Agenda

1) Painter's partition
2) Aggressive cows

} Hard (VVIP)

Q.1 **Painter's partition**

Given an **array** containing **length of boards**. There are **K painters** available and each of them takes **T units of time** to paint 1 unit of the board.

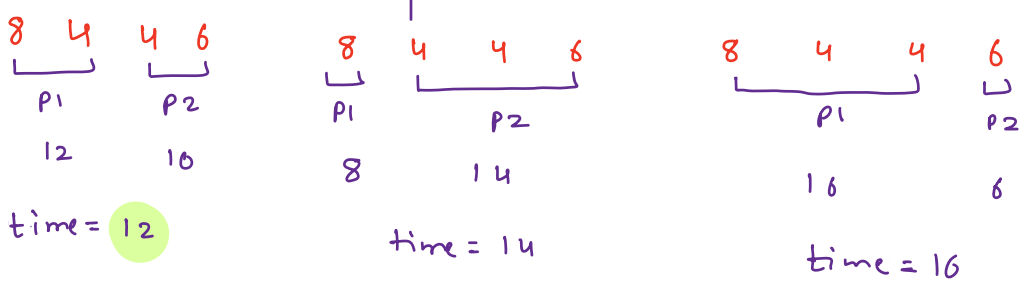
Calculate **minimum time** required to paint all the boards, by keeping following rules in mind:

i) 2 painters can't share a board to paint.

ii) A painter can only pick contigous boards.

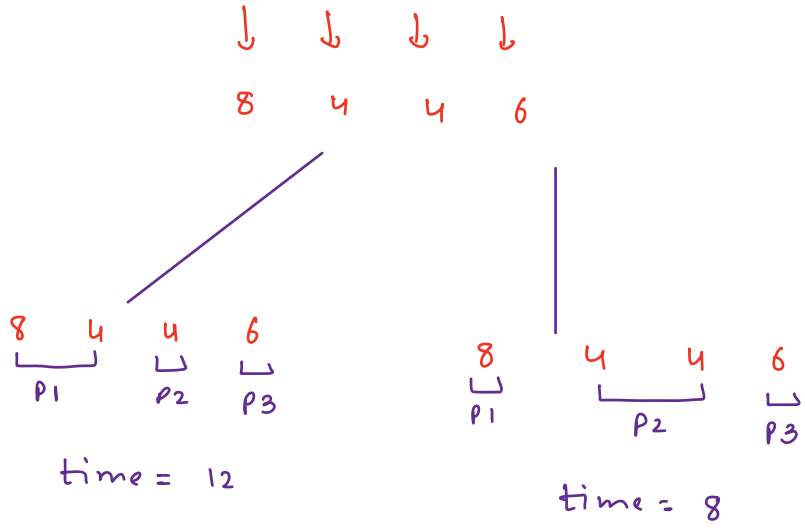
A[] = [4 2 2 3] T = 2, K = 2

8 4 4 6 {A[i] * T}



Ans = 12

A[] = [4 2 2 3] T=2, K=3



ans=8

A[] = [5 3 6 1 7] T=2, ~~K=3~~

10 6 12 2 14

lo = max workforce applied, painters = 5 (5 tasks)

give every painter a single task

10 | 6 | 12 | 2 | 14 => ans = 14
P1 P2 P3 P4 P5

do => max of times

hi = min workspace applied, painters = 1

10 + 6 + 12 + 2 + 14 => ans = 44
P1

Conclusion

ans is going to be in the range lo to hi, so we apply binary search on finding our answer.

lo = max value of time array

hi = sum of values of time array

A[] : [5 3 6 1 7] T = 2, K = 3

10 6 12 2 14

lo = 14 hi = 44 mid = 29

if time = 29 can you allocate the entire work to 3 painters

↳ yes hi = mid - 1

lo = 14 hi = 28 mid = 21

if time = 21 can you allocate the entire work to 3 painters

↳ yes hi = mid - 1

lo = 14 hi = 20 mid = 17

if time = 17 can you allocate the entire work to 3 painters

↳ yes hi = mid - 1

lo = 14 hi = 16 mid = 15

if time = 15: can you allocate the entire work to 3 painters

↳ no lo = mid + 1

lo = 16 hi = 16 mid = 16

if time = 16: can you allocate the entire work to 3 painters

↳ yes hi = mid − 1

lo = 16 hi = 15

ans = 16

A[] = [4 2 2 3]

T = 2, K = 2

t = 12

p1 p2

ans = ~~15~~ ~~13~~ 12

If time = mid, is it possible to allocate the entire work among K painters

lo = 8 hi = 22 mid = 15

time = 15 yes hi = mid − 1

lo = 8 hi = 14 mid = 11

time = 11 no lo = mid + 1

lo = 12 hi = 14 mid = 13

time = 13 yes hi = mid − 1

lo = 12 hi = 12 mid = 12

time = 12 yes hi = mid − 1

lo = 12 hi = 11

only part remaining isPossible

↓

with time = mid, is it possible to allocate the work among k painters

```
boolean isPossible (int [] arr, int time, int k) {
    int cnt = 1;
    int curr = 0;

    for(int i=0; i< arr.length; i++) {
        curr += arr[i];
        if(curr > time) {
            cnt++;
            curr = arr[i];
        }
        if( cnt > k) {
            return false;
        }
    }
    return true;
}
```

time = 11 k = 2

time array

arr: [8 4 4 6]

p1 p2 p

i	curr	cnt
0	8	1
1	~~12~~ 4	2
2	8	2
3	~~14~~ 6	3

cnt > k

(no)

Dry run

arr: [8 4 4 6] time = 12 K = 2

P1 P2

i	curr	cnt
0	8	1
1	12	1
2	~~16~~ 4	2
3	10	2

```
for(int i=0; i<arr.length; i++) {
    curr += arr[i];
    if(curr > time) {
        cnt++;
        curr = arr[i];
    }
    if( cnt > K) {
        return false;
    }
}
return true;
```

```
boolean isPossible (int[] arr, int time, int K) {
    int cnt = 1;
    int curr = 0;
    for(int i=0; i<arr.length; i++) {
        curr += arr[i];
        if(curr > time) {
            cnt++;
            curr = arr[i];
        }
        if( cnt > K) {
            return false;
        }
    }
    return true;
}
```

```
int solve(int[] A, int T, int K) {
    int n = A.length;
    int[] arr = new int[n];
    int max = Integer.MIN_VALUE;
    int sum = 0;

    for(int i=0; i<n; i++) {
        arr[i] = A[i] * T;
        max = Math.max(max, arr[i]);
        sum += arr[i];
    }

    int lo = max, hi = sum;
    int ans = 0;
    while(lo <= hi) {
        int mid = (lo+hi)/2;
        if(isPossible(arr, mid, K) == true) {
            ans = mid;
            hi = mid-1;
        }
        else {
            lo = mid+1;
        }
    }

    return ans;
}
```

arr

time array:
contains time taken for painting ith board.

Converting length array to time array and find max & sum of time array

itr = $\log_2 R * n$

$\rightarrow (\log_2 R) * n$

TC: $O(n * \log_2 R)$

R: search space range

↳ sum - max + 1

A[] = [4 2 2 3] T=2, K=2

arr[] = [8 4 4 6] max = 8, sum = 22

ans = ~~15~~ ~~13~~ 12

```
int do = max, hi = sum;
int ans = 0;
while (do <= hi) {
    int mid = (do + hi)/2;
    if (isPossible(arr, mid, k) == true) {
        ans = mid;
        hi = mid - 1;
    }
    else {
        do = mid + 1;
    }
}
return ans;
```

do	hi	mid	isPossible
8	22	15	true
8	14	11	false
12	14	13	true
12	12	12	true
12	11		

Stops

Q.2 **Aggressive cows**

Given **k cows** and **n stalls**. There is an array denoting locations of the stall. Place all k cows in such a way that min distance between any two cows is as large as possible. What is the largest minimum distance.

Note: i) In a stall only 1 cow can be present.
ii) All cows have to be placed.

A = [1 2 4 8 9] k = 3

min distance blw any two cows

=) dist = 1

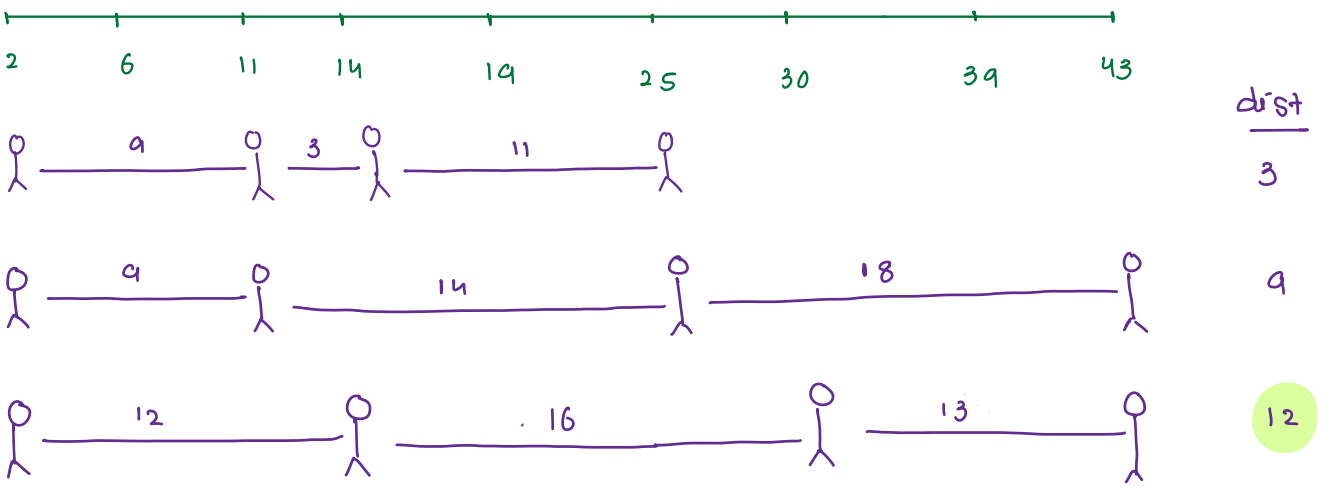
ans = 3

=) dist = 3

=) dist = 1

maximise the min distance blw any two cows

A = [2 6 11 14 19 25 30 39 43] K = 4



If shelter location array is not sorted, please sort it.

find ans

lo = 1 hi = A[n-1] - A[0]

A = [1 2 4 8 9]

K = 3

ans = ~~7~~ 3

Is it possible to place K cows in shelters such that min dist blw two cows = mid

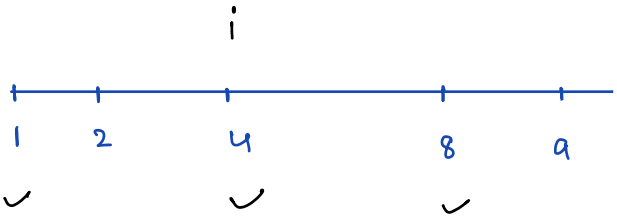
lo = 1 hi = 8 mid = 4 NO, hi = mid - 1

lo = 1 hi = 3 mid = 2 Yes, lo = mid + 1

lo = 3 hi = 3 mid = 3 yes, lo = mid + 1

lo = 4 hi = 3

Is it possible to place K cows in all shelters such that min distance blw any two cows is atleast = mid.



K = 3

dist = 2

cnt = ~~2~~ ~~2~~ 3

```java
static int solve(int[]A,int k) {
    //maximise the min distance b/w any two cows
    int n = A.length;
    Arrays.sort(A);

    int lo = 1, hi = A[n-1]-A[0];
    int ans = 0;

    while(lo <= hi) {
        int mid = (lo + hi) / 2;

        if(isPossible(A,mid,k) == true) {
            ans = mid;
            lo = mid + 1;
        }
        else {
            hi = mid - 1;
        }
    }

    return ans;
}
```

A = [1 2 4 8 9] K = 3

ans = ~~2~~ 3

lo	hi	mid	isPossible
1	8	4	false
1	3	2	true
3	3	3	true
4	3		

stops

```java
static boolean isPossible(int[]A,int dist,int k) {
    int cnt = 1;
    int i=0;

    for(int j=1; j < A.length;j++) {
        //next cow can be placed at A[j]
        if(A[j] - A[i] >= dist) {
            cnt++;
            i = j;
        }

        if(cnt == k) {
            return true;
        }
    }

    return false;

}
```

A = [1 2 4 8 9] K = 3

i

j

dist = 3

cnt = ~~1~~ ~~2~~ 3

return true

Post class content / extra reference material (today's class)

↳ video added (A^{th} magical no.)

Contest on tuesday

Syllabus of contest → sorting , searching

↓ ↳ i) Binary search

i) algo's

ii) custom sorting

9 pm to 10:30 PM → contest

10:30 Pm onwards → contest discussion