ painters = 2

time



(time) taken by each painter

mid = 25 , painters = 2

mid = 17 , painters = 4

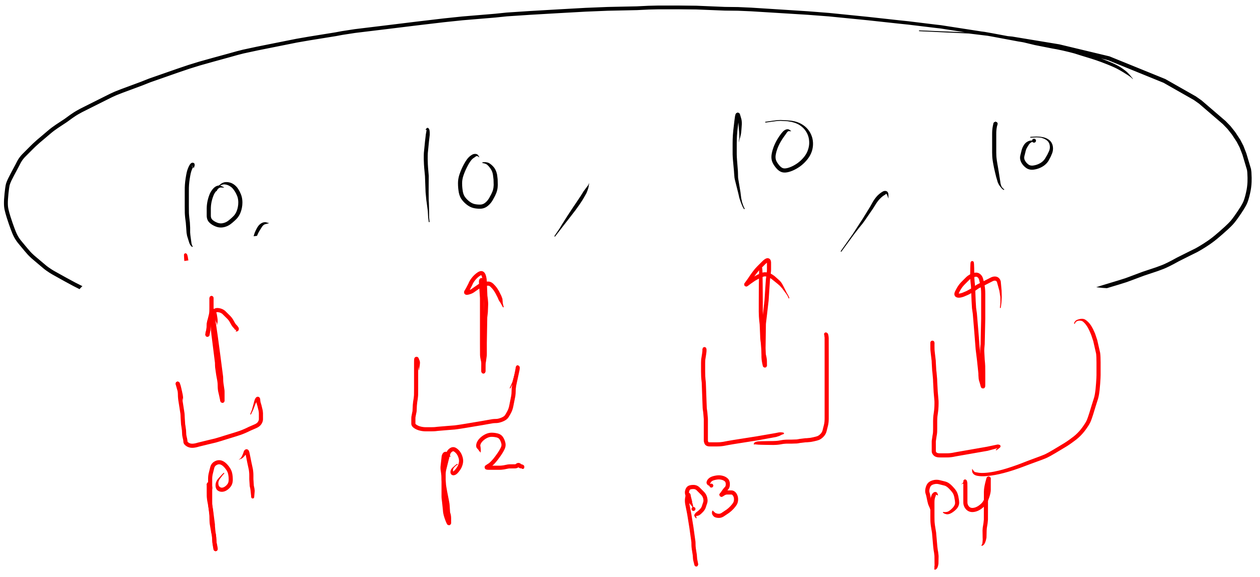
mid = 21 , painters = 2

mid = 19 , painters = 4

mid = 20 , painters = 2

ans = si

pointers = ~~1~~ ~~2~~ ~~3~~ 4

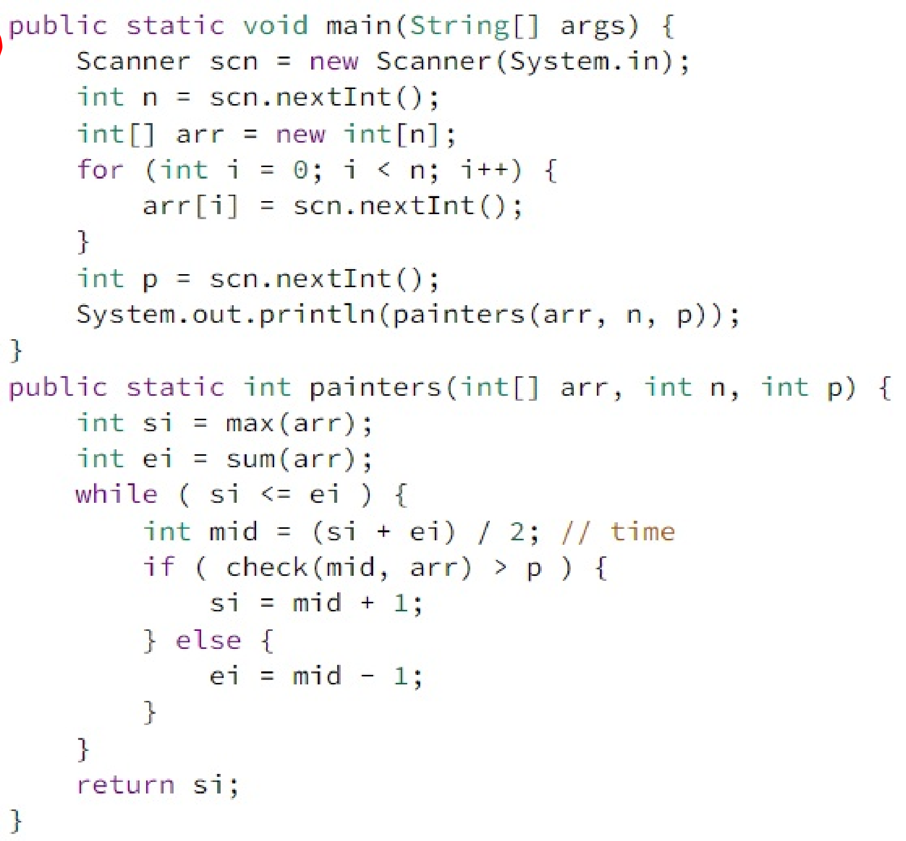


time = 18

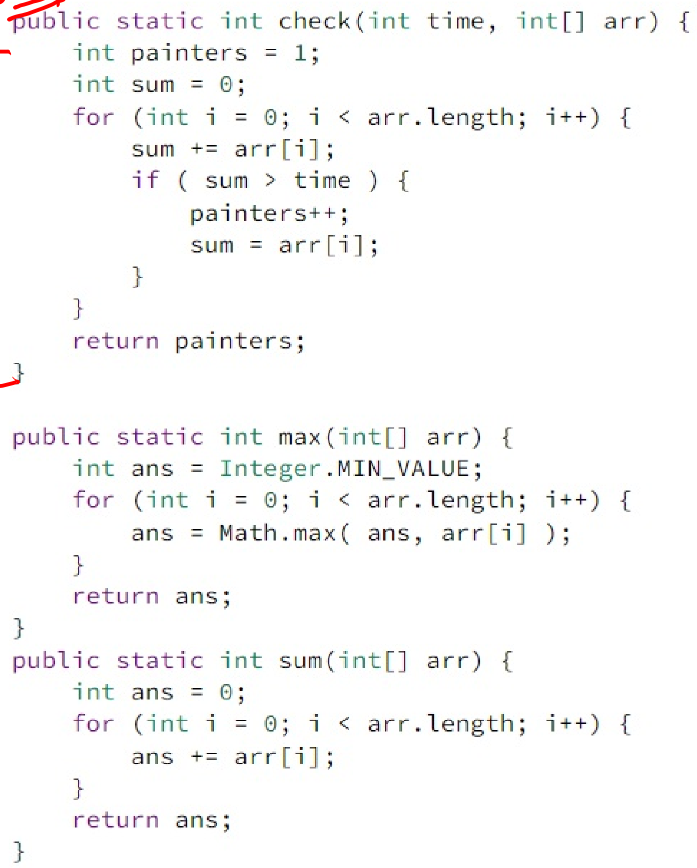
sum = ~~0+10+10+10~~

~~10+10~~

10+10



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 p = scn.nextInt();
    System.out.println(painters(arr, n, p));
}
```
2)
```java
public static int painters(int[] arr, int n, int p) {
    int si = max(arr);
    int ei = sum(arr);
    while ( si <= ei ) {
        int mid = (si + ei) / 2; // time
        if ( check(mid, arr) > p ) {
            si = mid + 1;
        } else {
            ei = mid - 1;
        }
    }
    return si;
}
```



Imp

3)
```java
public static int check(int time, int[] arr) {
    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 = Integer.MIN_VALUE;
    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 += arr[i];
    }
    return ans;
}
```