Today's Content

Pointers Partition

Agressive cows.

Painters Partition Problem.

Ques

We have to paint n boards of length $A_1, A_2 \dots A_n$. There are k painters available and each takes 1 unit of time to paint 1 unit of board.

find min time to get the job done.

→ 1 painter will paint only continuous sections of the board.

e.g 1) A = {10, 10, 10, 10} time ⇒ 20.

k = 2

e.g 2) A = {10, 20, 30, 40} time ⇒ 60

k = 2

0 1 2 3

P_1 → 60

P_2 → 40

Ex

N = 15 . K = 3 .

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

idea :- divide into $\frac{\text{totaltime}}{K}$ Ans = 100.

arr[6] = 1 1 1 2 1 100 , K = 2

$\frac{106}{2}$ 53 → not possible to divide like this.

Ex

N = 15 . 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

w_1 w_2 w_3

w_1 w_2 w_3 w_4

Can we do all task in 30 mins ?.

ans

30 31 32 33 34

goto left.

Can we finish all tasks in 10 mins?.

7 8 9 10

x x x x

goto right

first True.

F F F F (T) T T T T

7 8 9 10 . . 30 31 32 33

x x x x ✓ ✓ ✓

Target :- min time to get the job done.

SearchSpace :- $\log n$.

lo → Max of Array

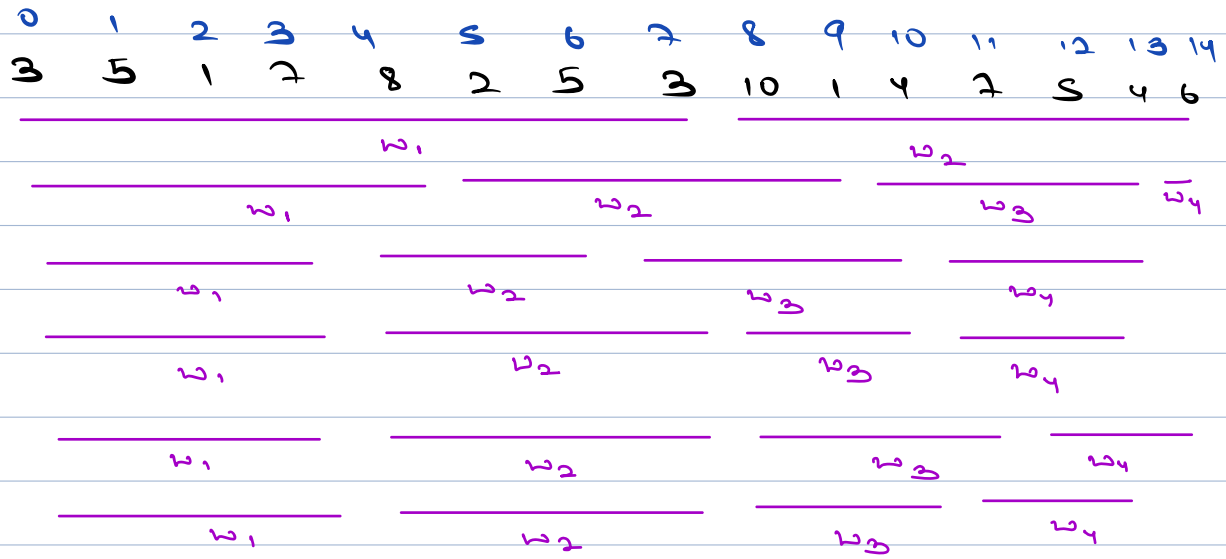
hi → Sum of Array.

Ex

N = 15.

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



lo	hi	mid	
10	71	40	Ans = 40, goto left
10	39	24	Ans = 24, goto left
10	23	16	goto right
17	23	20	goto right
21	23	22	Ans = 22, goto left
21	21	21	goto right.
22	21		break.

Pseudo code :- Binary search on Answer

```
int workers (int time[], int n, int k) {
    lo = max(arr), hi = sum of (arr)
    ans = hi;
    while (lo <= hi) {
        m = lo + (hi - lo) / 2
        if (check (m, time, n, k)) {
            ans = m;
            hi = m - 1;
        }
        else {
            lo = m + 1
        }
    }
    return ans;
}
```

k = 4

O (log (sum of arr - max of arr) x n) + n

log of max & sum.

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

O(n)

```
bool check (int m, int time[], int n, int k){
    s = 0, c = 1,
    for (i=0; i<n; i++) {
        s += time[i];
        if (s>m) {
            c = c+1;
            s = time[i];
            if (c>k) {
                return false
            }
        }
    }
    return True;
}
```

Task todo

two three parameters.

1-2 constraints

see what is asked.

maximize or minimize something.

1) Painters Partition
2) Aggressive cows
3) Allocate books.

T T T T T T T f f f f f f
f f f f f f f f T T T T T T

Ques) Given N cows & M stalls, all M stalls are on x axis at diff. locations, place all N cows such a way that min dist b/w any 2 cows is maximized.

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

Note 2:- All cows have to be placed.

Ex 1:-

C = 3

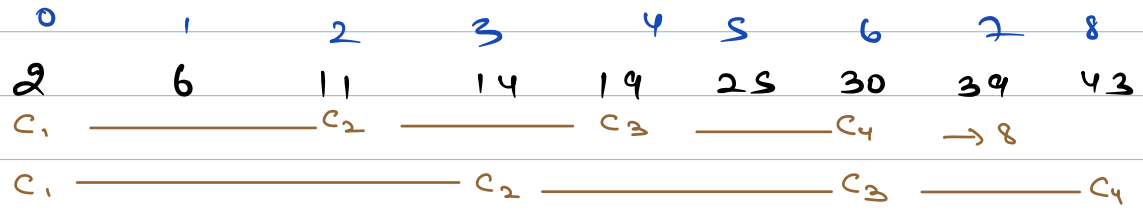
0	1	2	3	4	Min dis
1	2	4	8	9	
C_1	C_2	C_3			1
C_1		C_2		C_3	3
C_1			C_2	C_3	1

we can place cows atleast 3 dist away.

Ex 2 stalls = 9

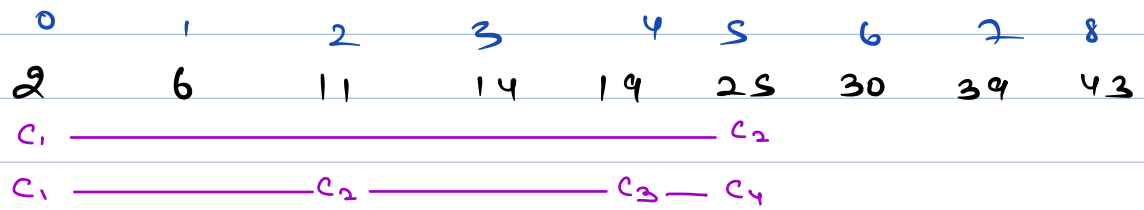
cows = 4

atleast → 12



stalls = 9

cows = 4



Can we place the cows atleast 20 dist away?

20 21 22 23

X X X X

Can we place the cows 5 dist away?

3 4 5

✓ ✓ ✓

3	4	5	6	...	20	21	22
T	T	T	T		F	F	F

T T T T T ... F F F

↓

last true.

Target → least dist b/w 2 cows should be max.

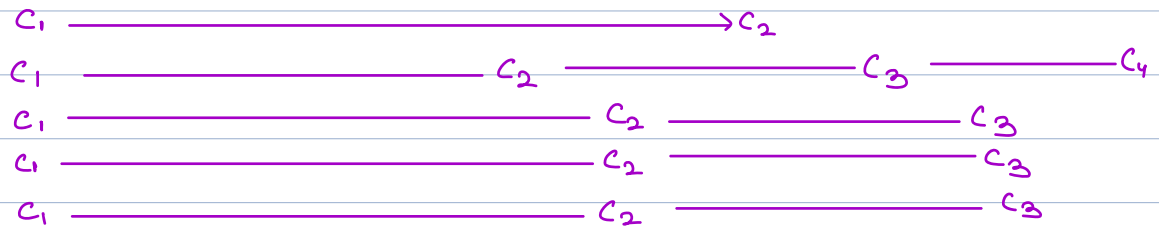
Search Space →

lo → min diff b/w adj cells.

hi → a[n-1] - a[0]

cows = 4.

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



lo	hi	m	
3	41	22	goto left.
3	21	12	ans = 12, goto right.
13	21	17	goto left.
13	16	14	goto left
13	13	13	goto left
13	12		break

Pseudo Code :- $O(n \log(hi - lo))$

```
int moo(int dist[], int n, int c){
    lo = min adj dist in dist[]
    hi = arr[n-1] - arr[0];
    ans = 0;
    while (lo <= hi){
        m = lo + (hi - lo)/2
        if (check(m, dist, n, c)){
            ans = m;
            lo = m+1
        }
        else{
            hi = m-1;
        }
    }
    return ans;
}
```

cows = 4 . m = 12 c = 2

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

→ dm).

```
bool check (int m, int dist[], int n,
                                int cows) {
    last_placed = dist[0];
    count = 1;
    for (i=1; i<n; i++) {
        if (dist[i] - last_placed >= m) {
            c = c+1;
            last_placed = dist[i];
            if (c == cows) {
                return True;
            }
        }
    }
    return false;
}
```

Monotonic nature

↳ problem is feasible till a certain point & after it is not.

or vice versa