Searching
- → Linear Search $TC = \underline{O(N)}$ ✓
- → Organised → Binary Search

Search Space }
Target } ✓

Binary Search

search space (indices 4 to 9)

	0	1	2	3	4	5	6	7	8	9
A =	[3	6	9	12	14	19	20	23	25	27]

x ← 20 > 12

(sorted)

target = 20

start from mid ⇒ search space is half.

	~~l~~				~~m~~	~~l~~ ~~m~~	l r	m ~~m~~		~~r~~
	0	1	2	3	4	5	6	7	8	9
A =	[3	6	9	12	14	19	20 ✓	23	25	27]

target = 20 20 > 14 20 > 19 20 < 23

```
l = 0        r = N-1
while (l <= r) {
    m = (l+r)/2      // l+(r-l)/2
    if (A[m] == target) return m
    if (A[m] < target)  l = m+1
    else    r = m-1
}
return -1  // Not found
```

$N \rightarrow \frac{N}{2} \rightarrow \frac{N}{4} \; . \; . \; . \; \frac{N}{2^K} = 1$

$\Rightarrow K = \underline{\log_2(N)}$

$TC = \underline{O(\log_2(N))}$

$SC = \underline{O(1)}$

Q → (Find first mail of particular date)
Given a sorted integer array (with duplicates), find first occurrence of a number.

$$
\begin{array}{cccccccccccc}
 & & \cancel{l} & \cancel{m} & \cancel{l} & \cancel{r} & \cancel{m} & & & & & \cancel{r} \\
 & & & & & l & & & & & & \\
 & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
A = & [& 3 & 3 & 9 & 12 & 12 & 12 & 20 & 23 & 23 & 27] \\
 & & & & \cancel{m} & m & & & & & &
\end{array}
$$

target = 12

Ans = 3

$l = 0 \quad r = N-1$

```
while (l <= r) {
    mid = (l+r)/2
    if (A[m] == target &&
        (m == 0 || A[m-1] != target))
            return m
    if (A[m] < target)   l = m+1
    else   r = m-1        // = or >
}
return -1
```

$TC = O(\log_2(N))$

$SC = O(1)$

3 steps →
1) Define search space
2) check if mid is answer
3) Decide going left/right

HW → Find last occurrence.

Q → Given an integer array where every element occurs twice except for 1 element, find that unique element. Duplicate elements are adjacent to each other.

$$
\begin{array}{ccccccccc}
 & & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
A = & [& 8 & 8 & 5 & 5 & 6 & 2 & 2]
\end{array}
$$

unsorted
target → condition

Ans = $\forall i$ ^A[i] (XOR)	TC = O(N)	SC = O(1)

l = 0, r = 6

	0	1	2	3	4	5	6
A =	8	8	5	5	6	2	2

$\neq$ $\neq$

3 steps →

1) Define search space ✓
2) check if mid is answer ✓
3) Decide going left/right ←

	0	1	2	3	4	5	6
A =	8	8	5	5	6	2	2
	l			m	l, m, r	m	r

(even, odd) (odd, even)

(m-1, m) / (m, m+1)

0, 1   2, 3   4, 5

```
l = 0        r = N-1
while (l <= r) {
    m = (l + r) / 2
    if ((m == 0 || A[m] != A[m-1]) &&
        (m == N-1 || A[m] != A[m+1]))   ✓
            return A[m]
    if (m != 0 && A[m] == A[m-1]) {   // m-1, m  ✓
        if (m % 2 == 0)   r = m-2      // odd, even
        else              l = m+1      // even, odd
    } else {              // m, m+1  ✓
        if (m % 2 == 1)   r = m-1      // odd, even
        else              l = m+2      // even, odd
    }
}
```

TC = O(log (N))   SC = O(1)

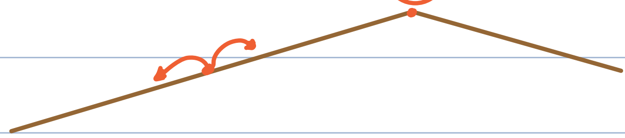
Q → Given an increasing decreasing array with distinct elements, find max element.

	0	1	2	3
A =	1	3	5	2

Ans = 5

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = [& 1 & 3 & 5 & 10 & 15 & 12 & 6 \;] \end{array}$$

Ans = 15



```
l = 0          r = N-1
while (l <= r) {
    m = (l+r) / 2
    if ((m == 0 || A[m] > A[m-1]) &&
        (m == N-1 || A[m] > A[m+1]))
            return A[m]
    if (m != 0 && A[m] > A[m-1])
            l = m+1
    else    r = m-1
}
```

(m-1) m → go right

$$\begin{array}{ccccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ A = [& 3 & 5 & 8 & 10 & 12 & 15 \;] \end{array}$$

(Ans)

TC = O(log (N))   SC = O(1)

Q → Given an array with distinct elements.
Google
Find any one local minima.

$A[i-1] > A[i] < A[i+1]$

$$\begin{array}{ccccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ A = [& 6 & 1 & 0 & 9 & 15 & 8 \;] \end{array}$$

Ans = 0 / 8

$$\begin{array}{cccc} & 0 & 1 & 2 \\ A = [& 5 & 8 & 10 \;] \end{array}$$

Ans = 5

$$\begin{array}{cc} & 0 \\ A = [& 20 \;] \end{array}$$

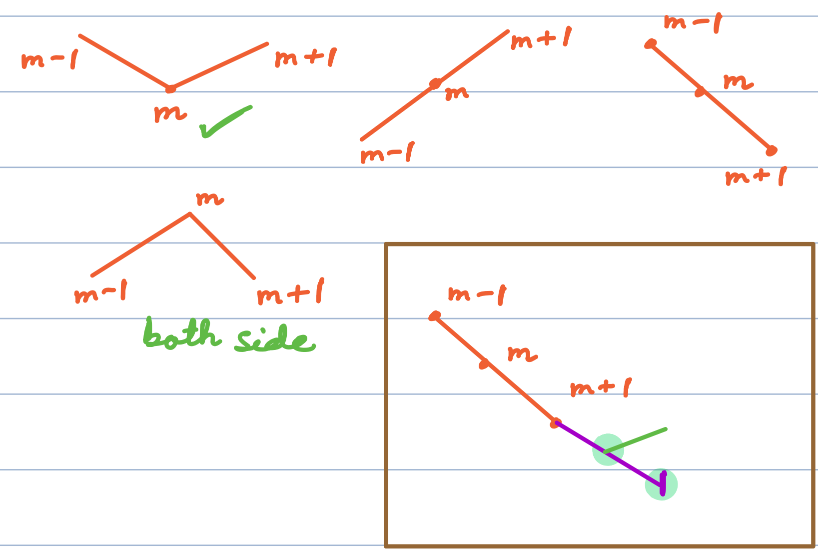
Ans = 20

Bruteforce → TC = O(N)   SC = O(1)

Binary Search

3 steps →

1) Define search space
2) check if mid is answer
3) Decide going left/right




```
l = 0        r = N-1

while ( l <= r ) {
    m = (l + r) / 2
    if ((m == 0 || A[m] < A[m-1]) &&
        (m == N-1 || A[m] < A[m+1]))
            return A[m]
    if ( m != 0 && A[m] > A[m-1] )
            r = m - 1
    else    l = m + 1
}
```

(m-1) m

TC = O(log (N))   SC = O(1)

[3 7 10 8 5 2 1]

... 12 3 7 10 8 5 2 4 6

Q → Given a rotated sorted array, find index of element K. If not present return -1. (unique elements)

	l			m				r
	0	1	2	3	4	5	6	7
A =	[32	38	40	2	5	10	25	26]

K = 5

Ans = 4

Linear Search → TC = O(N)

Search space → l = 0   r = N-1

check if mid is ans → if (A[m] == K) return m

Decide left / right →

	P1			P2				
	0	1	2	3	4	5	6	7
A =	[32	38	40	2	5	10	25	26]

K = 5

	l			~~m~~	~~l~~	~~m~~	~~l~~ ~~m~~	l r
	0	1	2	3	4	5	6	7
A =	[10	12	14	15	22	28	1	3]

K = 3

check if x is in part 1/2 →

if (x >= A[0]) ⇒ Part 1

else ⇒ Part 2

l	r	m
0	7	3
4	7	5
6	7	6
7	7	7

```
l = 0     r = N-1
while (l <= r) {
        m = (l + r)/2
        if (A[m] == K) return m
        if (K >= A[0]) {          // K → Part 1
            if (A[m] >= A[0]) {        // A[m] → Part 1
                if (A[m] < K)  l = m+1
```

```
                else    r = m-1
            } else {                // A[m] → Part 2
                r = m-1
            }
        } else {            // K → Part 2
            if (A[m] < A[0]) {  // A[m] → Part 2
                if (A[m] < K)   l = m+1
                else    r = m-1
            } else {            // A[m] → Part 1
                l = m+1
            }
        }
}
```

$TC = \underline{O(\log_2(N))}$ $SC = \underline{O(1)}$

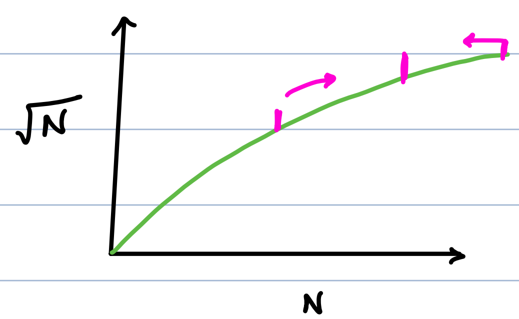
Q → Find sqrt(N) → only integer part $\underline{N >= 1}$

$N = 10$ Ans $= \underline{3}$ $N = 49$ Ans $= \underline{7}$

$\boxed{1 <= \sqrt{N} <= N}$

$N = 49 \longrightarrow$ $5 < \sqrt{N}$

$5^2 < N$




```
l = 1       r = N
while (l <= r) {                    Binary Search on Answer
    m = l + (r-l)/2
    if ( m*m <= N && (m+1)*(m+1) > N ) // m^2 <= N < (m+1)^2
            return m                        m <= √N < (m+1)
```

```
    if (m * m > N)    r = m-1
    else    l = m+1
}
```

$TC = O(\log(N))$ $SC = O(1)$

N^{th} Magical Number

1) $gcd(x, y)$ ✓ $TC = O(\log(x))$

2) $lcm(x, y) = \frac{x * y}{gcd(x, y)}$ ✓

3) Count of numbers <= K, divisible by X → $\frac{K}{X}$

$K = 20$ $X = 3$ {3, 6, 9, 12, 15, 18}   Ans = 6

$K = 50$ $X = 10$ Ans = 5

4) Count of numbers <= K, divisible by X or Y or both

$$\boxed{\frac{K}{X} + \frac{K}{Y} - \frac{K}{lcm(X, Y)}}$$

$K = 20$

$X = 3$ {3, 6, 9, 12, 15, 18}   Ans = 20/3 + 20/5

$Y = 5$ {5, 10, 15, 20}   = 6 + 4 = ~~10~~ 9

$K = 60$

$X = 6$ {6 12 18 24 30 36 42 48 54 60}

$Y = 5$ {5 10 15 20 25 30 35 40 45 50 55 60}

$Ans = \frac{60}{6} + \frac{60}{5} - 2 = 10 + 12 - 2 = 20$

$2 \rightarrow \frac{K}{lcm(X, Y)}$

Q → Find N^{th} number which is divisible by X or Y or both.

$N = 5$

$X = 6$ } 6 10 12 18 <u>20</u> (Ans)

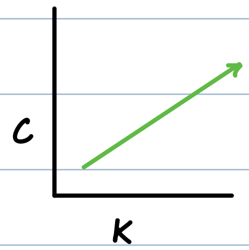
$Y = 10$

(i % X == 0 || i % Y == 0)

count of numbers <= K, divisible by X or Y or both = C

$K \searrow$ Ans $C \downarrow$ N

$K \propto C$



$N = 5$

$X = 6$ { 6 12 18 24 30 }

$Y = 10$ { 10 20 30 40 50 }

Range → [min (X, Y) N * min (X, Y)]

check if T is ans →

count (T) = T/X + T/Y − T/lcm (X, Y) = N

⇒ Ans = <u>T</u> ~~X~~

```
l = min (X, Y)        r = N * l
lcm = X * Y / gcd (X, Y)
while (l <= r) {
    m = l + (r−l) / 2
    cnt = m / X + m / Y − m / lcm
```

if (cnt == N && (m % x == 0 || m % y == 0))
    return m
if (cnt < N)   l = m+1
else   r = m - 1
}

TC = $\underline{O(\log(N * \min(x,y)))}$

SC = $\underline{O(\log(\min(x,y))}$ (GCD)

N = 6   lcm = 4*6/2 = $\underline{12}$

x = 4   l = ~~4~~ ~~15~~ ~~17~~ 18

y = 6   r = ~~24~~ 18

m = ~~14~~ ~~19~~ ~~16~~ ~~17~~ 18 ✓

cnt = 18/4 + 18/6 − 18/12 = 4 + 3 − 1 = $\underline{6}$

Q → Find $\underline{\text{median}}$ of given array.

middle element in sorted order

[1 3 5] Ans = 3

[3 5 1] Ans = 3

even → Average of 2 mid

[1 2 6 10]

smaller mid

[18 4 5 2 1]

1 2 4 5 18

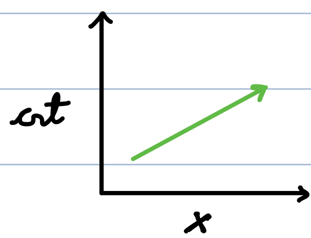
(Ans)

Sort & find middle

TC = $\underline{O(N \log(N))}$

Find $\underbrace{\text{cnt elements}}_{N/2}$ <= X.

⇒ Ans = $\underline{X}$



half = (N+1)/2

l = min (A[i])   r = max (A[i])

while (l <= r) {

    m = (l + r) / 2

cnt = count(m)   // # elements <= m in A[]

if (cnt == half &&   TC = O(N)

count(m-1) < half)   1 2 5 10 15

return m   half = (5+1)/2 = 3

if (cnt < half)   l = 1   r = 15

l = m+1   m = 8   cnt = 3

else   r = m-1   m-1 = 7   count(7) = 3

}   ⇒ 8 is not in A[]

	0	1	2	3	4	5
A = [	2	10	3	5	1	12]

half = (6+1)/2 = 3

l	r	m	cnt
1	~~12~~	6	4
1	5	3	3

count(2) = 2

Ans = 3

TC = O(N log(A[i]))   SC = O(1)

HW. → Find median of 2 sorted array.
TC < O(N)

A = [1 3 5 6]

B = [2 8 10]

1 2 3 5 6 8 10

(Ans)

// # elements <= m in A[]

TC = ~~O(N)~~ → O(log(N)) (sorted array)

Binary Search - 3

TABLE OF CONTENTS

Painter's Partition Problem

Given N boards with length of each board

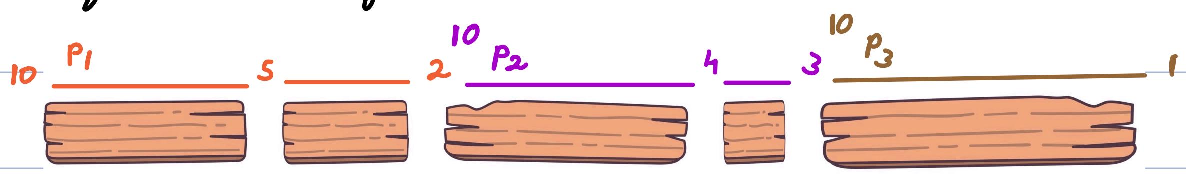
a) A painter takes 1 unit of time to paint 1 unit of length.

b) A board can only be painted by 1 painter.

c) A painter can only paint boards placed next to each other (i.e continuous segment).

A → Find min # painters required to paint all the boards in T unit of time. If not possible return -1.

N = 5



5 3 6 1 9

T = 7 Ans = -1

✓ T = 10 Ans = 3 { 5+3 , 6+1 , 9 }

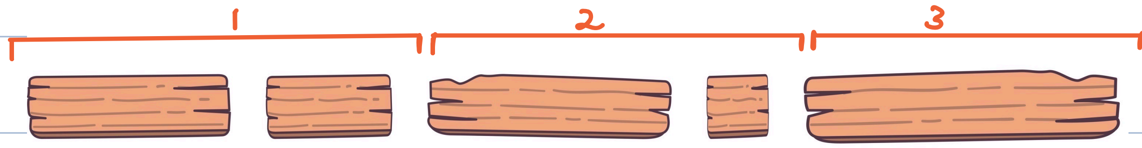
T = 20 Ans = 2 { 5+3+6+1 , 9 }

```
cnt = 1        t = T   → i/p

for i → 0 to (N-1) {
    if ( A[i] > T )  return -1    // INT_MAX
    if ( A[i] <= t )   t -= A[i]
    else {   cnt++
             t = T - A[i]  }
}  return cnt
```

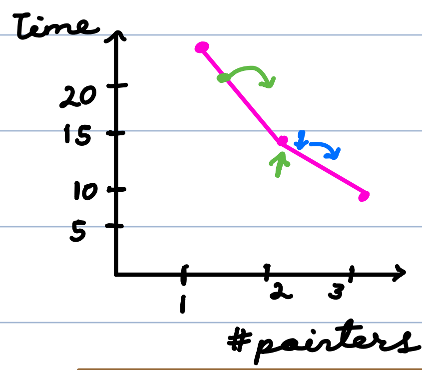
TC = O(N)   SC = O(1)

B → Find min time required to paint all boards if K painters are available.



[5 3 6] [1 9]

# Painters	Minimum Time
1	$5+3+6+1+9 = \underline{24}$
2	$\max(5+3+6,\ 1+9) = \underline{14}$
3	$\max(5+3,\ 6+1,\ 9) = \underline{9}$



$\boxed{\text{Time} \propto 1/_{\#\text{painter}}}$

→ Binary Search on Answer

$A = [\underbrace{1\quad 2\quad 3\quad 4}\ \ \underbrace{100}]$ K = 2

$\max(10\ ,\ 100) = \underline{100}$ (Ans)

$l = \max\limits_{\forall i}(A[i])$ $r = \sum\limits_{\forall i}(A[i])$ // sum ∀i (A[i])

$$\begin{array}{lccccc} & 0 & 1 & 2 & 3 & 4 \\ A = [& 10 & 10 & 10 & 10 & 10\ ] \end{array}$$

K = 4

m = 20 → cnt = 3

m-1 = 19 → cnt = 5

```
while (l <= r) {
    m = l + (r-l)/2
    cnt = countOfPainters(m, A)   // TC = O(N)
    if (cnt <= K && countOfPainters(m-1, A) > K)
        return m
```

if (cnt > K)   l = m+1
else   r = m-1

}

$TC = \underline{O(N \log(\Sigma A[i]))}$

$N <= 10^5$

$A[i] <= 10^9$

$\Sigma A[i] <= 10^9 * 10^5 = \underline{10^{14}}$ long

Example :

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

[3 5 1 7 8 2 5 3 10 1 4 7 5 4 6],   K = 4

$\underbrace{3\ 5\ 1\ 7}_{P1\ 16}\ \underbrace{8\ 2\ 5\ 3}_{P2\ 18}\ \underbrace{10\ 1\ 4}_{P3\ 15}\ \underbrace{7\ 5\ 4}_{P4\ 16}\ \underbrace{6}_{P5}$

l	r	mid	Is it possible to paint all boards by P painters in time = mid?
10	71	40	cnt = 2
10	39	24	cnt = 4 count (m-1) = 4
10	23	16	cnt = 5
17	23	20	cnt = 5
21	23	22 (Ans)	cnt = 4 count (m-1) = 5

Situation:

Imagine you are tasked with developing a system for evenly distributing the workload among a team of email response handlers in a customer service department. Each email is assigned a 'complexity score' which represents the estimated time and effort required to address it. The complexity scores are represented as an array, where each element corresponds to a single email.

Task

The goal is to divide the array into K contiguous blocks (where K is the number of email handlers), such that the maximum sum of the complexity scores in any block is minimized. This approach aims to ensure that no single email handler is overwhelmed with high-complexity emails while others have a lighter load.

$$[\overbrace{3\ 5}^{8}\ \overbrace{1\ 7\ 8}^{16}\ \overbrace{2}^{2}\ \overbrace{5\ 3\ 10\ 1\ 4\ 7\ 5\ 4\ 6}^{45}]$$

$$[\underbrace{3\ 5\ 1\ 7}_{16}\ \underbrace{8\ 2\ 5\ 3}_{18}\ \underbrace{10\ 1\ 4\ 7}_{22}\ \underbrace{5\ 4\ 6}_{15}], \qquad K = 4$$

Sol → Same as above

maximise the minima /

minimise the maxima ⟹ Binary Search

(observation)

Aggressive Cows

< Question > : Farmer has built a bar with N stalls.

A[i] → location of ith stall in sorted order.

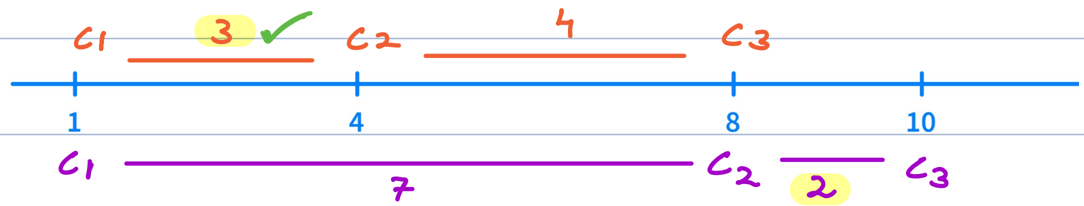
\# Cows ↑ K

$2 \le \cancel{M}\,K \le N$

Cows are aggressive towards each other. So, farmer wants to maximise the minimum distance between any pair of cows. → $|A[i] - A[j]|$

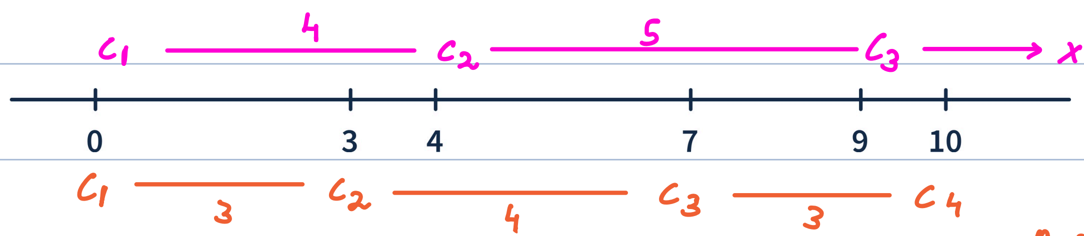
Find $\boxed{\text{max possible}}$ minimum distance.

$$\begin{array}{lcccc} & 0 & 1 & 2 & 3 \\ A[\ ] \to [& 1 & 4 & 8 & 10\ ], \end{array} \qquad \cancel{M}\,K = 3$$



$$\begin{array}{lcccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ A = [& 0 & 3 & 4 & 7 & 9 & 10\ ] \end{array}$$

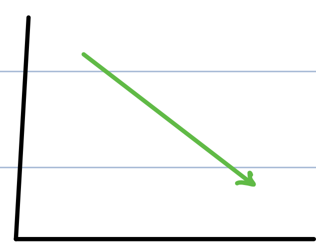
Quiz :



K = 4

Ans = 3

$\boxed{\#\text{cows} \propto 1/\text{minDistance}}$



```
int maxCows(dist, A[]) {
    cnt = 1      p = A[0]
    for i -> 1 to (N-1) {
        if (A[i] - p >= dist) {
            cnt++      p = A[i]
        }
    }  return cnt
}
```

$TC = \underline{O(N)}$

```
l = 1                  r = A[N-1] - A[0]
while (l <= r) {
    m = l + (r-l) / 2      // distance
    cnt = maxCows(m, A)
    if (cnt >= K  &&  maxCows(m+1, A) < K)
            return m
    if (cnt < K)    r = m-1
    else    l = m+1
}
```

$TC = \underline{O(N \log(A[N-1] - A[0]))}$

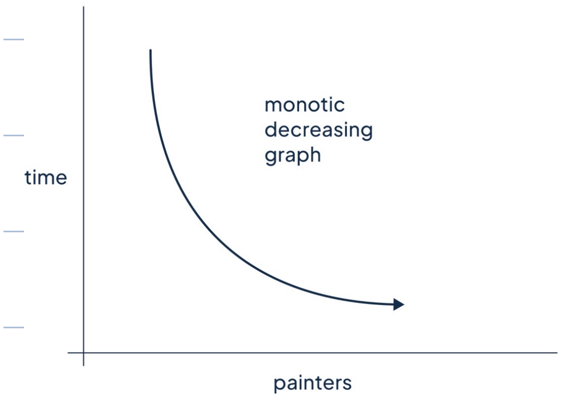
$SC = \underline{O(1)}$

	0	1	2	3	4	5	
A =	[1	3	4	7	11	14]	K = 4
	c_1	x	x	c_2	c_3	x	

l	r	mid	Can we place M cows with distance ≥ mid?	
1	13	7	cnt = 2	
1	6	3 (Ans)	cnt = 5 count(m+1) = 3	

- **Common observation for these two problems :**

Painters Partition



Aggressive – Cows