

Search insert position

$x = 5 \rightarrow ans = 2$?

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You must write an algorithm with $O(\log n)$ runtime complexity.

Sample Input 0

4
1 3 5 6
5

Sample Output 0

2

Sample Input 1

4
1 3 5 6
2

Sample Output 1

1

n = 4

1356

5

1345

0123

x = 2

x = 7

idx = 0

idx + 1 =

ans =

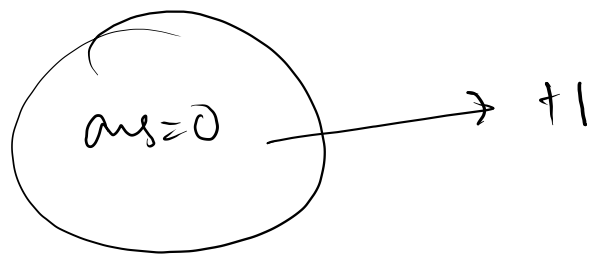
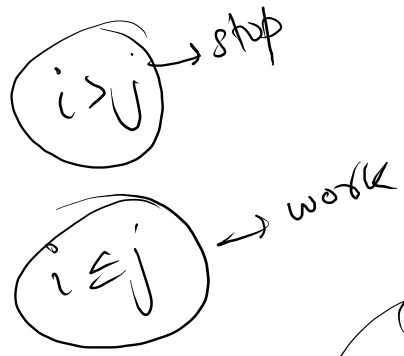
6

$x = 2$

$A[m] == x \rightarrow \text{return } m$

$A[m] > x \rightarrow \text{left}$

$A[m] < x$
↳ save & right



$x = 37$

$ans = \cancel{-1} \cancel{3} \cancel{6}$

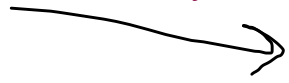
$i \leq j$

$i > j$
↳ stop

5	10	15	20	25	30	35	40
0	1	2	3	4	5	6	7
						j	

1. $A[m] == x \rightarrow m$

2. $A[m] > x \rightarrow \text{left}$

3. $A[m] < x \rightarrow \text{save \& right}$


i
m

```

10 for(int i = 0; i < n; i++){
11     A[i] = scn.nextInt();
12 }
13 int x = scn.nextInt();
14
15 int i = 0;
16 int j = n-1;
17 int ans = -1;
18 boolean isPresent = false;
19
20 while(i <= j){
21     int m = (i + j) /2;           // i + (j-i)/2
22     if(A[m] == x){
23         ans = m;
24         isPresent = true;
25         break;
26     }else if(A[m] > x){
27         //left
28         j = m-1;
29     }else{ //save and right
30         ans = m;
31         i = m + 1;
32     }
33 }
34 if(isPresent){
35     System.out.println(ans);
36 }else{
37     System.out.println(ans + 1);
38 }
39
40 }
41 }

```

```

5
6 public static void main(String[] args) {
7     Scanner scn = new Scanner(System.in);
8     int n = scn.nextInt();
9     int [] A = new int[n];
10    for(int i = 0; i < n; i++){
11        A[i] = scn.nextInt();
12    }
13    int x = scn.nextInt();
14
15    int i = 0;
16    int j = n-1;
17    int ans = -1;
18
19    while(i <= j){
20        int m = (i + j) /2;           // i + (j-i)/2
21        if(A[m] == x){
22            ans = m;
23            System.out.println(m);
24            return;
25        }else if(A[m] > x){
26            //left
27            j = m-1;
28        }else{ //save and right
29            ans = m;
30            i = m + 1;
31        }
32    }
33    System.out.println(ans + 1);
34 }
35 }

```

The banana challenge

Koko is fond of consuming bananas and is faced with n piles of bananas, where the i th pile has $piles[i]$ bananas. Meanwhile, the guards have temporarily left and are expected to return in h hours.

Koko has the freedom to determine her banana-eating speed per hour, which she can set to k . Every hour, she selects a pile of bananas and consumes k bananas from that pile. However, if the selected pile has less than k bananas, she finishes all the bananas in that pile and won't eat any more bananas in that hour.

Koko prefers to eat slowly but is still determined to finish consuming all the bananas before the guards come back.

Return the minimum integer k such that she can eat all the bananas within h hours.

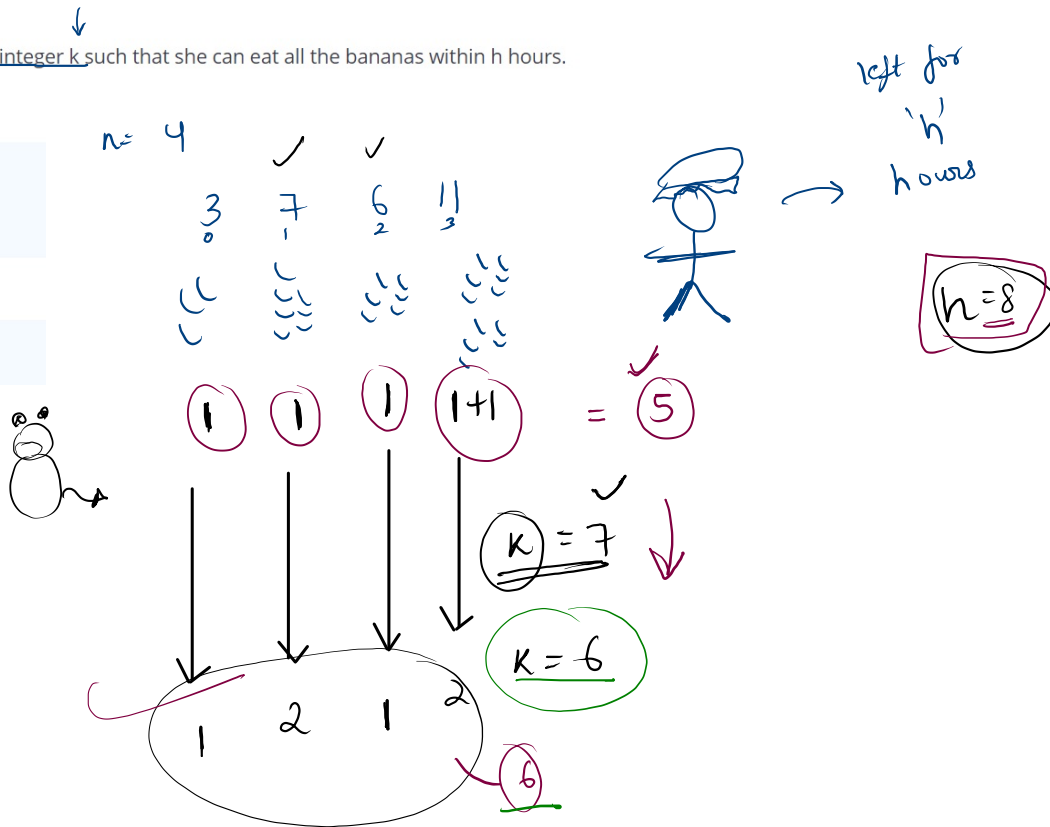
Sample Input 0

```
4
3 6 7 11
8
```

Sample Output 0

```
4
```

$k =$ banana - eating speed/hour



$$h=8 \checkmark$$

$$k=5$$

$$k=5 \checkmark$$

$$\boxed{k=4 \checkmark}$$

3

7

6

11

t

1

2

2

3

=

8

1

2

2

3

=

8

—

$\frac{3}{4}$

$\frac{7}{4}$

$\frac{6}{4}$

$\frac{11}{4}$

$\int_0 \dots$

$\int_1 \dots$

$\int_1 \dots$

$\int_2 \dots$

1

2

2

3

~~t~~

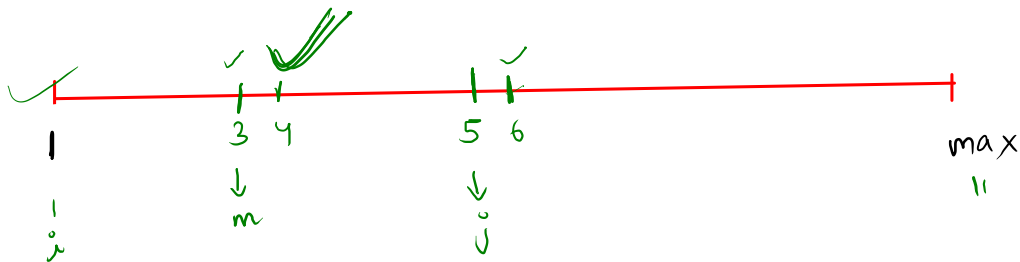
$h=8$

3 7 6 11
0 1 2 3

✓ $len \leq h$

(K)

func^c to check if she can eat
all banana in
 m hours
↳ 6



$x \leq h \rightarrow$ return true

```

4 public class Solution {
5     public static boolean isPossible(int [] A ,int h, int k){
6         int t = 0;
7         for(int i = 0; i < A.length; i++){
8             t += Math.ceil(1.0*A[i]/ k);
9         }
10        return t <= h;
11    }
12    public static void main(String[] args) {
13        Scanner scn = new Scanner(System.in);
14        int n = scn.nextInt();
15        int [] A = new int[n];
16        int max = Integer.MIN_VALUE;
17        for(int i = 0; i < n; i++){
18            A[i] = scn.nextInt();
19            max = Math.max(max, A[i]);
20        }
21        int h = scn.nextInt();
22        int i = 1;
23        int j = max;
24        int k = max;
25        while(i <= j){
26            int m = (i+j) / 2;
27            if(isPossible(A, h, m)){ //if k = m can koko eat all banana?
28                k = m;
29                j = m-1;
30            }else{
31                i = m+1;
32            }
33        }
34        System.out.println(k);
35    }

```