Lecture :- Binary Search on Answer

Agenda

- Painter partition problem
- Aggressive cows

Painter partition problem

We have to paint all n boards of length $[C_0, C_1, C_2, C_3 \dots C_n]$.

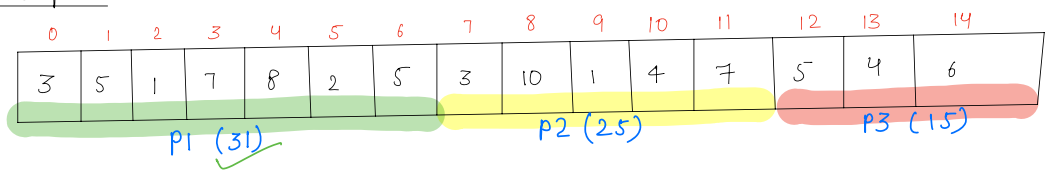
There are k painters available and each of them takes 1 unit of time to paint 1 unit of the board. Calculate and return minimum time required to get the job done.

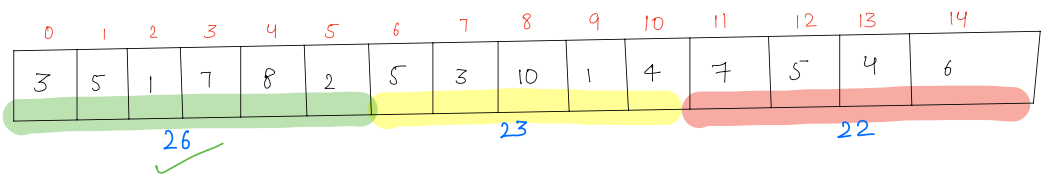
Constraints:

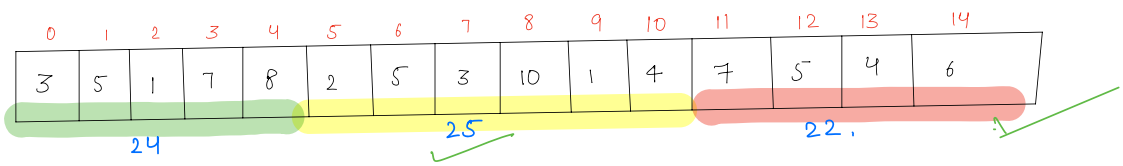
1) Two painters cannot share a board to paint. That is to say, a board cannot be partially painted by one painter, and partially by another.

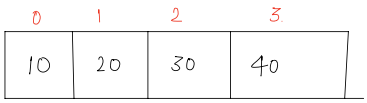
2) A painter will only paint continuous boards. This means a painter paints a continuous subarray of boards.

Example k=3 (no of painters)



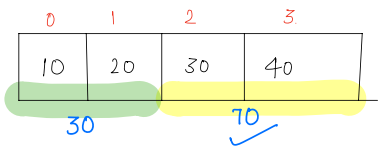


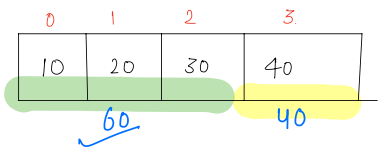




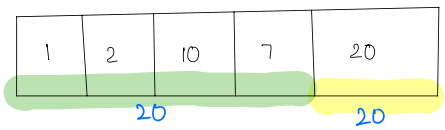
k = 2

ans = 60





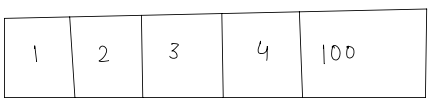
Approach1 Divide total time / total no of painters



K = 2.

Each painter will paint boards of length = $\frac{40}{2} = 20$

flaw :- It leads to partial painting



K = 2.

Each painter will paint boards of length = $\frac{110}{2} = 55$

Approach2

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

k = 4

Colour code of each painter :-

P1	
P2	
P3	
P4	

Search space

Min value of range : (Best case)

Say we have as many painters as the no. of boards, in which case each painter will paint one board.

Ans =

0	1	2	3
2	5	3	8
P1	P2	P3	P4

, k = 4

ans = 8 [max(arr)]

Max value of range : Worst case.

0	1	2	3
2	5	3	8
P1			

, k = 1

Ans = 18 [sum(array)]

Range ⟶ [max(array), sum(array)]

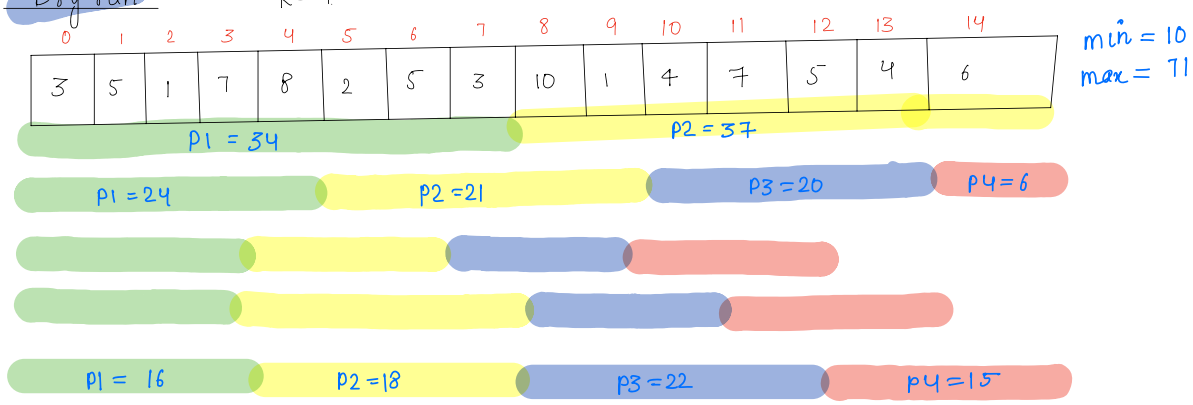
Target The max time to complete the task

Condition

- Say we calculated mid. How to decide whether mid is answer?
- We can check if we can complete the task within mid amount of time.
  - If yes, we should save that as the answer. and move to left to try for a lesser time.
  - Else, move to right (it means we will need more time to complete the task).

Dry run

k = 4



min = 10
max = 71

start	end	mid $\left(\frac{s+e}{2}\right)$	Action
10	71	40	ans = 40 Left e = mid - 1
10	39	24	ans = 24 Left e = mid - 1
10	23	16	Right s = mid + 1
17	23	20	Right , s = mid + 1
21	23	22	Ans = 22 left
21	21	21	Right
22	21		Break

Pseudocode

```
function painterPartition(A[], K) {
        n = A.length;
        start = max(A);
        end = sum(A);
        while(start <= end) {
            mid = (start + end)/2;
            if(isPossible(A, K, mid)) {        ---- O(n)
                ans = mid;
                end = mid-1;
            } else {
                start = mid+1;
            }
        }
    return ans;
}

function isPossible(A, K, mid) {        ---- O(n)
    sum=0, painters=1;
    for(int el: A) {
        sum+=el;
        if(sum> mid) {
            painters++;
            sum=el;
        }
    }
    if(painters <=K) {
        retur true;
    }
    return false;
}
```

max - min

TC: O(nlogn)
SC: O(1)

Break: 8:24 – 8:34

Aggressive cows

Given n cows and m stalls, all m stalls are located at different location at x-axis, place all the cows such that the minimum distance b|w any two cows is maximised

Constraints

1) There can only be one cow in a stall at a time.
2) We need to place all cows.

Ex1 stalls:

0	1	2	3	4
1	2	4	8	9

cows = 3

stall = 5

Ans: 3

1	2	4	8	9	min distance
c1	c2	c3			1
c1		c2		c3	3
c1			c2	c3	1

Ex2

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

cows: 4

stall: 9

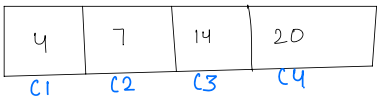
2	6	11	14	19	25	30	39	43	min distance.
c1	c2	c3	c4						3
c1		c2		c3		c4			8
c1			c2			c3		c4	12

Approach

Search space

Worst case: If there are same cows as the no. of stalls ?

min (diff b|w adjacent cells).

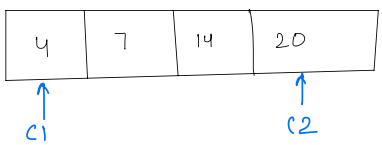


cows = 4.

ans = 3

Best case: There are 2 cows ?

Place them at corners

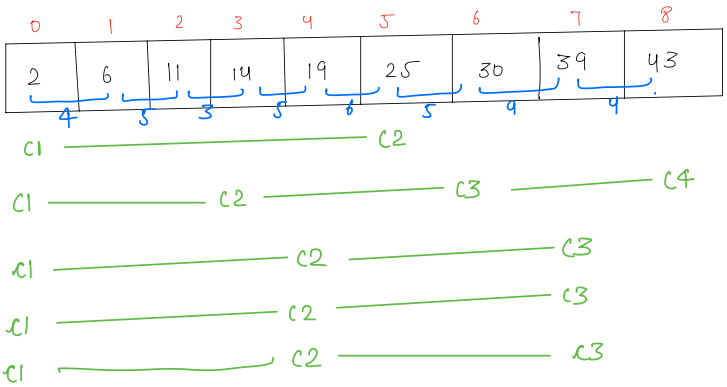


cows = 2

A[n-1] - A[0]

Range: [worst - best
(1) A[n-1] - A[0]]

Dry run:



cows = 4. min = 1
max = 43 - 2 = 41

start	end	mid	Action
~~1~~ 3	41	22	Left
3	21	12	ans = 12 Right
13	21	17	Left
13	16	14	Left
13	13	13	Left
13	12		stop

Pseudocode

```
boolean check( A[], int cows, int mid) {
        n = A.length;
        count = 1; // no of cows placed at mid distance.
        last = 0;
        for( i = 0; i<n; i++) {
          if ( A[i] - A[last] >= mid) {
                last = i;
                count++;
          }
          if ( count >= cows) {
                return true;
          }
    }
    return false;
}
```

Thankyou