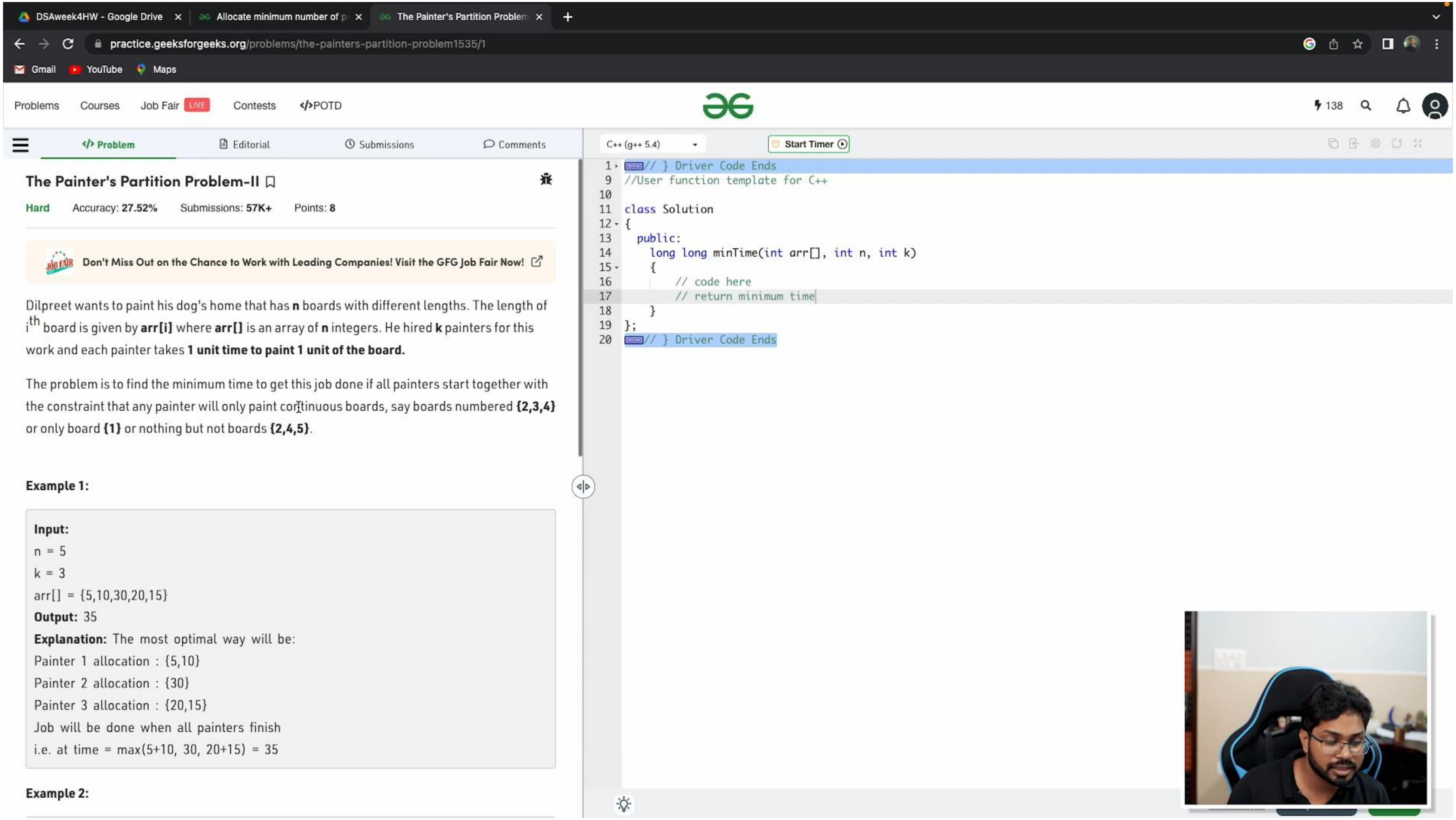

DSAweek4HW - Google Drive
Allocate minimum number of p
The Painter's Partition Problem
practice.geeksforgeeks.org/problems/the-painters-partition-problem1535/1
Gmail
YouTube
Maps
Problems
Courses
Job Fair LIVE
Contests
</>POTD
138
</> Problem
Editorial
Submissions
Comments
C++ (g++ 5.4)
Start Timer

The Painter's Partition Problem-II

Hard Accuracy: **27.52%** Submissions: **57K+** Points: **8**

Don't Miss Out on the Chance to Work with Leading Companies! Visit the GFG Job Fair Now!

Dilpreet wants to paint his dog's home that has **n** boards with different lengths. The length of i^{th} board is given by **arr[i]** where **arr[]** is an array of **n** integers. He hired **k** painters for this work and each painter takes **1 unit time to paint 1 unit of the board.**

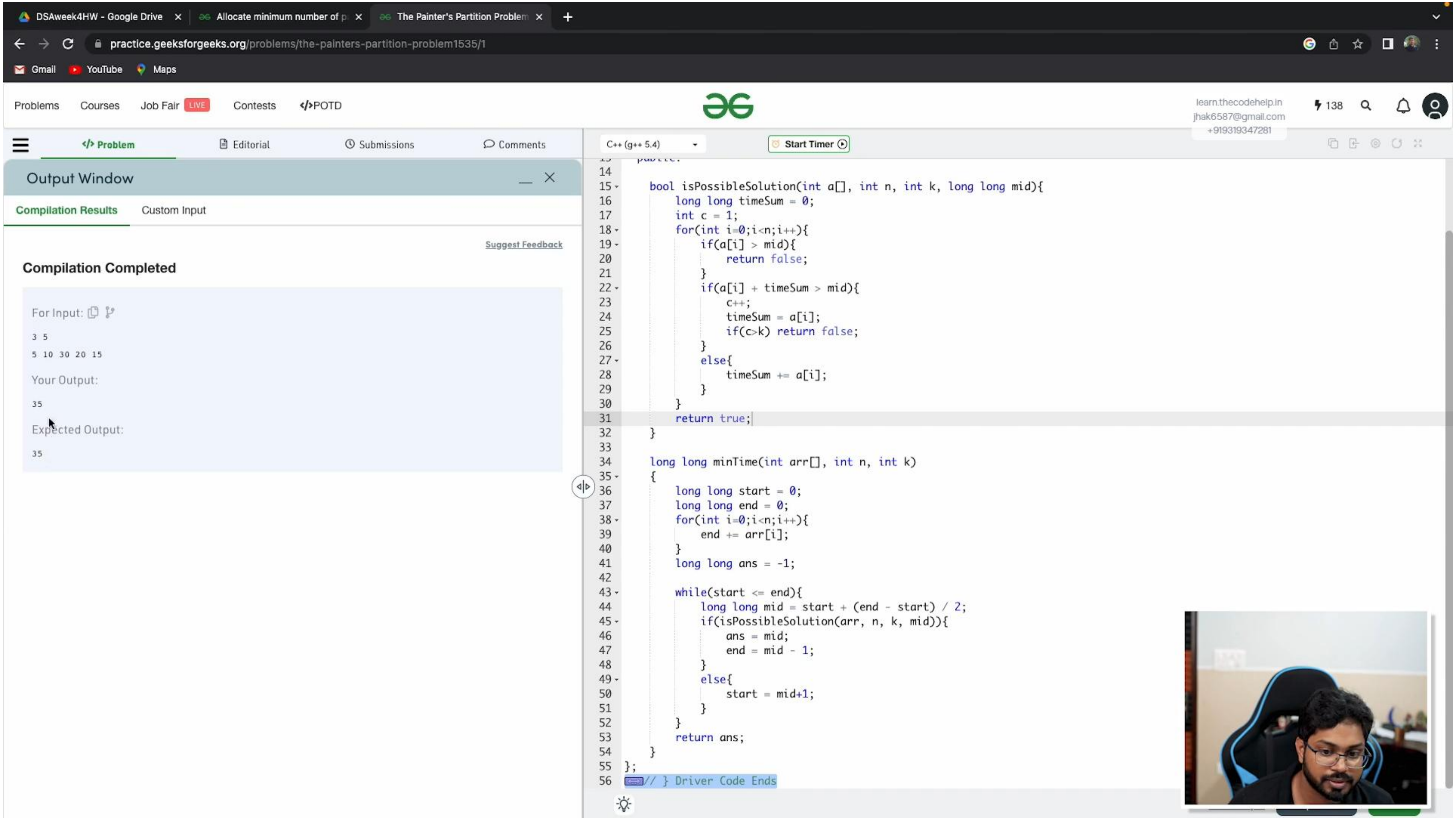
The problem is to find the minimum time to get this job done if all painters start together with the constraint that any painter will only paint continuous boards, say boards numbered **{2,3,4}** or only board **{1}** or nothing but not boards **{2,4,5}**.

Example 1:

```
Input:
n = 5
k = 3
arr[] = {5,10,30,20,15}
Output: 35
Explanation: The most optimal way will be:
Painter 1 allocation : {5,10}
Painter 2 allocation : {30}
Painter 3 allocation : {20,15}
Job will be done when all painters finish
i.e. at time = max(5+10, 30, 20+15) = 35
```

Example 2:

```
// } Driver Code Ends
//User function template for C++

class Solution
{
  public:
    long long minTime(int arr[], int n, int k)
    {
        // code here
        // return minimum time
    }
};
// } Driver Code Ends
```



Compilation Completed

For Input:

```
3 5
5 10 30 20 15
```

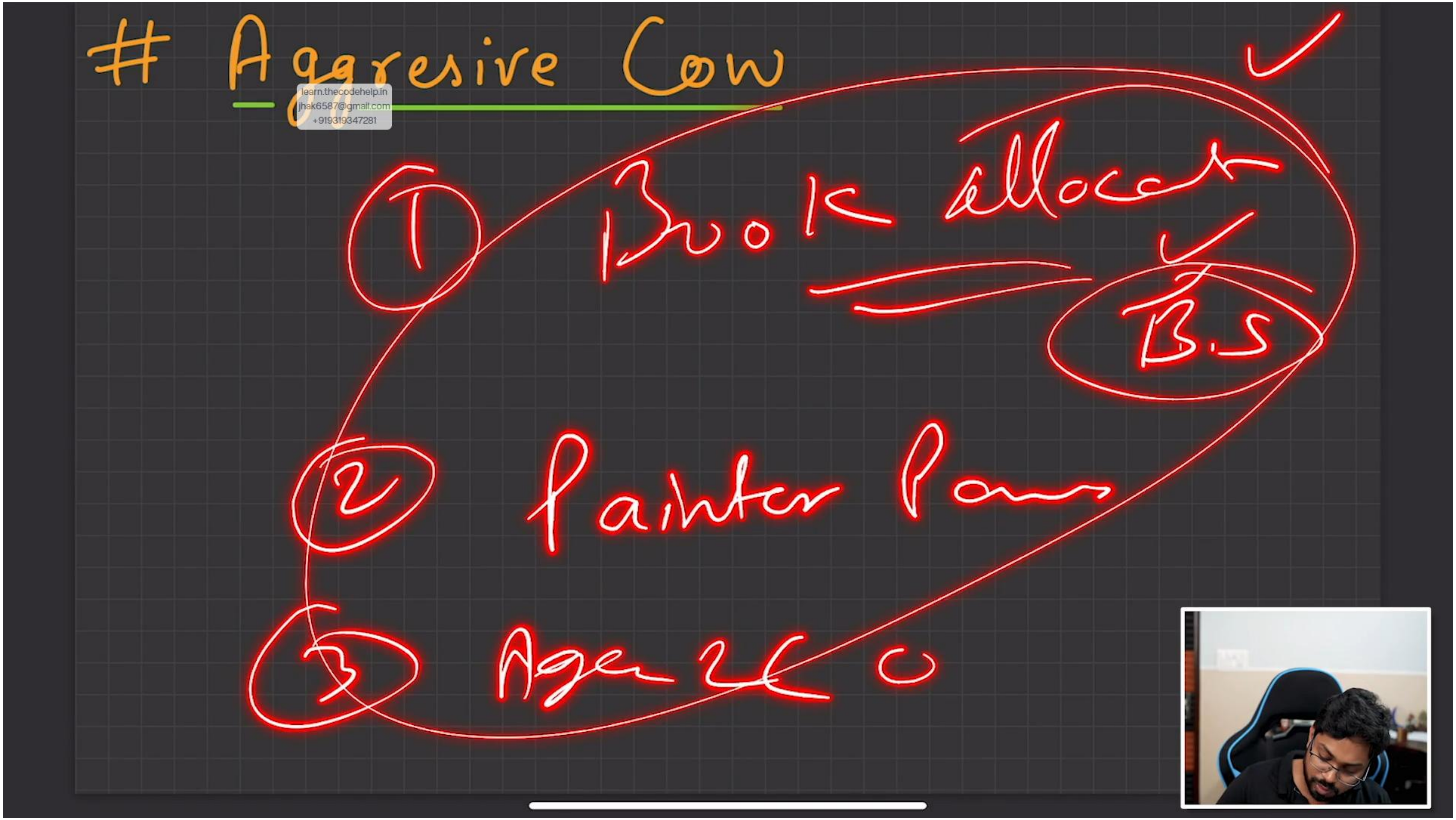
Your Output:

```
35
```

Expected Output:

```
35
```

```
  public:

    bool isPossibleSolution(int a[], int n, int k, long long mid){
        long long timeSum = 0;
        int c = 1;
        for(int i=0;i<n;i++){
            if(a[i] > mid){
                return false;
            }
            if(a[i] + timeSum > mid){
                c++;
                timeSum = a[i];
                if(c>k) return false;
            }
            else{
                timeSum += a[i];
            }
        }
        return true;
    }

    long long minTime(int arr[], int n, int k)
    {
        long long start = 0;
        long long end = 0;
        for(int i=0;i<n;i++){
            end += arr[i];
        }
        long long ans = -1;

        while(start <= end){
            long long mid = start + (end - start) / 2;
            if(isPossibleSolution(arr, n, k, mid)){
                ans = mid;
                end = mid - 1;
            }
            else{
                start = mid+1;
            }
        }
        return ans;
    }
};
// } Driver Code Ends
```



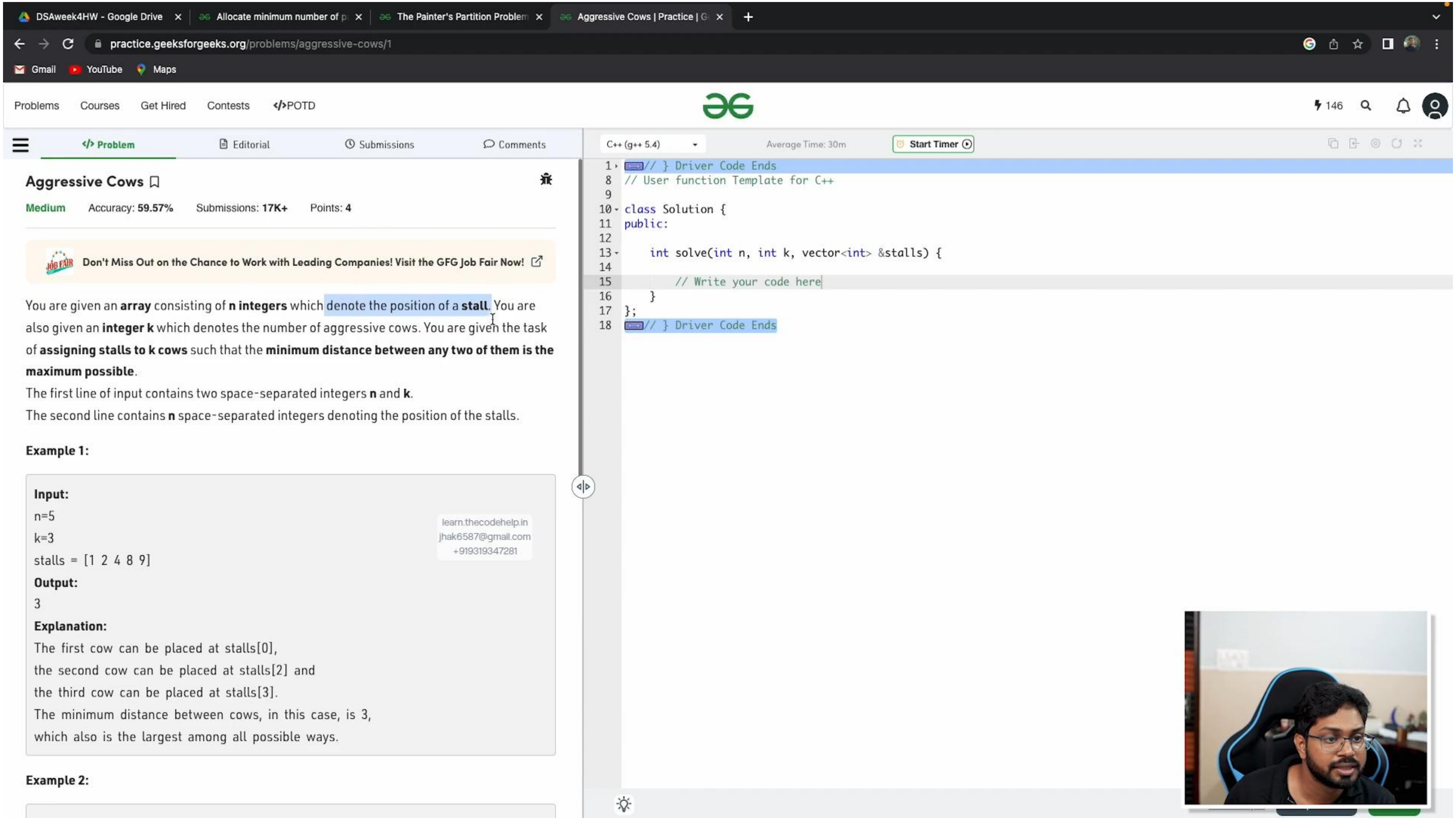
Aggresive Cow

1 Book allocat ✓

B.S

2 Painter Pans

3 Agr 2C o



DSAweek4HW - Google Drive
Allocate minimum number of p
The Painter's Partition Problem
Aggressive Cows | Practice | G
practice.geeksforgeeks.org/problems/aggressive-cows/1
Gmail
YouTube
Maps
Problems
Courses
Get Hired
Contests
</>POTD
146
</> Problem
Editorial
Submissions
Comments
C++ (g++ 5.4)
Average Time: 30m
Start Timer

Aggressive Cows

Medium   Accuracy: 59.57%   Submissions: 17K+   Points: 4

You are given an **array** consisting of **n integers** which denote the position of a **stall**. You are also given an **integer k** which denotes the number of aggressive cows. You are given the task of **assigning stalls to k cows** such that the **minimum distance between any two of them is the maximum possible**.

The first line of input contains two space-separated integers **n** and **k**.

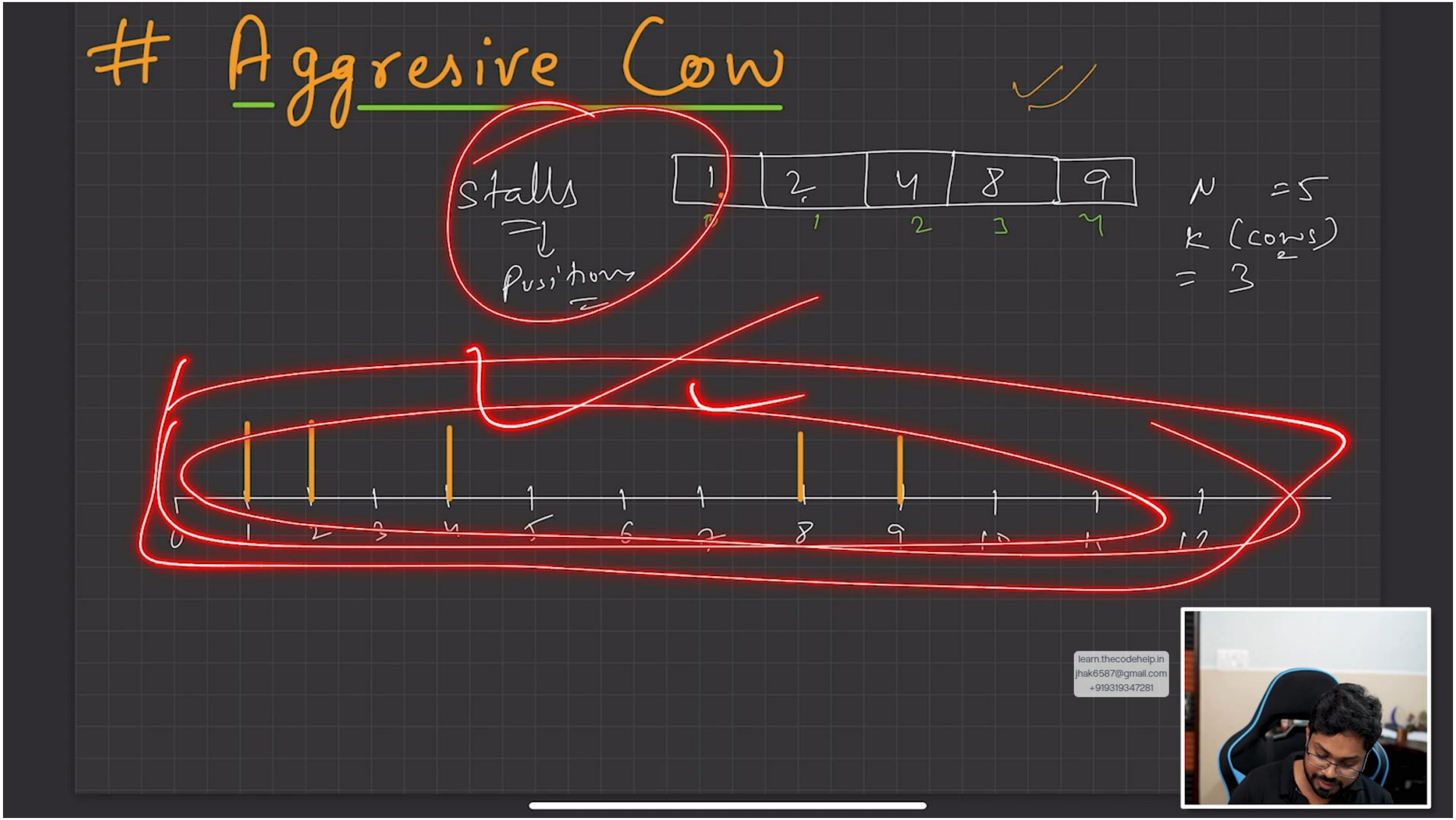
The second line contains **n** space-separated integers denoting the position of the stalls.

Example 1:

Input:
n=5
k=3
stalls = [1 2 4 8 9]
Output:
3
Explanation:
The first cow can be placed at stalls[0],
the second cow can be placed at stalls[2] and
the third cow can be placed at stalls[3].
The minimum distance between cows, in this case, is 3,
which also is the largest among all possible ways.

Example 2:

```
// } Driver Code Ends
// User function Template for C++

class Solution {
public:

    int solve(int n, int k, vector<int> &stalls) {

        // Write your code here
    }
};
// } Driver Code Ends
```



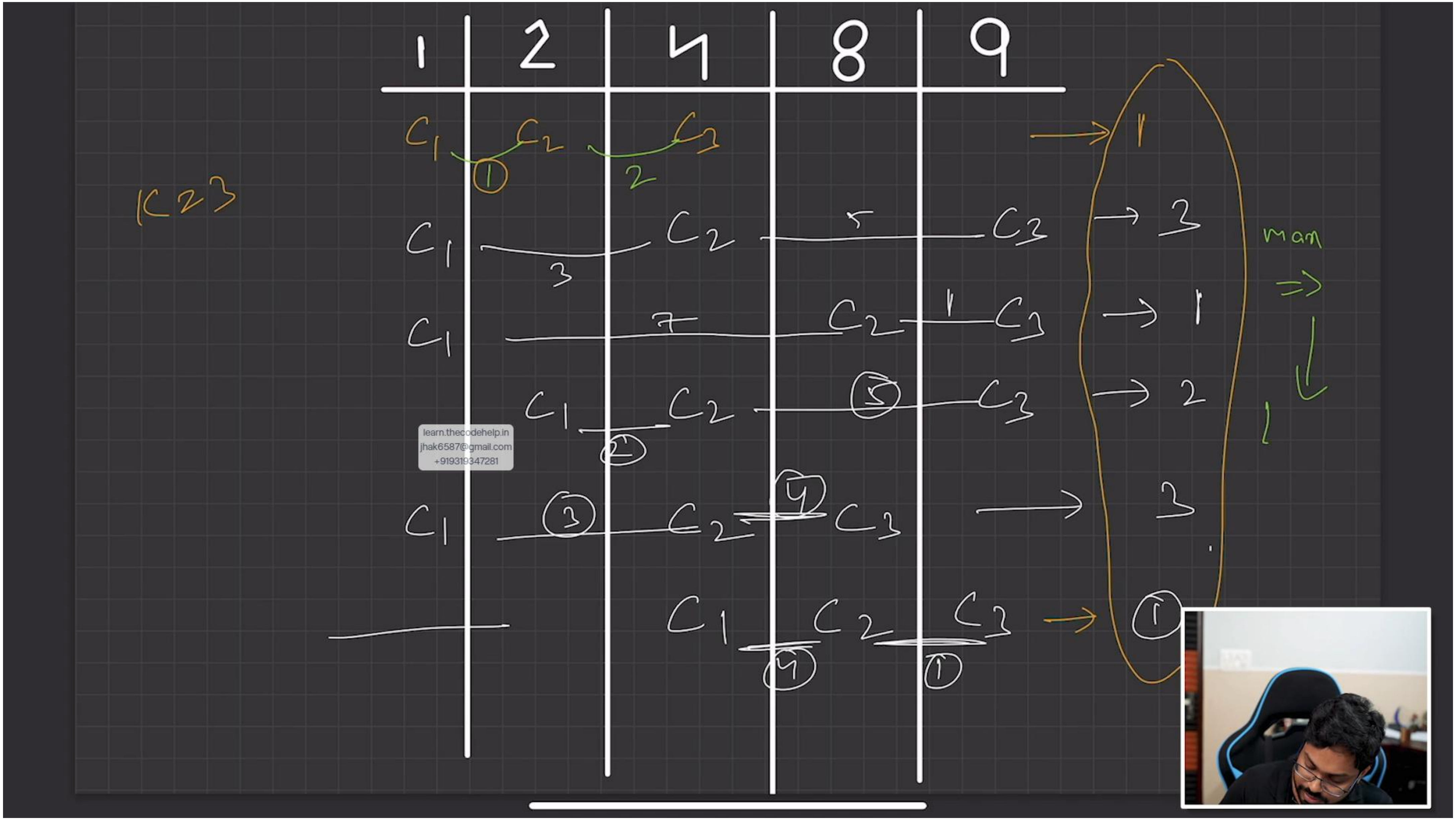
Aggresive Cow

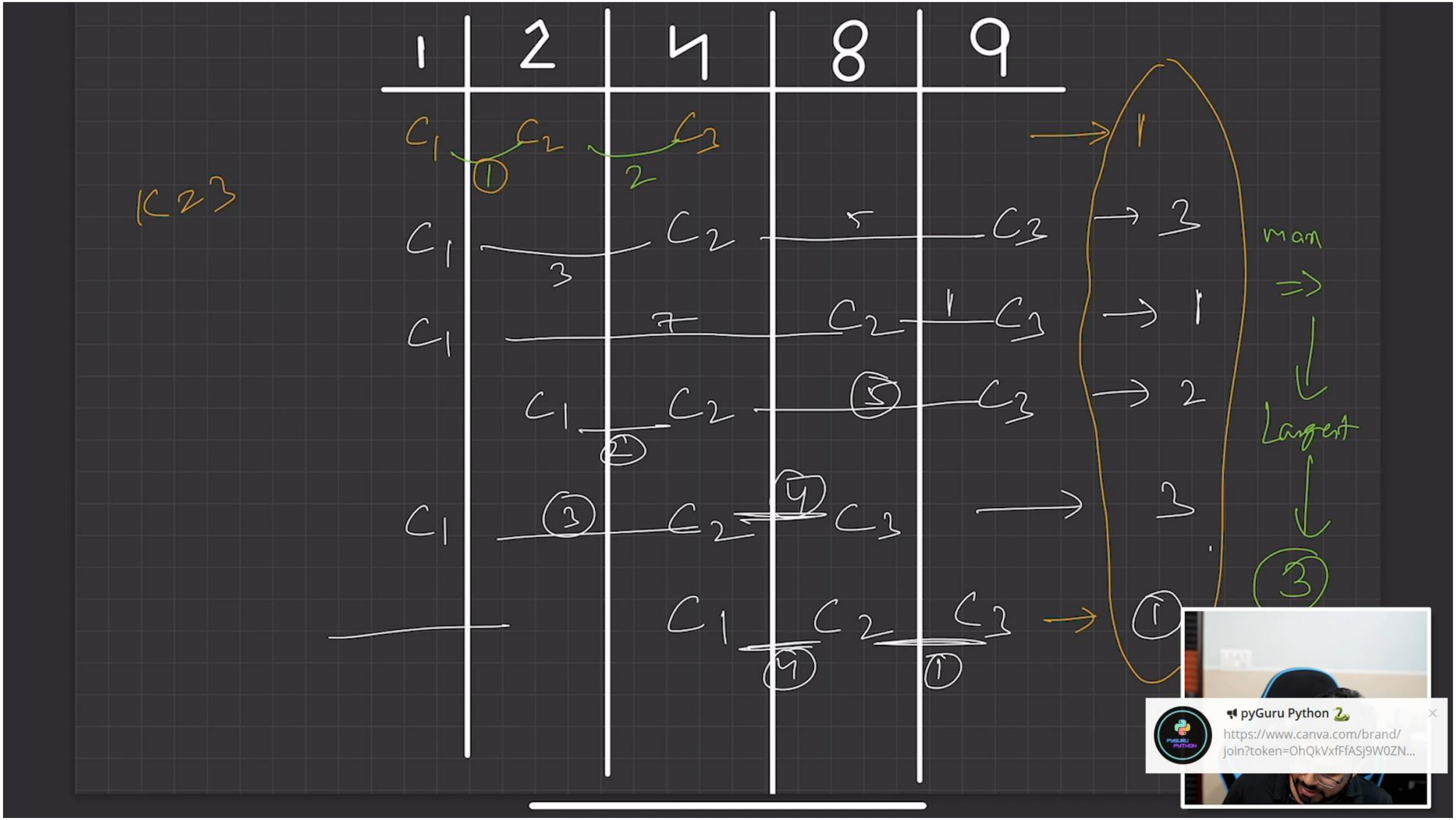
Stalls ⇒ Positions

1	2	4	8	9
0	1	2	3	4

N = 5

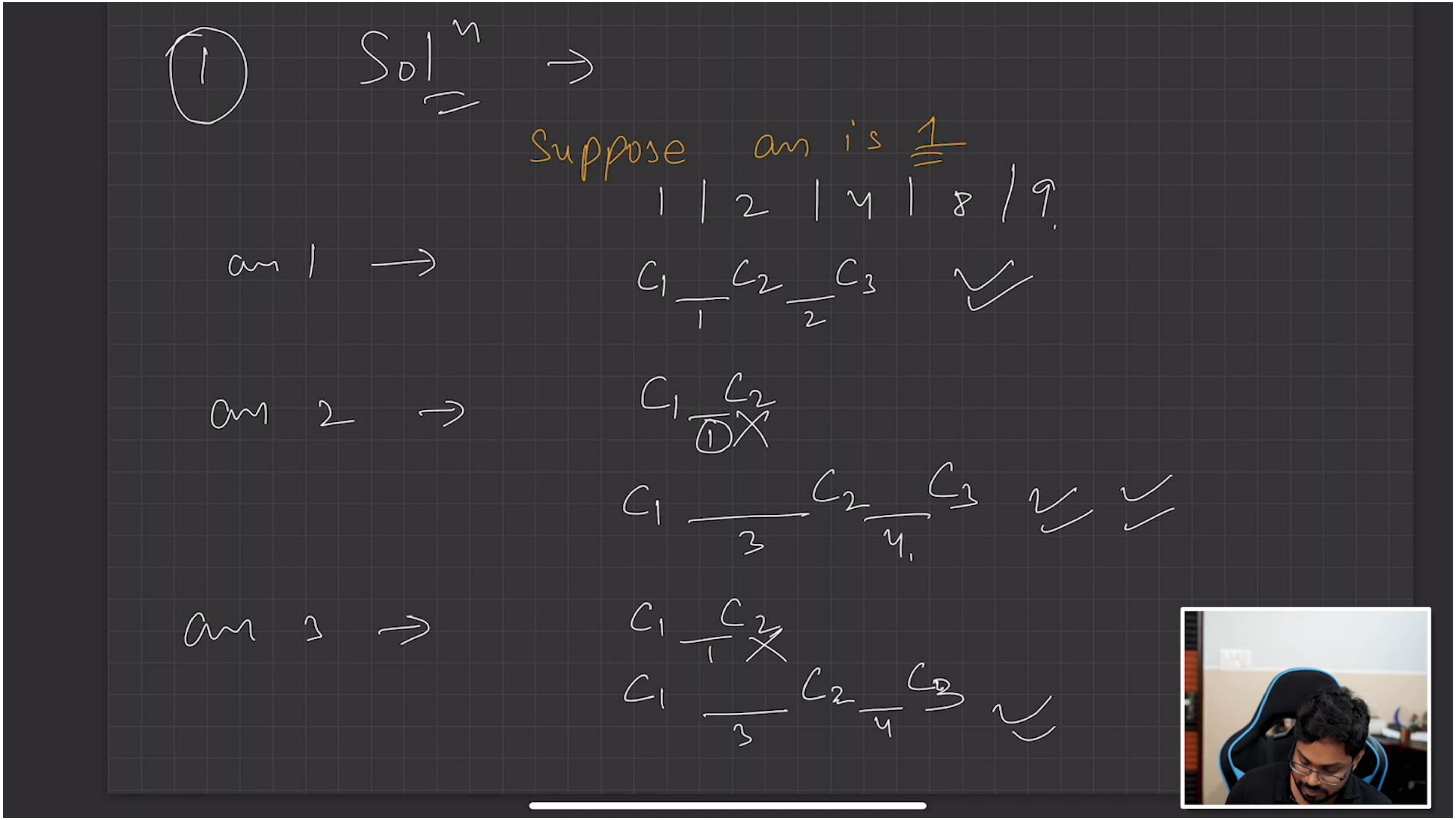
k (cows) = 3


1 | 2 | 4 | 8 | 9
1<2<3
C1 C2 C3
1 2
C1 C2 C3
3 5
C1 C2 C3
7 1
C1 C2 C3
2 5
C1 C2 C3
3 4
C1 C2 C3
4 1
1
3
1
2
3
1
max



pyGuru Python 🐍

https://www.canva.com/brand/join?token=OhQkVxfFfASj9W0ZN...



① Sol^n →

Suppose ans is 1

1 | 2 | 4 | 8 | 9.

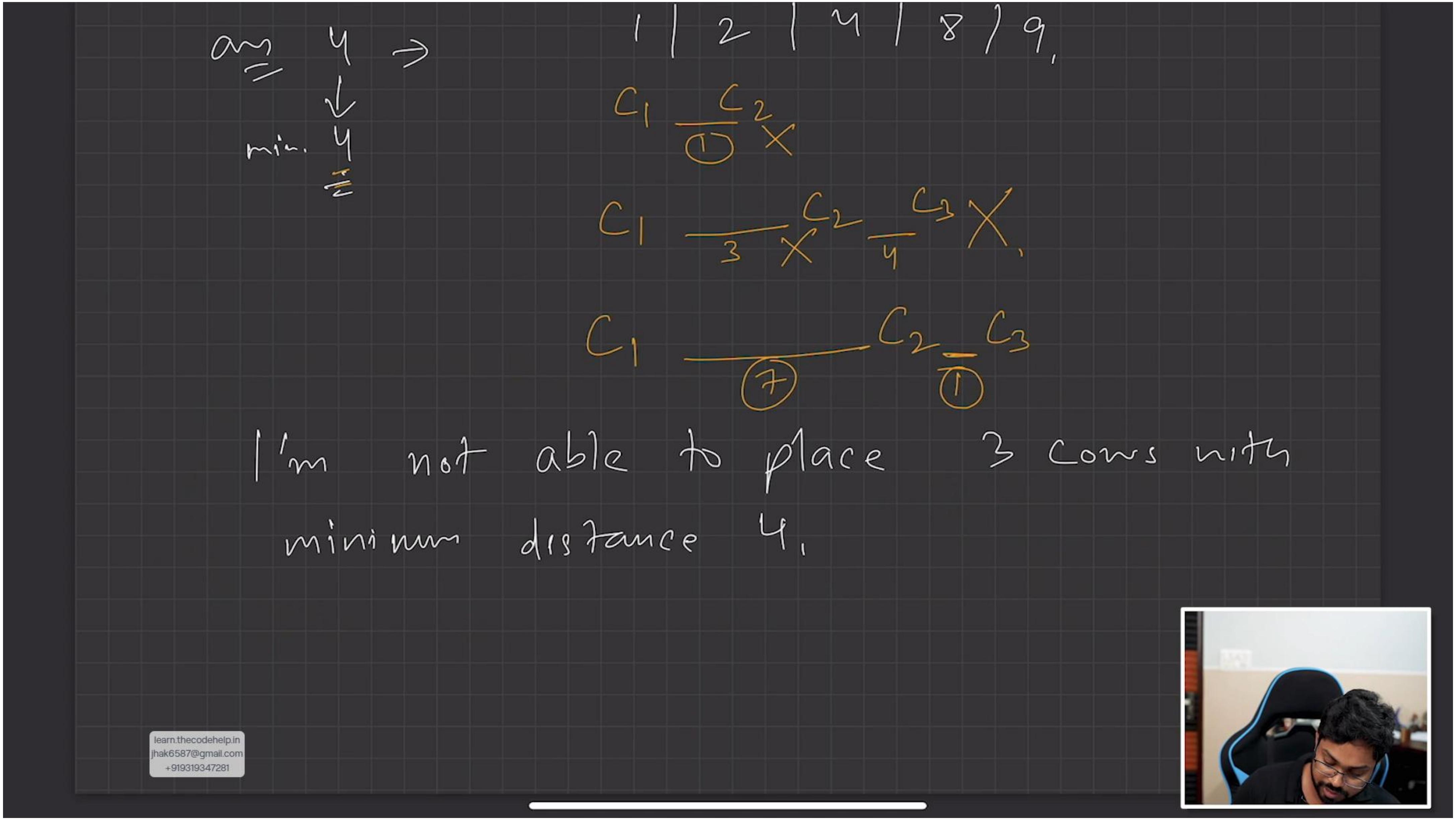
ans 1 → $C_1 \underset{1}{\overline{\quad\quad}} C_2 \underset{2}{\overline{\quad\quad}} C_3$ ✓

ans 2 → $C_1 \underset{\boxed{1}}{\overline{\quad}} C_2$ ✗

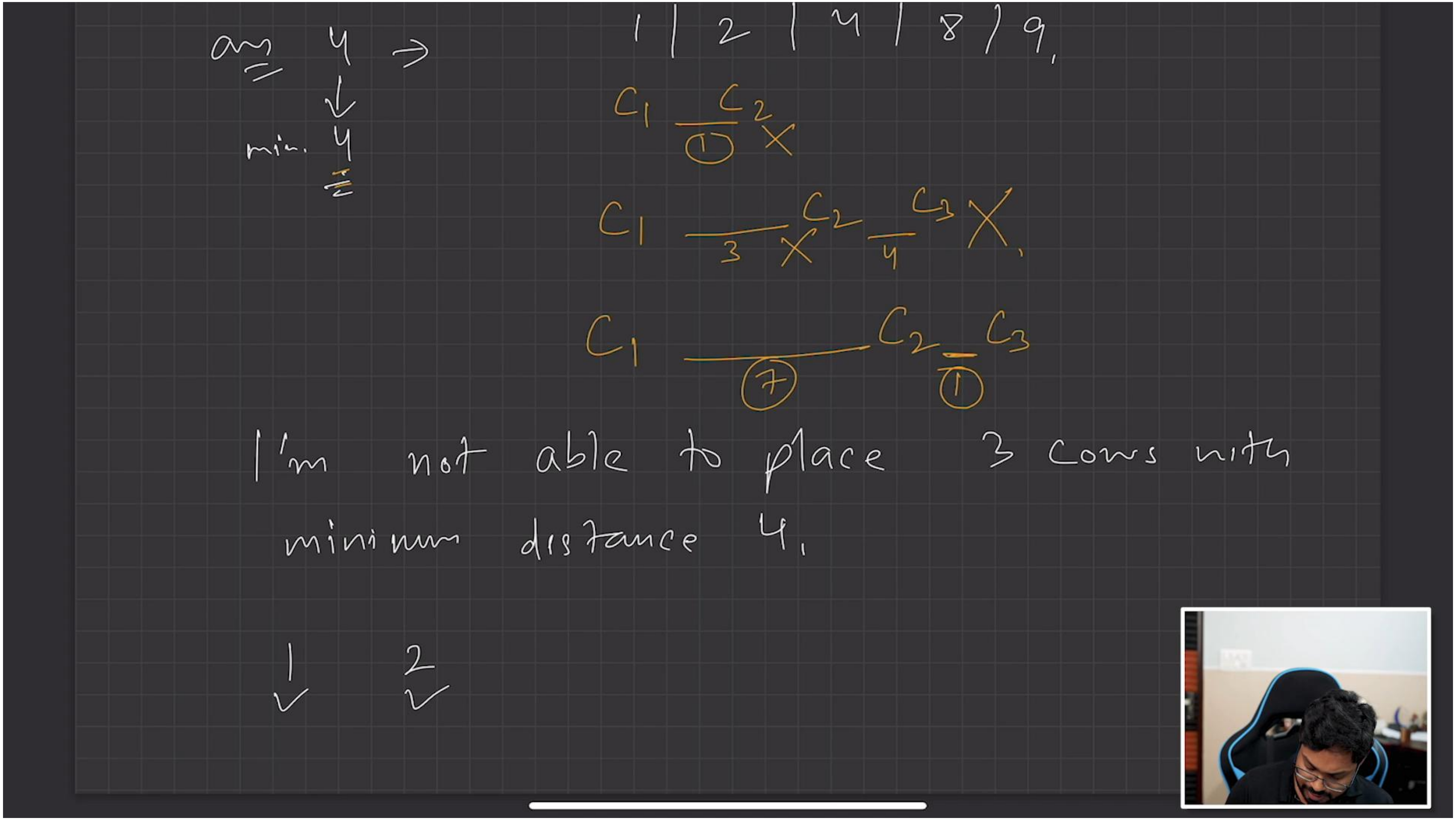
$C_1 \underset{3}{\overline{\quad\quad\quad}} C_2 \underset{4}{\overline{\quad\quad}} C_3$ ✓ ✓

ans 3 → $C_1 \underset{1}{\overline{\quad}} C_2$ ✗

$C_1 \underset{3}{\overline{\quad\quad\quad}} C_2 \underset{4}{\overline{\quad\quad}} C_3$ ✓



I'm not able to place 3 cows with minimum distance 4.



ans 4 →

↓

min. 4

1 | 2 | 4 | 8 | 9.

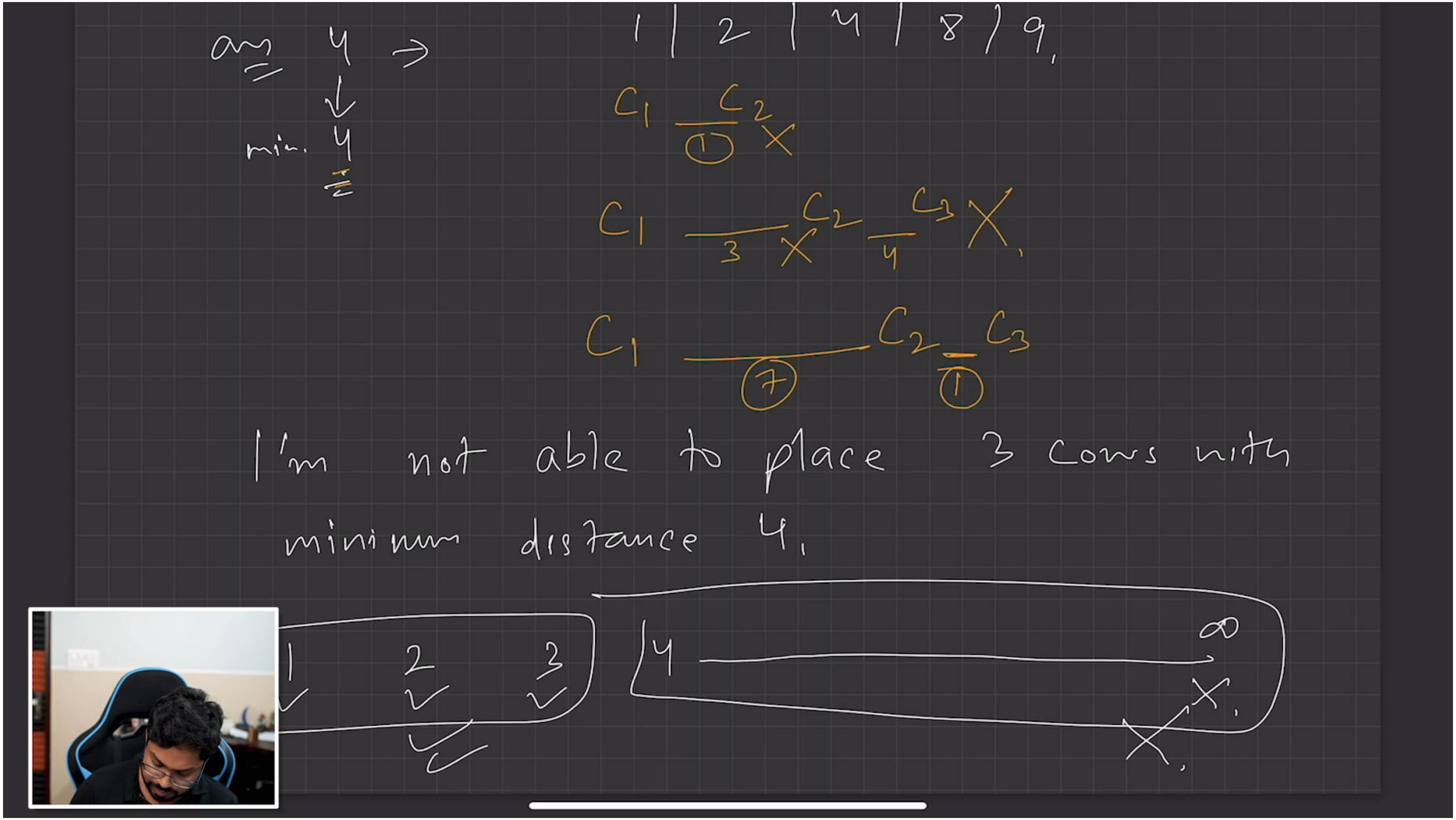
C1 — C2 X
1

C1 — X C2 — C3 X.
3 4

C1 — C2 — C3
7 1

I'm not able to place 3 cows with minimum distance 4.

1 ✓ 2 ✓



ans 4 →

↓

min. 4

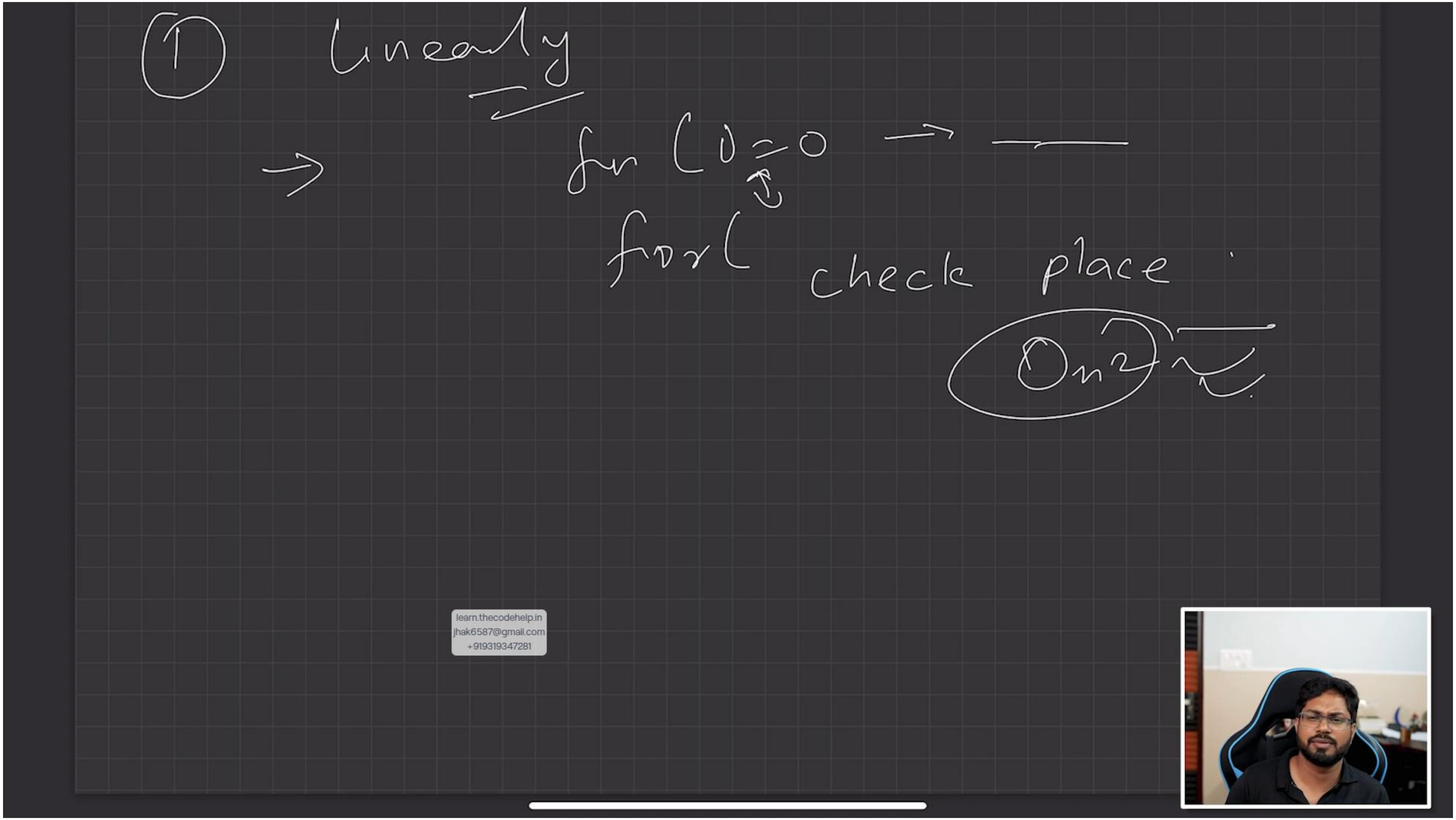
1 | 2 | 4 | 8 | 9,

C1 —①— C2 X

C1 —3— X C2 —4— C3 X.

C1 —⑦— C2 —①— C3

I'm not able to place 3 cows with minimum distance 4.



① linearly

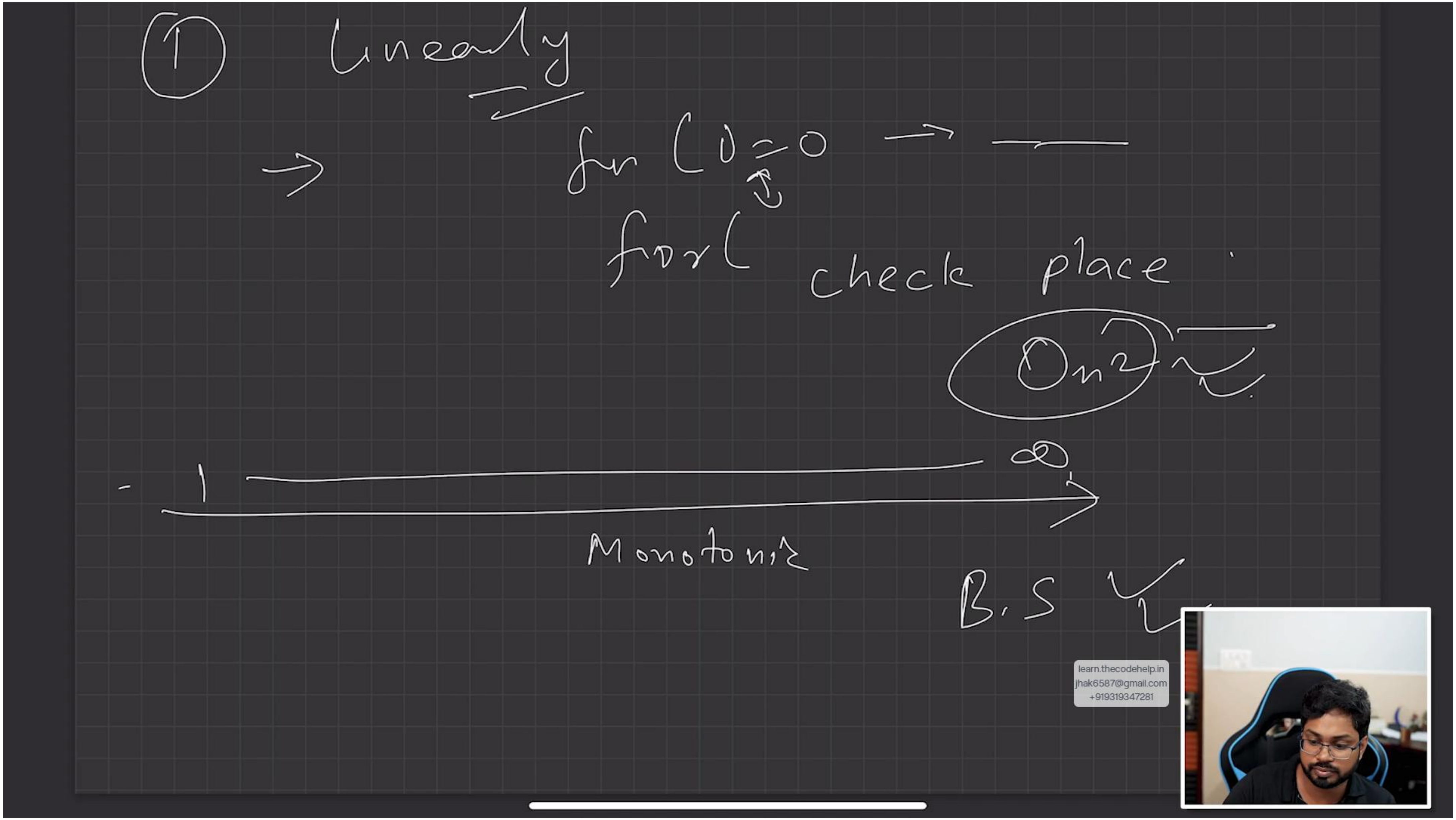
→

for (i = 0 → ____

for (

check place

$O(n^2)$



① Linearly

for (i = 0 →

for (

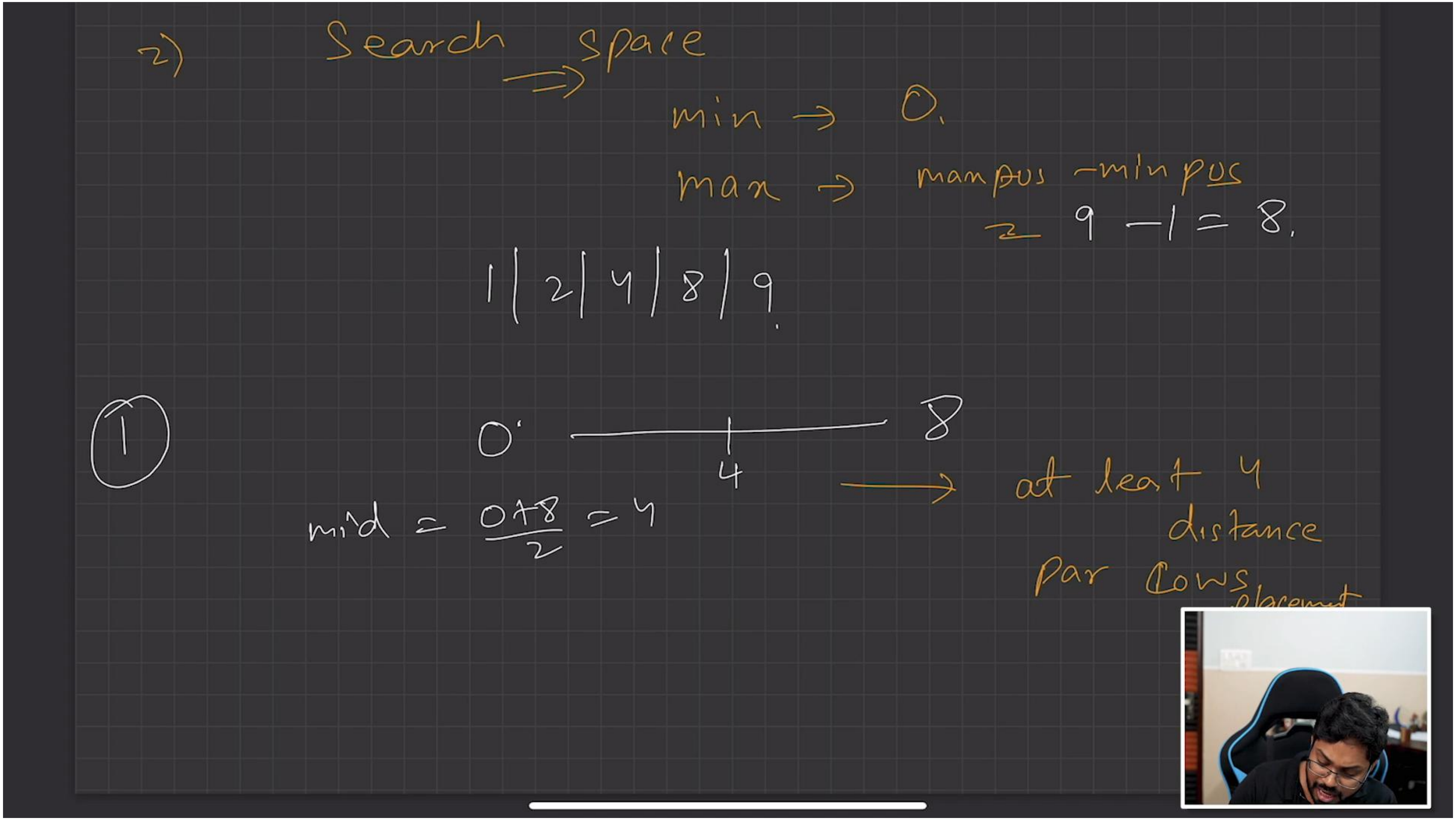
check place

$O(n^2)$

∞

Monotonic

B.S ✓



2) Search space

$\Rightarrow$ min → 0.

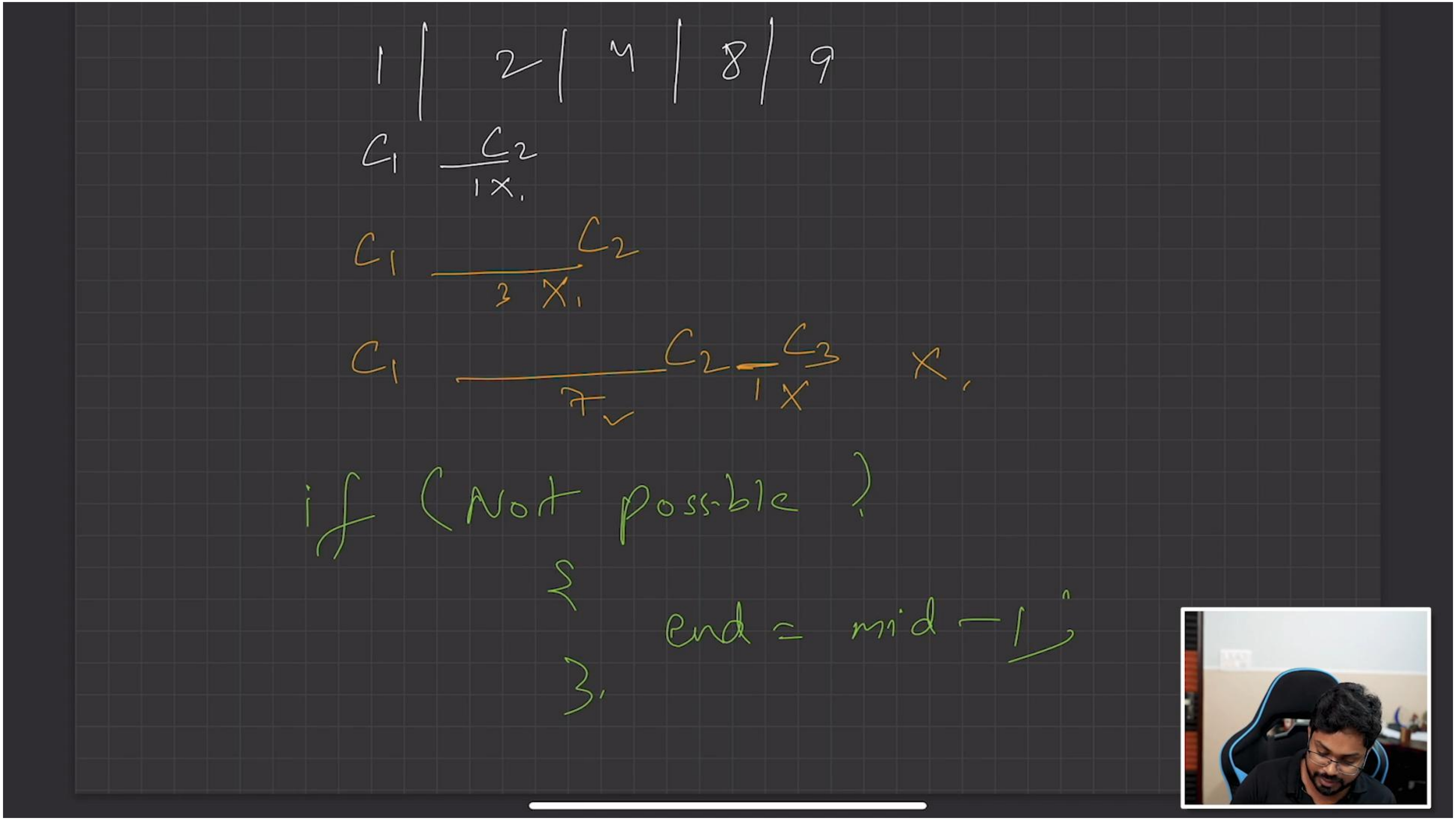
max → maxpos − minpos

= 9 − 1 = 8.

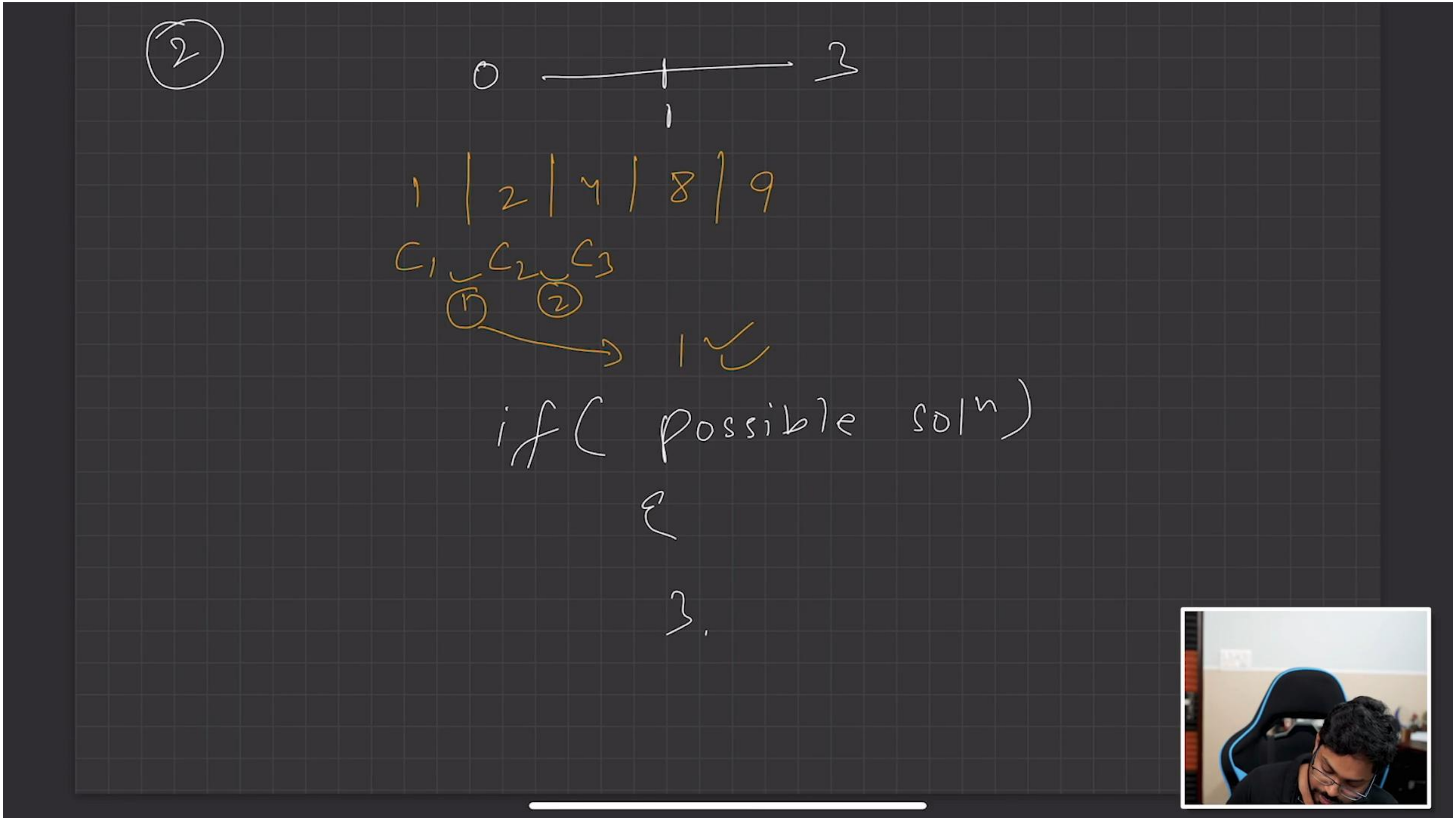
1 | 2 | 4 | 8 | 9

① 0 ——— 4 ——— 8 → at leat 4 distance par cows placement

$\text{mid} = \frac{0+8}{2} = 4$



```
if (Not possible)
    {
        end = mid - 1;
    }
```



2

0 — 1 — 3

1 | 2 | 4 | 8 | 9

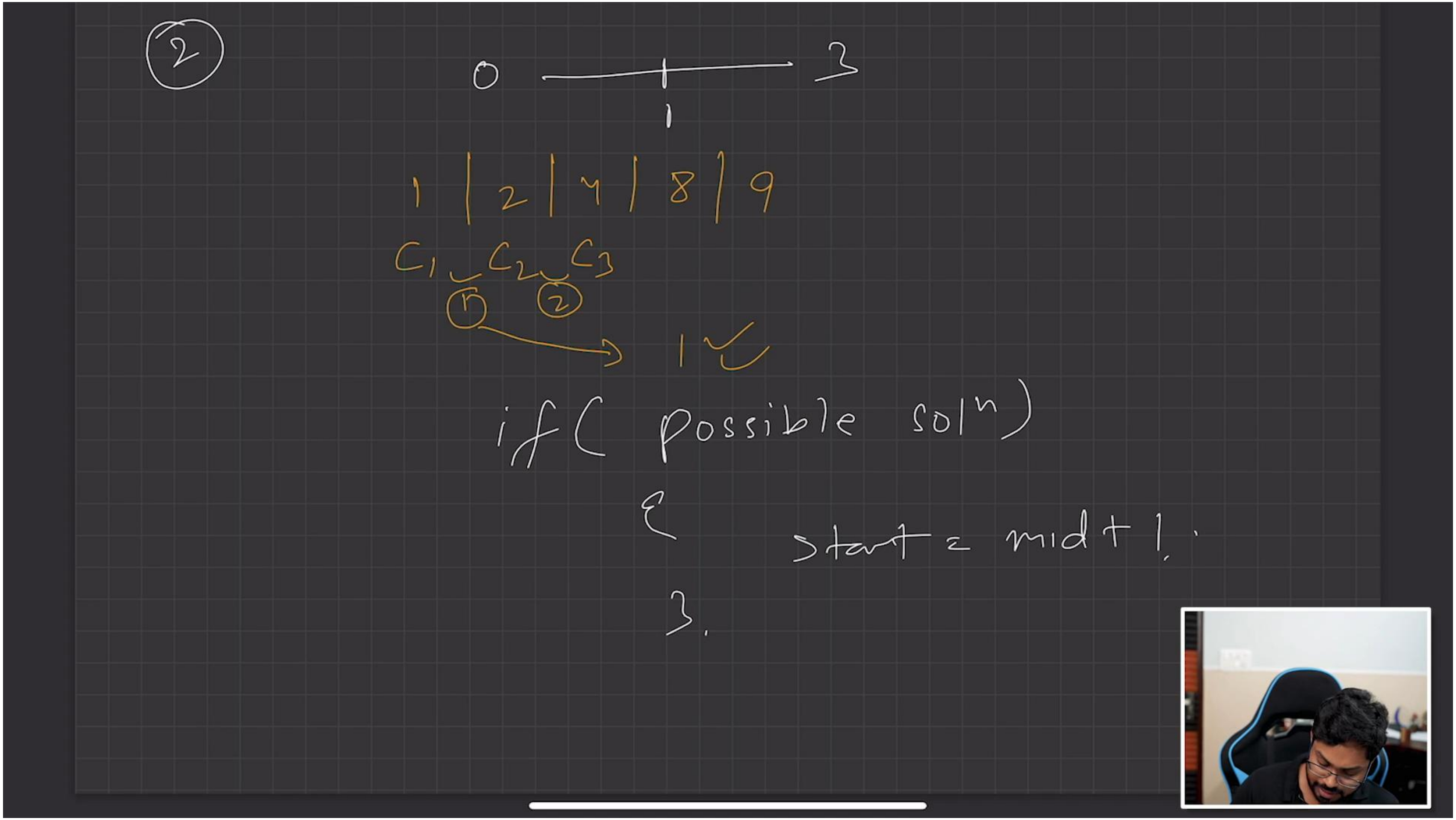
$C_1 \underbrace{\quad}_{1} C_2 \underbrace{\quad}_{2} C_3$

1 → 1 ✓

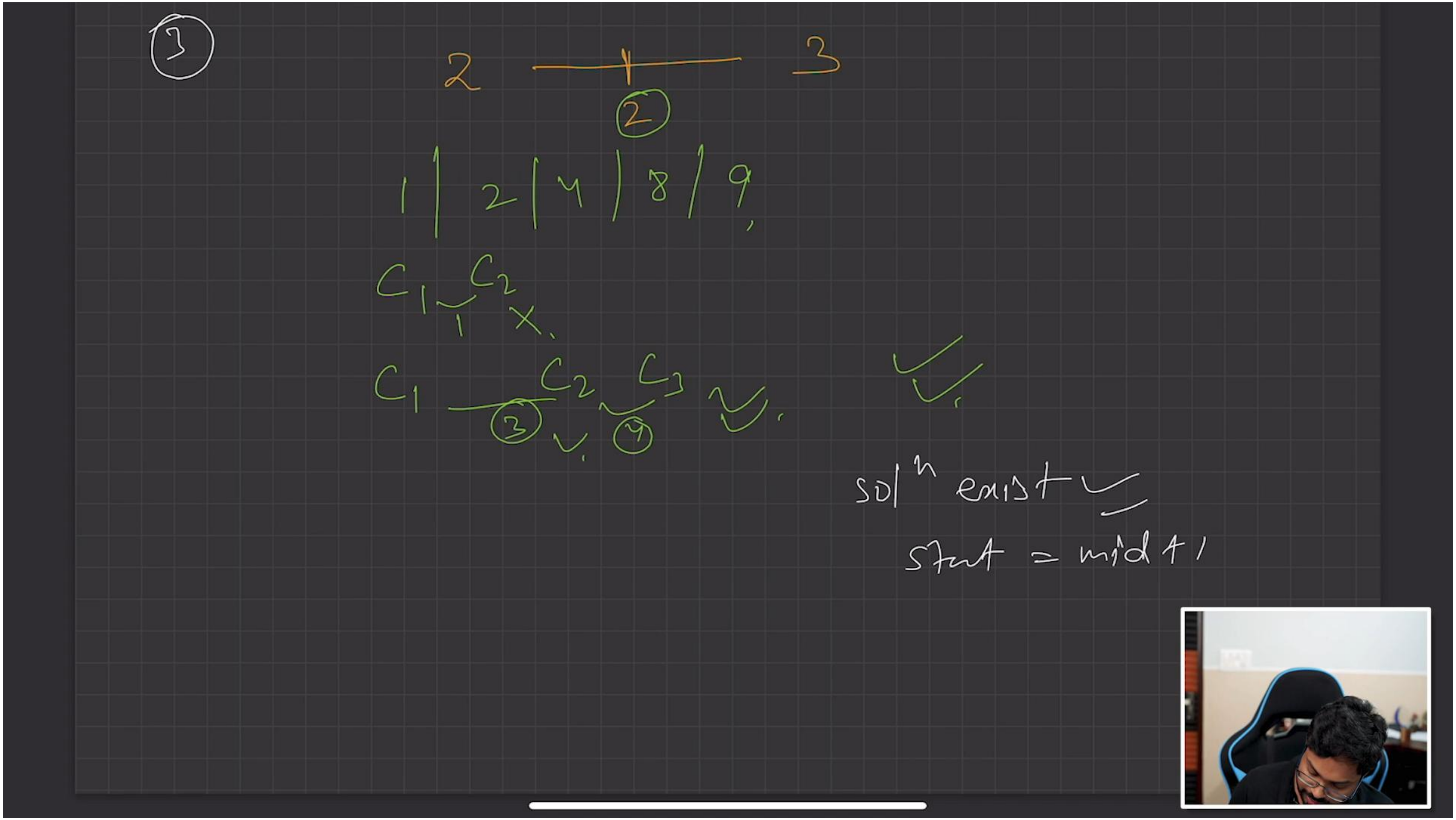
if(possible sol^n)

{

}.

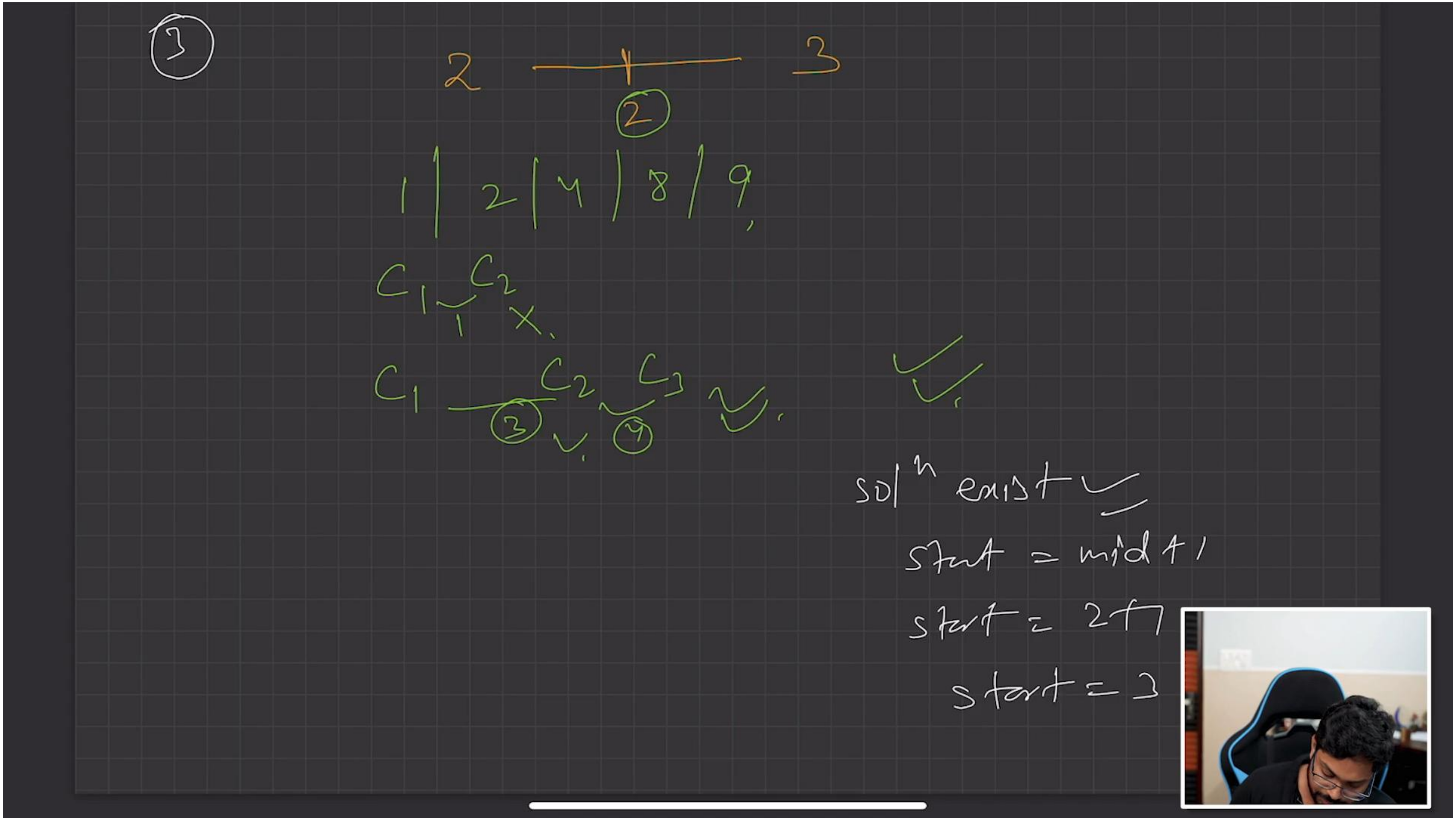


```
if( possible sol^n)
{
        start = mid+1.
}.
```



sol^n exist ✓

start = mid + 1

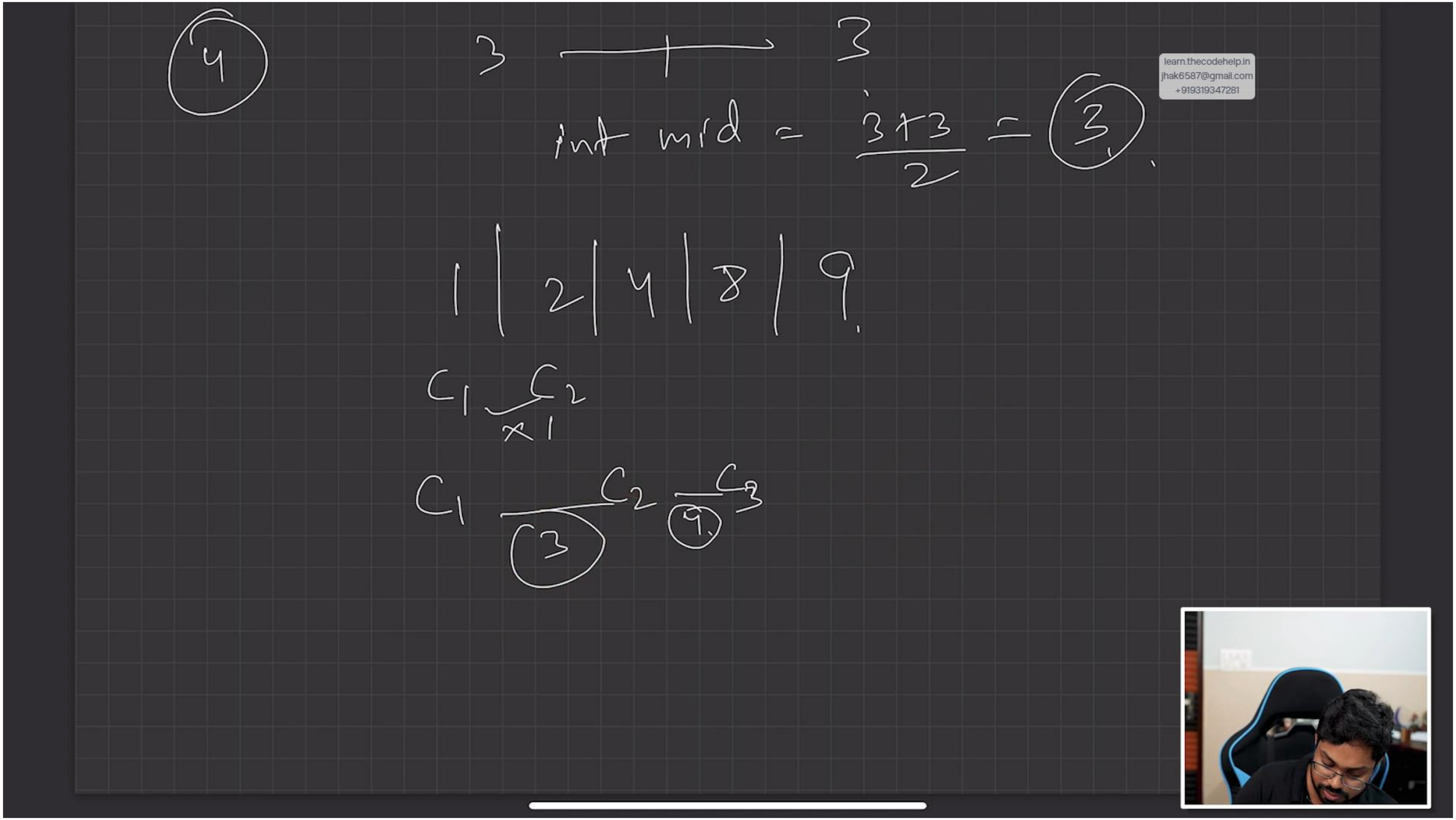


sol^n exist ✓

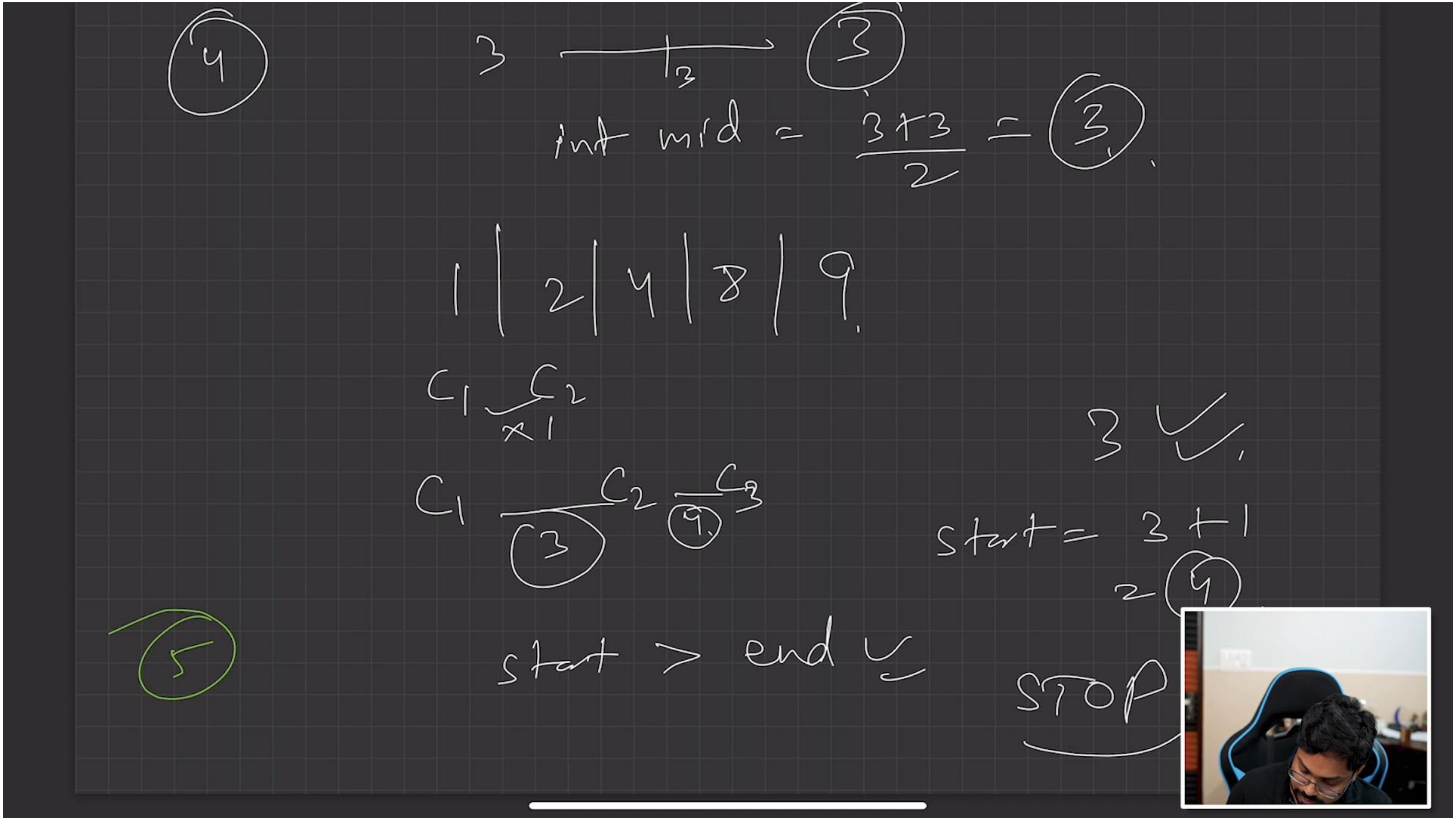
start = mid + 1

start = 2 + 1

start = 3



int mid = $\frac{3+3}{2}$ = $\boxed{3}$



$\boxed{4}$

$3 \xrightarrow{3} \boxed{3}$

int mid $= \frac{3+3}{2} = \boxed{3}$.

1 | 2 | 4 | 8 | 9.

$C_1 \underset{x1}{—} C_2$

$C_1 \underset{\boxed{3}}{—} C_2 \underset{\boxed{9}}{—} C_3$

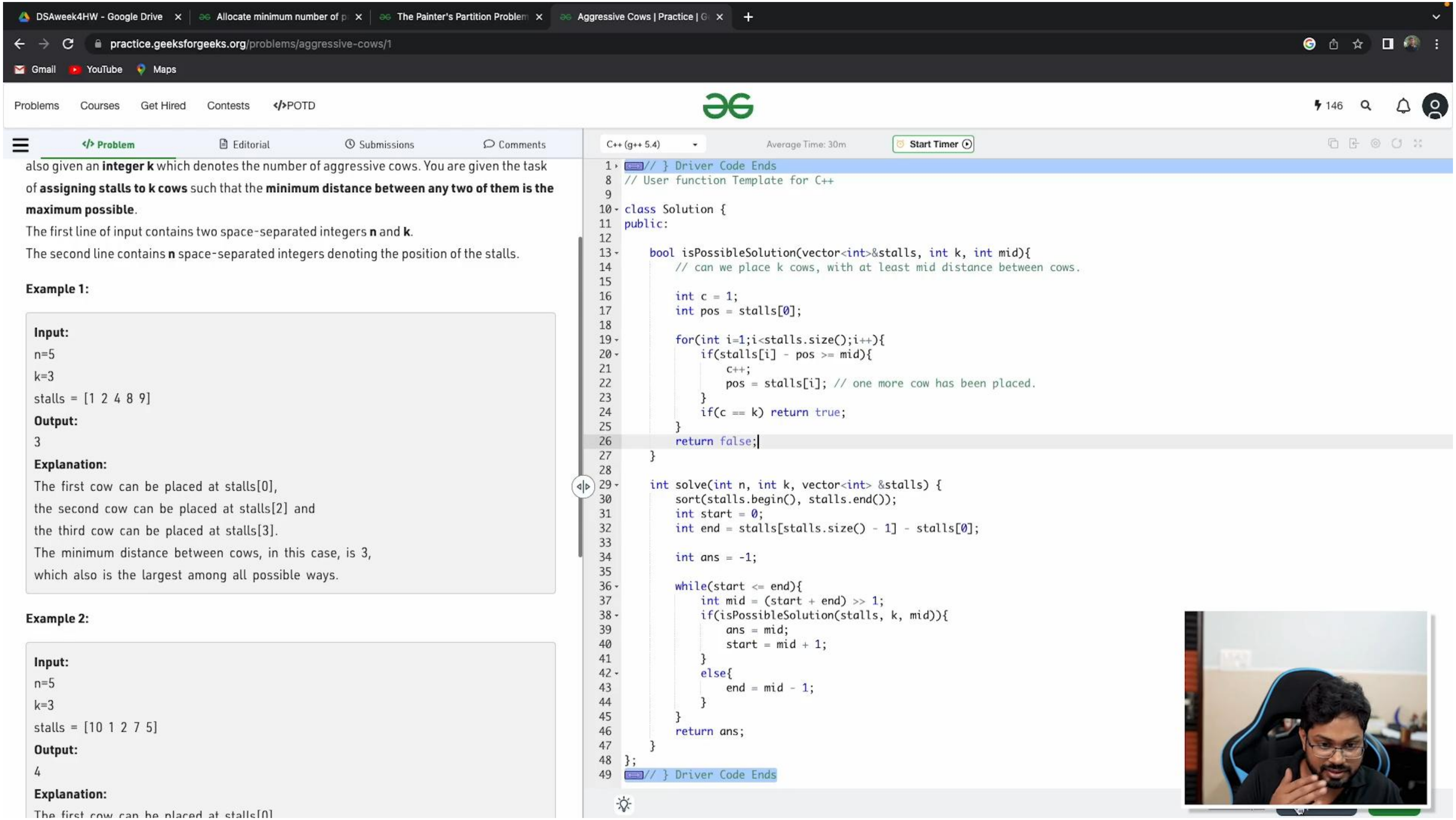
3 ✓✓

start $= 3+1$
$= \boxed{4}$

$\boxed{5}$

start > end ✓

STOP


DSAweek4HW - Google Drive
Allocate minimum number of p
The Painter's Partition Problem
Aggressive Cows | Practice | G
practice.geeksforgeeks.org/problems/aggressive-cows/1
Gmail
YouTube
Maps
Problems
Courses
Get Hired
Contests
</>POTD
146
</> Problem
Editorial
Submissions
Comments
C++ (g++ 5.4)
Average Time: 30m
Start Timer

also given an **integer k** which denotes the number of aggressive cows. You are given the task of **assigning stalls to k cows** such that the **minimum distance between any two of them is the maximum possible**.

The first line of input contains two space-separated integers **n** and **k**.

The second line contains **n** space-separated integers denoting the position of the stalls.

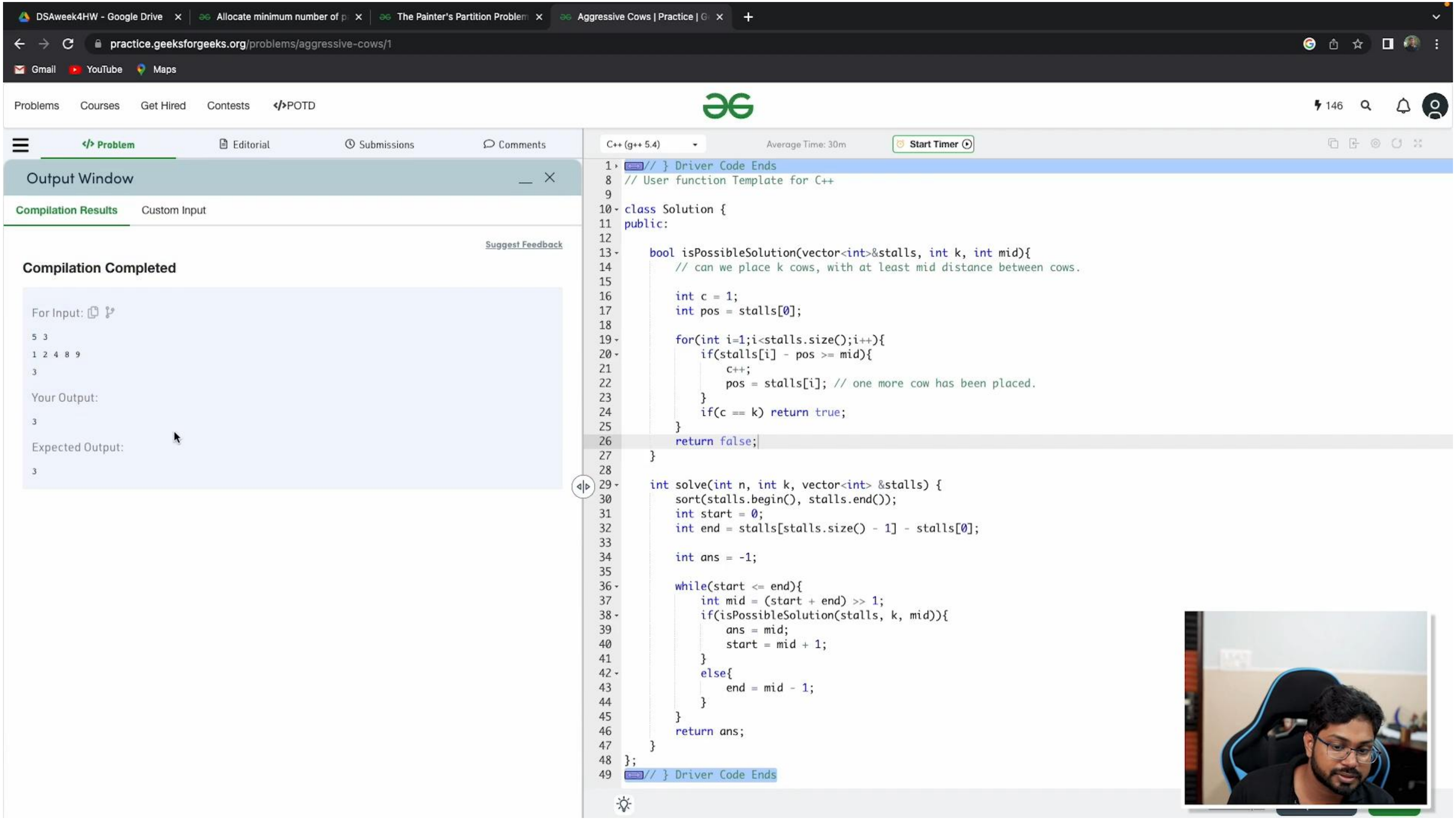
Example 1:

```
Input:
n=5
k=3
stalls = [1 2 4 8 9]
Output:
3
Explanation:
The first cow can be placed at stalls[0],
the second cow can be placed at stalls[2] and
the third cow can be placed at stalls[3].
The minimum distance between cows, in this case, is 3,
which also is the largest among all possible ways.
```

Example 2:

```
Input:
n=5
k=3
stalls = [10 1 2 7 5]
Output:
4
Explanation:
The first cow can be placed at stalls[0]
```

```cpp
// } Driver Code Ends
// User function Template for C++

class Solution {
public:

    bool isPossibleSolution(vector<int>&stalls, int k, int mid){
        // can we place k cows, with at least mid distance between cows.

        int c = 1;
        int pos = stalls[0];

        for(int i=1;i<stalls.size();i++){
            if(stalls[i] - pos >= mid){
                c++;
                pos = stalls[i]; // one more cow has been placed.
            }
            if(c == k) return true;
        }
        return false;
    }

    int solve(int n, int k, vector<int> &stalls) {
        sort(stalls.begin(), stalls.end());
        int start = 0;
        int end = stalls[stalls.size() - 1] - stalls[0];

        int ans = -1;

        while(start <= end){
            int mid = (start + end) >> 1;
            if(isPossibleSolution(stalls, k, mid)){
                ans = mid;
                start = mid + 1;
            }
            else{
                end = mid - 1;
            }
        }
        return ans;
    }
};
// } Driver Code Ends
```


DSAweek4HW - Google Drive
Allocate minimum number of p
The Painter's Partition Problem
Aggressive Cows | Practice | G
practice.geeksforgeeks.org/problems/aggressive-cows/1
Gmail
YouTube
Maps
Problems
Courses
Get Hired
Contests
</>POTD
146
</> Problem
Editorial
Submissions
Comments
C++ (g++ 5.4)
Average Time: 30m
Start Timer
Suggest Feedback

Output Window

Compilation Results   Custom Input

Compilation Completed

For Input:

```
5 3
1 2 4 8 9
3
```

Your Output:

```
3
```

Expected Output:

```
3
```

```cpp
// } Driver Code Ends
// User function Template for C++

class Solution {
public:

    bool isPossibleSolution(vector<int>&stalls, int k, int mid){
        // can we place k cows, with at least mid distance between cows.

        int c = 1;
        int pos = stalls[0];

        for(int i=1;i<stalls.size();i++){
            if(stalls[i] - pos >= mid){
                c++;
                pos = stalls[i]; // one more cow has been placed.
            }
            if(c == k) return true;
        }
        return false;
    }

    int solve(int n, int k, vector<int> &stalls) {
        sort(stalls.begin(), stalls.end());
        int start = 0;
        int end = stalls[stalls.size() - 1] - stalls[0];

        int ans = -1;

        while(start <= end){
            int mid = (start + end) >> 1;
            if(isPossibleSolution(stalls, k, mid)){
                ans = mid;
                start = mid + 1;
            }
            else{
                end = mid - 1;
            }
        }
        return ans;
    }
};
// } Driver Code Ends
```