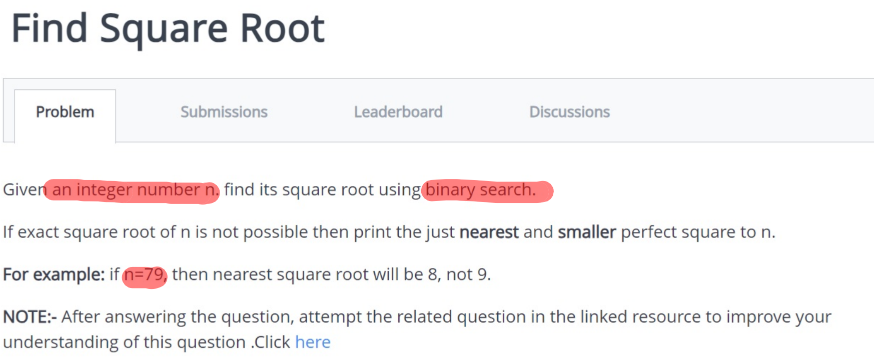


Find Square Root

Given an integer number n, find its square root using binary search.

If exact square root of n is not possible then print the just **nearest** and **smaller** perfect square to n.

For example: if n=79, then nearest square root will be 8, not 9.

NOTE:- After answering the question, attempt the related question in the linked resource to improve your understanding of this question .Click here

Sample Input 0

```
16
```

Sample Output 0

```
4
```

$1+17/2 = 9$

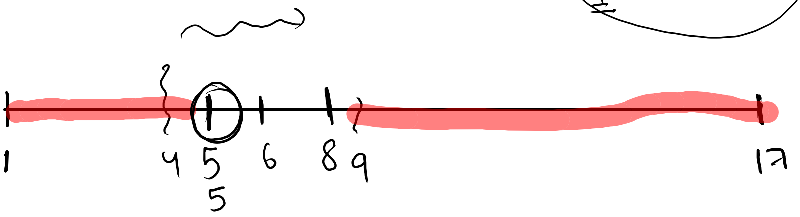
$n = 17 \rightsquigarrow 4$

$9 * 9 == 17$ ✗

$9 * 9 < 17$

$9 \times 9 > 17$

pot = 4



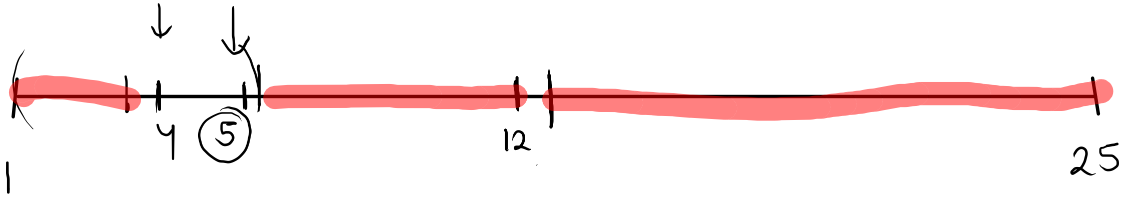
$6 \times 6 > 17$

$6 \times 6 < 17$ ✗

$6 \times 6 = 17$ ✗

n=2~~5~~4

m=5   5*5 ==25



right →

pot.= ~~3~~ 4

← left → not storing pot.

25+1/2 =13

3*3 ==25

3*3 < 25

X 13*13 ==25

X 13x13 < 25

✓ 13x13 > 25

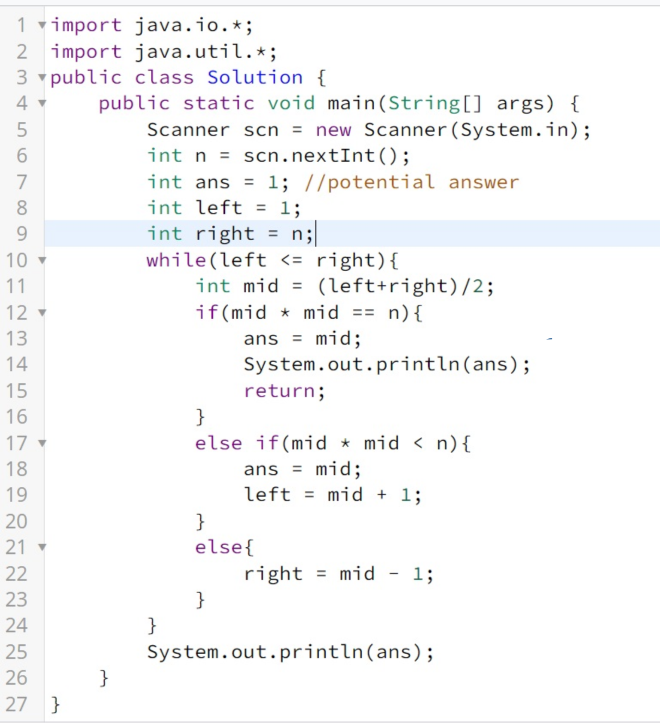
4*4 ==25

4*4 < 25

6*6 = 25

6*6 < 25

6*6 > 25

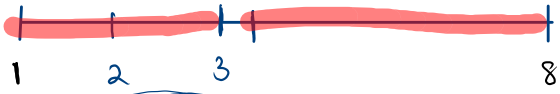


```java
import java.io.*;
import java.util.*;
public class Solution {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int ans = 1; //potential answer
        int left = 1;
        int right = n;
        while(left <= right){
            int mid = (left+right)/2;
            if(mid * mid == n){
                ans = mid;
                System.out.println(ans);
                return;
            }
            else if(mid * mid < n){
                ans = mid;
                left = mid + 1;
            }
            else{
                right = mid - 1;
            }
        }
        System.out.println(ans);
    }
}
```

n= 8

string ans.

$\boxed{ans = 2}$



r l

3 ✓

m = 2

2 * 2 == 8

2 * 2 < 8

3 ≤ 2 ✗

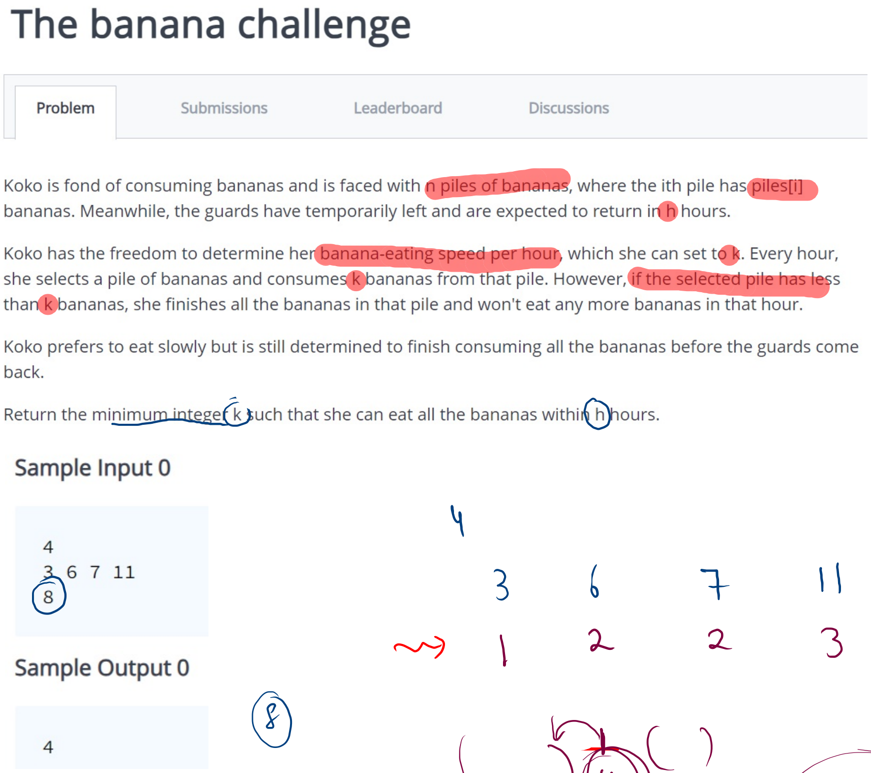
3 ≤ 3 ✓

3 * 3 == 8

3 * 3 < 8

$\sqrt{8} = 2$ ...

3 ✗



The banana challenge

Koko is fond of consuming bananas and is faced with n piles of bananas, where the ith 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
```

if she can eat all bananas if k=m
in given 'h'

eating speed / hr

h=8

k = ~~11~~ ~~6~~ ~~5~~ 4

~~5~~ 4

8

4

3 6 7 11

1 2 2 3

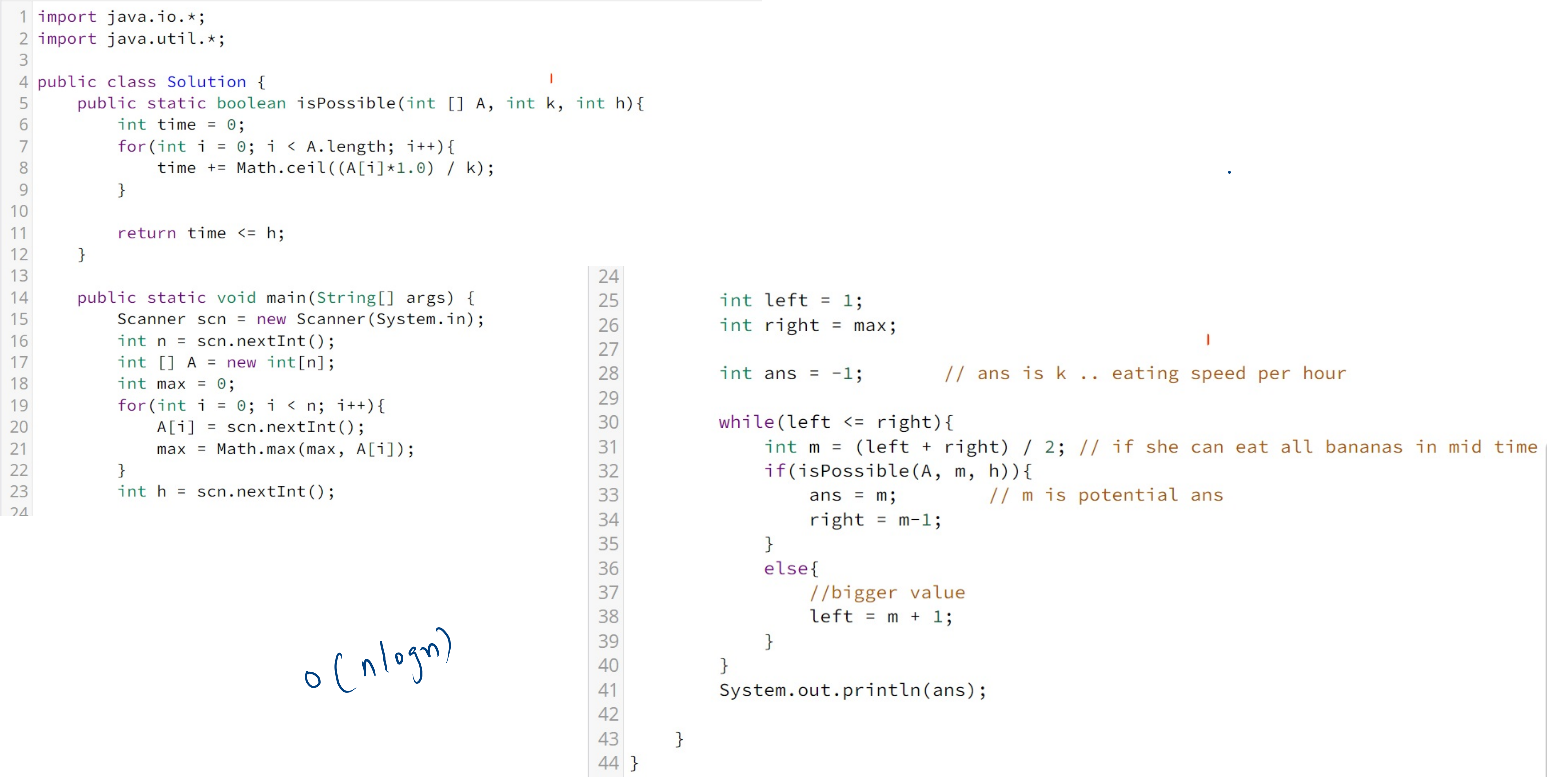
4

m=4

m = 6

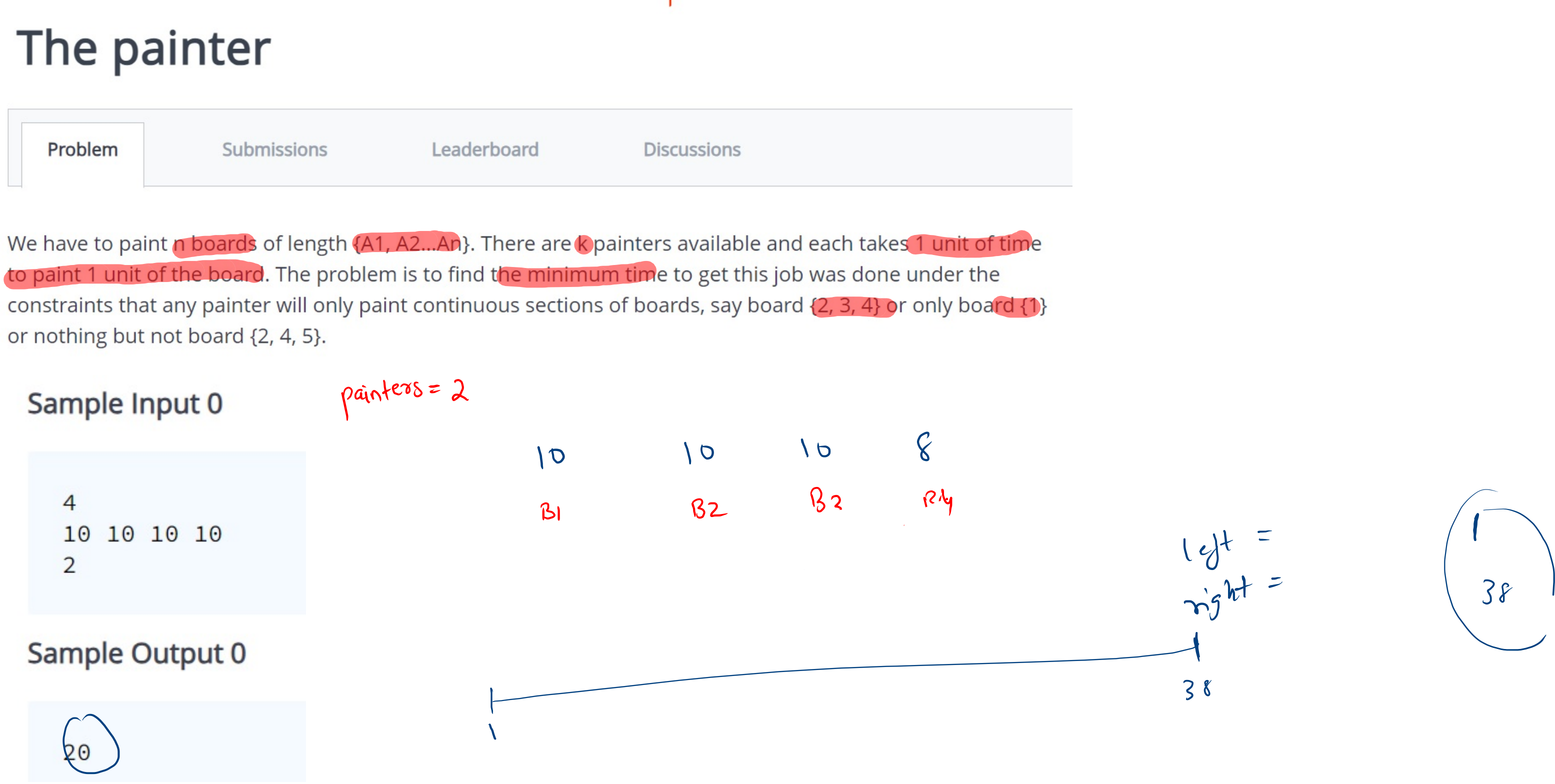
m= 3

m=5



```java
import java.io.*;
import java.util.*;

public class Solution {
    public static boolean isPossible(int [] A, int k, int h){
        int time = 0;
        for(int i = 0; i < A.length; i++){
            time += Math.ceil((A[i]*1.0) / k);
        }

        return time <= h;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        int max = 0;
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
            max = Math.max(max, A[i]);
        }
        int h = scn.nextInt();

        int left = 1;
        int right = max;

        int ans = -1;       // ans is k .. eating speed per hour

        while(left <= right){
            int m = (left + right) / 2; // if she can eat all bananas in mid time
            if(isPossible(A, m, h)){
                ans = m;        // m is potential ans
                right = m-1;
            }
            else{
                //bigger value
                left = m + 1;
            }
        }
        System.out.println(ans);

    }
}
```

o(nlogn)



The painter

We have to paint n boards of length {A1, A2...An}. There are k painters available and each takes 1 unit of time to paint 1 unit of the board. The problem is to find the minimum time to get this job was done under the constraints that any painter will only paint continuous sections of boards, say board {2, 3, 4} or only board {1} or nothing but not board {2, 4, 5}.

Sample Input 0

```
4
10 10 10 10
2
```

Sample Output 0

```
20
```

painters = 2

10 10 10 8

B1 B2 B2 B4

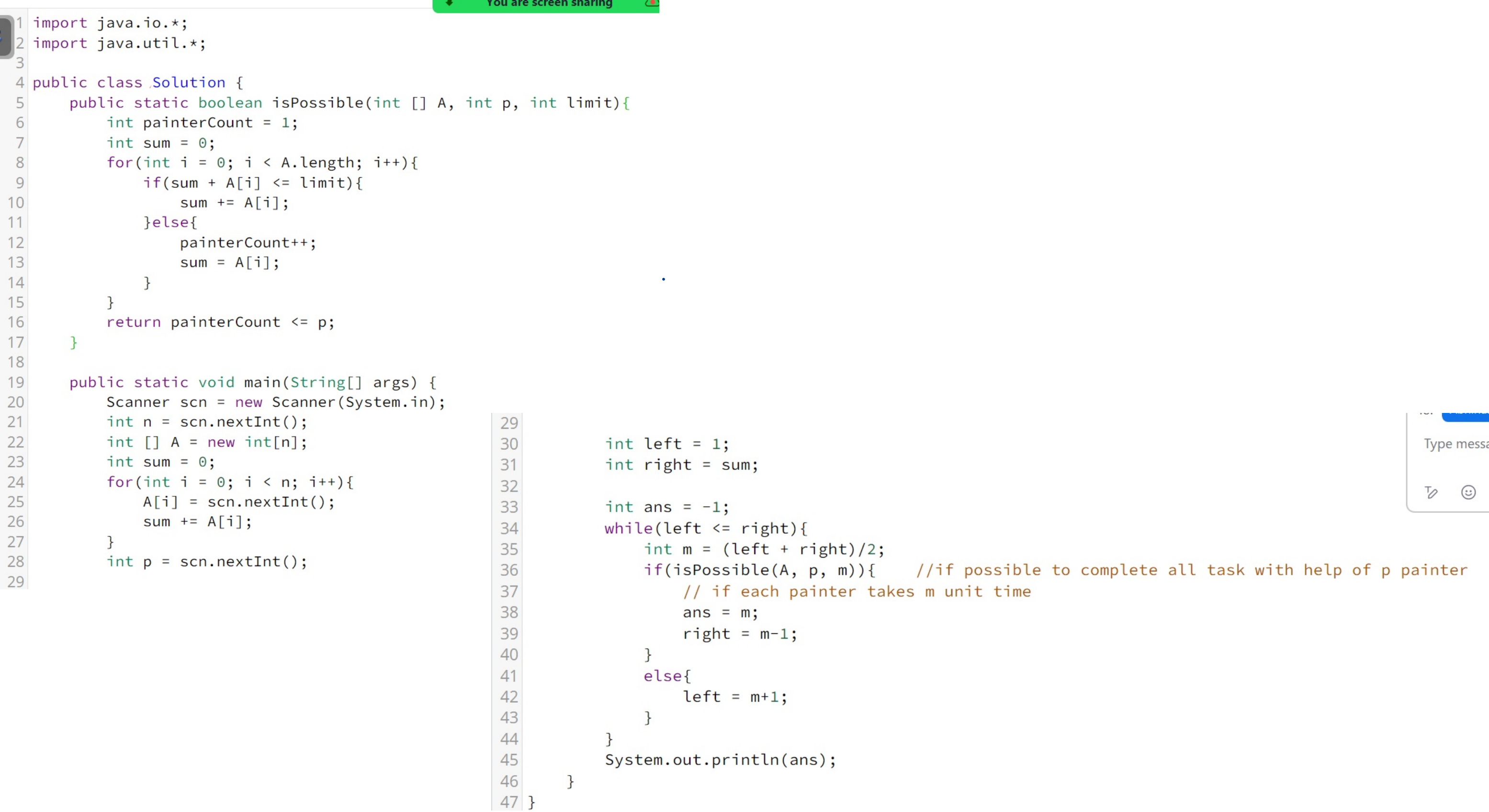
left =
right =

1

38

1

38



```java
import java.io.*;
import java.util.*;

public class Solution {
    public static boolean isPossible(int [] A, int p, int limit){
        int painterCount = 1;
        int sum = 0;
        for(int i = 0; i < A.length; i++){
            if(sum + A[i] <= limit){
                sum += A[i];
            }else{
                painterCount++;
                sum = A[i];
            }
        }
        return painterCount <= p;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        int sum = 0;
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
            sum += A[i];
        }
        int p = scn.nextInt();

        int left = 1;
        int right = sum;

        int ans = -1;
        while(left <= right){
            int m = (left + right)/2;
            if(isPossible(A, p, m)){    //if possible to complete all task with help of p painter
                // if each painter takes m unit time
                ans = m;
                right = m-1;
            }
            else{
                left = m+1;
            }
        }
        System.out.println(ans);
    }
}
```