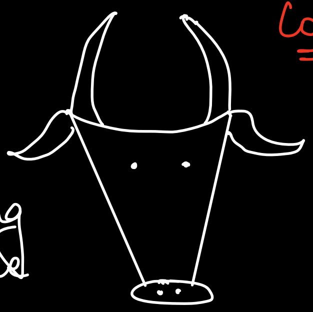
Q.1 Aggressive Cows.

Google

Given :


Cow

1) A sorted array +ve integers having the positions of the rooms where we can keep the cows. (size = N)

2) K ⇒ No. of cows.

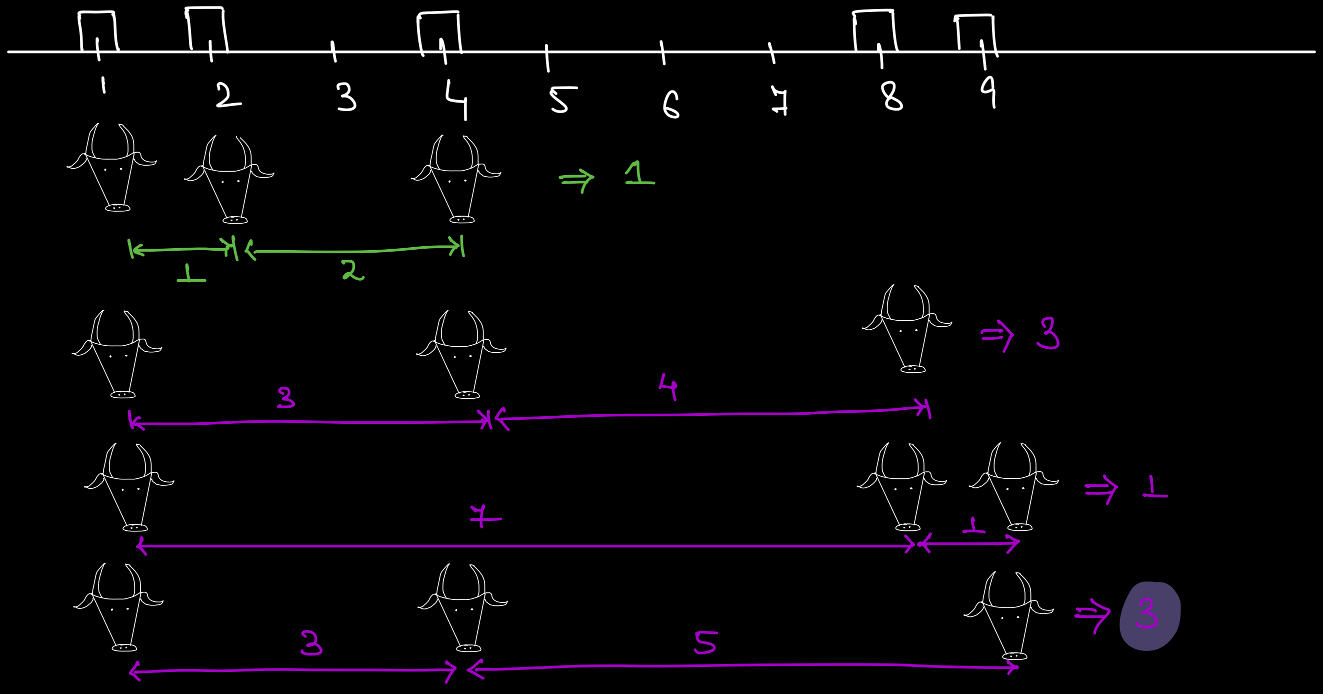
Cows are aggressive.

distance b/w any two cows.

Return the MAX value of $\boxed{\text{MIN distance}}$ possible b/w any two cows.

A : $\overset{0}{1}, \overset{1}{2}, \overset{2}{4}, \overset{3}{8}, \overset{4}{9}$ } Room positions.

K=3



⇒ Place (K) cows in (N) rooms such that the min distance b/w any 2 closest cows is MAXIMUM

Brute Force

Try out all the possible ways of placing K cows in N rooms.

↳ $^{N}C_{K}$ $\left(\frac{N!}{K!(N-K)!}\right)$

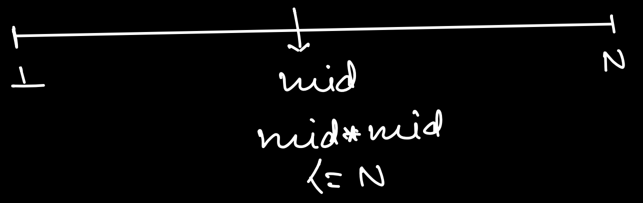
⇒ Iterate over all $^{N}C_{K}$ possibilities & keep updating the distance b/w any 2 closest cows.

$$^{N}C_{K} * N \simeq O(N!)$$

sqrt(N)

$i = 1$

$$\left.\begin{array}{l} 1\times 1 <= N \ ? \\ 2\times 2 <= N \ ? \\ 3\times 3 <= N \ ? \\ \vdots \\ n\times n <= N \end{array}\right\} \sqrt{N}$$



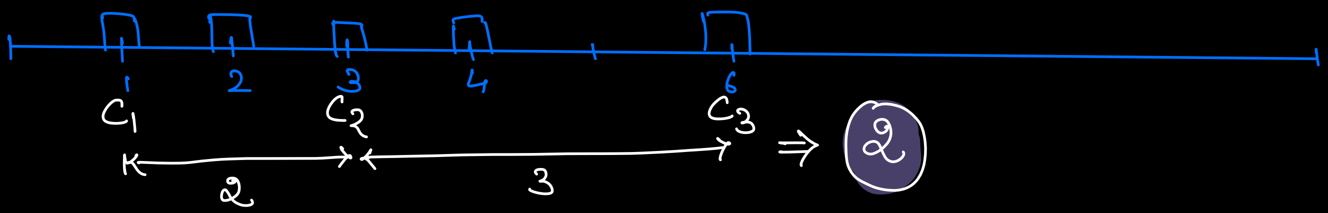
⇒ max value s.t.
$n*n <= N$.

Target ⇒ dist b/w 2 closest cows.

$ans_{max} \rightarrow A[N-1] - A[0]$	
$ans_{min} \rightarrow 1$	

$\Big\}$ $R \in [1, ans_{max}]$

$A : \{ \overset{0}{1}\ \ \overset{1}{2}\ \ \overset{2}{3}\ \ \overset{3}{4}\ \ \overset{4}{6} \}$ $K = 3.$



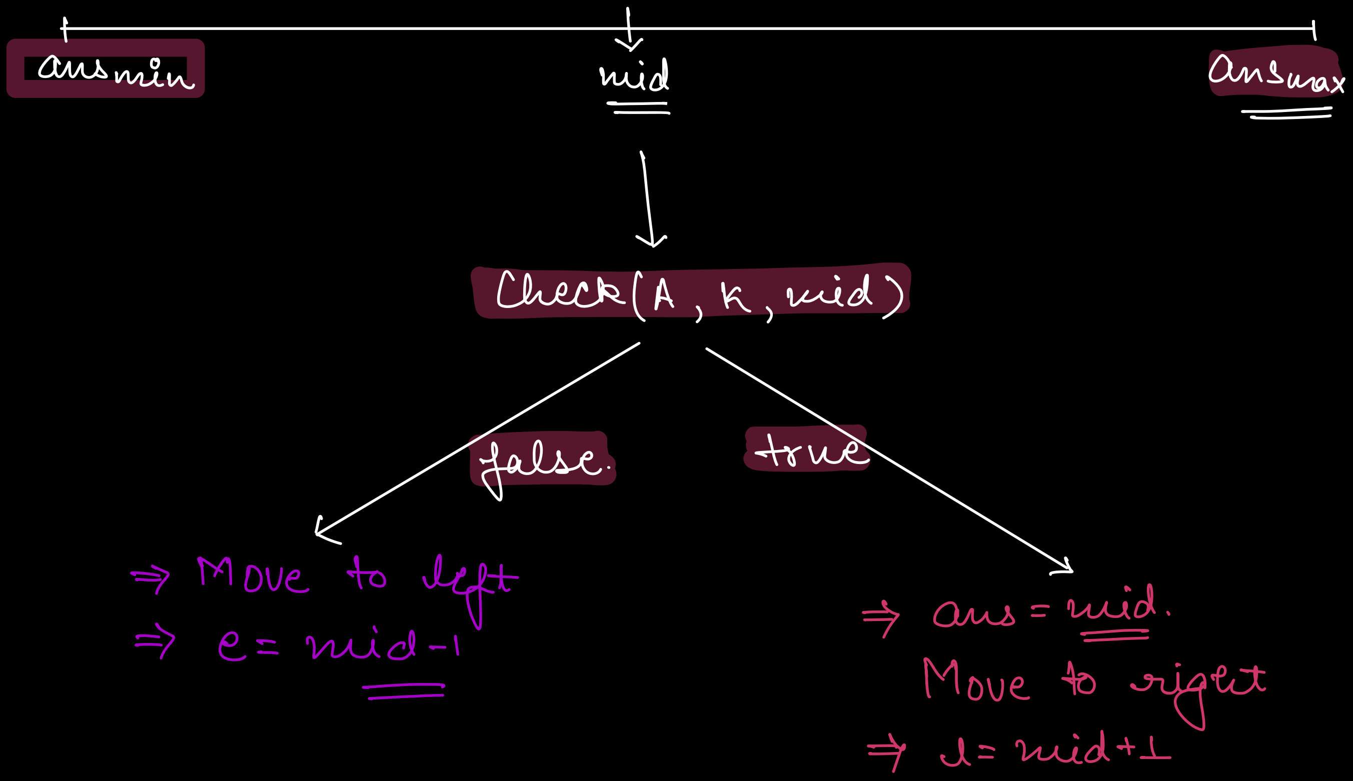
ans ∈ [1, 5]

d	check(A, k, d)
5	×
4	×
3	×
2	✓

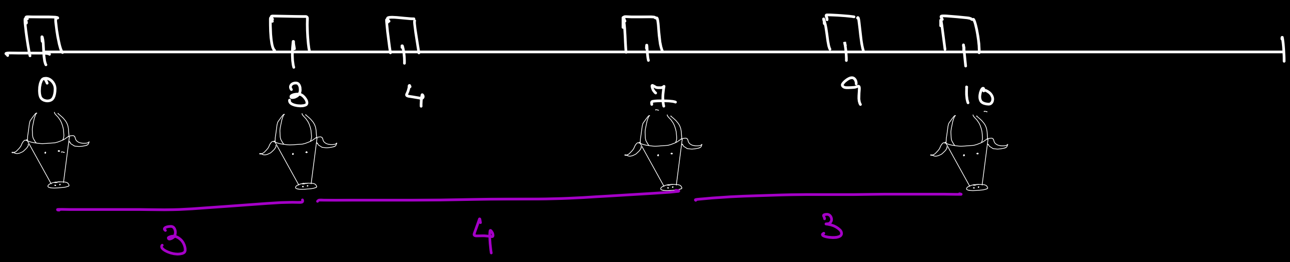
```
        Range (R)
for ( d = ansmax ; d >= 1 ; d-- ) {
    //Check if we can place k cows
    // maintaining a min dist of d.
    if(check(A, k, d)) {
        return d;
    }
}
    O(N)
```

TC : $O(K \cdot N)$

SC : $O(1)$



$$A : \left\{ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 0, & 3, & 4, & 7, & 9, & 10 \end{array} \right\} \quad K=4$$



ans $\in [1, 10]$

d	Check(A, K, d)
10	✗
9	✗
8	✗
7	✗
6	✗
5	✗
4	✗
3	✓

```
bool check(A, d, K){
    // returns true if it is possible to place
    // K cows maintaining the dist. b/w
    // any 2 cows >= d.
    int prevPos = A[0]
    int cowsPlaced = 1;
    for(i=1; i< N; i++){
            if( A[i] - prevPos >=d ){
                  cowsPlaced++;
                  prevPos = A[i]
            }
            if(cowPlaced == K){
                  return true;
            }
    }
    return false;
}
```

⇒ $\log R * N$

TC: $O(N \log R)$

SC: $O(1)$

Q.2 Given N tasks & K workers.
Google

- Array of size N.
- A[i] is the time required to complete i^{th} task.

1) One task can only be performed by one worker. (Worker's can't share a task)
2) A worker can only perform tasks which are contigous to each other.
3) All workers can do their tasks Parallely.

* Find the minimum amount of time required to complete all the tasks by the worker team.

size

N = 15 , K = 3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A:	3	5	1	7	8	2	5	3	10	1	4	7	5	4	6

$w_1 = 31$ $\quad w_2 = 25$ $\quad w_3 = 15$

$t = \max(w_1, w_2, w_3)$

$= \underline{\underline{31}}$

0 – 5 $\quad$ 6 – 10 $\quad$ 11 – 14

$W_1 = 26$ $\quad W_2 = 23$ $\quad W_3 = 22$

$t = \underline{\underline{26}}.$

0 – 4 $\quad$ 5 – 10 $\quad$ 11 – 14

$W_1 = 24$ $\quad W_2 = 25$ $\quad W_3 = 22$

$t = \underline{\underline{25}}$ ✓

Target ⇒ Min time to complete all the tasks.

Range ⇒ [max(A[i]), sum(A[])]

ans_{min} ⇒ max(A[i]) {N workers}

ans_{max} ⇒ sum(A[]) {1 worker}

⇒

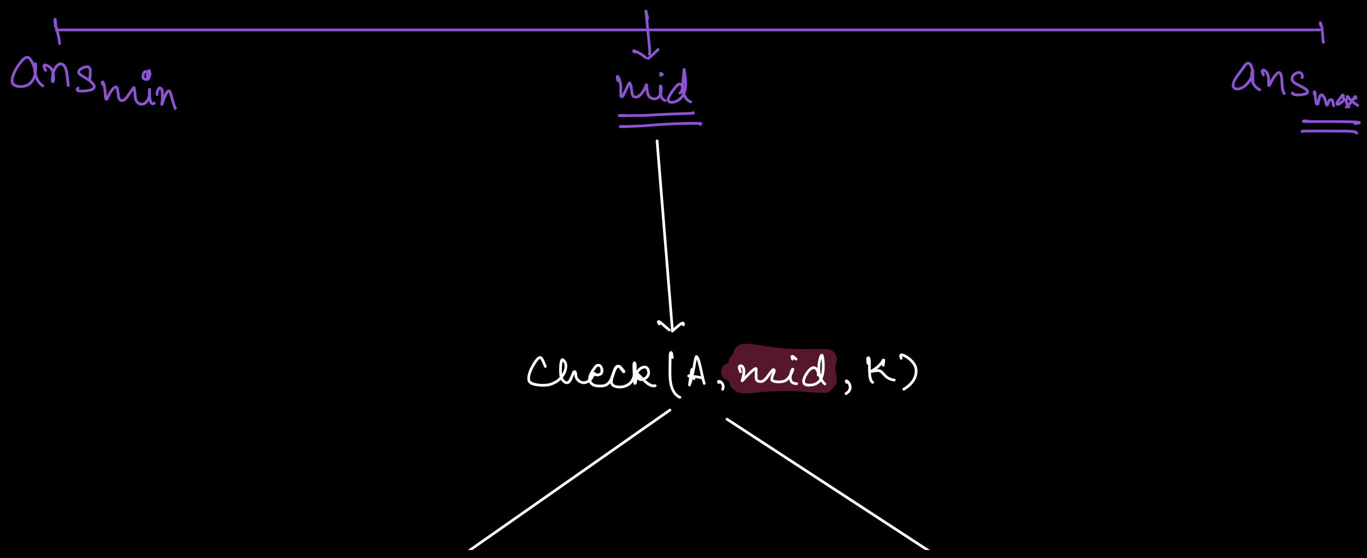
Linearly

```
for( t=ans_min ; t<= ans_max ; t++){
    if( check(A, t, k)){
        return t;
    }
}
```

O(N)

Check if it is possible to complete N tasks in t time using K workers.

⇒ TC: O(R·N)



True

⇒ Store mid as ans.

⇒ Move to left.

$\boxed{r = mid - 1}$

False

⇒ Move to right

⇒ $l = mid+1$

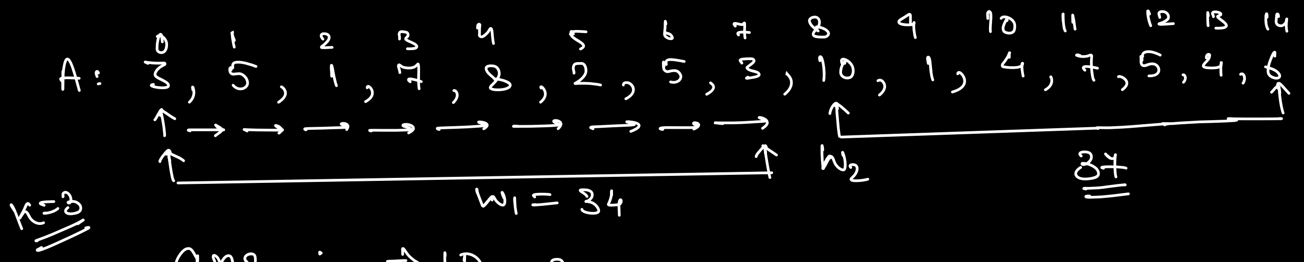
```
l = ansmin ]
r = ansmax ]  → log R
while(l <= r){
    m = (l+r)/2
    if(check(A, mid, K))
        ——
        ——
    }
    else{
        ——
        ——
    }
}
```

TC: $O(\log R \cdot N)$

$R \equiv [ans_{min}, ans_{man}]$

$R \equiv ans_{max} - ans_{min} + 1$

↑ sum(A)   ↑ man(A[])



ansmin ⇒ 10

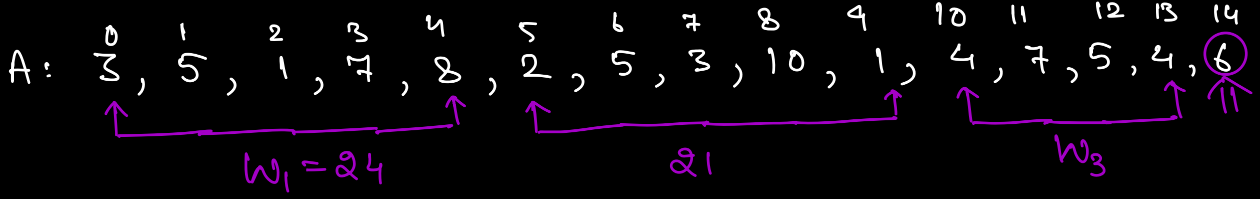
ansmax ⇒ 71

} mid = 40

ans = 40

⇒ [10, 39]

mid = 24 ×



⇒ [25, 39]

——— * ———

long

⇓

a = (long) b * c.