Wednesday - Binary Search - 4

Friday - 2-pointers (1 Session)

Sunday - 10 AM - (Problems Solving Session)

mostly { — }

Todays - { 2 - Problems / Rotation game }

18) Given N tasks, k workers & time taken for each task, find min time in which we can complete all tasks.

Note1: A single worker can only do continous set of tasks → sort not possible

Note2: All workers start their assigned tasks at same time

Note3: A task can only be assigned to 1 worker

Ex:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	time
N=15 :	3	5	1	7	8	2	5	3	10	1	4	7	5	4	6	

k=3 : (can be anything)

$W_1: 9$ $W_2: 22$ $W_3: 40$ → 40 mins

$W_1: 16$ $W_2: 18$ $W_3: 37$ → 37 mins

$W_1: 26$ $W_2: 23$ $W_3: 22$ → 26 mins

$W_1: 24$ $W_2: 25$ $W_3: 22$ → 25 mins

ans = 25

Ideas:

1) → divide into $\left\{\frac{\text{total Time}}{k}\right\}$ *

Ex1: ar[6] = 1 1 1 2 1 100

k=2 :

$W_1: 6$ $W_2: 100$

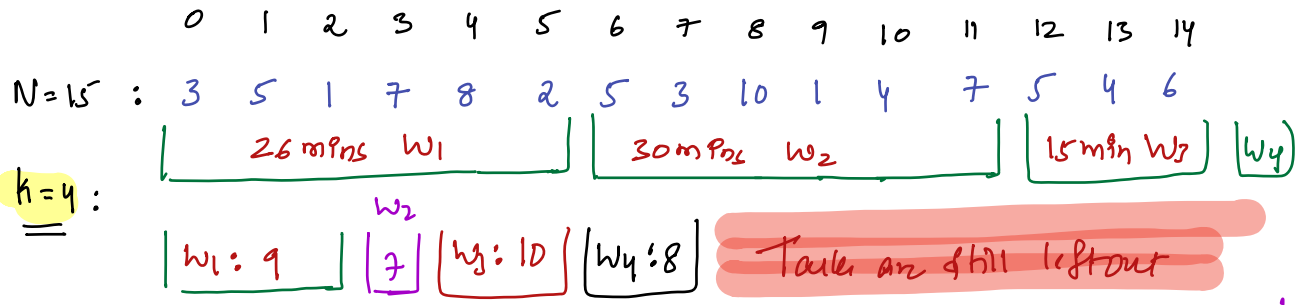
Total time = $\frac{106}{2}$ → 53 → avg time

min 100

5 6 7

days

10 ✓ 11 ✓ 12 ✓ 13 ✓



k = 4 :

$W_1 : 9$	W_2 7	$W_3 : 10$	$W_4 : 8$

Tasks are still leftout

I gave you this?

Can we finish all tasks 30 mins ?

30	31	32	33	. . . ———
T	T	T	T	. ———

I gave you 10

Can we finish all tasks 10 mins?

---- 8	9	10
F F F F F	F	F

Search :

→ Target : { Min time to finish all tasks }

l = 1 h = 7

→ SearchSpace : { Max of arr[] Sum of all Elements }

→ Divide Search Space ? ✓

Ex1:

	0	1	2	3
ar[4] =	3	2	8	9

→ Total

k=1 : 23

k=4 : w1 w2 w3 w4 → 3 2 8 9 = 9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
N=15 :	3	5	1	7	8	2	5	3	10	1	4	7	5	4	6

k=4 :

w1 : 34 | w2 : 37 | w3 w4

w1 : 24 | w2 : 21 | w3 : 20 | w4

w1 : 16 | w2 : 15 | w3 : 14 | w4 : 16 → Take

Same span style: [h-l+1]

w1 : 16 | w2 : 18 | w3 : 15 | w4 : 16 → Leftout

w1 : 16 | w2 : 18 | w3 : 22 | w4 : 15 → left

w1 : 16 | w2 : 18 | w3 : 15 | w4 : 16

[l	h]	m	check if m :
10	71	40	✓ ans = 40, goto left : h = m-1
10	39	24	✓ ans = 24 goto left : h = m-1
10	23	16	* 14 15 16, goto right : l = m+1
17	23	20	* 18 19 20, goto right : l = m+1
21	23	22	✓ ans = 22, goto left : h = m-1
21	21	21	x goto right, l = m+1

22 21 * {Break} → {ans = 22}

10:38 pm

Pseudocode: → BS → m time

tasks, workers

```
int workers(int time[], int N, int k){
    l = max of time[], h = sum of time[], ans = ___
    while(l <= h){
        m = (l+h)/2        // whether we can all task in m time
        if(check(m, time, N, k)){        // TC: O(N)
            // we can do in m time
            ans = m, h = m-1
        }
        else{
            // we cannot do in m time
            l = m+1
        }
    }
    return ans;
}
```

Total TC: check fn → Binary Search * N

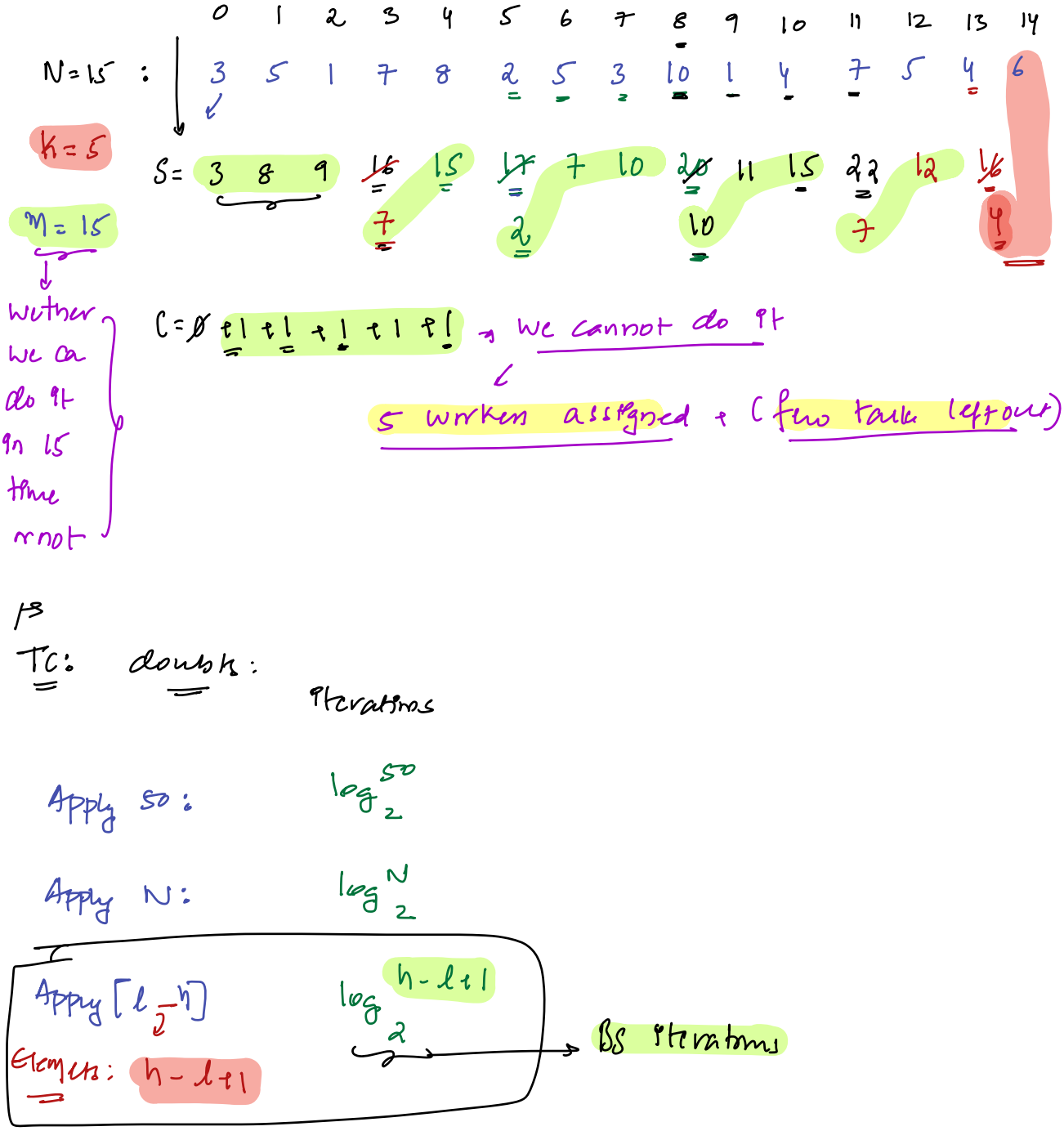
$\log(h-l+1) * N$ (base 2)

TC: $\log(\text{sum}() - \text{max}() + 1) * N + N$ (base 2)

SC: O(1) { Sum of all, Max of all }

```
bool check(int m, int time[], int N, int k){
    s=0, c=0
    for(i=0; i<N; i++){
        s = s + time[i]        // we are initially? task to previous worker
        if(s > m){ // last assigned task we need to add to new worker
            c = c+1
            s = time[i]
            if(c == k){return false;}        // we used all our workers & still tasks are left out
        }
    }
    return true
}
```

TC: O(N)



BS

TC: doubts:

Iterations

Apply SO: $\log_2^{SO}$

Apply N: $\log_2^{N}$

Apply $[l \frac{}{2} h]$ $\log_2^{h-l+1}$ → BS iterations

Elements: $h - l + 1$

28) Given N cows & M stalls, all M stalls are in x-axis at different locations, Place all N cows such a way min distance between any 2 cows is maximized?

Note1: In a stall only 1 cow can be present

Note2: All Cows have to be placed { stalls[] is sorted } ↳ if stall[] not sorted → Sort it

Ex1:

#stalls = 5

Cows = 3

0	1	2	3	4
1	2	4	8	9

$C_1 \leftrightarrow C_2 \leftrightarrow C_3$ (1, 2)

$C_1 \leftrightarrow C_2 \leftrightarrow C_3$ (3, 5)

$C_1 \leftrightarrow C_2 \leftrightarrow C_3$ (7, 1)

We can keep all the cows atleast at 3 distance away?

min distance:

1
3
1

maximize

ans = 3

min distance between cows

Ex2:

#stalls = 9

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

Cows = 4

We can keep all the cows atleast at k distance away.

- $C_1 \overset{4}{\leftrightarrow} C_2 \overset{5}{\leftrightarrow} C_3 \overset{3}{\leftrightarrow} C_4$ — 3
- $C_1 \overset{7}{\leftrightarrow} C_2 \overset{8}{\leftrightarrow} C_3 \overset{24}{\leftrightarrow} C_4$ — 8
- $C_1 \overset{12}{\leftrightarrow} C_2 \overset{16}{\leftrightarrow} C_3 \overset{13}{\leftrightarrow} C_4$ — 12
- $C_1 \overset{17}{\leftrightarrow} C_2 \overset{11}{\leftrightarrow} C_3 \overset{13}{\leftrightarrow} C_4$ — 11

Eg2:

	0	1	2	3	4	5	6	7	8
Stalls = 9	2	6	11	14	19	25	30	39	43
Cows = 4	c_1					c_2			c_3
	c_1		c_2		c_3	c_4			

y → left most

distance between adj → 20

Check if we can place cows at least 20 distance

goto left space ← | 20 | 21 | 22 | 23 | ... |
|---|---|---|---|---|
| F | F | F | F | |

distance between adj → 7 = 5

check if we can place cows atleast 5 distance

-	-	3	4	5
T	T	T	T	T

} update ans, goto right

Search:

Target: max of minimum dist between 2 cows

l_0 h_1

SearchSpace: { Min dist between 2 adj Stalls : last - first, $\{ar[N-1] - ar[0]\}$ }

Whether we can discard n not?

T T T T T T T T F F F F F F F F F

dis is low true

Ex1:

3 stalls: 0 1 2 — 3 4 8

2 cows: c_1 ⟷ c_2

5 → ans ✓

Ex2: 4 stalls 0 1 2 3 — 3 5 6 10

4 cows: c_1 — c_2 → c_3 — c_4 ans = 1

Stalls = 9	0	1	2	3	4	5	6	7	8
	2	6	11	14	19	25	30	39	43

Cows = 4

c_1 c_2 c_3 c_4

c_1 c_2 c_3 c_4

c_1 c_2 c_3 c_4

c_1 c_3 c_4

c_1 c_2 c_3 c_4

l	h	m	check m	
3	41	22	×	goto left, h = m-1
3	21	12	✓	ans = 12, goto right l = m+1
13	21	17	×	goto left, h = m-1
13	16	14	×	goto left, h = m-1
13	13	13	×	goto left h = m-1
13	12	Break		return ans = 12

Pseudocode

Stalls → N, # Cows → C

```
int maxDist(int dist[], int N, int C)
{
    l = { min adj dist in dist[] }   h = dist[N-1] - dist[0],  ans: ____

    while(l <= h){
        m = (l+h)/2          // m: distance; check if we can place cows atleast at m distance
        if(check(m, dist, N, C)){
            // if we can place them atleast at m dist
            ans = m;  l = m+1
        }
        else{
            // if we cannot place them atleast at m dist
            h = m-1
        }
    }
    return ans;
}
```

TC: Binary Search iterations * N → log(h-l+1) * N

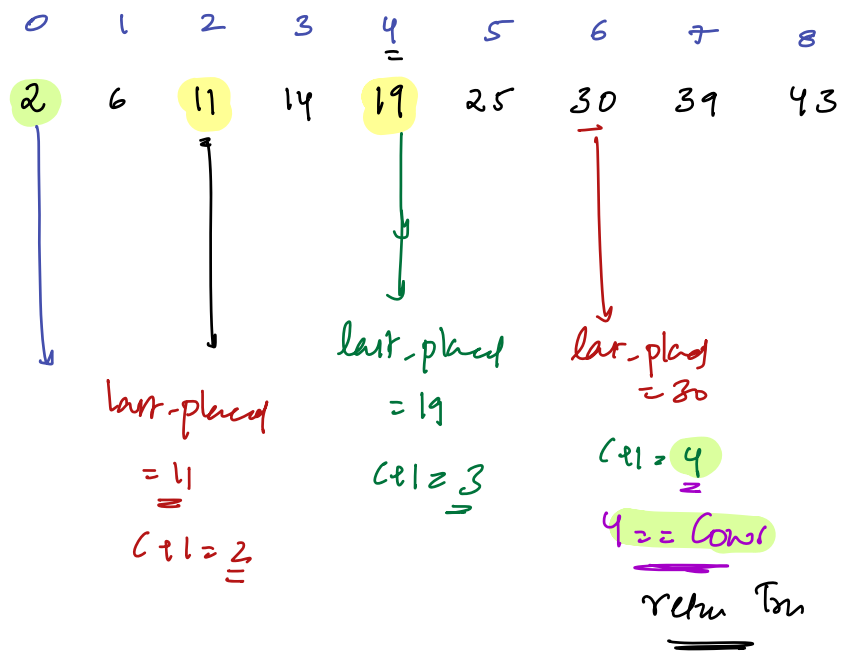
TC: log(h-l+1) * N + N (min adj dist)

SC: O(1)

```
bool check(int m, int dist[], int N, int Cows){
    last_placed = dist[0]
    cout = 1;
    for(i = 1; i < N; i = i+1){          // last_placed: dist[i]
        if(dist[i] - last_placed >= m){
            c = c+1;  last_placed = dist[i]
            if(c == Cows){return True}
        }
    }
    return false
}
```

TC: O(N)

Idea:



→ Wednesday → { 4th Binary Search }

↳ Tomorrow it will Create