Todays Content

a) Workers Allocation

b) Aggresive Cows

Apply Binary Search

a) Target

b) Search Space

c) Discard 1 half of search space

Q1 Given N tasks, k workers & time taken for each task, find min time in which we can complete all tasks.

Note: A single worker can only do continous set of tasks
We cannot change order of tasks
A task can only be assigned to 1 worker
A workers can take multiple tasks
All workers start their assigned tasks at same time

Eg:

	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 :	$W_1 = 34$									$W_2 = 11$	$W_3 = 26$					34
	$W_1 = 26$							$W_2 = 19$			$W_3 = 26$					26
	$W_1 = 24$							$W_2 = 25$					$W_3 = 22$			25

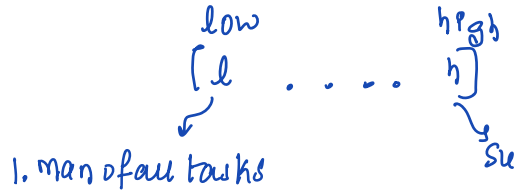
Eg2:

	0	1	2	3	4	5	Time Taken
ar[6] =	1	1	1	1	1	101	
k=2	$W_1 = 3$			$W_2 = 103$			103
	$W_1 = 5$			W_2 101			101

Let's Search ?

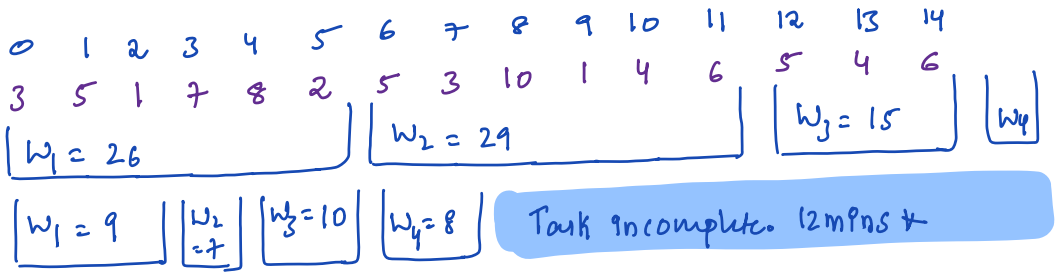
Target: Min time to finish all tasks

Search Space : Answer space : Range in which we are 100% Sure it will contain our ans.



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

K = 4



Can it finished in 12 mins ? → Eg:

. . . 10 11 12

F F F

goto right

Can it finished in 30 min ? → Eg:

30 31 32 33 34 . . .

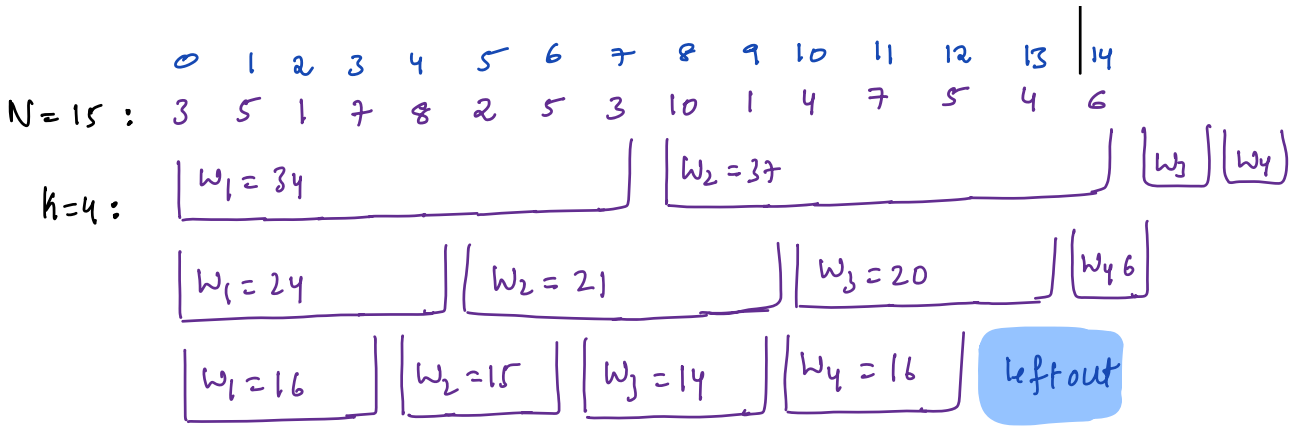
T T T T T

ans = 30, goto left

Discard:

if mid possible: ans = mid, goto leftside

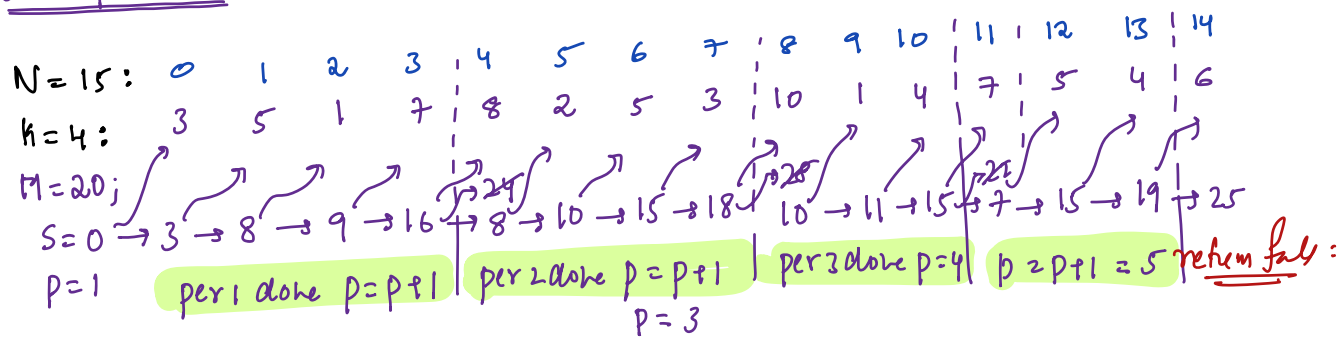
if mid not possible: goto right



↳ (10+71)/2 = 680/2 = 40.

l	h	m, Can we finish tasks in m time	ans = ____
10	71	40, 40 possible:	ans = 40, goto left
10	39	24 24 possible:	ans = 24, goto left
10	23	16 16 not possible	goto right
17	23	... continue process.	

Check func Idea:



Trace & Pseudocode

```
int minTime( int N, int k, int time[] ) {   // [l..h] ele = h-l+1
    int l = max of given time[]
    int h = sum of given time[]
    int ans = h; // Because it's minimal we assign opposite.
    while( l <= h ){
        int m = (l+h)/2 ; // Check if we can finish task within m time
                          with k workers.
        if( check( time, k, m ) {
            ans = m;
            h = m-1;
        }
        else{
            l = m+1;
        }
    }
}
```

// Tasks // Workers → Time for each task

Binary Search iterations = $\log_2(h-l+1) \approx N$

```
boolean check( int time[], int k, int m ) {   TC: O(N) : TODO
    int n = time.length;
    int s=0, p=1;
    for(int i=0; i<n; i++) {
        s = s + time[i]; // assing i^th task.
        if( s > m ) { // exceeding limit, re-assign i^th Task to new person
            p = p+1; // person completed goto new person
            s = time[i]
        }
        if( p > k ) { return false; }   // Used all k workers, yet some unfinished tasks are done, hence we cannot compute tasks.
    }
    return true;
}
```

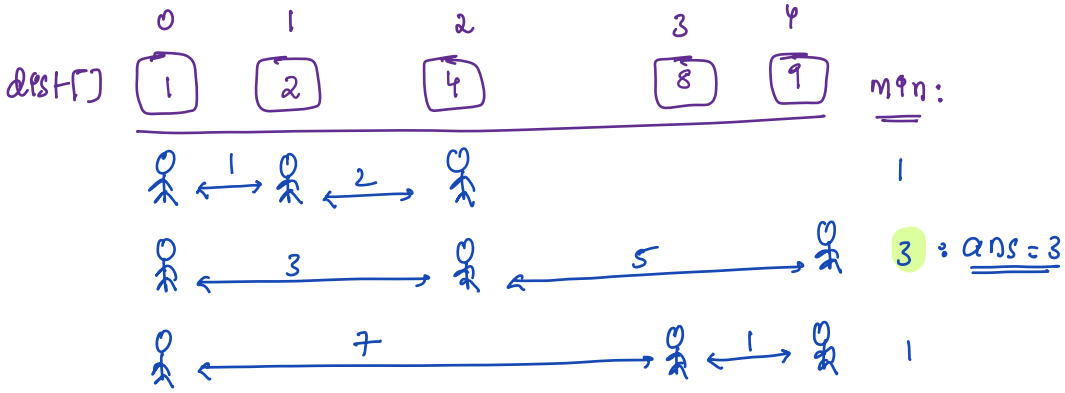
2d) Given K cows & N stalls, all stalls are on x-axis at different locations, Place all K cows such a way min distance between any 2 cows is maximized, Maximize min dist

Note1: In a stall only 1 cow can be present

Note2: All cows have to placed, N ≥ K, & stall pos are sorted
↳ if not sorted sort it

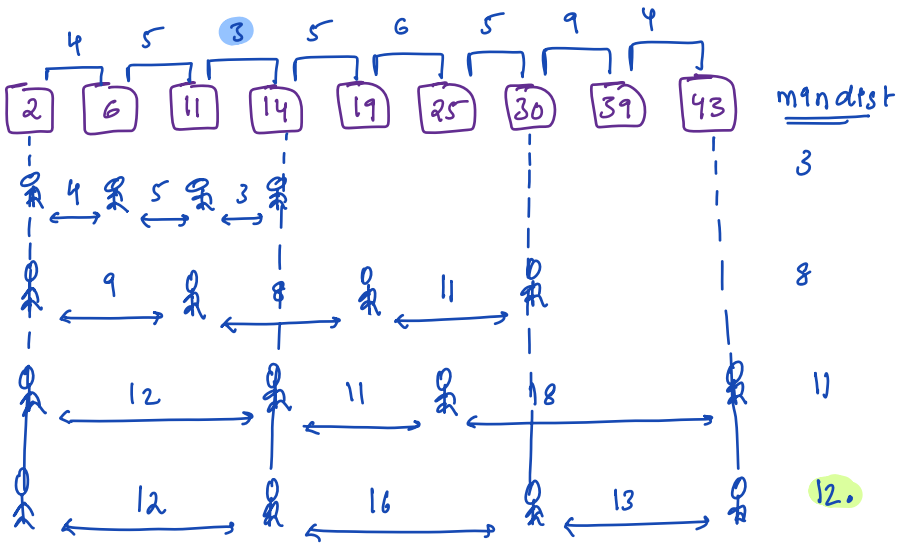
Ex1:

Stalls = 5
K = 3



Ex2:

Stalls = 9
Cows = 4



Idea: Target : maximize, minimum dist between any 2 cows.

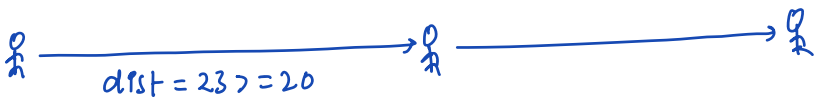
SearchSpace : Answer space : Range in which we are 100% sure it will contain our ans.

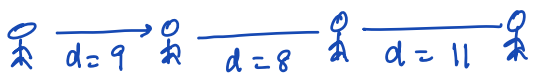
low high
[l .... h]

[min of all adj differences] [last_stall - first_stall]

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

Cows = 4





Atleast place them apart by 7

..3 4 5 6 7

T T T T T

ans=7

goto right

Atleast place them apart by 20

20 21 22 23....

F F F F

goto left

discard:

mid dist is poss: ans=mid goto right

mid dist not poss: goto left;

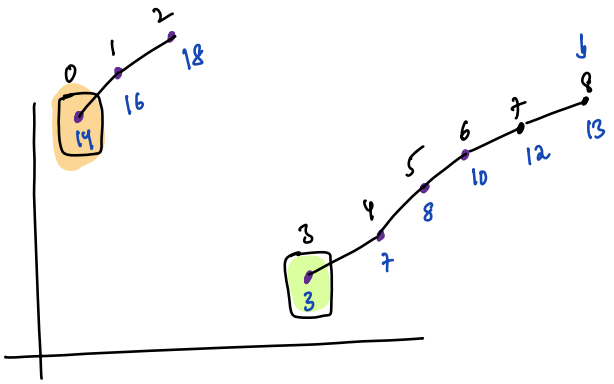
```
int maxdis(int N, int k, int dist[]){
    Arrays.sort(dist);
    int l = min adj diff;
    int h = dist[N-1] - dist[0];
    int ans = l // Because need to get max.
    while(l<=h){
        int m = (l+h)/2;
        Check if can place cows atleast m dist apart.
        if(check(m, dist, k)){
            ans = m;
            l = m+1
        }else{
            h = m-1;
        }
    }
}
```

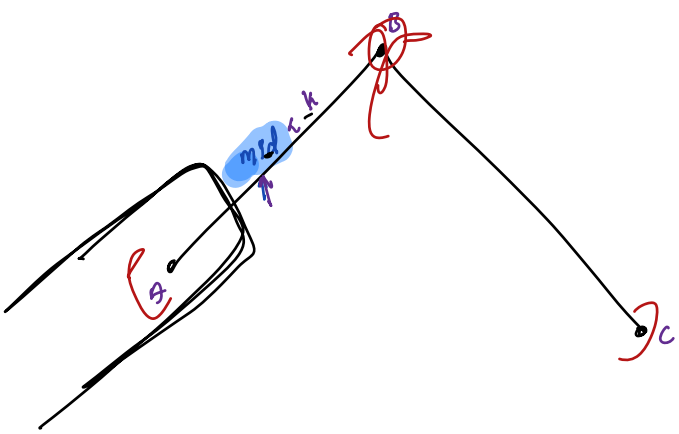
TODO

Check function:

	0	1	2	3	4	5	6	7	8
Stalls = 9	2	6	11	14	19	25	30	39	43
Cows = 4	↓			↓		↓		↓	
m = 10	cows = 1			cows = 2		cows = 3		cows = 4.	

Note: Compare current pos with last placed.


0
1
2
18
16
14
3
4
5
6
7
8
9
13
12
10
8
7
3


B
A
C