The painter

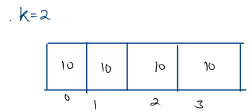
We have to paint n boards of length {A1, A2...An}. There are k painters available and each takes 1 unit of time to paint 1 unit of the board. The problem is to find the minimum time to get this job was done under the constraints that any painter will only paint contiguous sections of boards, say board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

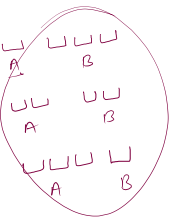
Sample Input 0

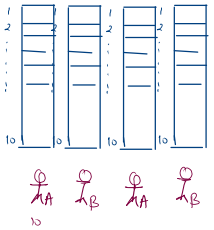
```
4
10 10 10 10
2
```

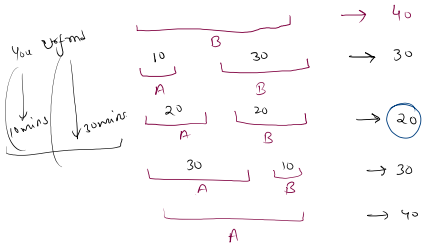
Sample Output 0

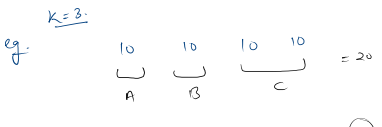
```
20
```

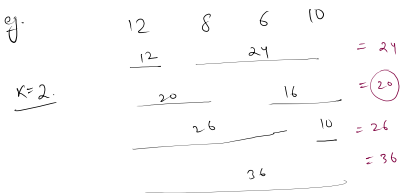


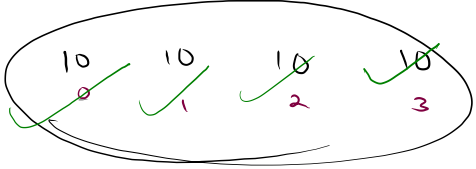








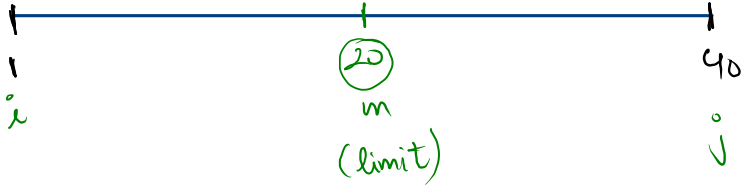


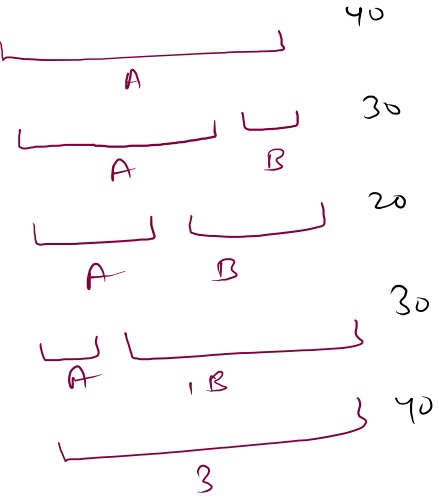


K=2

1 unit → 1 time

take limit & tell how much painters do you need to complete task





limit = 20

A = 0 1 2 3 4 5 6 7 8 9 10

B = 11 12 13 14 15 16 17 18 19 20

painter = A B

painter <= k

avg = 23 ~~19~~ ~~17~~ 16.

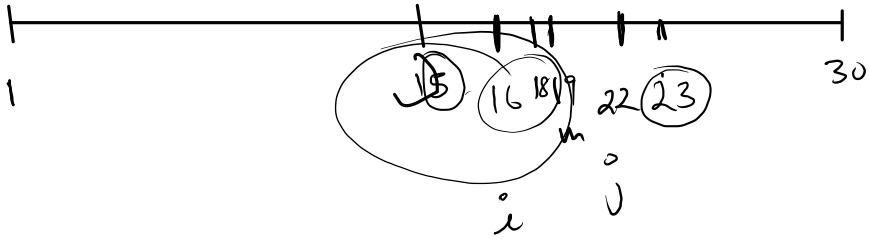
12 4 8 6

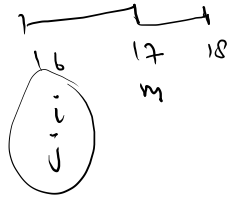
k = 2

avg = 16

m = i+j/2

= 15





limit = 15

A = ∅ 12

B = ∅ ~~4~~ 12

C = ∅ 6

limit = 23

A = ∅ ~~12~~ 16

B = ∅ ~~8~~ 14

limit = 19

A = 16

B = 14

limit = 17

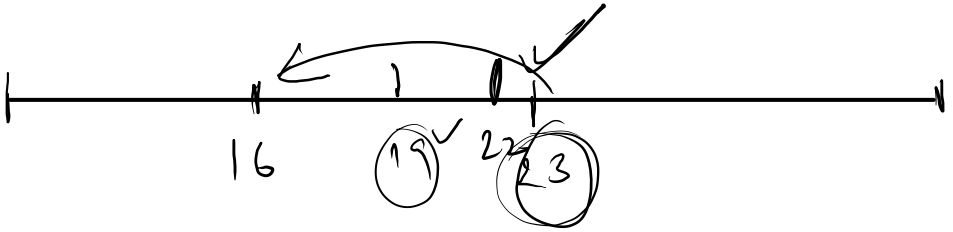
A = 16

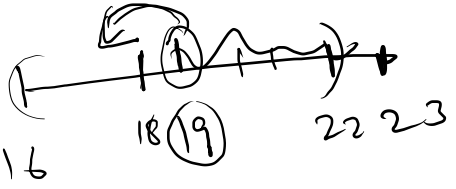
B = 14

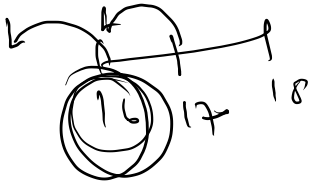
limit = 16

A = 16

B = 14


16
19
22
23


16
18
19
22
23


16
17
18

limit=15 k=2

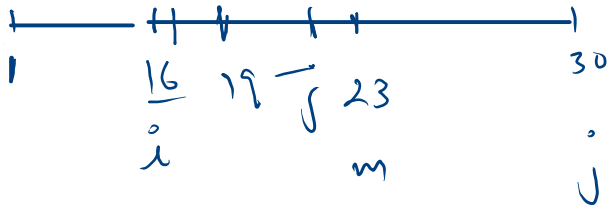
```java
public static boolean isPossible(int [] A, int limit, int k){
    //k is total given painter
    int painter = 1;
    int work = 0;   //work of current painter
    for(int i = 0; i < A.length; i++){
        if(work + A[i] <= limit){
            work += A[i];
        }else{
            painter++;
            work = A[i];
        }
    }
    return painter <= k;
}
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 k = scn.nextInt();

    int i = 1;
    int j = sum;
    int ans = sum;

    while(i <= j){
        int m = (i + j)/2;
        if(isPossible(A, m, k)){
            ans = m;
            j = m-1;
        }else{
            i = m+1;
        }
    }
    System.out.println(ans);
```

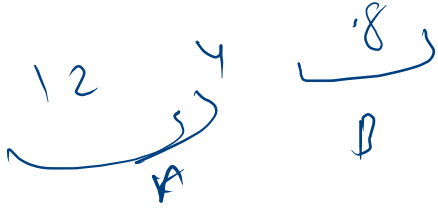
You are scree

12 4 8 6 k=2



painter = 2

work = 8



Array list → Dynamic Array → no need to define size.

Data Structure

Array → define size.

```
int [] A = new int [5];
```

Any data structure.

- initialize it
- add
- remove
- get
- size.

```java
import java.util.ArrayList;
import java.util.*;

public class Main
{
    public static void main(String[] args) {
        //init
        ArrayList<Integer> arr = new ArrayList<>();
        //add

        arr.add(10);
        arr.add(20);
        arr.add(30);

        System.out.println(arr);
    }
}
```

Integer

arr = { 10 20 30 }

0 1 2

Main.java

```
4  public class Main
5  {
6      public static void main(String[] args) {
7          //init
8          ArrayList<Integer> arr = new ArrayList<>();
9          //add
10
11         arr.add(10);
12         arr.add(20);
13         arr.add(30);
14         arr.add(40);
15
16         //print
17         System.out.println(arr);
18
19         //get(idx)
20         System.out.println(arr.get(2));
21
22         arr.add(50);
23
24         //size()
25         System.out.println(arr.size());
26
```