Searching - 3

Today's Content:

- Painters partition
- Aggressive Cows

} Binary search on answer

Question: Painters Partition

We have to paint n boards of lengths $A_1, A_2, A_3 \ldots A_n$. There are k painters available and each takes 1 unit of time to paint 1 unit of board. Find the min time to get the job done.

Note: 1 painter will paint only continuous sections of the board.

ex: $A = \{\overbrace{10, 10}^{①}, \overbrace{10, 10}^{②}\}$ $k = 2$

Time = 20 units.

ex: $A = \{\underbrace{10, 20}_{①}, \underbrace{30, 40}_{②}\}$ $k = 2$

① ② ⇒ 70 units.

① ② ⇒ 60 units ⇒ Ans

Idea 1:

Take $\frac{\text{sum of lengths}}{\text{Painters}}$

$= \frac{100}{2} = 50$ units ⇒ Wrong Answer

$[\underbrace{1, 2, 3, 4}_{①}, \underbrace{100}_{②}]$ $k = 2$

① ② ⇒ 100 units

$k = 4$

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

A = { (3, 5, 1, 7, 8, 2) , (5, 3, 10, 1, 4, 7) , (5, 4, 6) }

(1) (2) (3)

① ② ③ ④

Can we do the task in 30 mins?

So each painter should be assigned max 30 units length to complete in max 30 mins.

Yes => Find a even lesser time

Can we do the task in 10 mins?

No

7	8	9	10
x	x	x	x

30	31	32	33
✓	✓	✓	✓

↓ ans

* Binary search can be used.
  - Target : Job completion
  - Search space : [Max element, Sum of elements]
  - Condition : Check for job completion.
    If yes → store ans, go left.
    If no → go right.

$k = 4$

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

$A = \{ \underbrace{3, 5, 1, 7, 8, 2, 5, 3}_{P_1}, \underbrace{10, 1, 4, 7, 5, 4, 6}_{P_2} \}$

$P_1 \quad P_2 \quad P_3 \quad P_4$

$P_1 \quad P_2 \quad P_3 \quad P_4 \quad \times \times$

$P_1 \quad P_2 \quad P_3 \quad P_4 \quad \times$

$P_1 \quad P_2 \quad P_3 \quad P_4 \quad \times$

Low	High	mid	Result
10	71	40	ans = 40, Go left.
10	39	24	ans = 24, Go left.
10	23	16	Go right.
17	23	20	Go right.
21	23	22	ans = 22, Go left.
21	21	21	Go right.
22	21	⇒ Break.	

```
func painterPartition(int[] time, int n, int k){
    l = max(arr)
    h = sum(arr)
    ans = -1
    while (l <= h) {
        m = l + (h-l)/2
        if (check(m, time, k)) {
        // Job can be complete in m time
            ans = m;
            h = m - 1; // Go left.
        }
        else {
            l = m + 1; // Go right.
        }
    }
    return ans;
}
```

$T.C = O(\log(sum(arr) - max(arr)) * T.C(check))$

$\boxed{T.C = O(N * \log(sum(arr) - max(arr))}$

```
func check (int cur, int[] time, int k){
    used_painters = 1
    time_cons = 0
    for (int i=0; i<time.length; i++){
        if (time_cons + time[i] <= cur){
            time_cons += time[i]
        }
        else {
            used_painters++;
            time_cons = time[i]
        }
        if (used_painters > k)  return false;
    }
    return true;
}
```

time = { 3, 5, 1, 7, 8, 2, 5, 3, 10, 1, 4, 7, 5, 4, 6 }

time_cons = 16 | time_cons = 18 | time = 15 | time = 16 | time = 6

Can I do the task in 'cur' (20) time?

T.C (check) = O(N)

Question: Aggressive Cows

Given N cows and M stalls, all M stalls are on X-axis at different locations given in stalls[].
Place all N cows in such a way that the minimum distance between any 2 cows is maximised.

Note 1: In 1 stall only one cow can be accomodated.

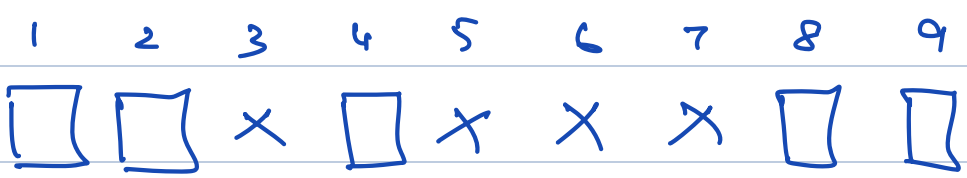
Note 2: And all the cows have to be placed.

ex: stalls = { 1 (0), 2 (1), 4 (2), 8 (3), 9 (4) } C = 3

Min dist. b/w any 2 cows

Minimum distance is maximised

C_1	C_2	C_3	X	X	1
C_1	X	C_2	C_3	X	3
C_1	X	X	C_2	C_3	1
X	C_1	X	C_2	C_3	1
X	X	C_1	C_2	C_3	1



stalls = [0 3 4 7 9 10] $C = 4$

Min distance

C_1	C_2	C_3	C_4	X	X	1
C_1	X	C_2	C_3	X	C_4	(3)
C_1	C_2	X	C_3	X	C_4	(3)

Cows = 4

ex: stalls = [2, 6, 11, 14, 19, 25, 30, 39, 43]
(0 1 2 3 4 5 6 7 8)

C_1 x x C_2 x x C_3 x C_4

C_1 x x x C_2 x x C_3 x C_4?

Can we place the cows atleast 12 distance apart?

Yes → Store ans, go right.

9 10 11 12

✓ ✓ ✓ ✓

Can we place the cows atleast 15 distance apart?

No → Go left.

9 10 11 12

✓ ✓ ✓ ✓

15 16 17 18

x x x x

Use binary search

- Target - Minimum distance
- Search space - [Min dist., Max dist.]
  = [1, arr[N-1] - arr[0]]
- Condition → If allocation is possible → Go right.
  else → Go left.

Cows = 4

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

C_1 x x x x C_2 x x x C_3 C_4?

C_1 x x C_2 x C_3 x C_4

C_1 x x x C_2 x x C_3 x C_4?

C_1 x x C_2 x x C_3 x C_4

C_1 x x x C_2 x x C_3 x C_4?

low	high	mid	Result
1	41	21	Go left.
1	20	10	ans = 10, go right.
11	20	15	Go left.
11	14	12	ans = 12, go right.
13	14	13	Go left.
13	12	=>	Break.

```
func aggressiveCows(int[] stalls, int cows){
    l = 1; h = stalls[N-1] - stalls[0]
    ans = -1;
    while (l<=h) {
        m = l + (h-l)/2;
        if (check(m, stalls, cows)) {
            ans = m;
            l = m+1;
        }
        else {
            h = m-1;
        }
    }
    return ans;
}
```

$$T.C = O(T.C(Check) * \log(stalls[N-1] - stalls[0]))$$

$$\boxed{T.C = O(N * \log(stalls[N-1] - stalls[0]))}$$

```
func check (int cur, int[] stalls, int cows){
    used_cows = 1;
    last_cow = stalls[0];
    for (int i=1; i<stalls.length; i++){
        if (stalls[i] - last_cow >= cur){
            used_cows++;
            last_cow = stalls[i];
        }
        if (used_cows >= cows) return true;
    }
    return false;
}
```

How to identify binary search problems?

- Search in an array
  - ↳ Binary search on Array.
- Search for a number to satisfy a condition
  - ↳ Binary search on number.
- Search for an answer
  - ↳ Binary search on answer.

- Conditions to get an answer. (1 - 2)
- Answers look like:

TTTTTTFFFFFFF

Job can be done | Job can't be done

FFFFFFFTTTTTTT

Job can be done | Job can't be done

} Return best answer.

Contest Information

- 2nd August – 9 PM to 10:30 PM.
- Contest Discussion – 10:30 PM to 11:30 PM.
- Topics: Hashing, Searching, Sorting, Two Pointers
- DSA 3 starts on 26th July.

Doubts

For rotated array search, use this logic to find the max element.

[10, 11, 1, 2, 3, 4]

If arr[m] > arr[N-1] → Go right.
else go left.

- Merge numbers in an array to form largest number.

  3 , 39.

  - Convert to string.
  - Concatenate (i1 + i2 = m1), (i2 + i1 = m2)
  - Convert m1 & m2 to int.
  - Compare m1, m2.