## Today's Content

- Painters Partition Problem
- Aggressive Cows.

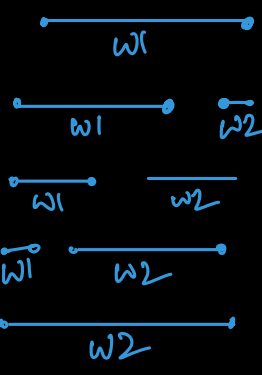
## Qn: Painters Partition Problem

We have to paint n boards of length $A_1, A_2, A_3 \ldots A_N$. There are K painters available & each takes 1 unit of time to paint 1 unit of board.

Find min. time to get the job done.

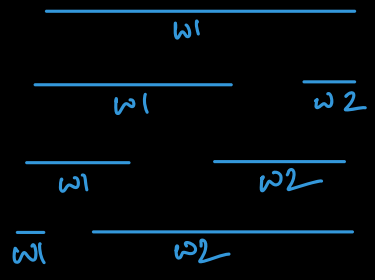
* Note: One painter will paint only continuous section of the board.

Eg- $A = \{10, 10, 10, 10\}$, $K = 2$




| W1 | W2 | |
|---|---|---|
| 40 | 0 | ⇒ 40 |
| 30 | 10 | ⇒ 30 |
| 20 | 20 | ⇒ 20 |
| 10 | 30 | ⇒ 30 |
| 0 | 40 | ⇒ 40 |

$A = 10\ 20\ 30\ 40$, $K = 2$

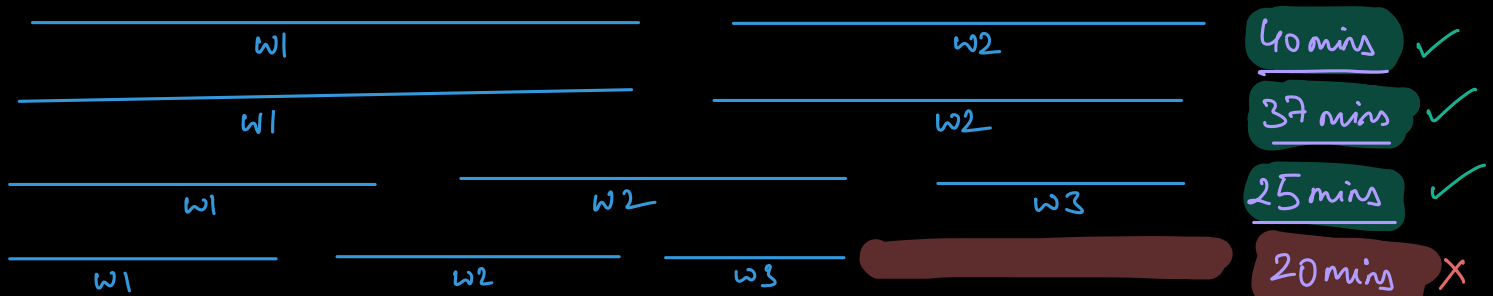



| W1 | W2 | |
|---|---|---|
| 100 | 0 | ⇒ 100 |
| 60 | 40 | ⇒ 60 |
| 30 | 70 | ⇒ 70 |
| 10 | 90 | ⇒ 90 |

"K=3"

Eg:

| 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 |




Ans is somewhere between [20 - 25 mins]

idea: Divide total time by K

a[]: 1 1 1 2 1 100 K=2

$\frac{106}{2} = 53$ → Can't divide work evenly.

Eg: K=4

| 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 |

w1 w2 w3

w1 w2 w3 w4

30 mins?

10 mins? ×

if work can be done in 30 mins

| time | 31 | 32 | 33 | 34 .... |
|---|---|---|---|---|
| possible | T | T | T | T .... |

| time | 7 | 8 | 9 | 10 |
|---|---|---|---|---|
| possible | F | F | F | F |

| time | 7 | 8 | 9 | 10 ..... | 30 | 31 | 32 | 33 ... |
|---|---|---|---|---|---|---|---|---|
| possible | F | F | F | F | T | T | T | T ... |

Job is to find first True

idea: Use binary search

Target: min. time to get job done

Search space: Not the array. We are searching for time.

Range: [min time to max time].

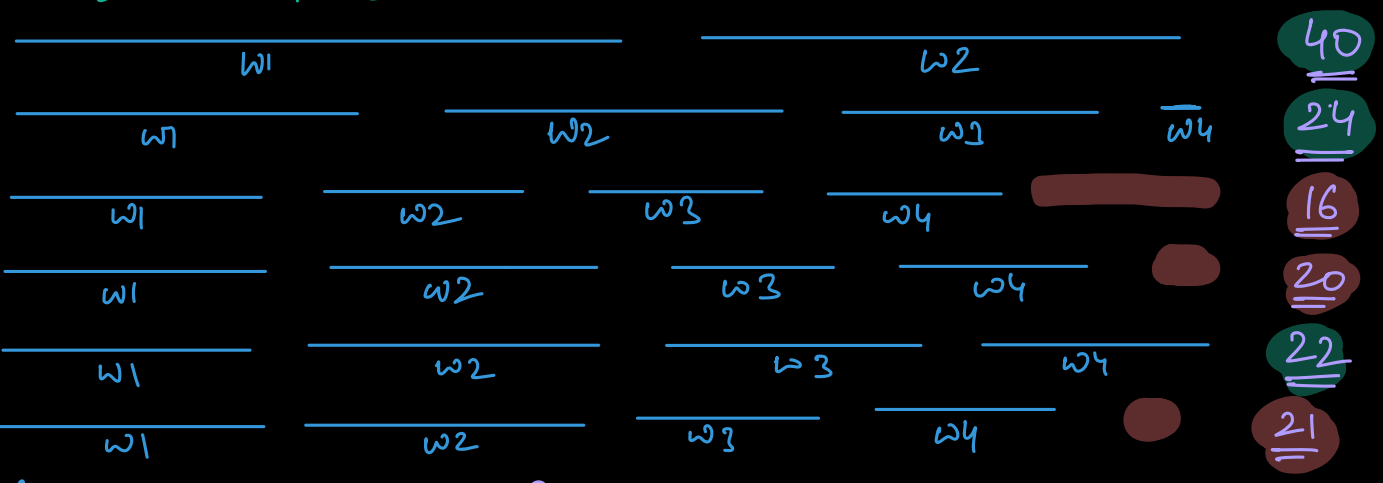
A: 3 2 8 9

∞ - workers ⇒ Max of array. ⇒ Min time.

1 worker ⇒ Sum of array ⇒ Max time.

Eg: K=4

| 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 |




| l | r | m | Possible ans | |
|---|---|---|---|---|
| 10 | 71 | 40 | ans = 40 | ∴ go left r = m-1 |
| 10 | 39 | 24 | ans = 24 | ∴ go left r = m-1 |
| 10 | 23 | 16 | × | ∴ go right l = m+1 |
| 17 | 23 | 20 | × | ∴ go right l = m+1 |
| 21 | 23 | 22 | ans = 22 | ∴ go left r = m-1 |
| 21 | 21 | 21 | × | ∴ go right l = m+1 |
| 22 | 21 | STOP! [l > r] | | |

Pseudocode:

```
int workers ( int time[] , int N, int K) {
    l = max( time[] ) , r = sum ( time[] )
    ans = r
    while ( l <= r) {
        m = (l+r)/2
        if ( check ( m , time, N, K)) { // Possible ans, go left
            ans = m,  r = m-1
        }
        else { // not possible
            l = m+1
        }
    }
    return ans
}
```

```
bool check(int m, int time[], int n, int K) {
    s = 0, no of painters = 1
    for(i = 0; i < n; i++) {
        s = s + time[i]
        if (s > m) {
            no of painters ++
            s = time[i] // give new worker, new time.
            if (no of painters > K) {
                return false
            }
        }
    }
    return true
}
```

↳ time to do the job in

TC: O(n)

SC: O(1)

TC: $O(n + n * \log(r - l)) = O(n \log(r - l))$.

($r$ → sum of time[], $l$ → max of array)

SC: O(1)

**Qn**: Given N cows & M stalls. All M stalls are on x-axis at diff. locations. Place all N cows in such a way that min diff. b/w any 2 cows = maximized.

Note 1: In a stall, only 1 cow can be present.

Note 2: All cows have to be placed.

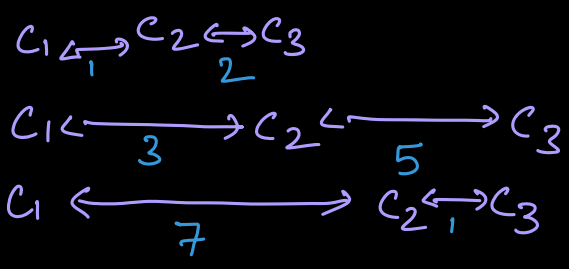
{ stalls[] is sorted }

**Eg:-**

stalls = 5

cows = 3

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 2 | 4 | 8 | 9 |




min dist

1

3

1

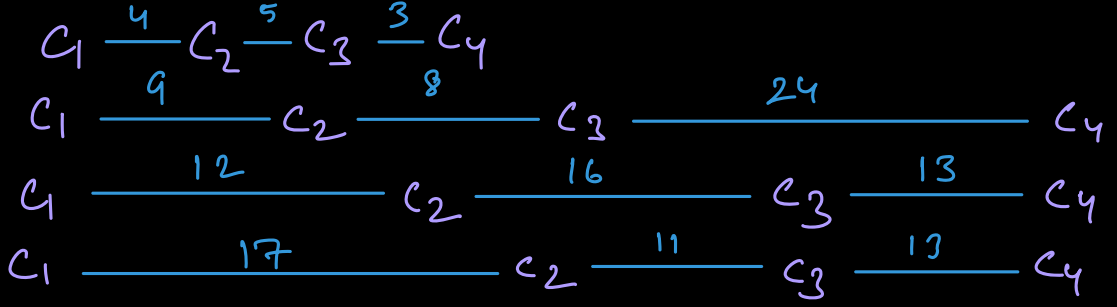
{ We can Keep 3 cows at 3 dist. away }

stalls = 9

cows = 4

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 2 | 6 | 11 | 14 | 19 | 25 | 30 | 39 | 43 |

**Eg2:**




min dist

3

8

12

11

We can Keep the cows atmost 12 dist. away.

stalls = 9

cows = 4

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Eg2: 2 | 6 | 11 | 14 | 19 | 25 | 30 | 39 | 43 |

$C_1$ ——— $C_2$ (23) ✗

$C_1$ — $C_2$ — $C_3$ — $C_4$ ✓

if we can't place cows 23 dist. away:

| dist: | 23 | 24 | 25 | 26 | 27 ... |
|---|---|---|---|---|---|
| Possible: | F | F | F | F | F |

We can place the cows 5 distance apart:

| dist: | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Possible: | T | T | T | T | T |

| dist: | 1 | 2 | 3 | 4 | 5 | ... | 23 | 24 | 25 | 26 | 27... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Possible: | T | T | T | T | T | ... | T | F | F | F | F | F |

↳ Last occurence of T

Binary Search: Target: Max. of min dist

Search Space: Range.

| stalls: | 3 | 4 | 8 |
|---|---|---|---|
| 2 cows | $C_1$ | | $C_2$ |
| | $C_1$ | $C_2$ | |

r = Diff b/w (max - min) in stall[]

l = min. of diff b/w adjacent elements in stall[]

stalls = 9

cows = 4

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Eg2: | 2 | 6 | 11 | 14 | 19 | 25 | 30 | 39 | 43 |
| 22 | $C_1$ | | | | | $C_2$ | | | X |
| 12 | $C_1$ | | | $C_2$ | | | $C_3$ | | $C_4$ |
| 17 | $C_1$ | | | | $C_2$ | | | $C_3$ | X |
| 14 | $C_1$ | | | | $C_2$ | | | $C_3$ | X |
| 13 | $C_1$ | | | | $C_2$ | | | $C_3$ | X |

| l | r | m | Can place cows? |
|---|---|---|---|
| 3 | 41 | 22 | X ∴ go left ⇒ r = m-1 |
| 3 | 21 | 12 | ans = 12, go right : [l = m+1] |
| 13 | 21 | 17 | X ∴ go left ⇒ r = m-1 |
| 13 | 16 | 14 | X ∴ go left ⇒ r = m-1 |
| 13 | 13 | 13 | X ∴ go left ⇒ r = m-1 |
| 13 | 12 | STOP! [l > r] | |

```
int moo(int stall[], int n, int c) {
    l = min of adj elements in stall[]      → O(n)
    r = stall[n-1] - stall[0], ans = l      → O(1)
    while(l <= r) {                         → O(n * log(r-l))
        m = (l+r)/2
        if(check(m, stall, n, c)) { // can place c cows
            ans = m, l = m+1                // m dist away.
        }
        else { // Can't place cows
            r = m-1
        }
    }
    return ans
}
```

```
bool check(int m, int stall[], int n, int cows) {
    lastPlaced = stall[0]
    count = 1
    for(i=1; i<n; i++) {
        if(stall[i] - lastPlaced >= m) {
            lastPlaced = stall[i]
            count++
            if(count == cows) {
                return true
            }
        }
    }
    return false.
}
```

TC: O(n), SC: O(1)

Doubt

l → 1 2 3 ..... INT_MAX ← r

T T T . . . F ✓

$\log_2(INT\_MAX)$

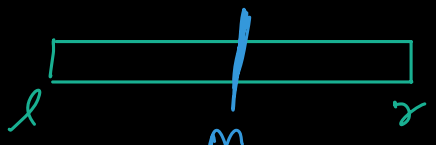
$\log_2(2^{32}) = 32 \approx O(1)$

$\log_2(2^{64}) = 64 \approx O(1)$

$m = (l+r)/2$

$(1 + INT\_MAX)/2$ → Overflow!

$m = l + (r-l)/2$




$l = 15$, $r = 17$ $\left.\begin{array}{l}\\\\\end{array}\right\}$ $\frac{l+r}{2} = 16$

$\left(\frac{r-l}{2}\right) + l = 1 + l = 16$