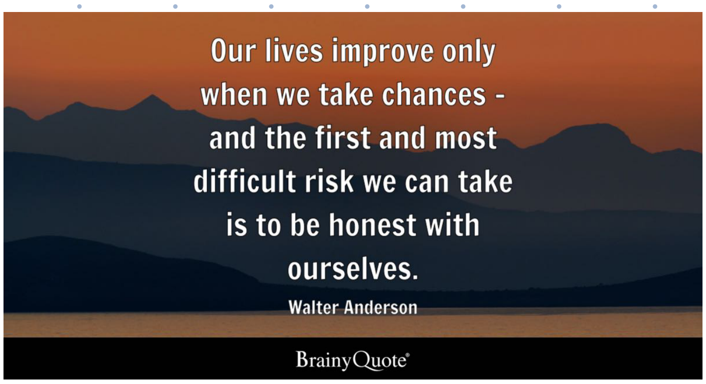
Binary Search 2



Good Morning

Content

01. Painter's Partition I
02. Painter's Partition II
03. Aggressive cows

* Maximum shipping capacity
* A* Magical No.
* koko eating Bananas
* Allocate min no. pages / Allocate books

01. Painter's partition problem – I

Given N boards with length of each board

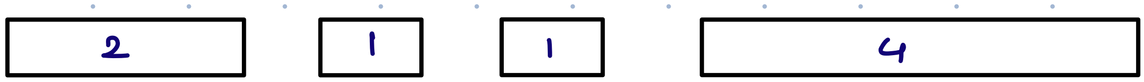
(a) A painter takes T unit of time to paint 1 unit of length

(b) A board can only be painted by 1 painter

c) A painter can only paint boards placed next to each other (i.e continous segment)

Q Find minimum no. of painters required to paint all boards in X unit of time. Return −1 if not possible.

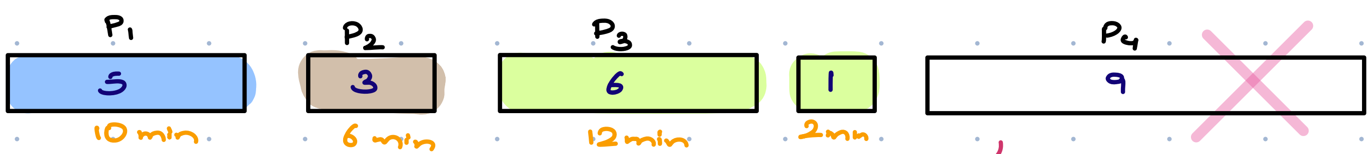
N = 4



T = 3 mins to paint 1 unit length of board

X = Total time for painters

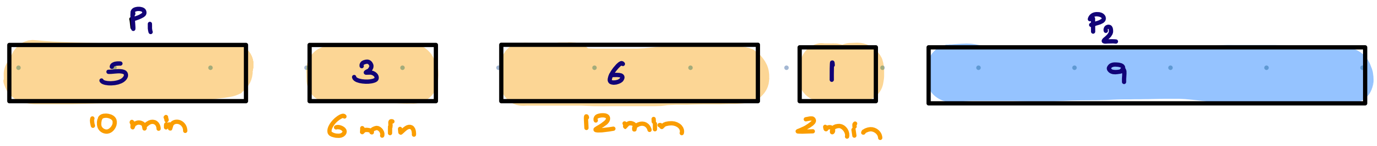
N = 5 A[ ] = {5 3 6 1 9} T = 2mins X = 15 min



T = 2 mins = 1 length

X = 15 min

For board of length 9, we atleast require 18 minutes. but the maximum time we have is 15 mins, so not possible to paint & hence return −1;



T = 2 mins = 1 length

X = 30 mins

Ans = 2 painters

T = 2 mins = 1 length

X = 20 mins

Ans = 3 painters

```
int minpainters ( int [] A , int T , int x )
    painter = 1     Timeleft = X
    for ( i = 0 ; i < N ; i++ ) {
        if ( A[i] * T > x ) return -1;
        if ( A[i] * T ≤ Time left )
            Timeleft = Timeleft - A[i] * T .
        }
        else {
            painter ++;
            Time left = X - A[i] * T
        }
    }
    return painter
}
```

* Painters partition II

Find min time to paint all the boards if P painters are available

5	3	6	1	9

T = 2 min → 1 unit length

P = 1 painter

$$\text{Min time} = 5*2 + 3*2 + 6*2 + 1*2 + 9*2$$

$$\Rightarrow (5+3+6+1+9)*2$$

$$\Rightarrow 24*2$$

$$\Rightarrow 48 \text{ mins}$$

5	3	6	1	9

T = 2 min → 1 unit length

P = 2 painter

Case 1

$$\left.\begin{array}{l} P_1 = 1 \text{ board} \Rightarrow 5*2 = 10 \text{ min} \\ P_2 = 4 \text{ board} \Rightarrow 19*2 = 38 \text{ min} \end{array}\right\} \text{Time} = 38 \text{ min}$$

Case II

$$\left.\begin{array}{l} P_1 = 2 \text{ boards} \Rightarrow 8*2 = 16 \text{ min} \\ P_2 = 3 \text{ boards} \Rightarrow 16*2 = 32 \text{ min} \end{array}\right\} \text{Time} = 32 \text{ mins}$$

Case III

P_1 = 3 boards ⇒ 14 * 2 = 28 min

P_2 = 2 boards ⇒ 10 * 2 = 20 min

Time = 28 min

Case IV

P_1 = 4 boards ⇒ 15 * 2 = 30 min

P_2 = 1 boards ⇒ 9 * 2 = 18 min

Time = 30 min

$A[\,] = \{ \underbrace{1 \quad 2 \quad 3 \quad 4}_{P_1} \quad \underbrace{100}_{P_2} \}$ K = 2

Ans = 100 min

Target → Min time

Search space ⇒ { Maxele * T (Minimum time) … Σ A[i] * T (Maximum time) }

Min time	M	Mid	Max time

Mid → represent time

For mid value, find min no. of painters required to paint all boards

< P: Increase painters, Decrease time ⇒ hi = mid − 1

= P: ans = mid, hi = mid − 1

> P: Decrease no. of painter, Increase time ⇒ lo = mid + 1

5	3	6	1	9

T= 2 min → 1 unit length

P= 2 painter

lo	hi	mid	Min no. of painters req for mid time	
18	48	33	2	ans= 33 min hi= mid - 1
18	32	25	3	Decrease painters Increase time lo = mid + 1
26	32	29	2	ans= 29 hi= mid - 1
26	28	27	3	Decrease painters Increase time lo = mid + 1
28	28	28	2	ans= 28 hi= mid - 1
28	27	→ stop		

```
int  mintime ( int[] A , int T , int P )
{
    lo = maxEle * T
    hi = sumof all Ele * T
    ans = -1
    while ( lo ≤ hi )
    {
        mid = ( lo + ( hi - lo ) / 2 )

        int painters = minpainter (A , T, mid)

        if ( painters ≤ P )
        {
            ans = mid
            hi = mid - 1 ;
        }
        else {
            lo = mid + 1
        }
    }
}
```

N time for this

TC : O(log (search space) * N) + N

SC : O(1)

```
int minpainters ( int [] A , int T , int x )
    painter = 1    Timeleft = X
    for ( i = 0 ; i < N ; i++ ) {
        if ( A[i] * T > x ) return -1;
        if ( A[i] * T ≤ Time left )
            Timeleft = Timeleft - A[i] * T .
        }
        else {
            Painter ++ ;
            Time left = X - A[i] * T
        }
    }
    return painter
}
```

8:17 AM → 8:27 AM

Aggressive cows

Farmer has build a barrier with N stalls

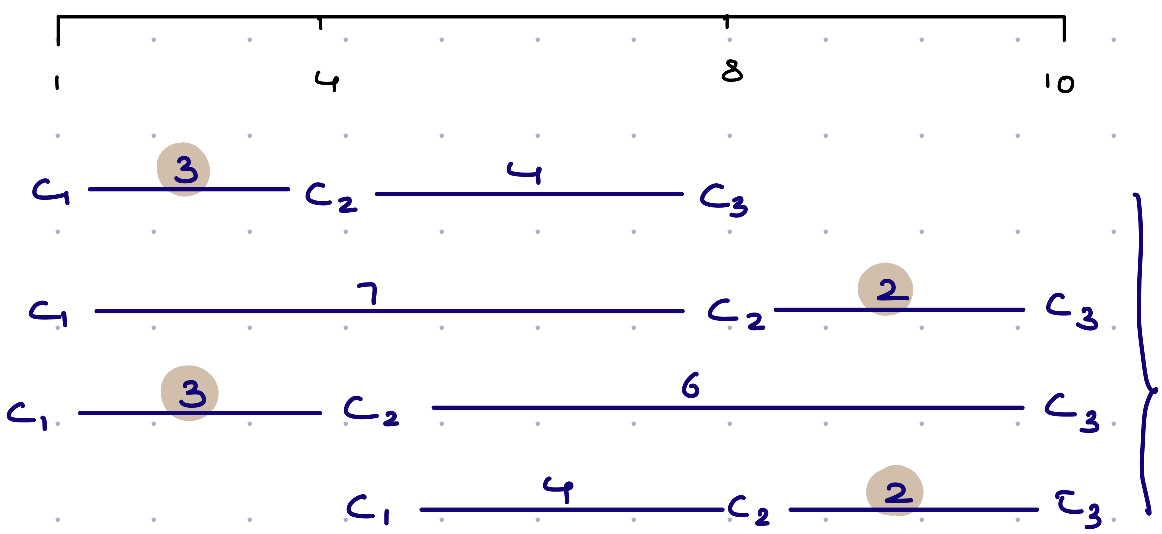
A[i] → location of i^{th} stall in increasing order

M → no. of cows the farmer has. $\boxed{2 \le M \le N}$

Cows are aggressive towards each other so, farmer wants to maximise the minimum distance b/w any pair of cows. Find max possible min distance

N = 4

$A = \{1, 4, 8, 10\}$ M cows = 3

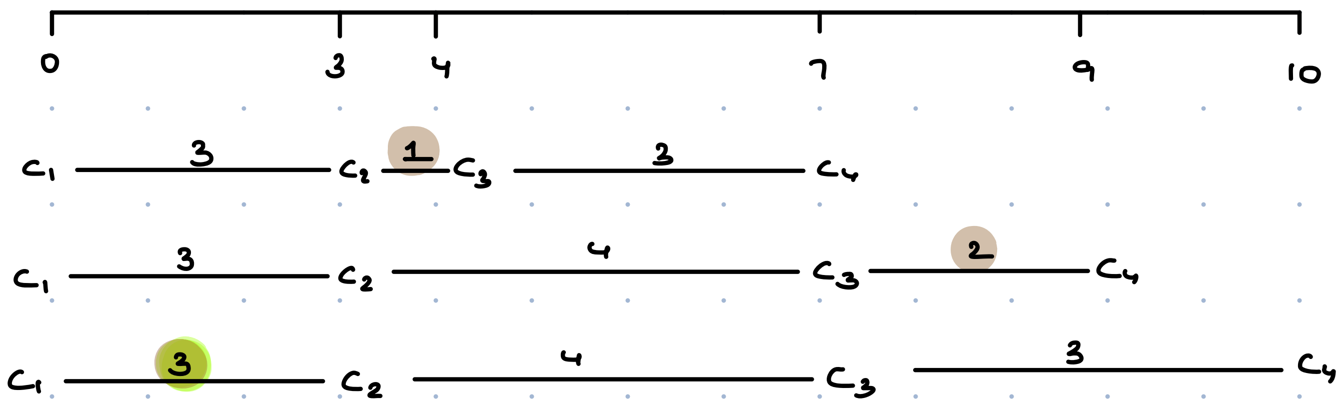


Maximise the min possible distance : 3

What will be the maximum value of the distance between the closest cows in this case?

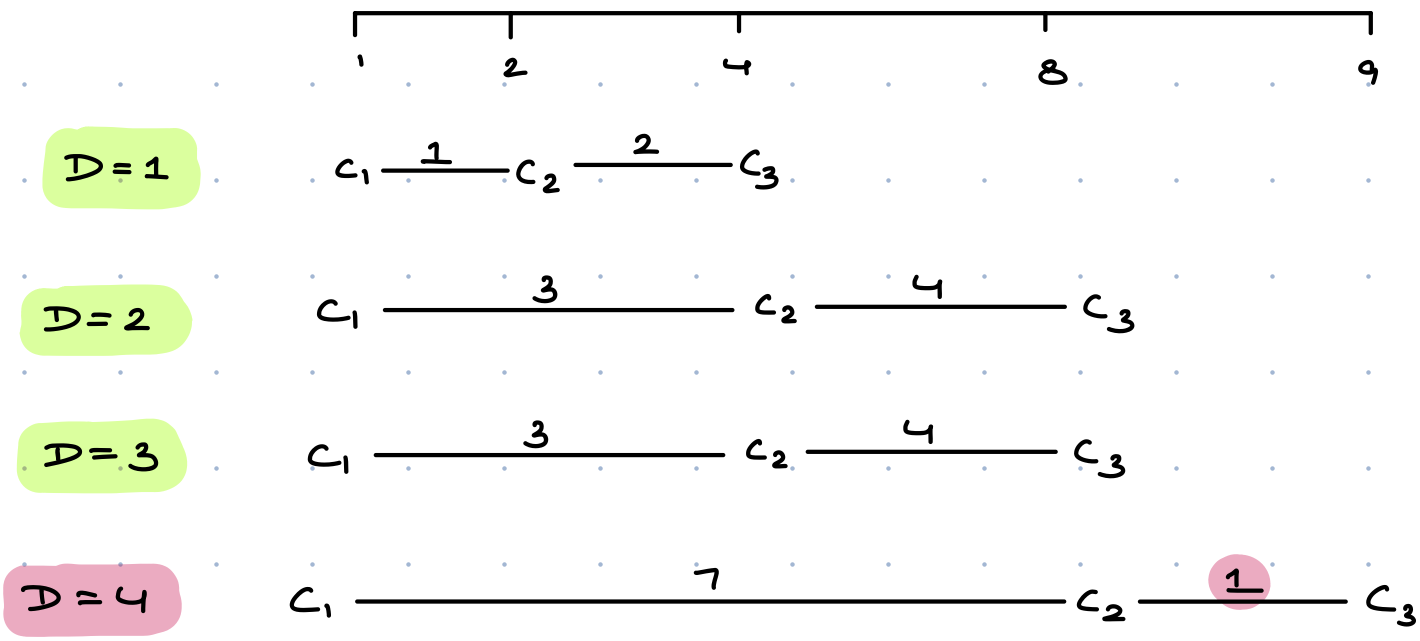
A: 0, 3, 4, 7, 9, 10

No. of cows ← **K = 4**



Ans = 3

A[ ] = { 1 2 4 8 9 } C = 3



Ans = 3

Target ⇒ Maximum value of D

Search space $\left\{ \text{min dis } 1 \quad \quad \text{Max dis } A[N-1] - A[0] \right\}$

min of all differences b/w two adjacent stall

lo	mid	hi

mid

↓

distance

For mid distance, check if it is possibble to place all cows atleast mid distance apart

Yes

ans = mid

lo = mid + 1

No

decrease the distance

hi = mid − 1

C_1			C_2		C_3		C_4	
2	6	11	14	19	25	30	39	43
							last pos	

$\underline{\underline{C = 4}}$

lo	hi	mid	Can we place 4 cows atleast mid dis apart
1	41	21	No, hi = mid−1
1	20	10	ans = 10, lo = mid+1
11	20	15	No, hi = mid−1
11	14	12	ans = 12, lo = mid+1
13	14	13	No, hi = mid−1
13	12		→ Exhausted search Space

```
lo = 1
hi = A[N-1] - A[0]
while (lo ≤ hi)
|    mid = (lo + (hi-lo)/2)
|    if (check (A, mid, c) == true)
|    |   ans = mid;
|    |   lo = mid+1
|    }
|    else hi = mid-1;
}
```

TC : O(log (searchspace) * N)

SC : O(1)

```
boolean check ( int []A , int dis , int c )
{
    cows = 1
    lastpos = A [0]
    for ( i = 1 ; i < n ; i ++ )
    {
        if ( A [i] - lastpos ≥ dis )
        {
            cows ++ ;
            lastpos = A [i] ;
            if ( cows == c ) return true ;
        }
    }
    return false
}
```

Binary Search Problems Identification

FIGQLUT

- These type of problems generally has following characteristics :-
    - There are two or three parameter & constraints
    - Requirement is to maximize or minimize the given parameter
- One tricky point is to find search space which is generally the parameter asked to maximize or minimize.
- The problem should be **Monotonic** in nature i.e after one point it is not feasible to solve or vice versa.