Todays Content:

→ Workers allocatm

→ Aggresive Cows

→ ∞ Array[]

1Q Given N tasks, k workers & time taken for each task, find min time in which we can completed all tasks

Notes : A single worker can only do continous set of tasks → no sorting

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

				Time Taken
k = 3 :	W1 = 9	W2 = 17	W3 = 45	45
k = 3 :	W1 = 24	W2 = 21	W3 = 26	26
k = 3 :	W1 = 24	W2 = 25	W3 = 22	25

Ex2:

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

k = 2 W1 = 5 W2 = 101

Ideas:

→ avg time

→ N/k length subarrays

Idea: In N-1 Empty Slots, We need to keep k-1 Sticks

$\#ways = {}^{N-1}C_{k-1} * N$ → Time taken to get amount of work assigned to each perm

To distribute tasks to workers

→ opt: TC: $N + {}^{N-1}C_{k-1} * k$

Idea2: Binary Search:

a) :Target →: Min time to finish tasks

b) Search Space : $\begin{bmatrix} \underline{\text{low}} & \underline{\text{high}} \\ \text{max of time} & \text{sum of all times} \end{bmatrix}$

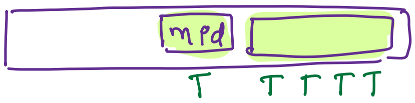
Ex1: ar[4] = {10 2 9 8}

k=1 , Time taken = 29

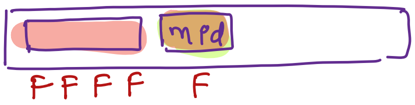
Ex2: ar[4] = $\{\underbrace{10}_{w_1}\ \underbrace{15}_{w_2}\ \underbrace{11}_{w_3}\ \underbrace{13}_{w_4}\}$

k=4 : Time taken = 15

c) discard:



Case-I: If we can do task in mid time :
ans = mid goto left



Case-II If we cannot do task in mid time :
goto right

	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 $\quad$ W1 = 26 $\quad$ W2 = 30 $\quad$ W3 = 15 $\quad$ W4 = { }

k=4 $\quad$ W1 = 9 $\quad$ 7 $\quad$ W3=10 $\quad$ W4=8 $\quad$ Tasks are left out

Can it finished in 10 min?

. . . . . 8 9 10

F F F F F

Can it finished in 30 min?

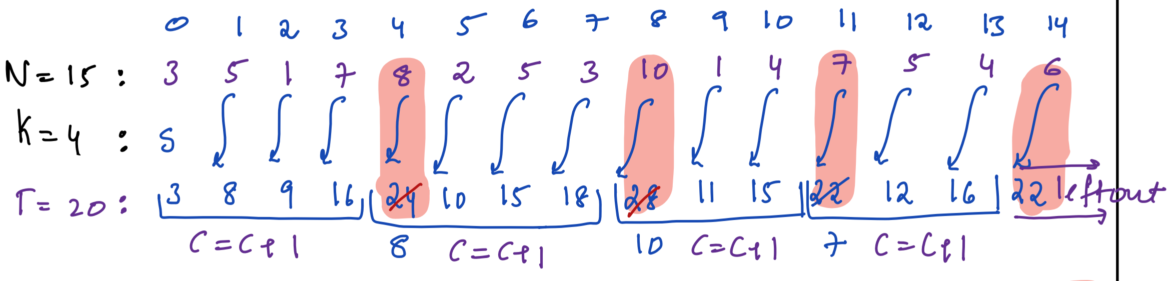
30 31 32 33 . . .

T T T T . . .

Trace & Pseudocode

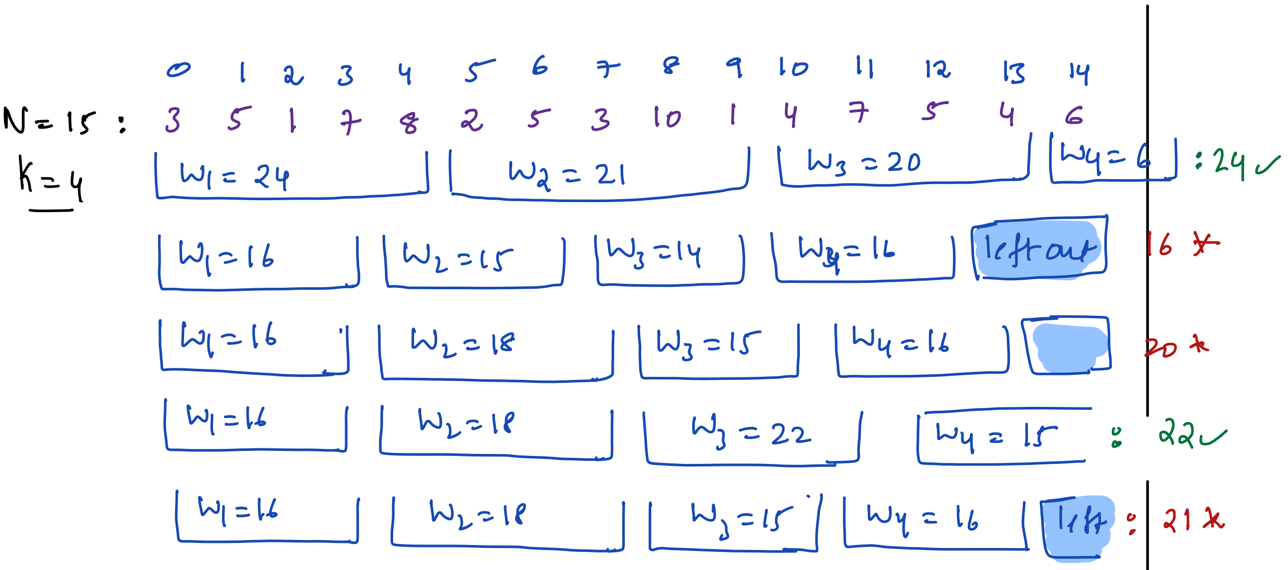
```
                              → // Tasks   → // workers   → Time for each task
int mintime( int N, int k, int time[] ){     TC: log_2(ele) * N
    l = max of arr[]    h = sum of arr[]    ans = ____
    while( l <= h ){                         #ele on which we
                                             apply BS = h-l+1
        m = (l+h)/2                          ele ≈ [ sum of arr[] ele
        // Can we finish N tasks in m time           - max of arr[] ]
        if( check( N, k, time[], m) == True ){
            ans = m;   // goto left
            h = m-1
        }
        else { // not possible goto right
            l = m+1
        }
    }
    return ans
}

bool check( int N, int k, int time[], int T ){     TC: O(N)  SC: O(1)
    c = 0, s = 0
    i = 0; i < N; i++){
        s = s + ar[i]   // add task time
        if( s > T ){ // re-assign ar[i] task to another worker
            c = c+1,  [ s = ar[i] ]
            if( c == k ){ // utilize k worker but task left?
                return False                  S≠0, some work
            }                                 is to be done
        }
    }
    return True
}
```

Check func Idea:



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 : S
T = 20 : 3 8 9 16 ~~24~~ 10 15 18 ~~28~~ 11 15 ~~22~~ 12 16 22 leftout
C = C + 1
8
C = C + 1
10
C = C + 1
7
C = C + 1

obs: Even after utilizing 4 people tasks are leftout return False



l	h	m,	Can we finish tasks in m time	ans = ___
10	71	40	✓	ans = 40, goto left h = m - 1
10	39	24	✓	ans = 24, goto left h = m - 1
10	23	16	✗	goto right l = m + 1
17	23	20	✗	goto right l = m + 1
21	23	22	✓	ans = 22, goto left h = m - 1
21	21	21	✗	goto right l = m + 1

22 21 { Break } return ans = 21

2Q) 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 please sort

Ex1:

	0	1	2	3	4	mindistance
Stalls = 5	1	2	4	8	9	
Cows = 3						

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

$C_1 \xleftrightarrow{2} C_2 \xleftrightarrow{4} C_3 \longrightarrow 2$

$C_1 \xrightarrow{3} C_2 \xrightarrow{5} C_3 \rightarrow 3$

$C_1 \xrightarrow{7} C_3 \xrightarrow{1} C_4 \rightarrow 1$

ans=3

Ex2:

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

$C_1 \xleftrightarrow[4]{} C_2 \xleftrightarrow[5]{} C_3 \xleftrightarrow[3]{} C_4 \dashrightarrow 3$

$C_1 \xrightarrow{9} C_2 \xrightarrow{8} C_4 \xrightarrow{11} C_6 \dashrightarrow 8$

$C_1 \xleftrightarrow{12} C_2 \xrightarrow{16} C_3 \xleftrightarrow{18} C_4 \quad 12$

ans=12

Idea1: Given N Stalls, ways to choose K Stalls for K Cows

$\underbrace{{}^{N}C_{K}}_{\#ways} * \underbrace{K}_{}$ ↳ // To cal min distance for a selection

Idea2: Binary Search

a) Target: max min distance between 2 cows

b) Search space: $\left[\begin{array}{cc} \text{low} & \text{high} \\ \text{min adj diff} & \text{stall[N-1] - stall[0]} \end{array} \right]$

Ex1:

stalls[4] = { 3 8 14 20 }

k=2 Cows

$\overset{\uparrow}{c_1}$ at 3 ⟷ $\overset{\uparrow}{c_2}$ at 20 = 17

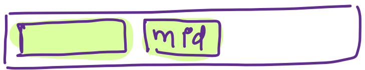
stalls[4] = { 3 8 14 20 }

k=4 Cows = 5

$c_1 \xleftrightarrow{5} c_2 \xleftrightarrow{6} c_3 \xleftrightarrow{6} c_4$

c) discard:

Case-I:

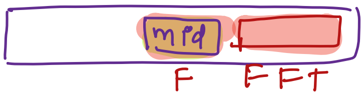


T T T T   T

Say we can place all cows atleast at mid dist apart

ans = mid, goto right

Case-II:



F   F F F

Say cannot place all cows atleast at mid dist 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 $c_1 \xleftrightarrow{23} c_2 \longrightarrow c_3^* \; c_4^*$: 20 ✗

$c_1 \xrightarrow{9} c_2 \xrightarrow{8} c_3 \xrightarrow{6} c_4$: we can place atleast 5 dist apart

Can we place cows atleast at 5 dis apart?

Can we place cows atleast at 20 disapart?

1	2	3	4	5	20	21	22	23	24 . . .
. . .		T	T	T	F	F	F	F	F

```
             // stalls  // cows            // position of all stalls
                                                    // no: of ele on which we apply BS
int maxdis(int N, int k, int stalls[]){ TC: log_2(ele) * N
    l = min adj diff in stalls[] , h = stalls[N-1]-stalls[0], ans = __
                                                    #ele = h-l+1
    while(l<=h){
        m= (l+h)/2
        // If we can place k cows atleast at mid distance apart
        if(check(N, k, stalls[], m) == True){   TODO
                                                TC: O(N)
            ans = mid; // goto right
            l = m+1
        }
        else // cannot place cows at mid distance apart
            h=m-1
        }
    }
    return ans;
}
bool    check(        ){ TODO

}
```

final obs on BS

Check function : T T T T T . . T $\boxed{T}$ F F F . . . F

If we land on True
ans = mid
goto right

If we land on False
goto left

Check function : F F F . . . F $\boxed{T}$ T T . . . T

If we land on False
goto right

If we land on True
ans = mid
goto left

38: Search in a ∞ sorted array, search k = 32

	0	1	2	3	4	5	6	7	8	
arr[ ] = {	2	4	10	13	19	24	29	30	33	......

// Say target is at some index p position

Idea1: linear search on arr[ ] : p steps

Idea2:

l	h	
1	2	: k is > ar[h] not present in range
↓	↓	
2	4	: k is > ar[h] not present in range
↓	↓	
4	8	: k is <= ar[h] possible apply Binary Search

After every step we double l & h : log p + log p ≈ log p

→ it will log p steps to elements to be range →

→ Again apply BS in given range, to check if p is there are not ≈ log p