→ Arrays → 1D static array =

Leetcode →

Ques → Optimal X

→ Brute force

vector → D.S

dynamic array

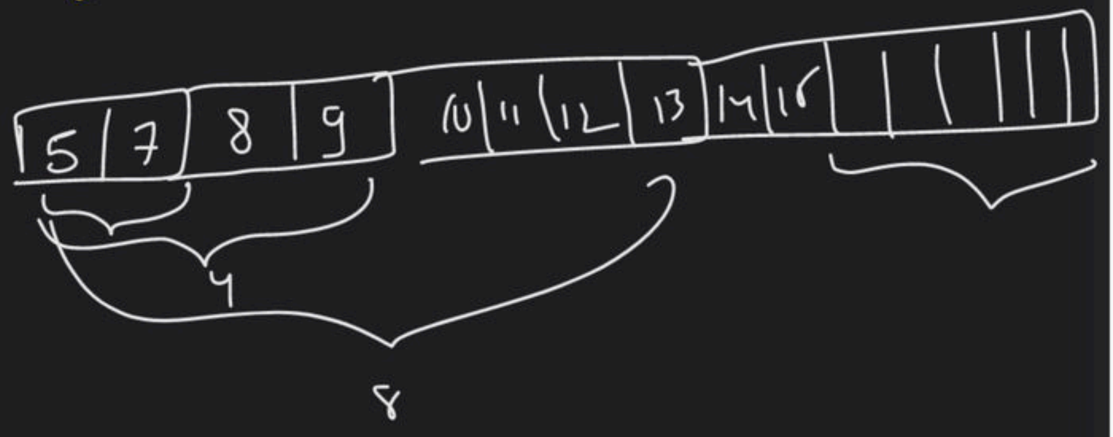
declare

```
vector<int> arr;
```

default = 0 → verify → 0/2

initialise

```
vector<int> arr(10,20,30);
vector<int> arr(10, -1);
```



```
int n;
cin>>n;
vector<int> arr(n);
```

`int arr[n]` ✗

`sizeof(arr)/sizeof(int)` → 6

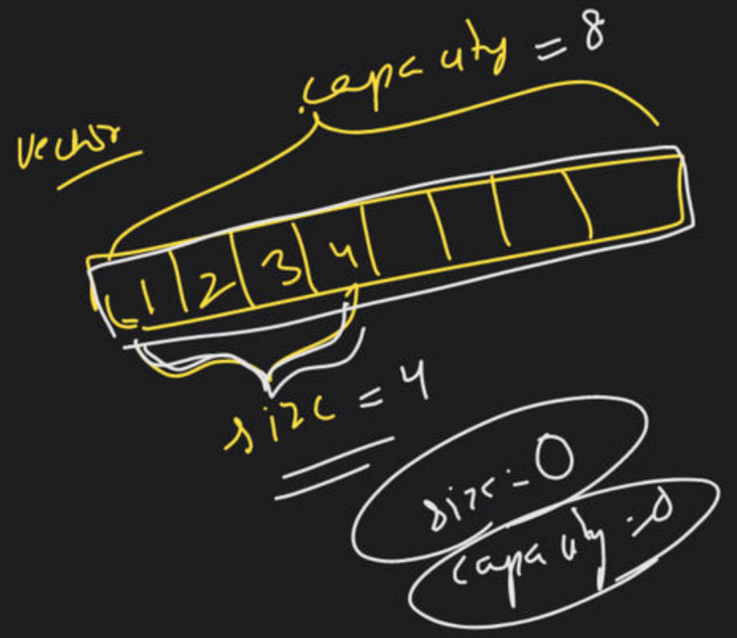
Input int vector →

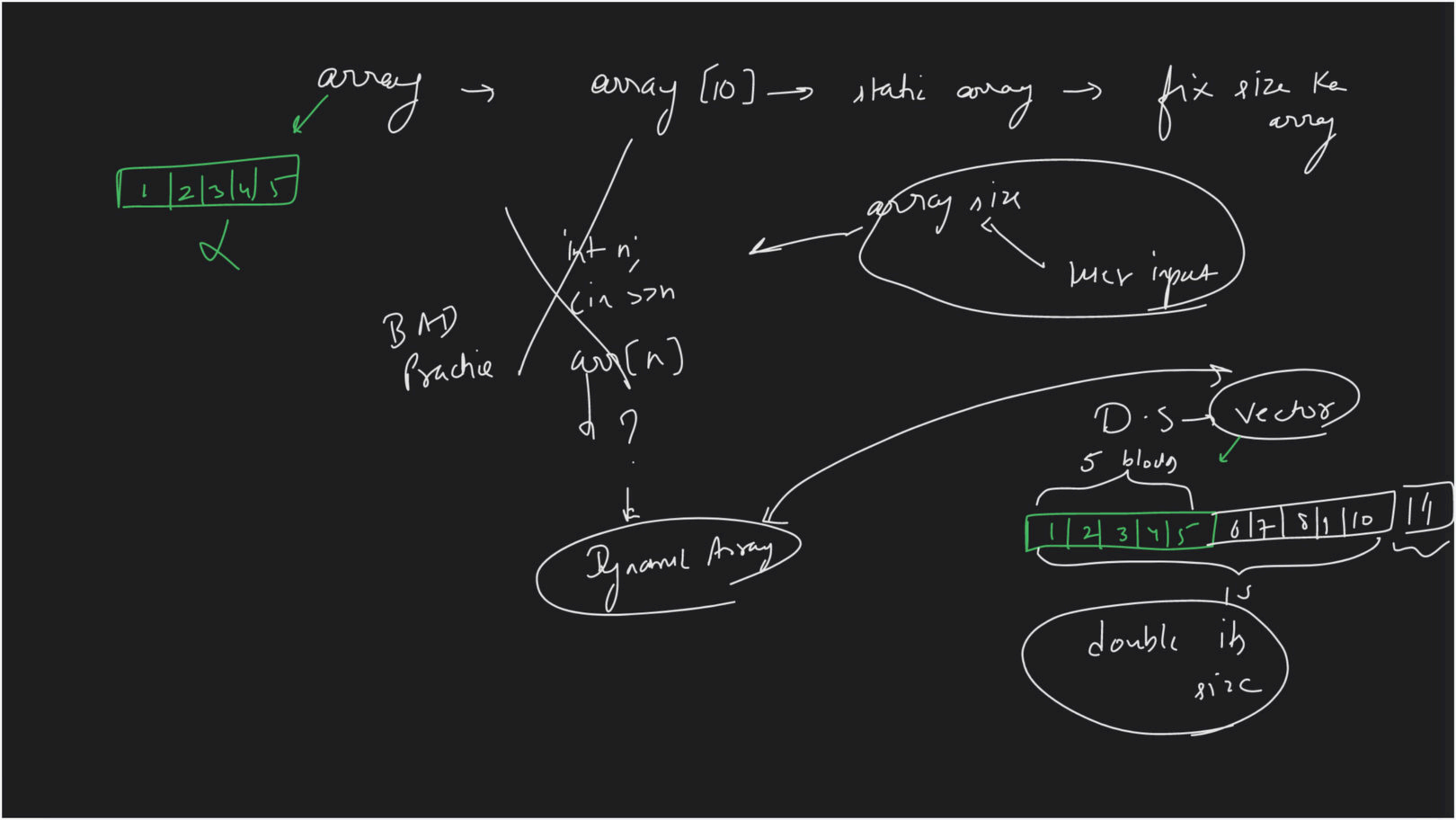
```
arr.push_back(5);
arr.push_back(7);
```

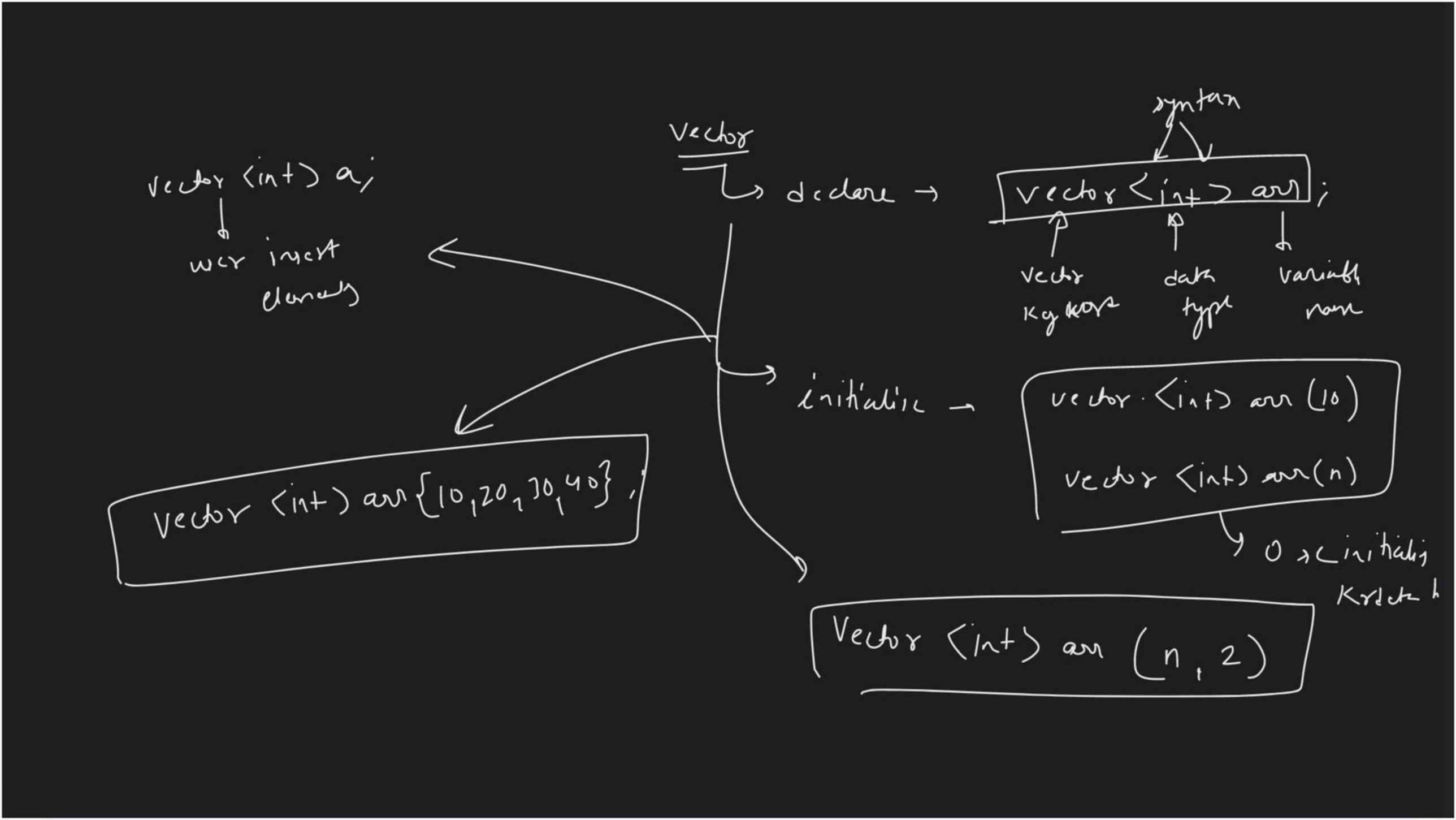
empty → `arr.empty();`

remove → `arr.pop_back();`

size → `arr.size();`




array → array[10] → static array → fix size ka array
1 2 3 4 5
array size
user input
int n;
cin >> n
arr[n]
BAD Practice
Dynamic Array
D.S → Vector
5 blocks
1 2 3 4 5 6 7 8 9 10 11
5
double ki size


Vector
declare →
syntax
vector <int> arr ;
vector keyword
data type
variable name
vector <int> a;
use insert elements
initialize →
vector <int> arr (10)
vector <int> arr(n)
0 → initialize
Vector <int> arr {10, 20, 30, 40} ;
Vector <int> arr (n, 2)

insert → ~~Vector~~ push_back()

vector<int> arr;

~~arr.push~~

arr.push_back(5);

arr.push_back(6);

remove → pop_back()

arr.pop_back()

size -> arr.size();

(no of element)

capacity -> arr.capacity()

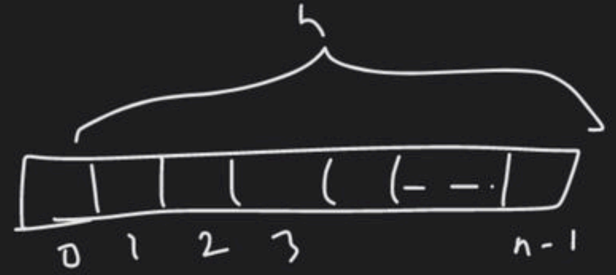
(how many elements it can store)

dot

empty -> arr.empty()

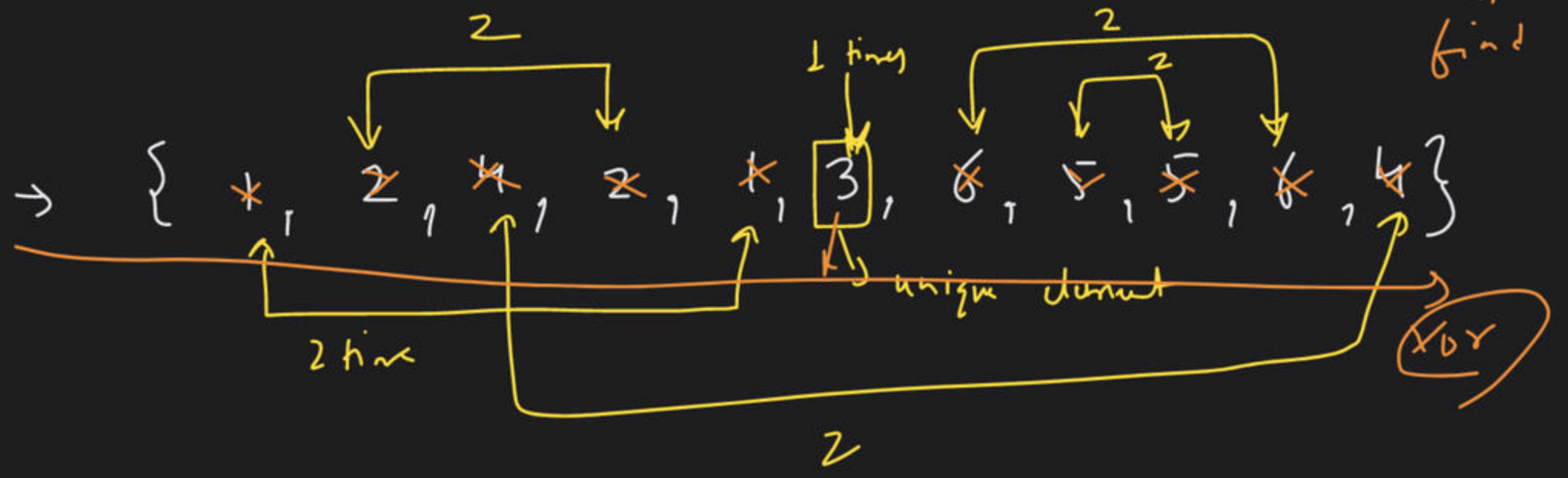
T F

how to access nth element in vector ?



① Find Unique Element

every element occurs twice except one → find



$1 \wedge 1 \rightarrow 0$

$2 \wedge 2 \rightarrow 0$

$\cancel{4} \wedge \cancel{4} \wedge 3 = 3$

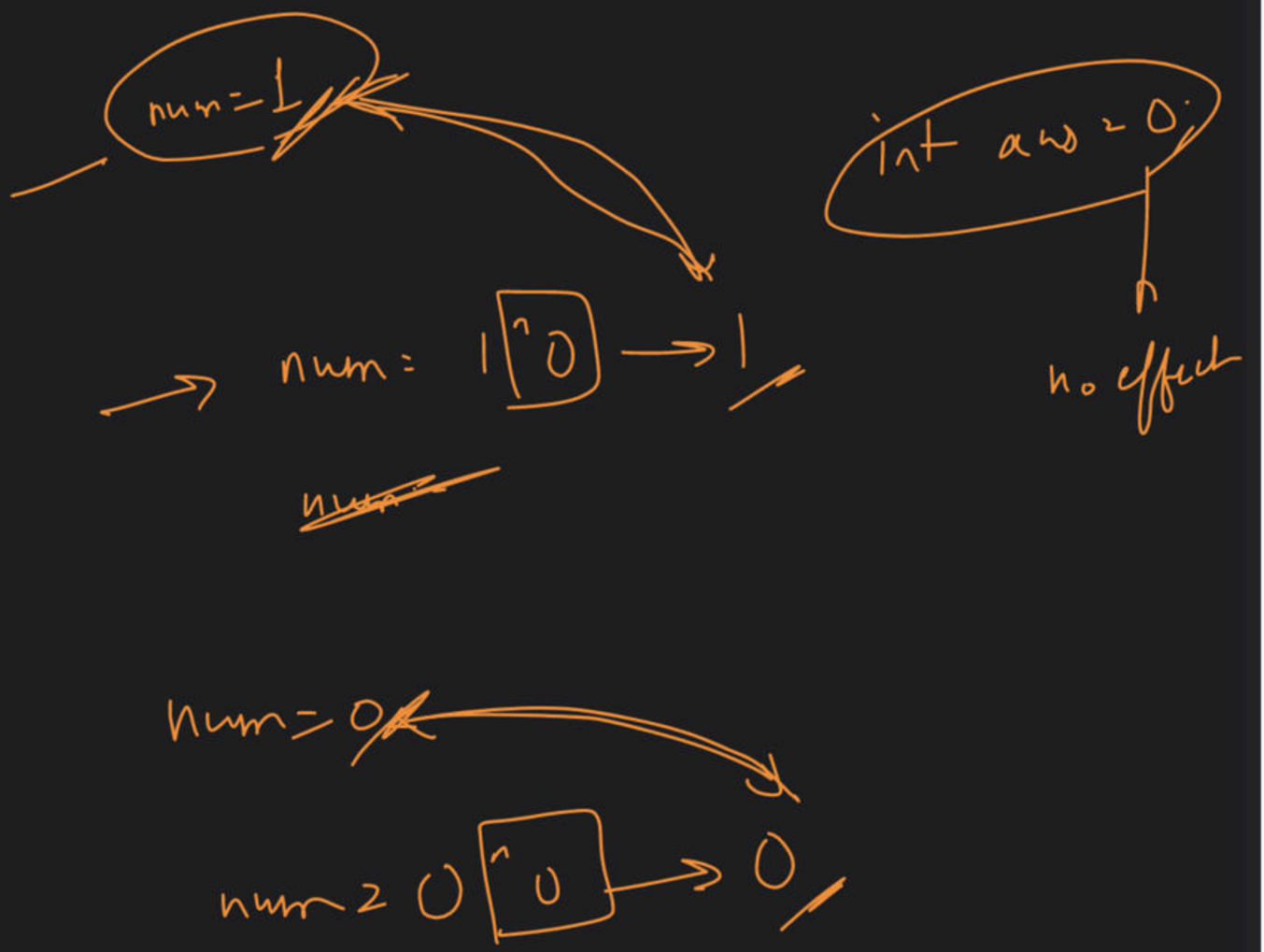
$\cancel{2} \wedge \cancel{2} \wedge \cancel{1} \wedge \cancel{1} \wedge 3 = 3$

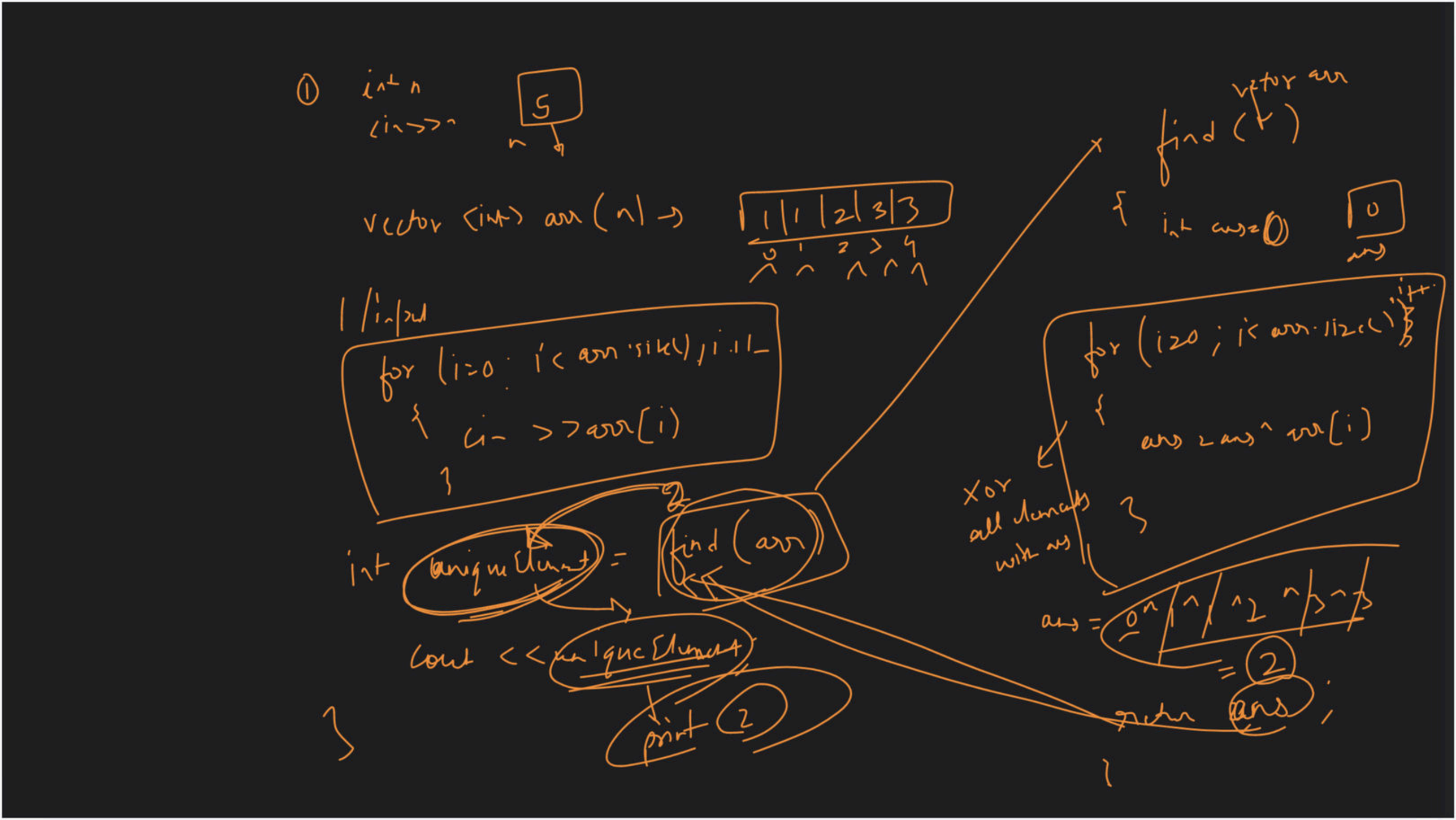
$\cancel{1} \wedge \cancel{1} \wedge \cancel{2} \wedge \cancel{2} \wedge (3) \wedge \cancel{4} \wedge \cancel{4} \wedge \cancel{5} \wedge \cancel{5} \wedge \cancel{6} \wedge \cancel{6} = (3)$

same element → cancel out

XOR → same → 0

Operator → diff → 1


num=1
int ans=0;
num: 1 0 → 1
n
no effect
num=0
num: 0 0 → 0


① int n
cin >> n
5
n
vector<int> arr(n) →
1 1 2 3 3
0 1 2 3 4
// input
for (i=0 ; i< arr.size() ; i++)
{ cin >> arr[i]
}
int uniqueElement = find(arr)
cout << uniqueElement
print (2)
}
find (vector arr)
{ int ans=0
0
ans
for (i=0 ; i< arr.size() ; i++)
{
ans = ans ^ arr[i]
}
XOR all elements with ans
ans = 0^1 / ^1 ^2 ^3^3
= 2
return ans ;
}

Union of 2 arrays

i/p ① a[] → {2, 4, 6, 8}

② b[] → {1, 3, 7}

algo:-

→ create an ans array/vector

→ put all element of a[] into ans array

→ put all ~~~~ b[] ~~~~

→ done

assum: ~~no~~ duplicate

duplicate exists

arr → {1, 2, 3}

brr → {4, 6}

ans[] → {1, 2, 3, 4, 6}

sorted ans

2 week

50+ code

10 Ques → 30

extra class

7-9 → Recl class

30 → Recorded solut

H/W

Union → (with duplicates)

intersection

a → { 1, 2, 4, 6, 8, 10 }

b → { 3, ~~4~~, 5, ~~6~~ }

INT_MIN INT_MIN → Union

{ INT_MIN, 1, 2, 4, 6, 8, 10, 3, 5, INT_MIN }

Union-1 old algo

{ 1, 2, 4, 6, 8, 10, 3, 5, 6 }

we want no repeated number

~~INT_MIN~~ ~~INT_MIN~~

```
if (value != INT_MIN)
{
ans.push_back(value);
}
```

Intersection of 2 arrays:-

a() intersection b()

i/p

a = {1, 2, 3, 4, 6, 8}

b = {3, 4, 9, 10}

out -> {3, 4}

common element

ans() -> {3, 4}

algo:-

for loop on a

for loop -> {

for loop on b

entire b array traverse

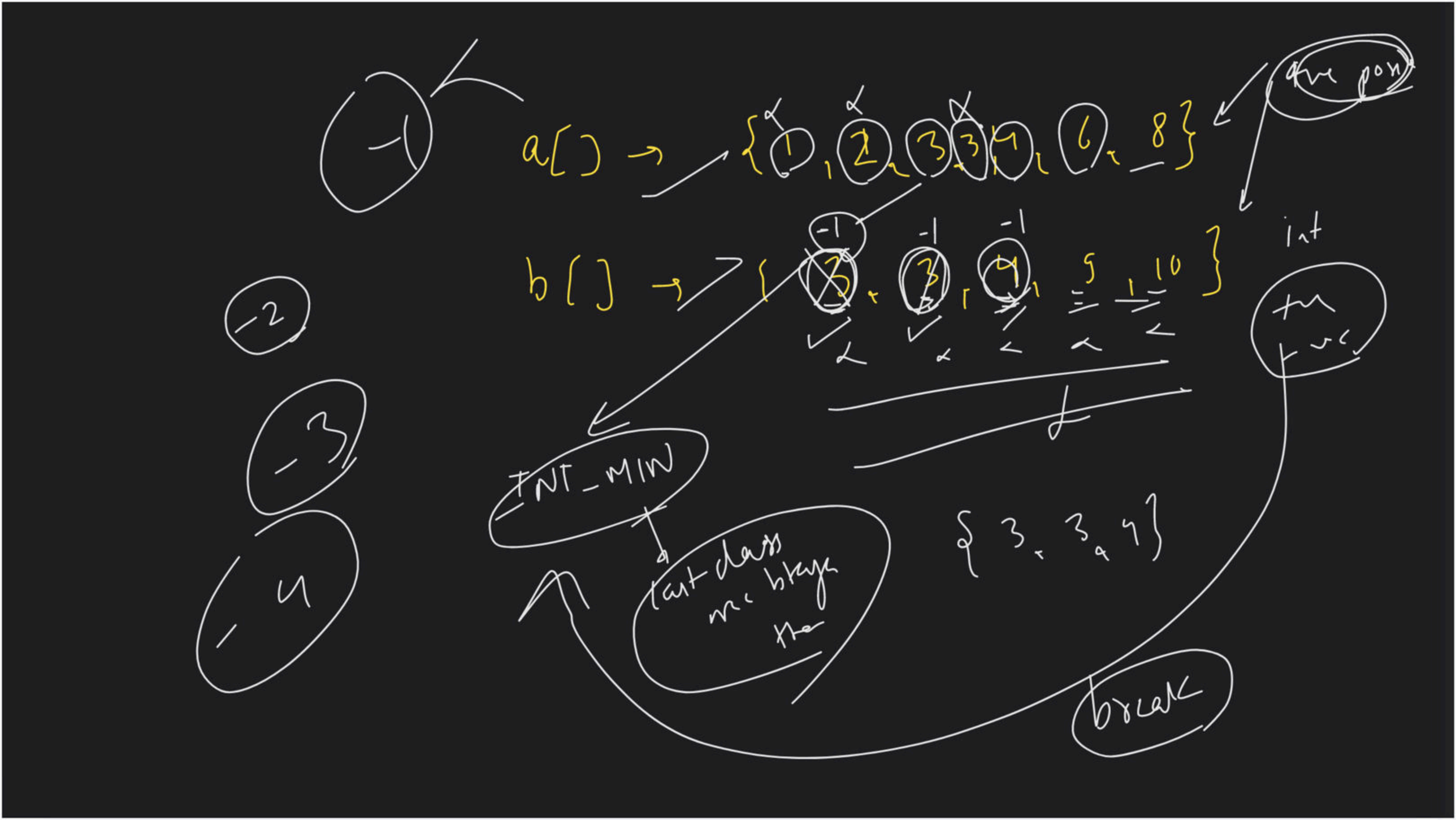
for loop on b

for loop on b

for loop on b

for loop on b

for loop on b


-1
a[] →
{1, 2, 3, 3, 7, 6, 8}
give pos
-1
-1
-1
b[] →
{3, 3, 4, 5, 10}
int
fun
-2
-3
-4
INT_MIN
last class
me bkya
tha
{3, 3, 4}
break

Pair Sum :-

i/p → a[] → {1, 3, 5, 7, 2, 4, 6}

sum = 9

find a pair that upon addition gives value equal to sum

Brute force

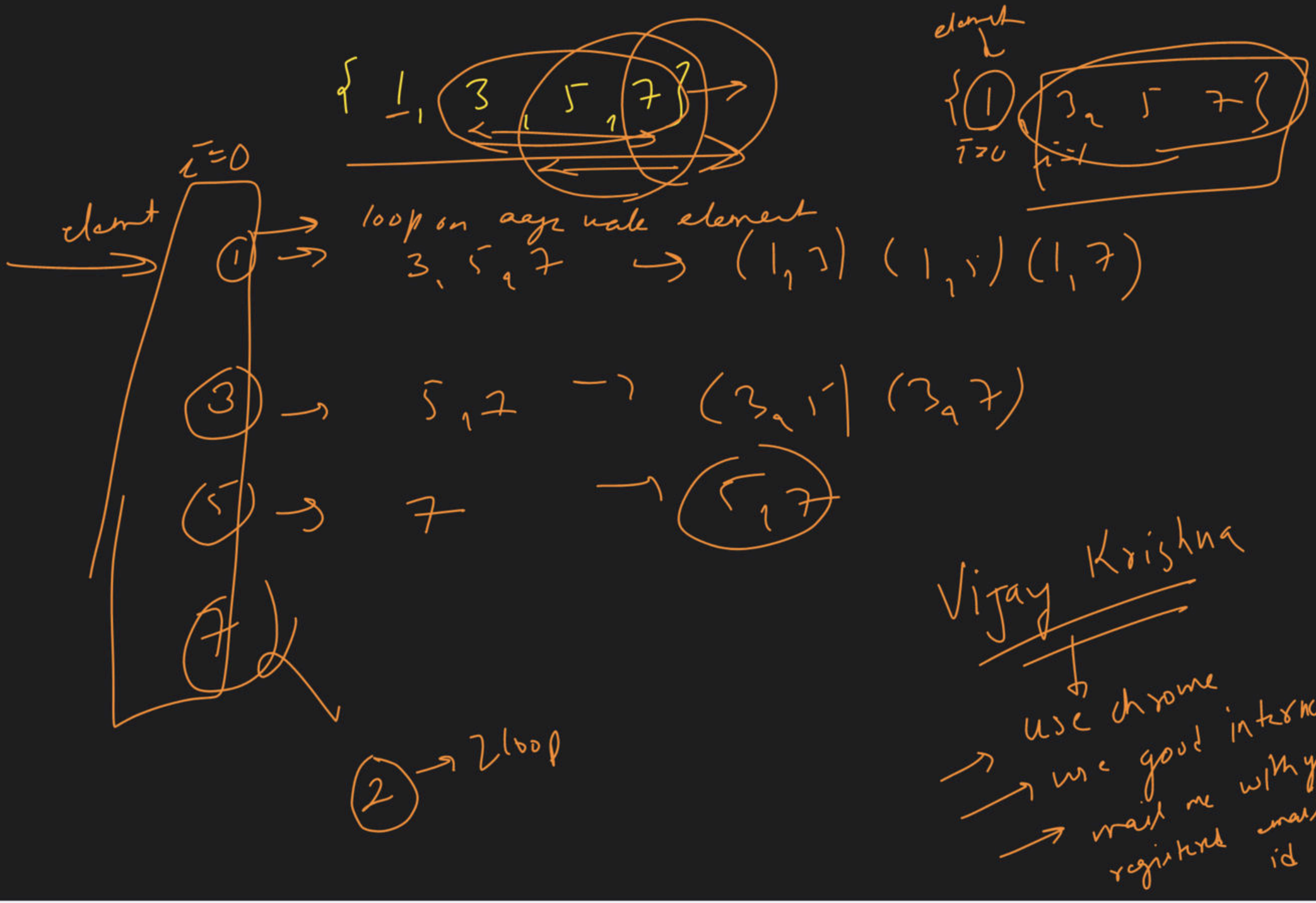
find all pair

(1,3) (3,5) (5,7) (7,7) (2,4) (4,6)
(1,5) (3,7) (5,1) (7,4) (2,6)
(1,7) (3,2) (5,4) (7,6)
(1,2) (3,4) (5,6)
(1,4) (3,6)
(1,6)

ans

any

O(n)



Vijay Krishna

→ use chrome
→ use good internet
→ mail me with your registered email id

arr[] → { 10, 20, 30, 40 }

Sum = 10

element = 10

inner loop (j+1 → arr.length)

20 30 40 50

30 40 50

arr found

Inner For Loop

20

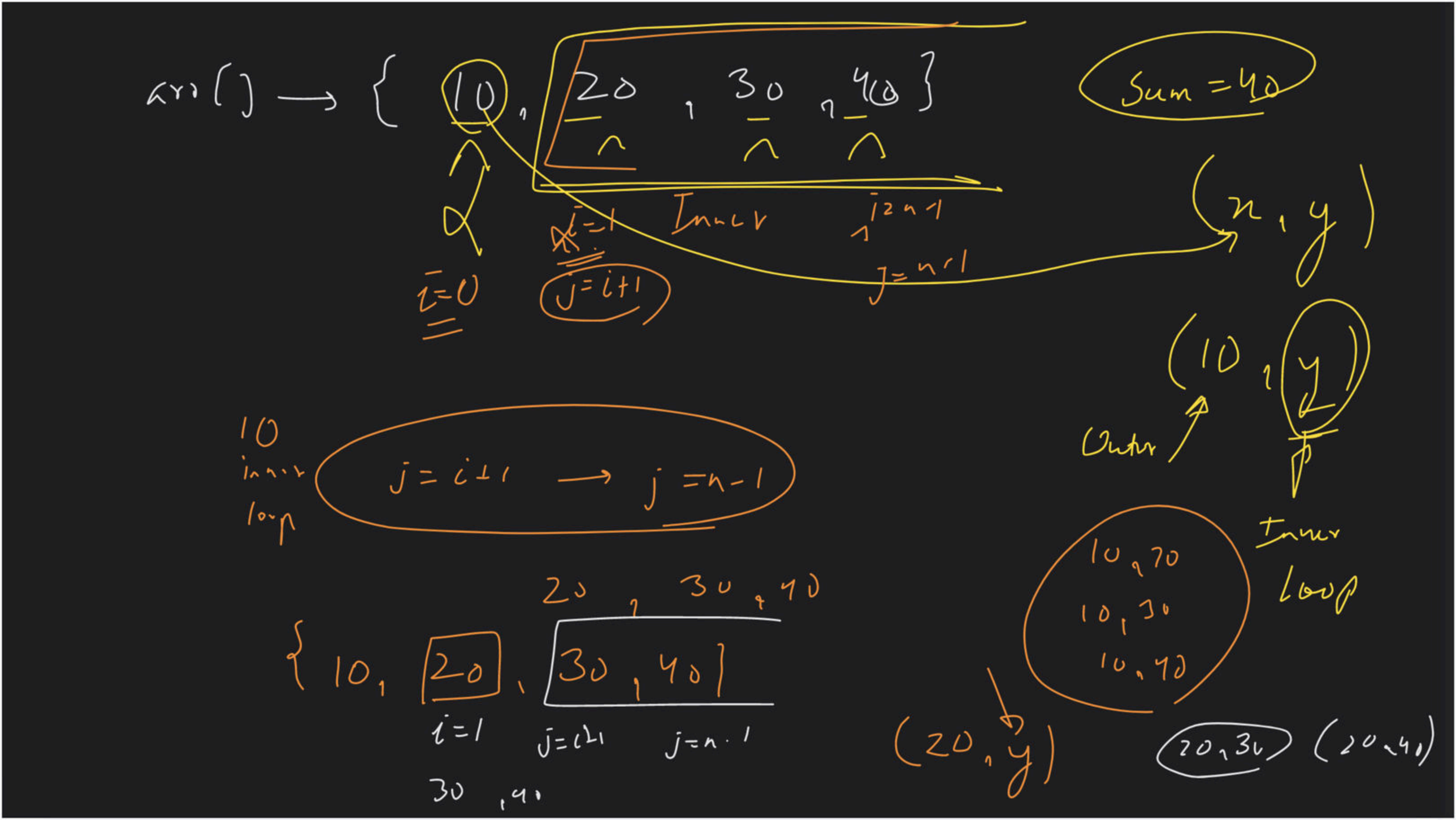
30 40

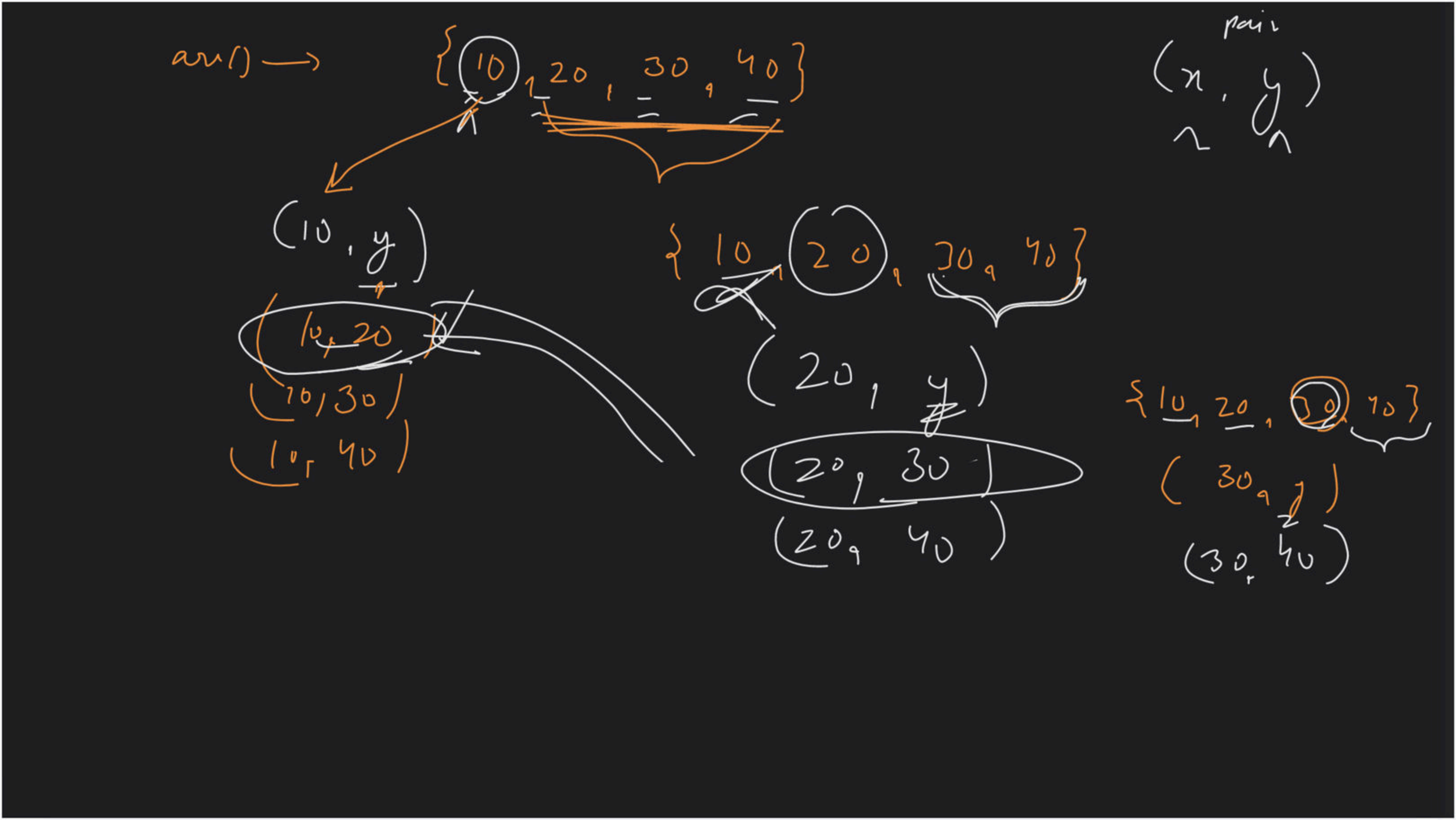
50 60

30

40

70


arr[] → { 10 , 20 , 30 , 40 }
Sum = 40
i=0
i=1
j=i+1
Inner
i≥n-1
j=n-1
(x , y)
(10 , y)
Outer
Inner Loop
10 inner loop
j = i+1 → j = n-1
20 , 30 , 40
{ 10 , 20 , 30 , 40 }
i=1
j=i+1
j=n-1
30 , 40
10 , 20
10 , 30
10 , 40
(20 , y)
20 , 30
(20 , 40)


arr() →
{10, 20, 30, 40}
pair
(x, y)
(10, y)
(10, 20)
(10, 30)
(10, 40)
{10, 20, 30, 40}
(20, y)
(20, 30)
(20, 40)
{10, 20, 30, 40}
(30, y)
(30, 40)

a[] → {10, 20, 30, 40}

{10, 20, 30} {10, 30, 40}

{10, 20, 40} → 20 ✗

{20, 30, 40}

bigger

(10, 30, 40) → sum = 80

arr

triplet

3 number

(x, y, z)

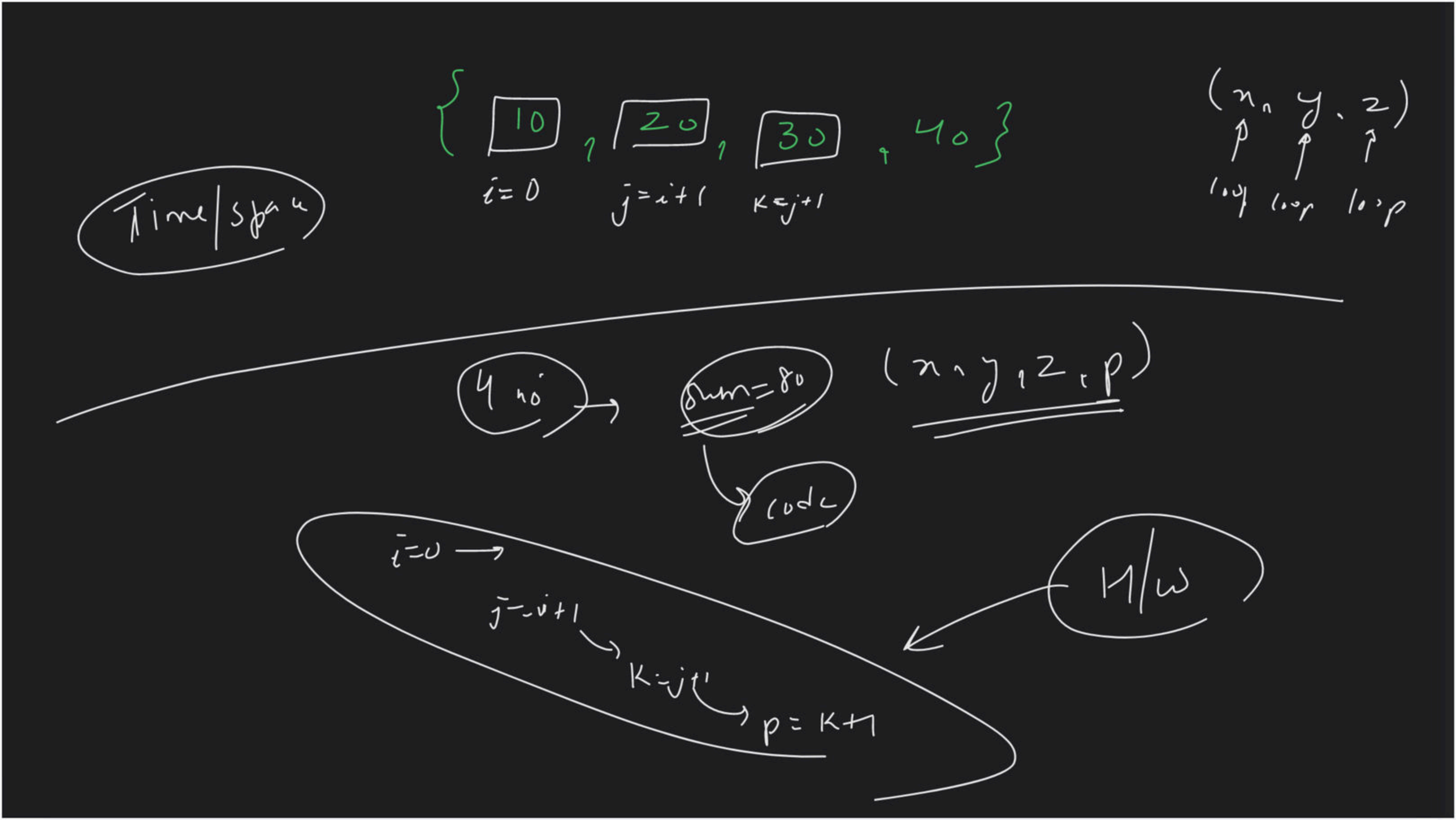
sum = 80

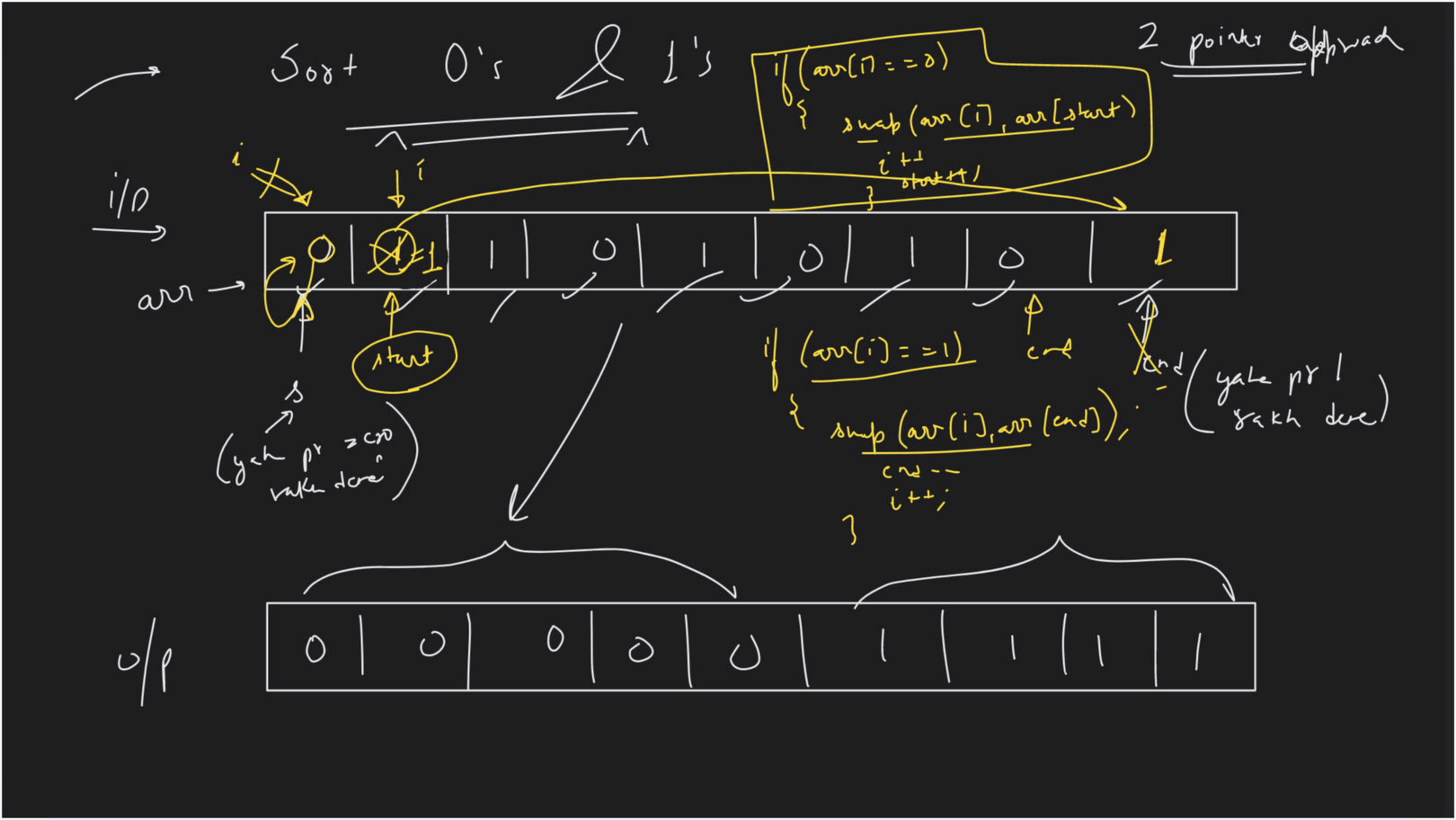
find a triplet that upon addition give value equal to sum

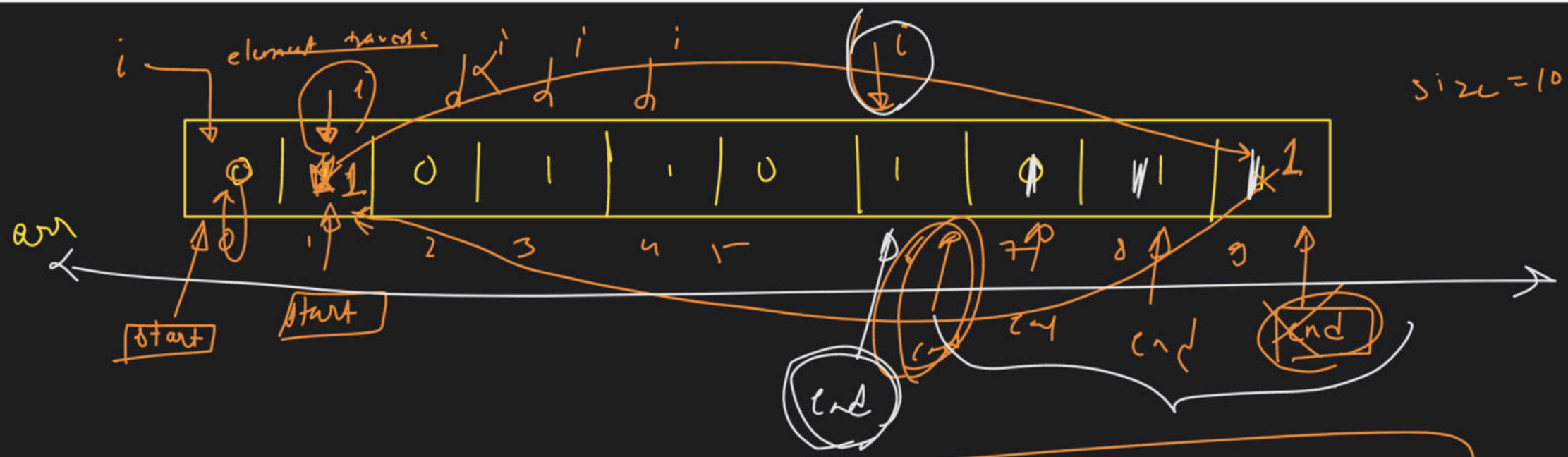
doublet/pair

triplet

3 loop


{ 10 , 20 , 30 , 40 }
i=0
j=i+1
K=j+1
(x, y, z)
p
loop loop loop
Time/space
4 no
sum=80
(x, y, z, p)
code
i=0 →
j=i+1
K=j+1
p=K+1
H/W

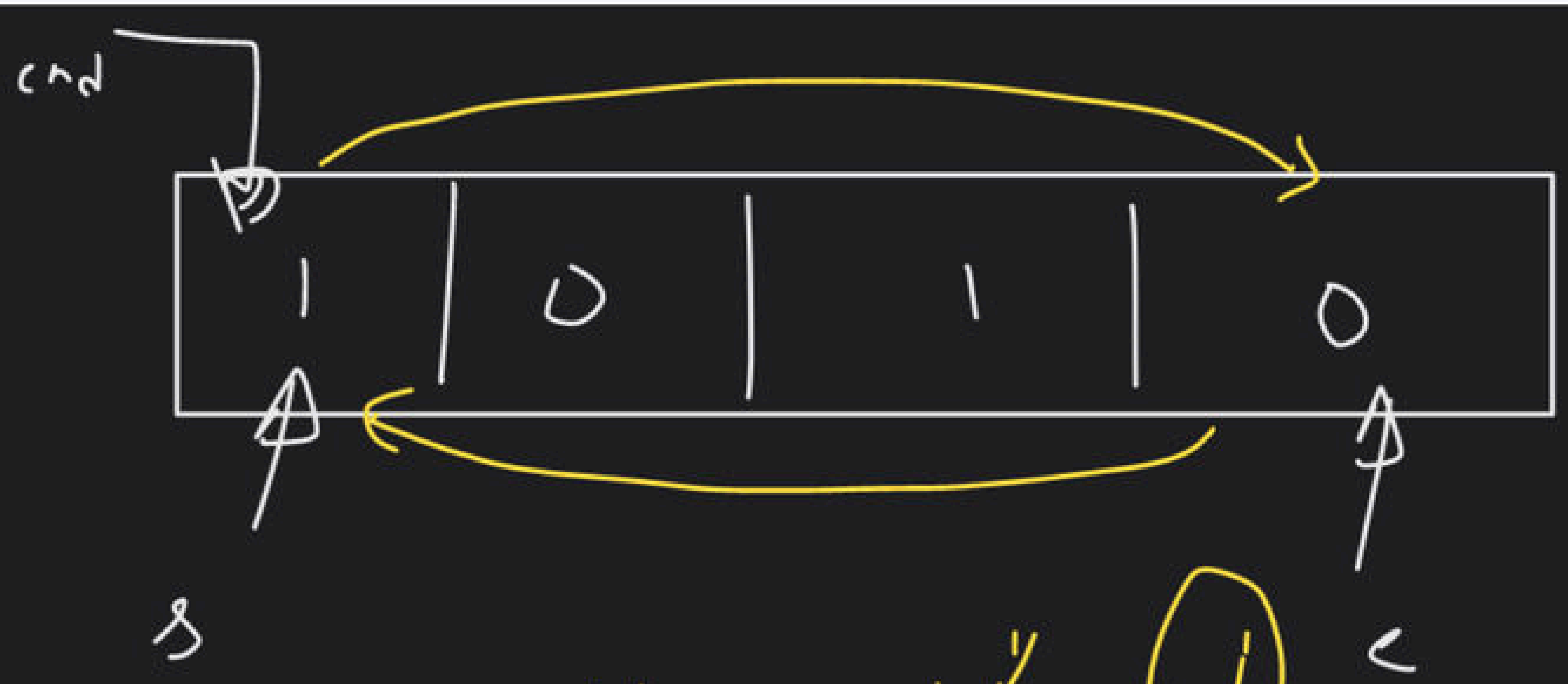

Sort 0's & 1's
2 pointer approach
i/p
arr
i
start
end
if (arr[i]==0)
{
swap(arr[i], arr[start])
i++
start++;
}
if (arr[i]==1)
{
swap(arr[i], arr[end]);
end--
i++;
}
(yaha pr = 0 wala done)
(yaha pr 1 wala done)
o/p
0 0 0 0 0 1 1 1 1




```
if (arr[i] == 0)
{
    swap (arr[start], arr[i]);
    i++;
    start++;
}
```

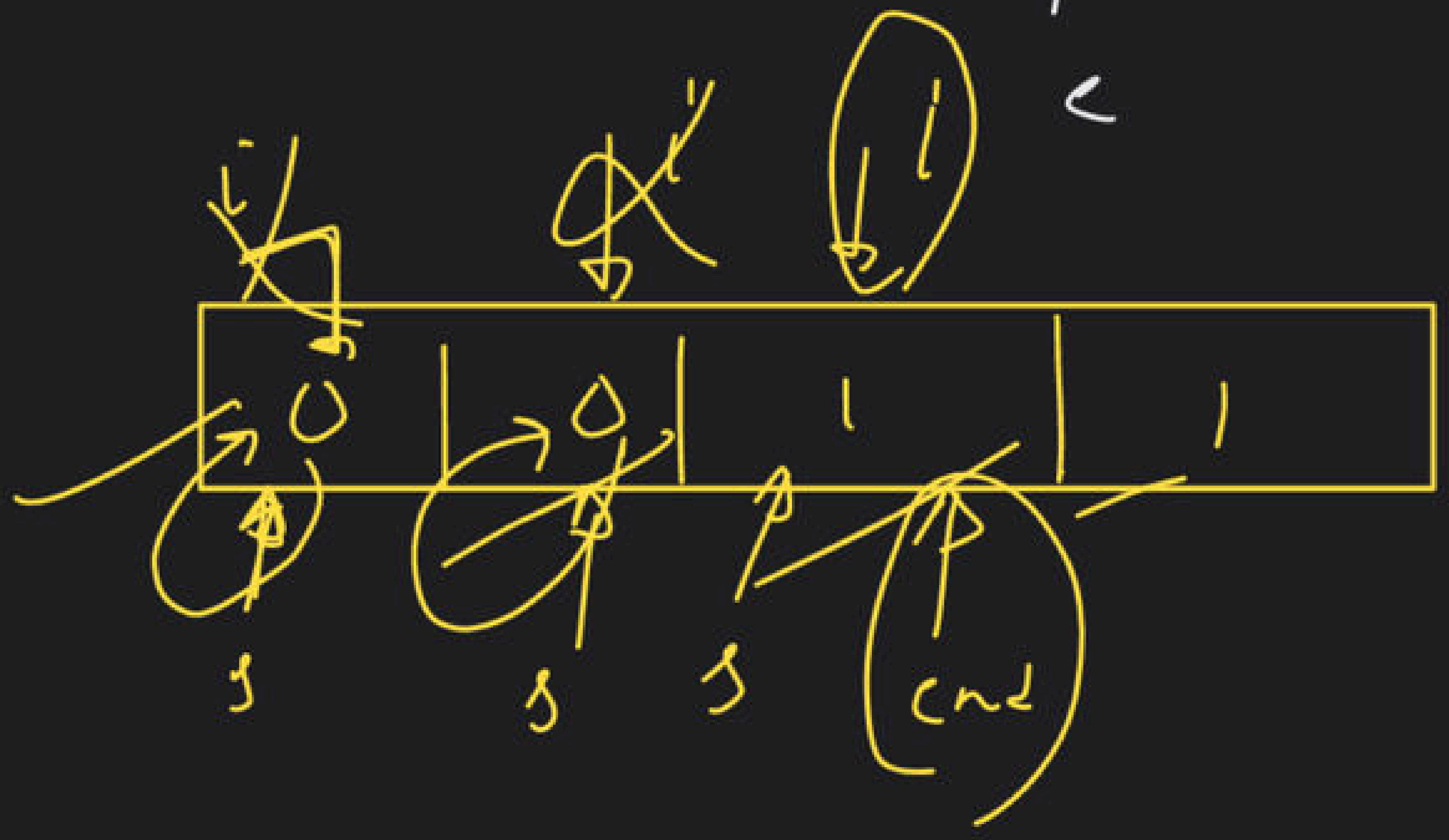
```
if (arr[i] == 1)
{
    swap (arr[end], arr[i]);
    end--;
}
```

i != end → loop chala rkte hn



i = 0

arr(i) = arr[0] → 1



i == end

arr[i] = arr[0] = 0

i++

s++

arr[i] = arr[1] = 0

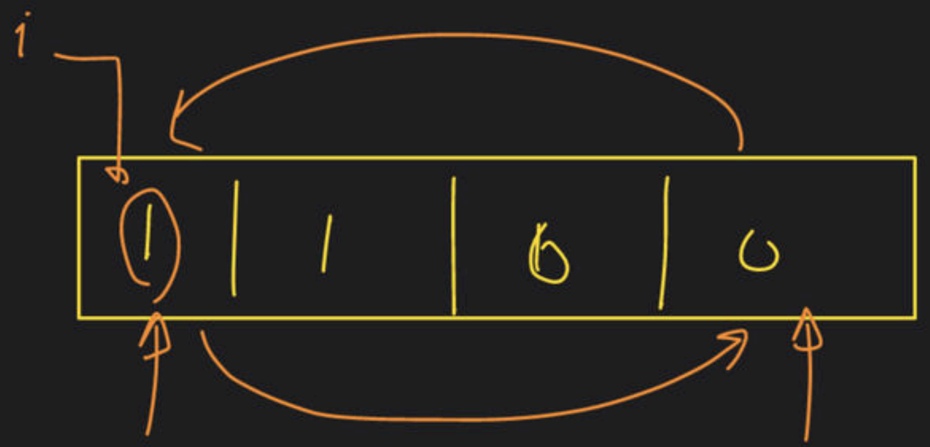
swap(arr[i], arr[s])

i++

s++

arr sort ho chuka h

break



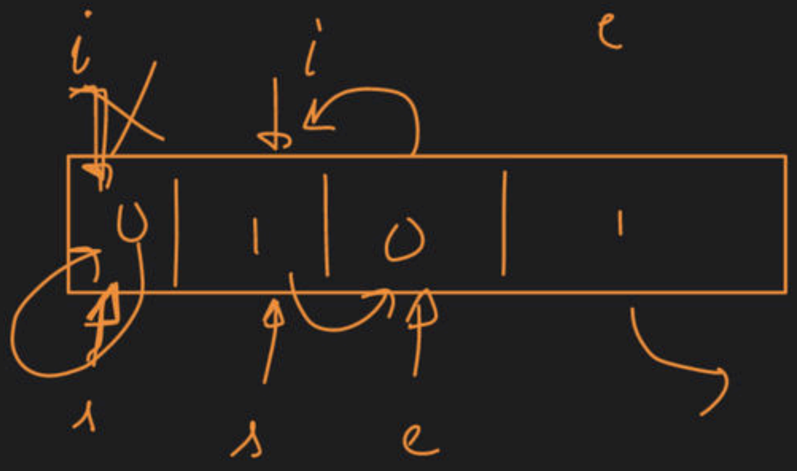
arr[i] = 1

arr[i] = 0

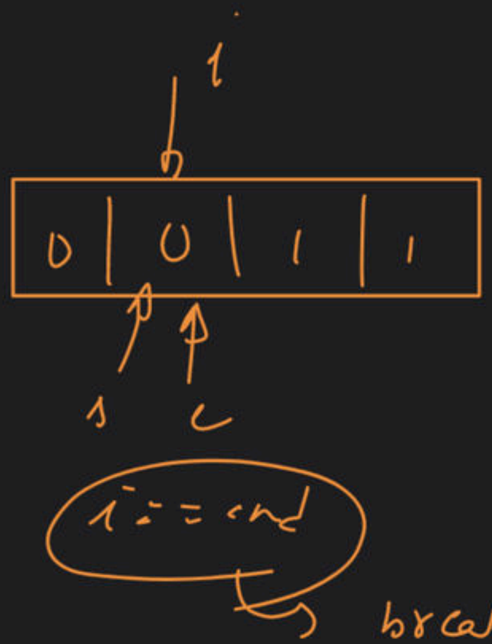
arr[i] arr[i

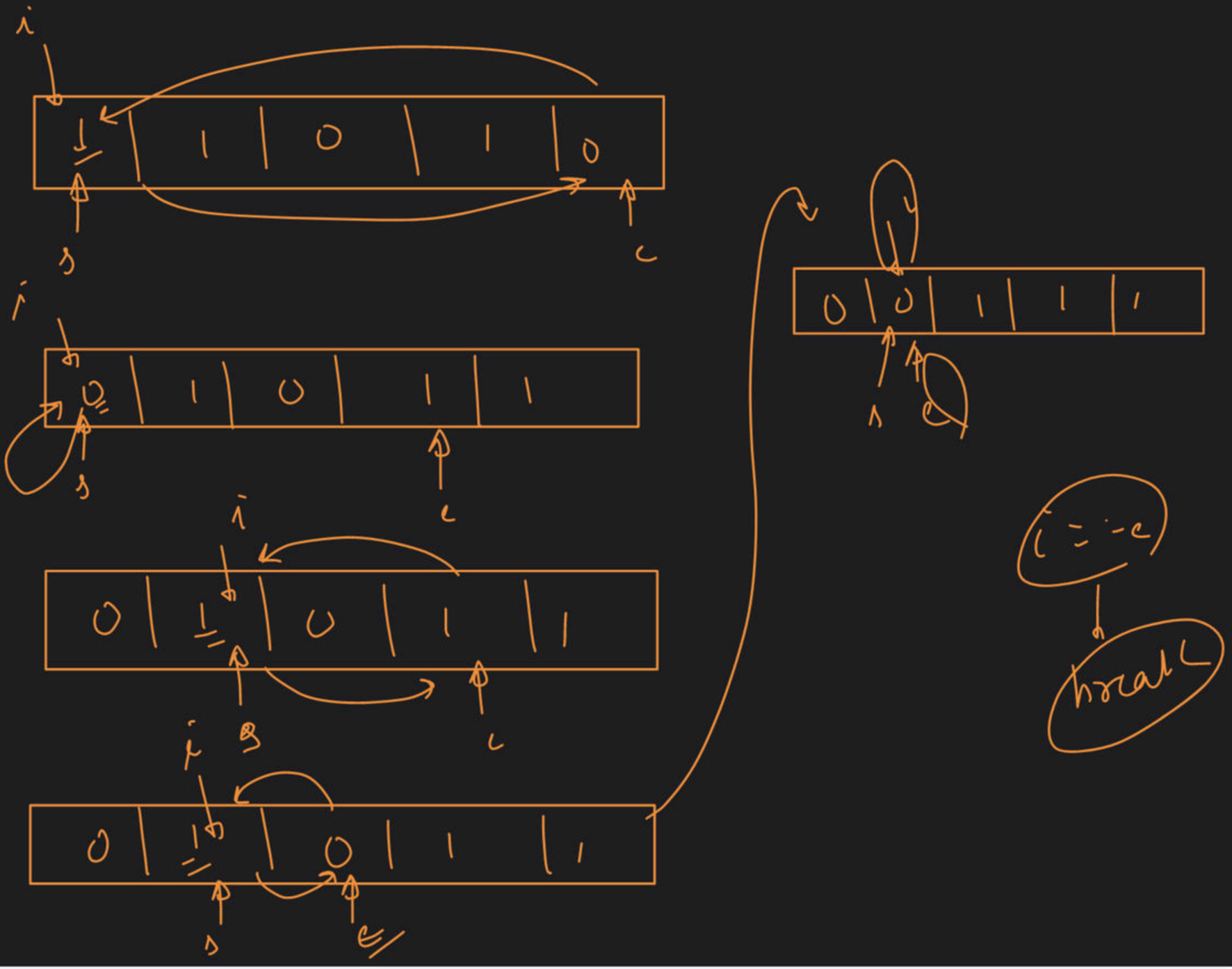
i++

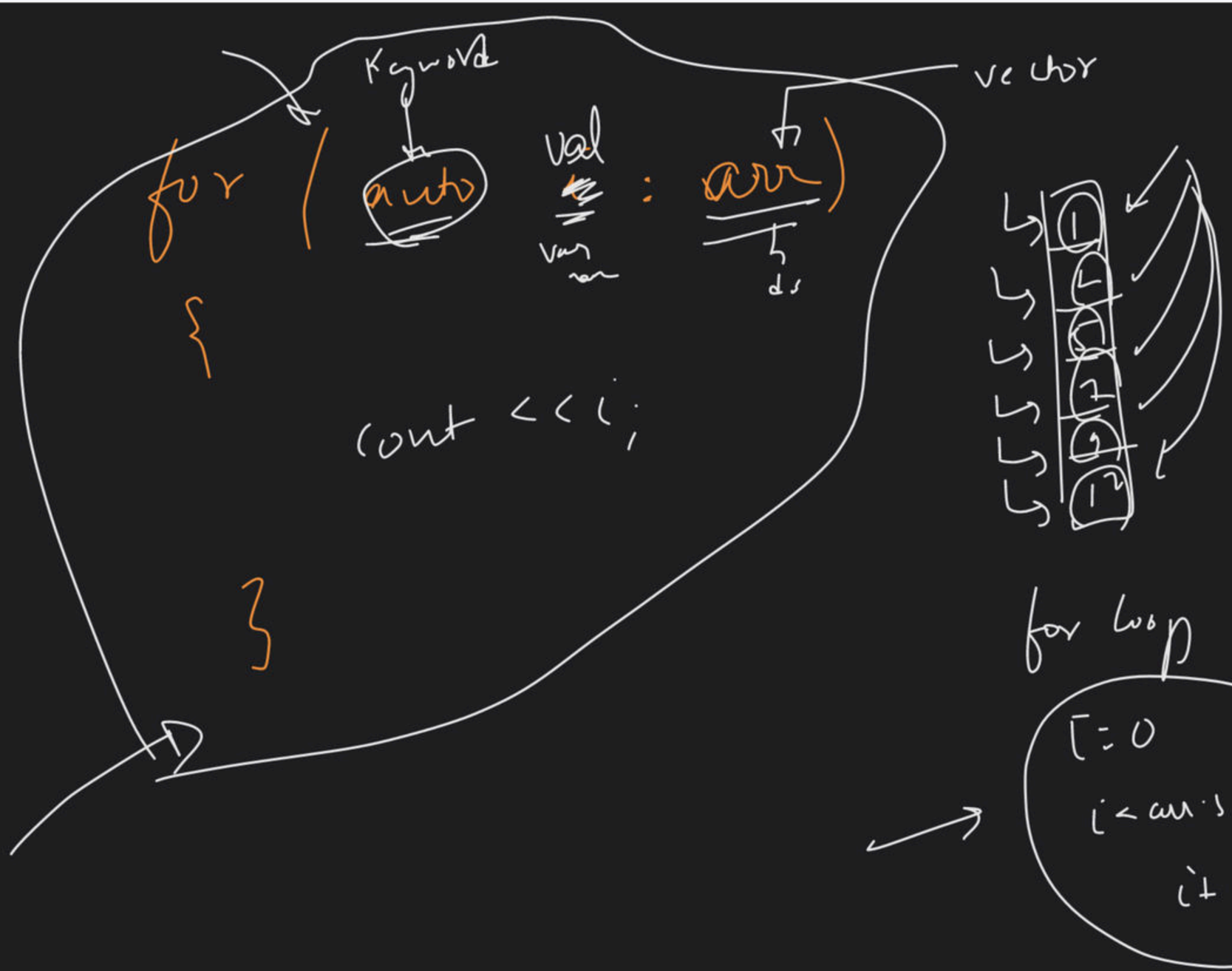
s++

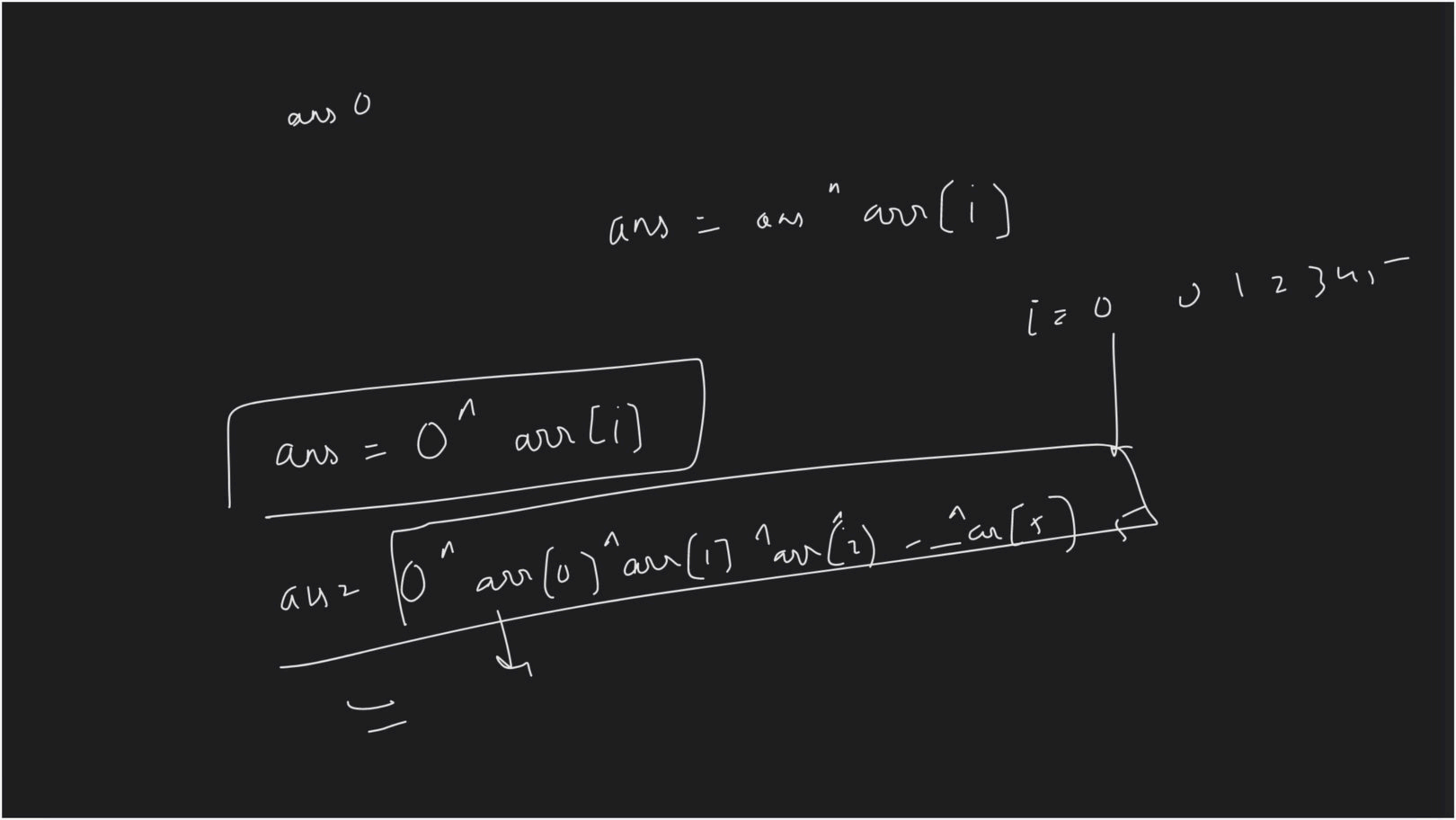


arr[i] = 1




i
1
0
1
0
j
c
i
0
1
0
1
1
j
c
i
0
1
0
1
1
j
c
i
0
1
0
1
1
j
c
0
0
1
1
1
i
c
hrall

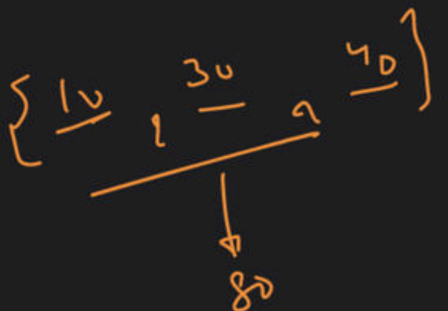

Keyword
vector
for (auto val : arr)
{
cout << i;
}
for loop
i = 0
i < arr.size()
i++

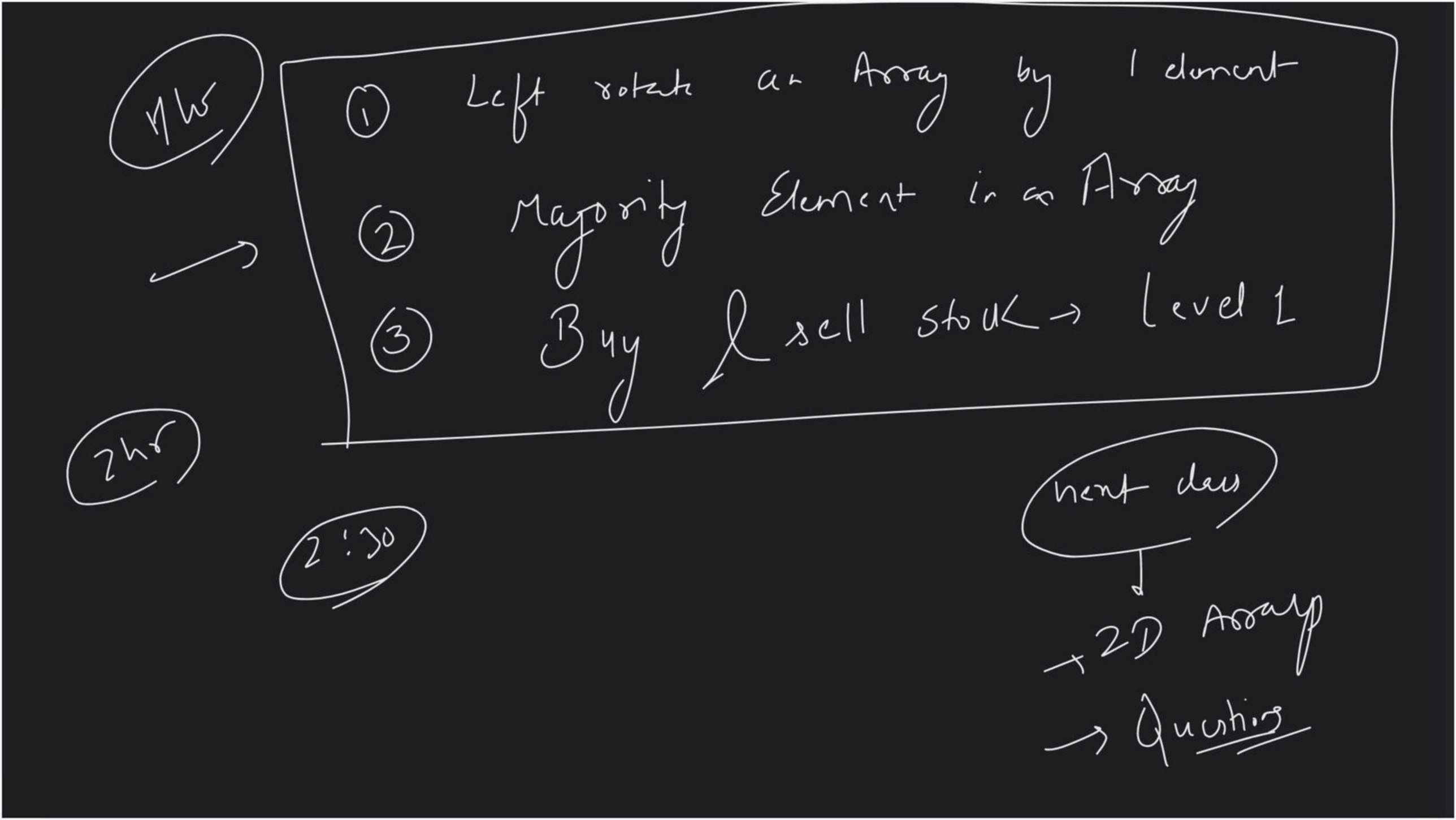

ans 0
ans = ans ^ arr[i]
i = 0 0 1 2 3 4 ...
ans = 0 ^ arr[i]
ans = 0 ^ arr[0] ^ arr[1] ^ arr[2] ^ ... ^ arr[i]

Triplet sum

int a[] → {10, 20, 30, 40, 70}

Sum = 80




HW
1) Left rotate an Array by 1 element
2) Majority Element in an Array
3) Buy & sell stock → level 1
2hr
2:30
next class
→ 2D Arrays
→ Questions