Todays Content:

- → Workers allocation
- → Aggresive Sheeps

	W	Thu	Fri	Sat	Sun	M	Tue	Wed	Thur	Friday
Date:	19th	20th	21st	22nd	23rd	24th	25th	26th	27th	28th
	✓		✓							✓

Ques: Given N tasks, k workers & time taken for each task, find **min time** in which we can completed all tasks

→ no sorting can be done, order lost

Notes: A single worker can only do continious set of tasks

All workers start their assigned tasks at same time

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 taken															
N = 15 :	3	5	1	7	8	2	5	3	10	1	4	7	5	4	6

k = 3

$W_1 = 44$ | $W_2 = 12$ | $W_3 = 15$ → = 44 mins

$W_1 = 24$ | $W_2 = 25$ | $W_3 = 22$ → = 25 mins

→ min time to finish all tasks = 25 mins

Ex2:

	0	1	2	3	4	5
ar[6] =	1	1	1	1	1	101

k = 2 : $W_1 = 5$ | $W_2 = 101$

ans = 101

avg = 53 ≠ ans = 101

Ideal: Given N tasks & k works * not working

Calculate avg time = Total time / k

Try to allocate avg slot to everyone

Idea2: a) Target: min time to finish all tasks

b) search space: [low = max of arr[] , high = sum of arr[]]

Eg1: ar[4] = { 3 9 2 8 } (indices 0 1 2 3)

k = 4

w_1 w_2 w_3 w_4

3 [9] 2 8

↳ 9 min to finish task

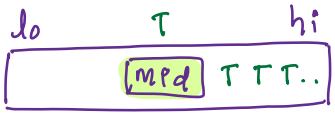
ar[4] = { 3 9 2 8 }

k = 1

w_1 = 22

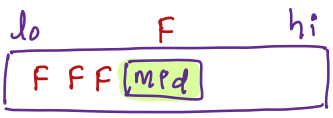
↳ 22 min to finish all tasks

c) discard:



if we can do the task in mid time

: ans = mid goto left



if we cannot do the task in mid time

: goto right

Idea:

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

k = 4

w_1 = 26 | w_2 = 30 | w_3 = 15 | w_4 = 9dle

w_1 = 9 | 7 | 10 | 8 | Task are remaining

w_1 w_2 w_3 w_4

Can it finished in 10 min?

.. 7 8 9 10

F F F F F

↑ : goto right

Can it finished in 30 min?

30 31 32 33

T T (T) T T T T

↑ update ans

goto left

Tracing

	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

$W_1 = 34$ | $W_2 = 37$ | $W_3 = W_4 = 40$ ✓

$W_1 = 24$ | $W_2 = 21$ | $W_3 = 20$ | 6 (W_4) ✓ Possible 24

$W_1 = 16$ | $W_2 = 15$ | $W_3 = 14$ | $W_4 = 16$ | 16 not poss

$W_1 = 16$ | $W_2 = 18$ | $W_3 = 15$ | $W_4 = 16$ | 20 ✗ not

$W_1 = 16$ | $W_2 = 18$ | $W_3 = 22$ | $W_4 = 15$ | 22 possible

$W_1 = 16$ | $W_2 = 18$ | $W_3 = 15$ | $W_4 = 16$ | 21 ✗ not poss

l	h	m, Can we finish tasks in m time	ans = ____
10	71	40 ✓ : ans = mid, goto left h = m-1,	ans = 40
10	39	24 ✓ : ans = mid, goto left h = m-1,	ans = 24
		24 25 26 27 . . .	
		T T T T	
10	23	16 * goto right l = m+1	14 15 16
			F F F
17	23	20 * goto right l = m+1	
21	23	22 ✓ : ans = mid, goto left h = m-1,	ans = 22
21	21	21 * goto right l = m+1	
22	21	: { l > h } break it	return ans = 22

Binary Search Iterations:

Elements	→	Iterations
N		$\log_2 N$
x		$\log_2 x$

If BS: [l h]

Elements: $h - l + 1$

→ Iterations: $\log_2 (h - l + 1)$

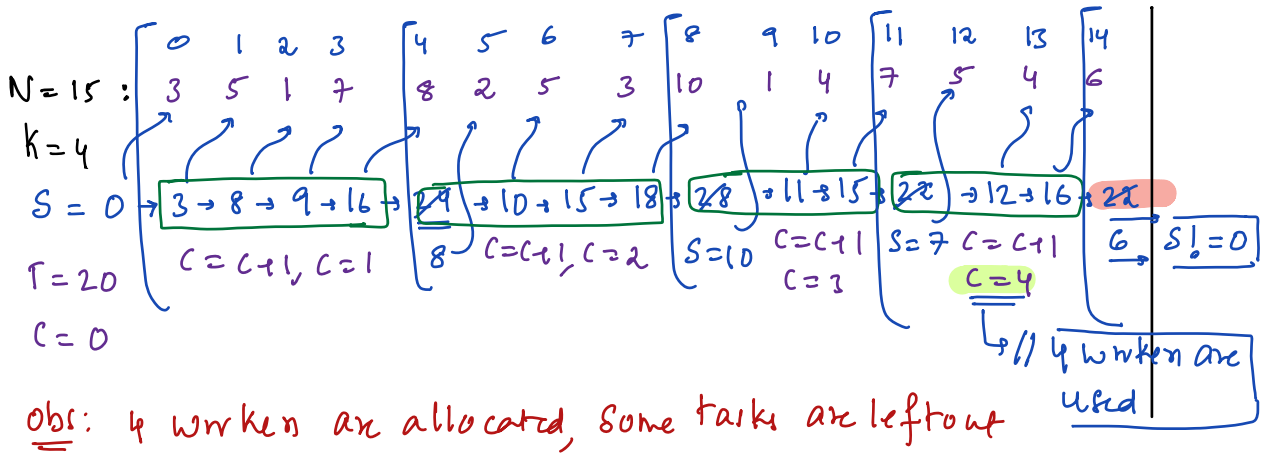
Trace & Pseudocode

```
int minTime(int N, int k, int time[]){   // N: tasks, k: workers, time[]: time for each task
    lo = max of time[]  hi = sum of time[]   } ⇒ TC: O(N)
    ans = _; // any initialization
    while(lo <= hi){
        m = (lo+hi)/2
        // Check if we can finish all tasks in m time
        if(check(N, k, time[], m) == True){
            // We can do tasks in m time, update & go left
            ans = m,
            hi = m-1
        }else{ // We cannot do tasks in m time, goto right
            lo = m+1
        }
    }
    return ans,
}
```

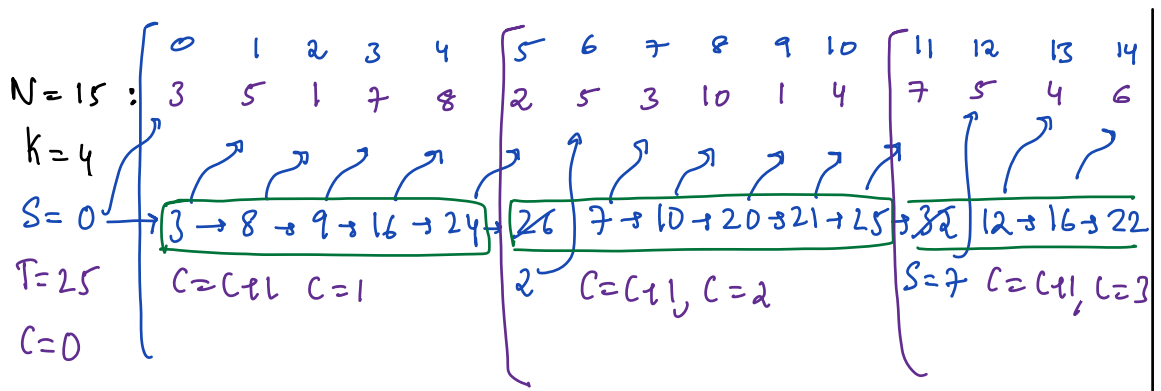
TC: $N + \log_2^{hi-lo+1} * O(N)$

TC: $O(N \log_2^{hi-lo+1})$ SC: $O(1)$

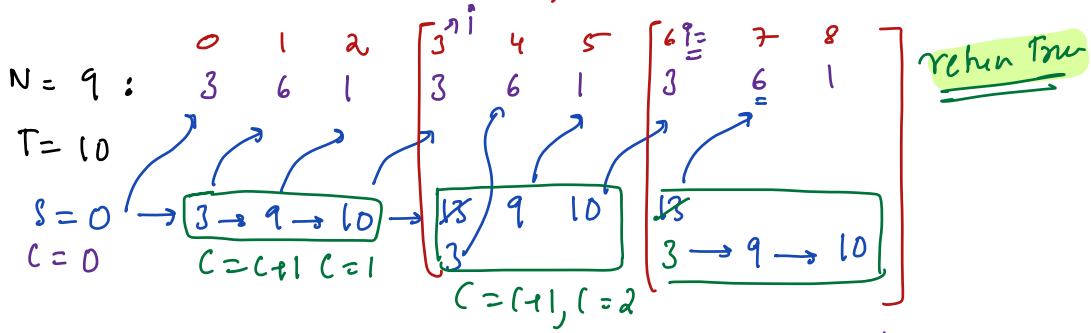
```
bool check(int N, int k, int time[], int T){   TC: O(N) SC: O(1)
    // We need to finish in all tasks <= T
    S=0, C=0
    i=0; i<N; i++){
        // allocate ith task to current worker
        S = S + time[i]
        if(S > T){ // ith task should be allocated to next person
            C=C+1, S = time[i]
            if(C == k && S > 0){ return False;   // not needed, it will work.
            // k workers used   // remaining task
        }
    }
    return True
}
```



Obs: 4 workers are allocated, some tasks are leftout



Obs: Able to do in 3 workers, return True



2@) Given K cows & N stalls, all stalls are on x-axis at different locations, Place all K cows such a way min distance between any {2 cows} is maximized, {Maximize min dist}

Note1: In a stall only 1 cow can be present

Note2: All cows have to placed, N>=K, & stall pos are sorted
stalls >= cows
↳ if not please sort

Ex1:

0	1	2	3	4	mindistance
1	2	4	8	9	

Stalls = 5
Cows = 3

$C_1 \xleftrightarrow{1} C_2 \xleftrightarrow{2} C_3$

$C_1 \xleftrightarrow{3} C_2 \xleftrightarrow{5} C_3$

$C_1 \xleftrightarrow{7} C_2 \xleftrightarrow{1} C_3$

1, Inc ?

3, 1 } increase min distance between cows

ans = 3

Ex2:

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

Stalls = 9
Cows = 4

$C_1 \xleftrightarrow{4} C_2 \xleftrightarrow{5} C_3 \xleftrightarrow{3} C_4$ → 3

$C_1 \xleftrightarrow{9} C_2 \xleftrightarrow{8} C_3 \xleftrightarrow{24} C_4$ → 8

$C_1 \xleftrightarrow{12} C_2 \xleftrightarrow{16} C_3 \xleftrightarrow{13} C_4$ → 12

// ans = 12

Idea2: Target : max dist we can keep them apart

SearchSpace : low : min adj diff in stalls[]

high : stall[N-1] - stall[0]

Eg1: ar[4] = { 2 6 9 14 }

cows = 4

C1 C2 C3 C4

2 ↔ 6 : 4 ; 6 ↔ 9 : 3 ; 9 ↔ 14 : 5

// ans = 3

ar[4] = { 2 6 9 14 }

cows = 2

C1 ↔ C2 : 12

// ans = 12

Discard:

T

lo T-1 T mid hi

: Say we can keep cows atleast mid distance apart : ans = mid, goto right

F

lo mid FFF hi

Say we cannot keep cows atleast mid distance apart : goto left

Trace :

	0	1	2	3	4	5	6	7	8
Stalls = 9	2	6	11	14	19	25	30	39	43
Cows = 4	C1					C2			C3 C4

C1 —9— C2 —8— C3 —11— C4 : min dist : 8, 7 ✓

Atleast place them apart by 7

... 5 6 7

T T T

↑ ans = mid

goto right

Atleast place them apart by 20

20 21 22 23 ....

F F F F

↑ : goto left

Trace:

	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_4 22 ✗

C_1 ←12→ C_2 →16→ C_3 →13→ C_4 12 ✓

C_1 →17→ C_2 →20→ C_3 → C_4 17 ✗

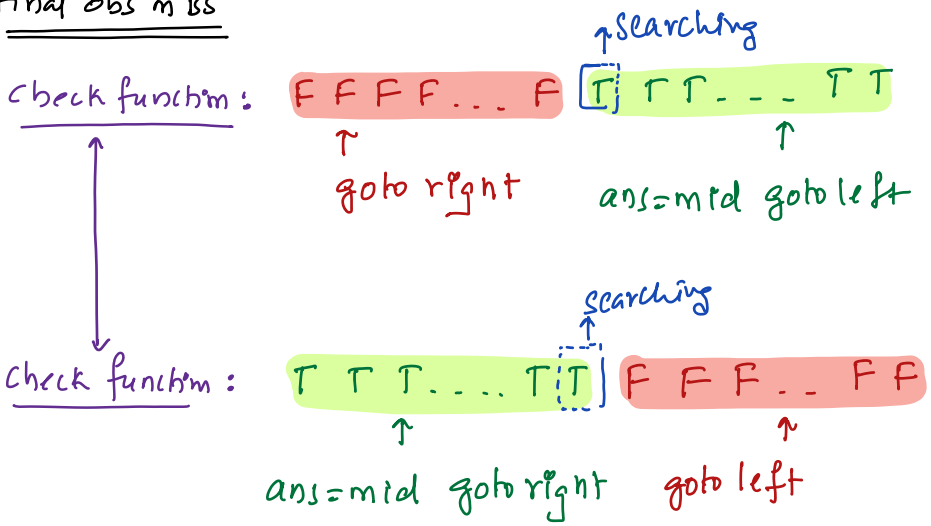
l	h	m : Can we place cows atleast m dist apart	ans
3	41	22 : ✗ 22 23 24 ... goto left h = m-1 (F F F)	
3	21	12 : ✓ ... 10 11 12 ans = m, goto right l = m+1 (T T T)	12
13	21	17 : ✗ goto left, h = m-1	
13	16	14 : ✗ goto left, h = m-1	
13	13	13 : ✗ goto left, h = m-1	
13	12 :	break return ans = 12	

```
int maxdis(int N, int k, int dist[]){   // N: # stalls, k: # cows, dist[]: position of all stalls
    l = min adj diff in dist[]   h = dist[N-1] - dist[0],  ans = ____

    while (l <= h){

        m = (l+h)/2
        //check if we can place all k cows atleast at
          m distance apart
        if (check(N, k, dist[], m) == true){
            // we can place them at m distance
            ans = m; goto right
            l = m+1
        }
        else // we cannot place them at m distance
            h = m-1
        }
    }
    return ans;
}

bool check(int N, int k, int dist[], int d){ TODO: TC: O(N)
                                              // N: Stalls, k: Cows, dist[]: post of N stalls, d: place cows atleast at d distance

}
```

TC: $N + \log_2^{h-l+1} * N$

TC: $O(N \log_2^{h-l+1})$ SC: $O(1)$

final obs n BS



Note: In general question say min/max/ first/ last, binary search might work

↳ But we can never say for sure

a) Target b) Search Space c) Discard

```
bool check(int N, int k, int dist[], int d){
                ↳ Stalls  ↳ Cows  ↳ post of N stalls
    last_placed = dist[0] // placing a cow at 1st stall
    c = 1 //
    i=1; i<N; i++){
        if( dist[i] - last_placed >= d){
            // we can place another cow at dist[i]
            last_placed = dist[i], c=c+1
            if(c==k){ // if we place k cows return True
                return True
            }
        }
    }
    return false
}
```