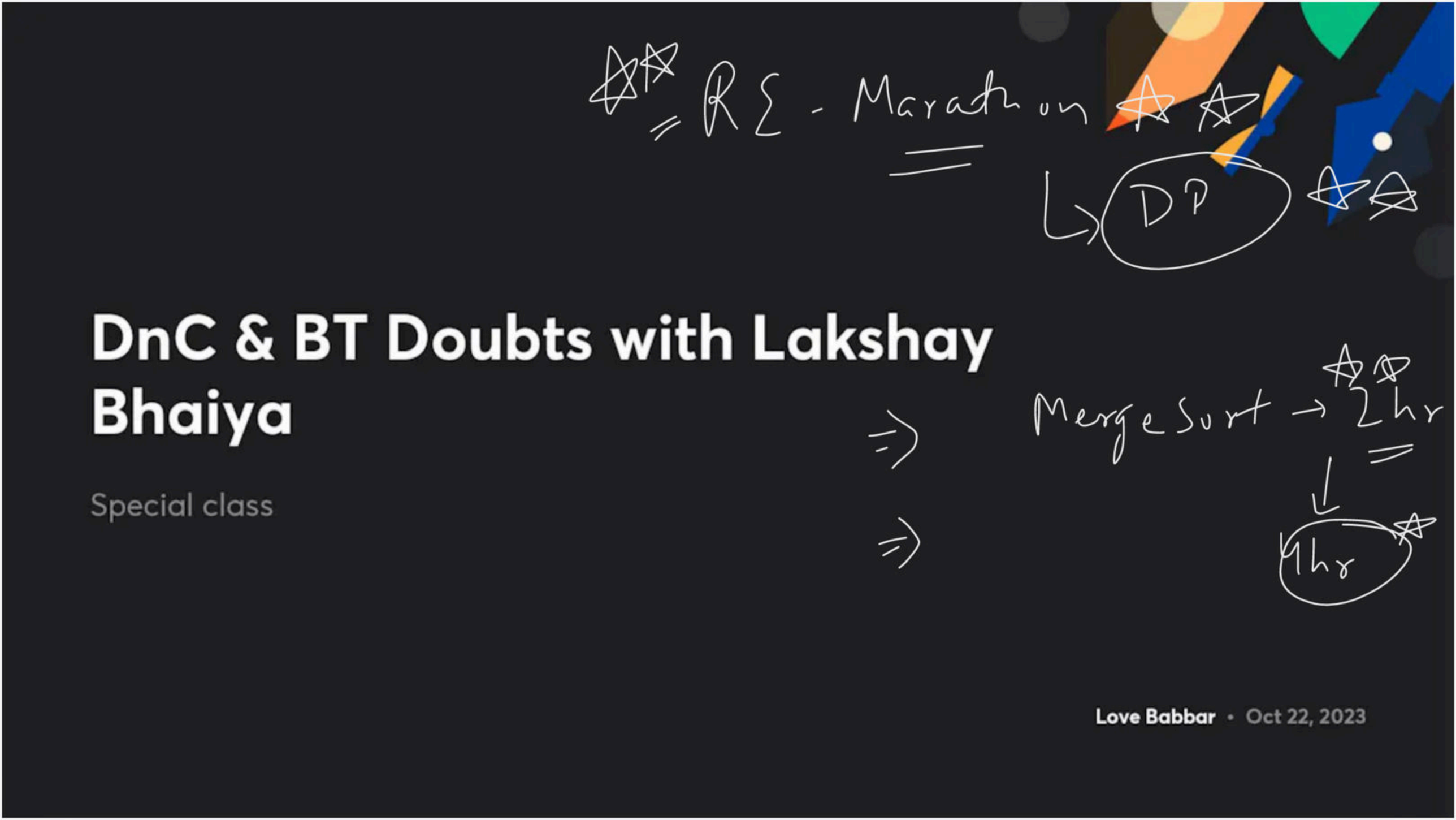


DnC & BT Doubts with Lakshay Bhaiya

Special class

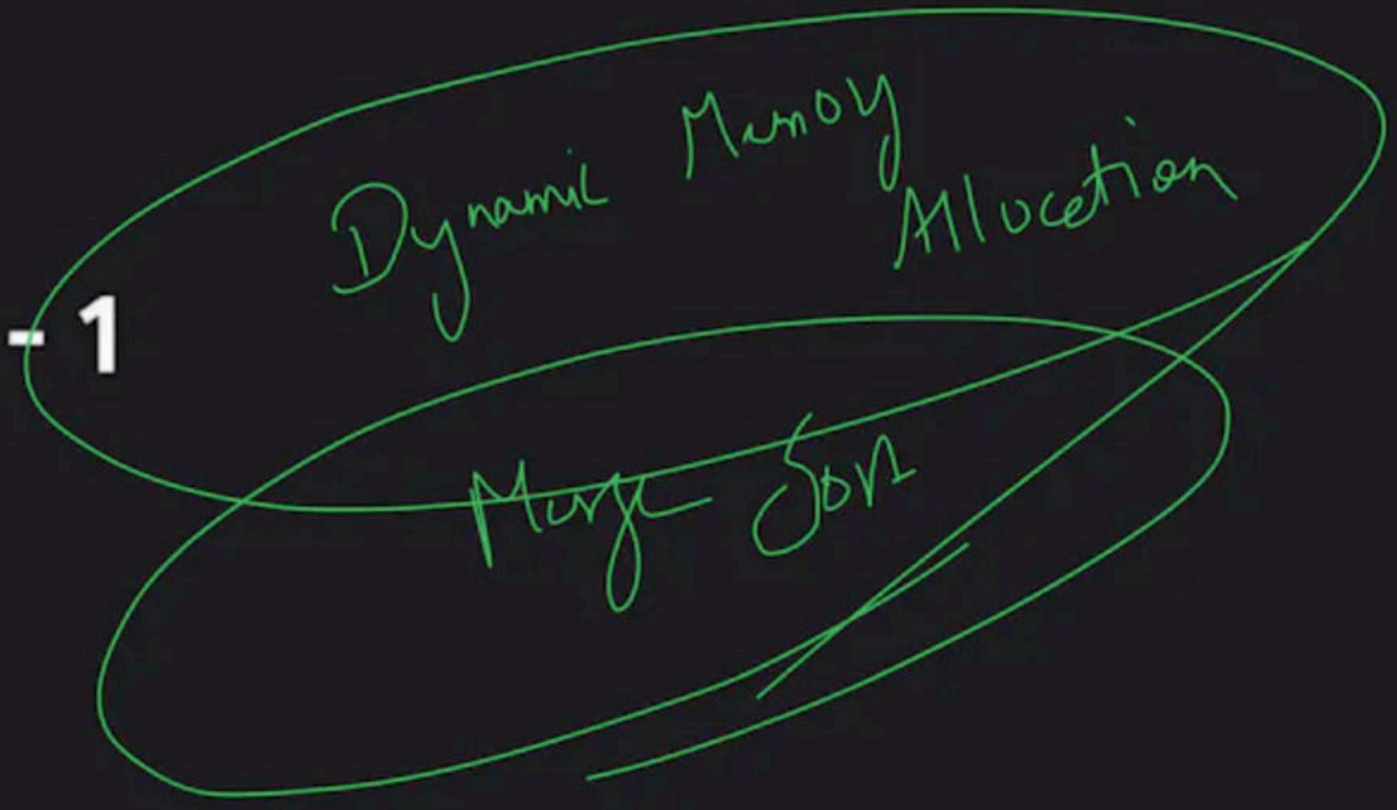
Love Babbar • Oct 22, 2023

1. Last classes Doubts ☆☆
2. Debugging in C++ ☆☆
3. Interview Question on Dynamic mem. alloc

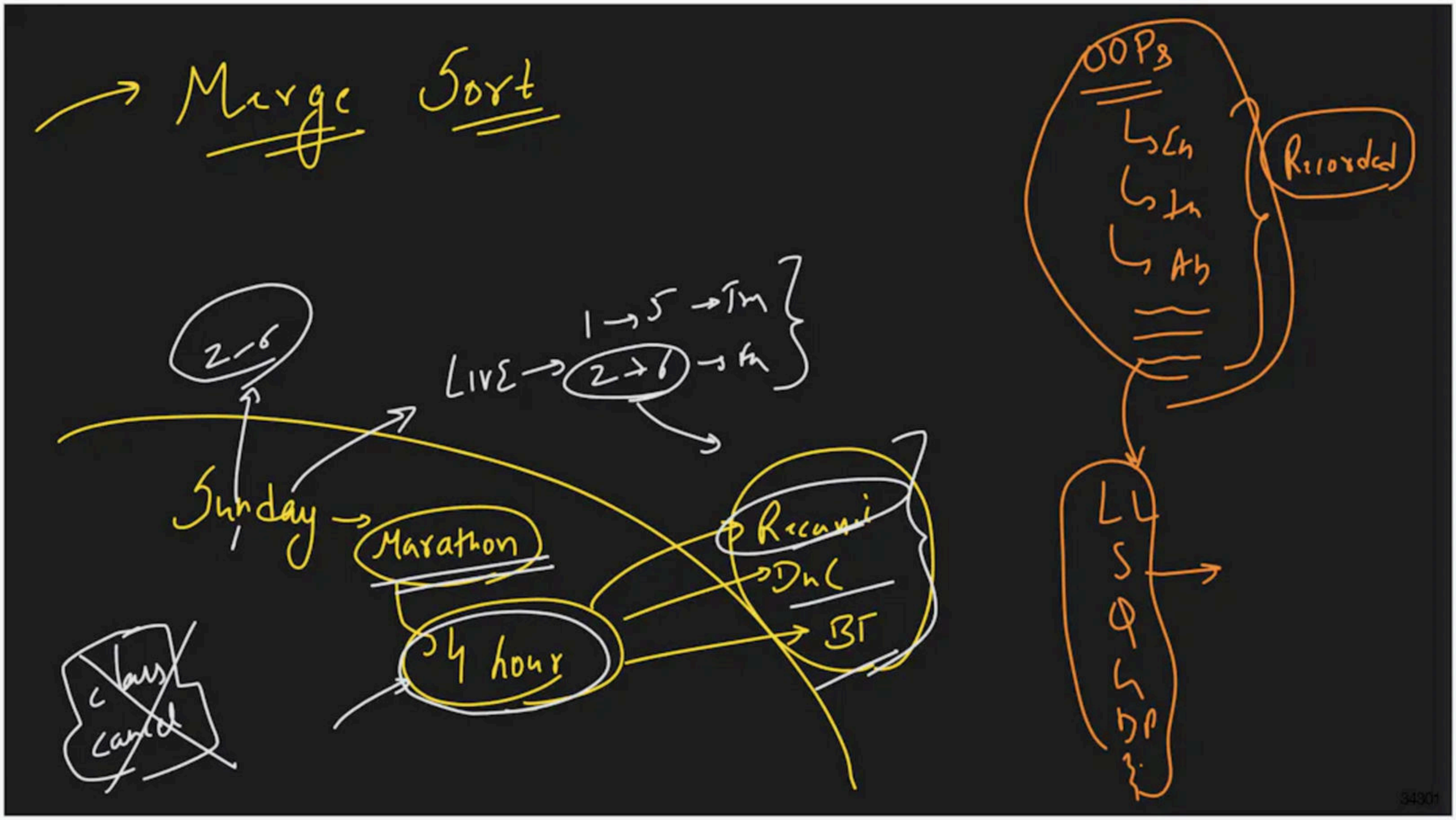
13+ ✗ O Σ ☆/☆

DnC Class - 1

Special class



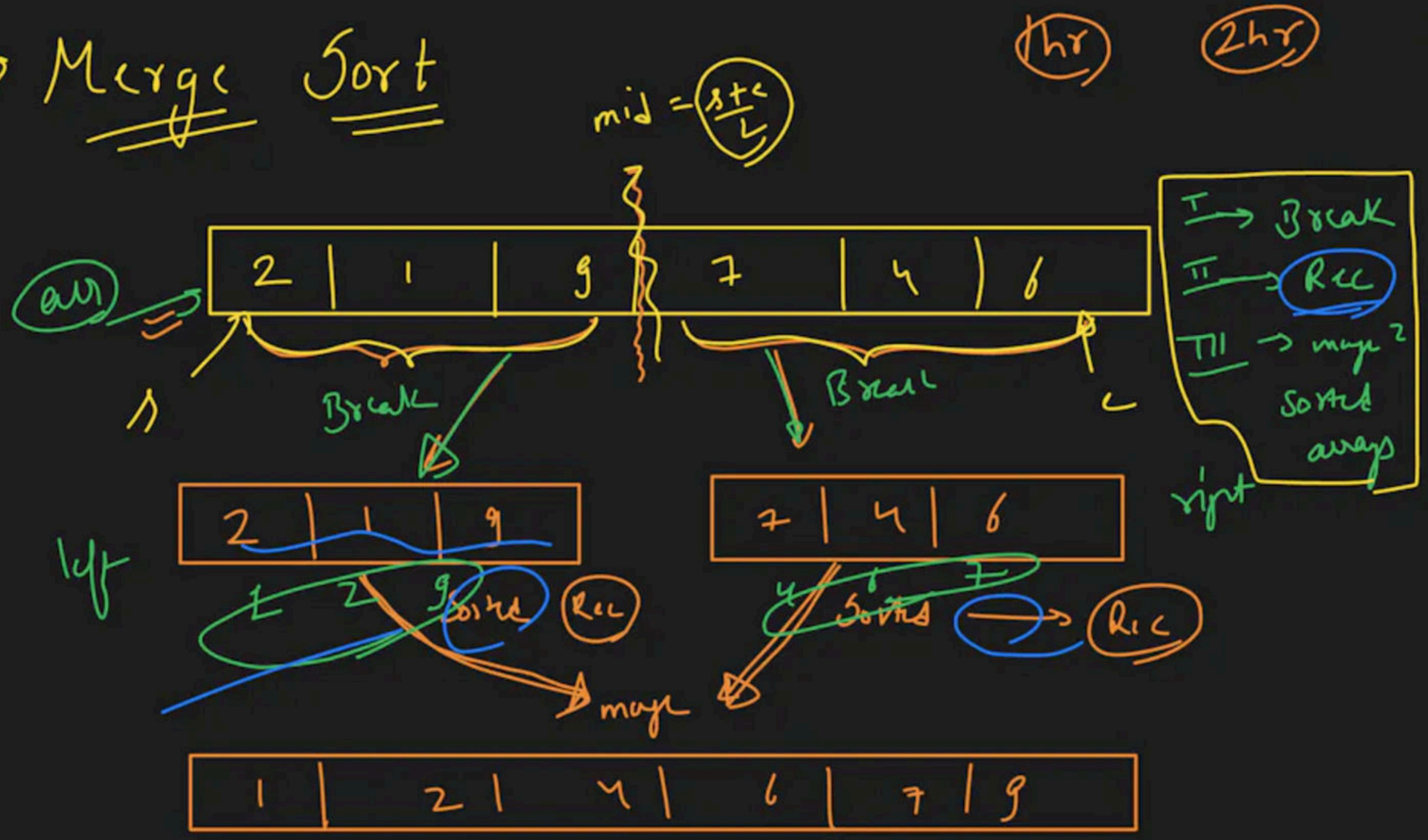
Love Babbar • Oct 18, 2023

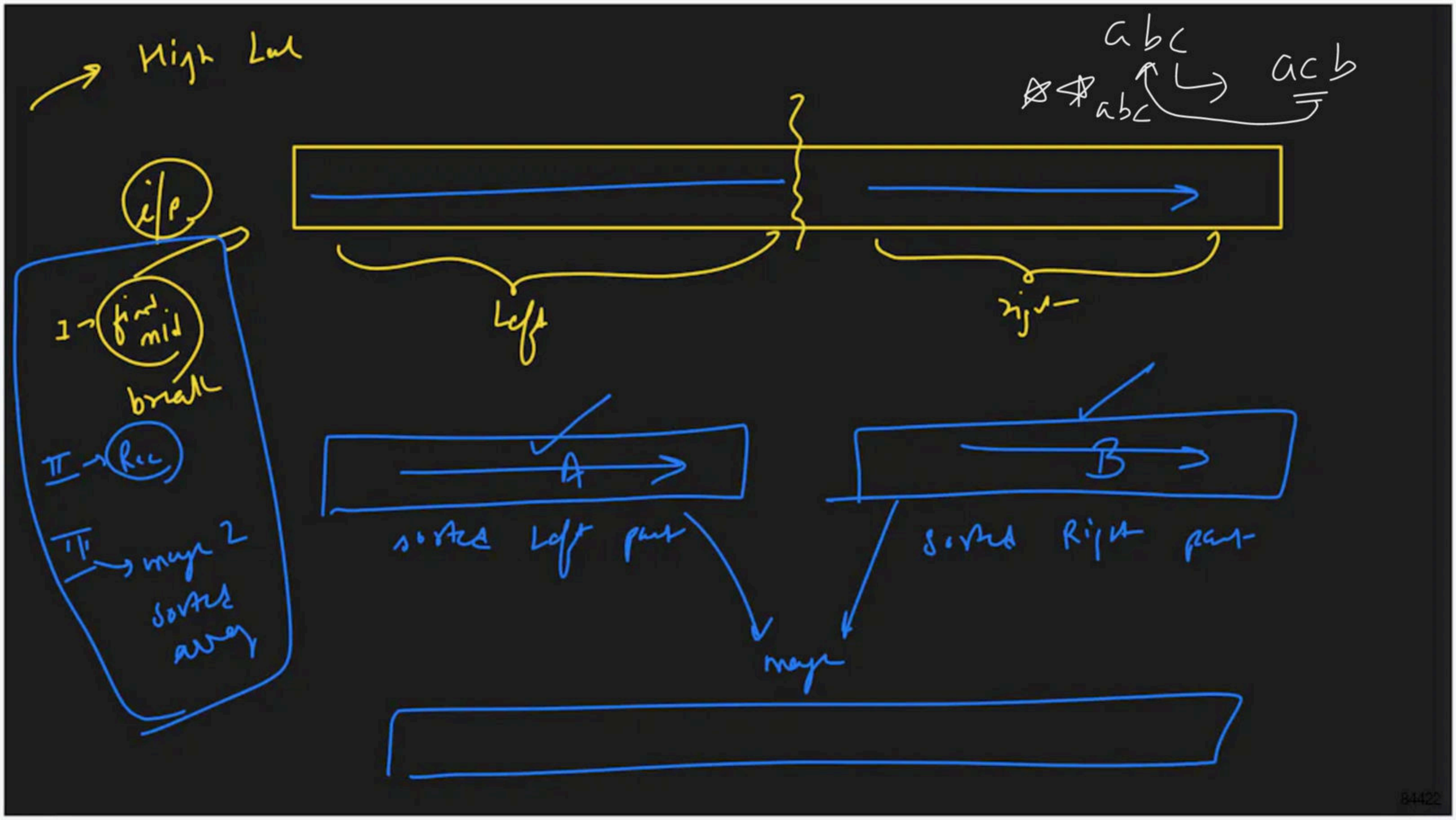

Merge Sort
2-6
Live
1→5→7m
2→6
→4m
Sunday
Marathon
4 hour
Recent
DnC
BT
OOPs
Recorded
LL
Stack

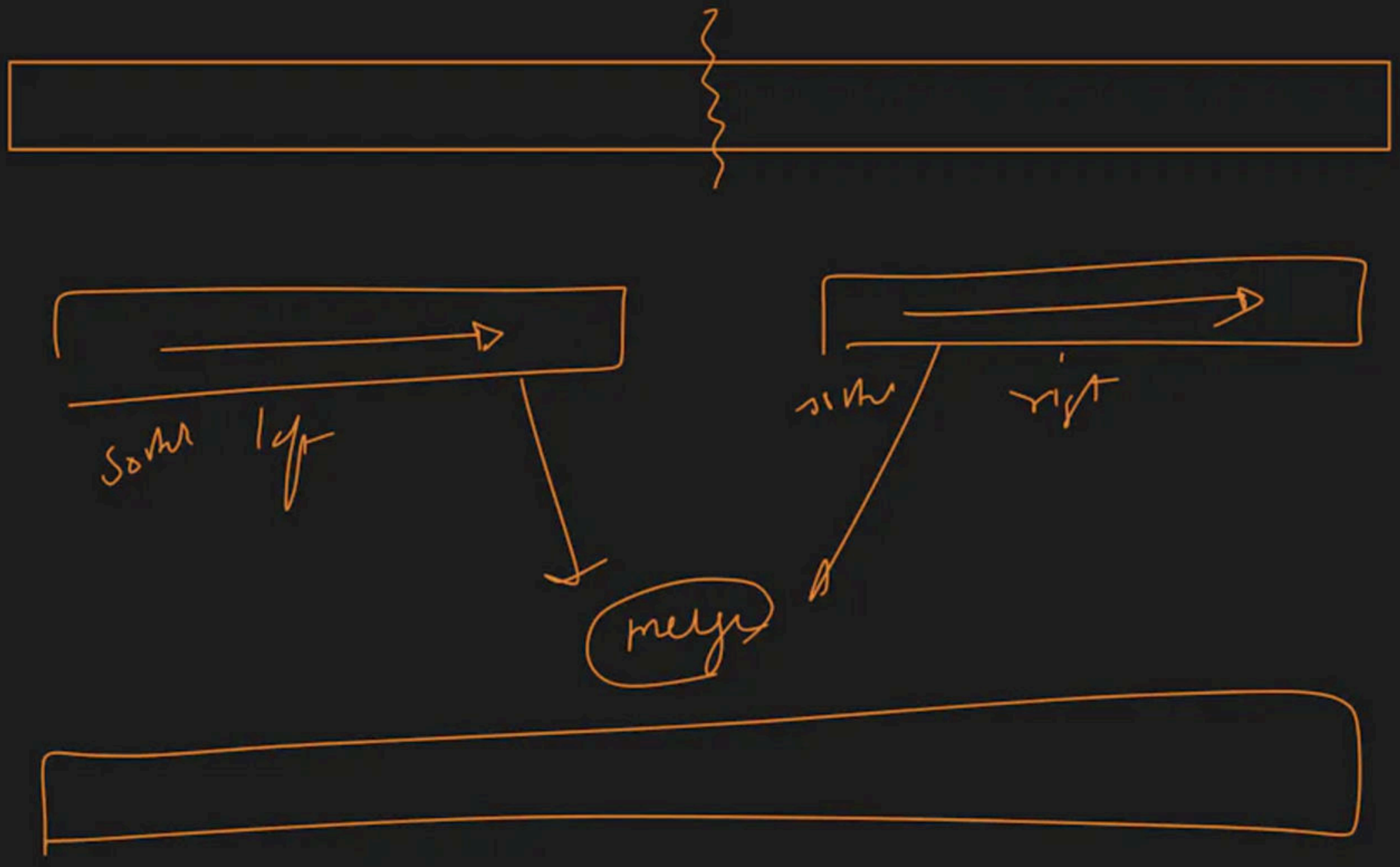

Merge Sort

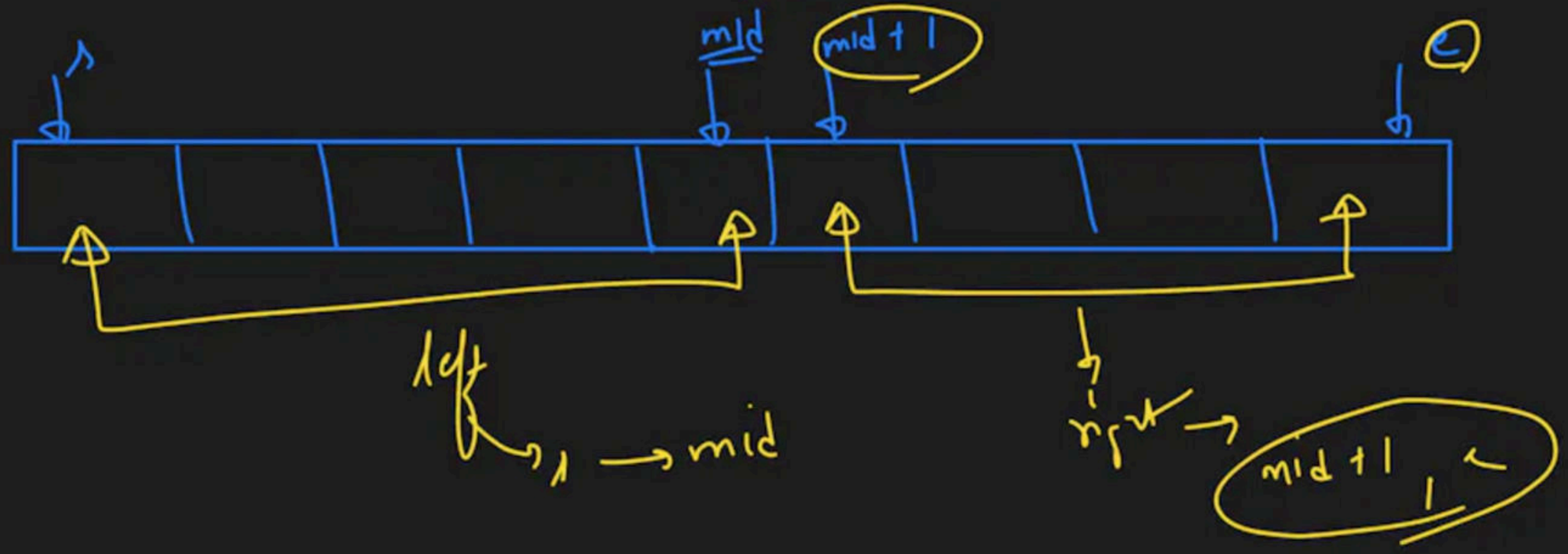
→ Merge Sort

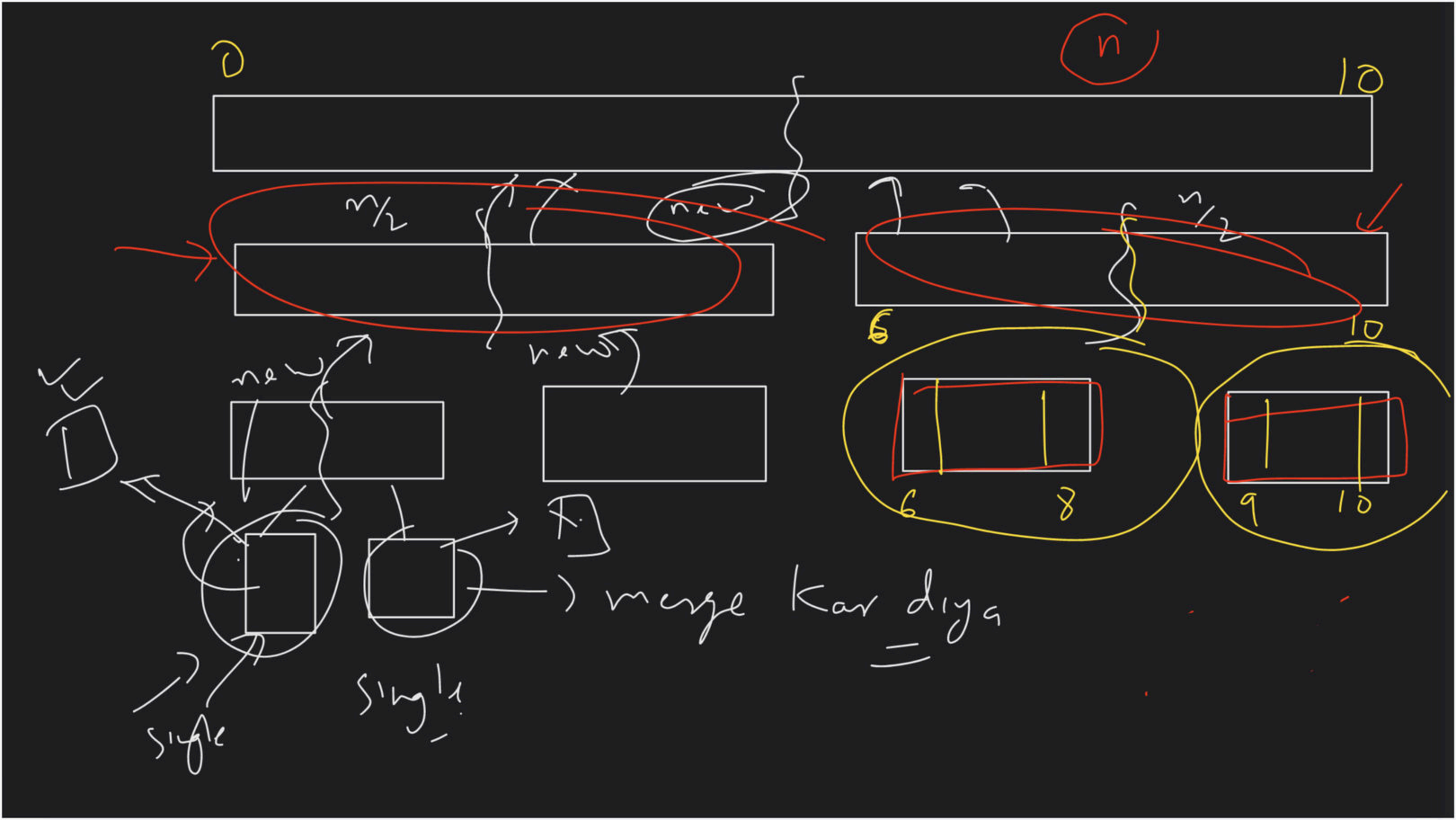
1hr 2hr

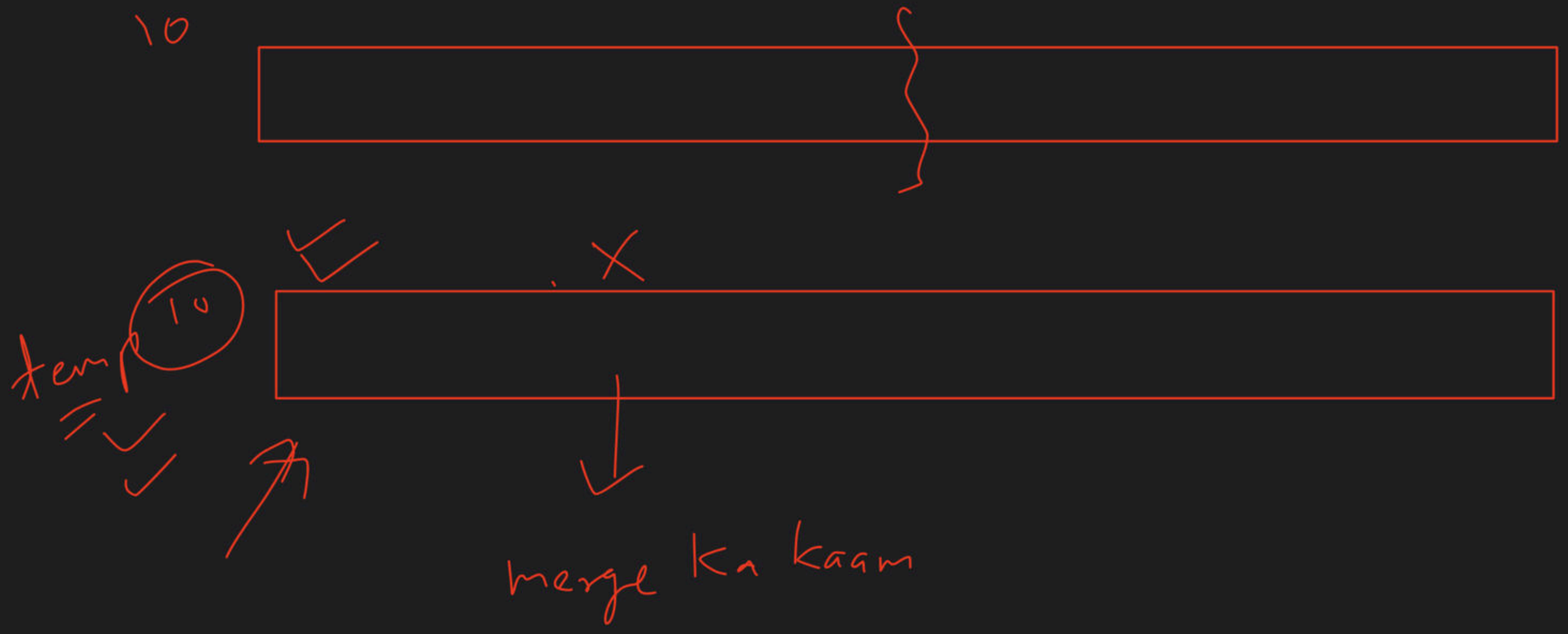


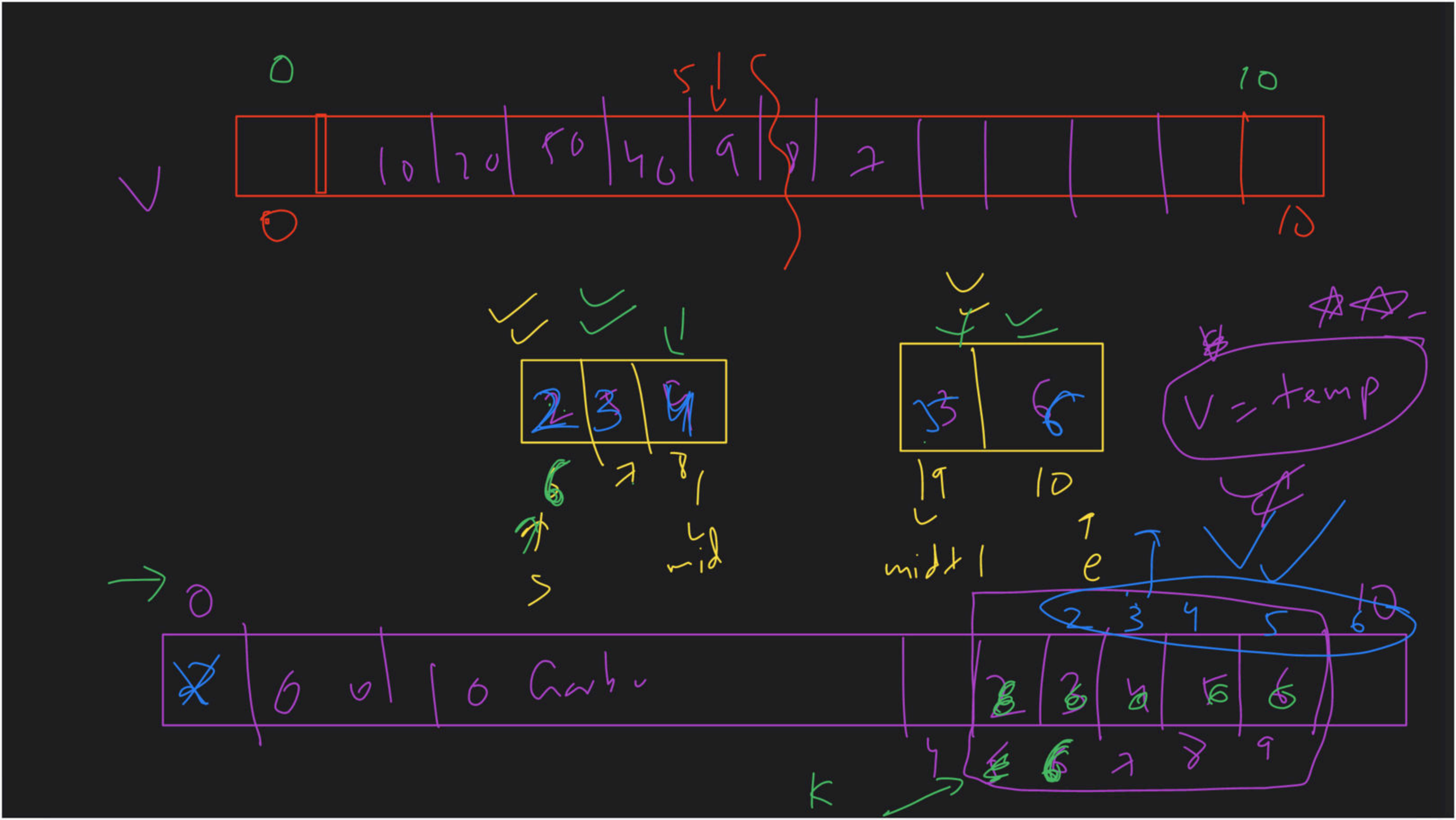

High Level
abc
abc
acb
i/p
find mid
broadcast
Left
Right
A
B
sorted Left part
sorted Right part
merge




mid
mid + 1
left
mid
right
mid + 1


0
n
10
n/2
n/2
5
10
6
8
9
10
single
single
merge kar diya

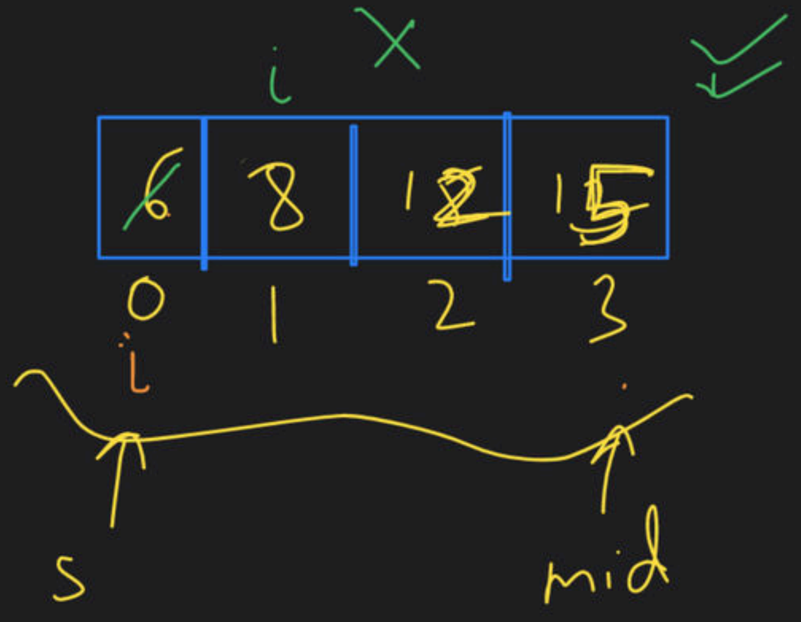

10
temp
10
merge ka kaam

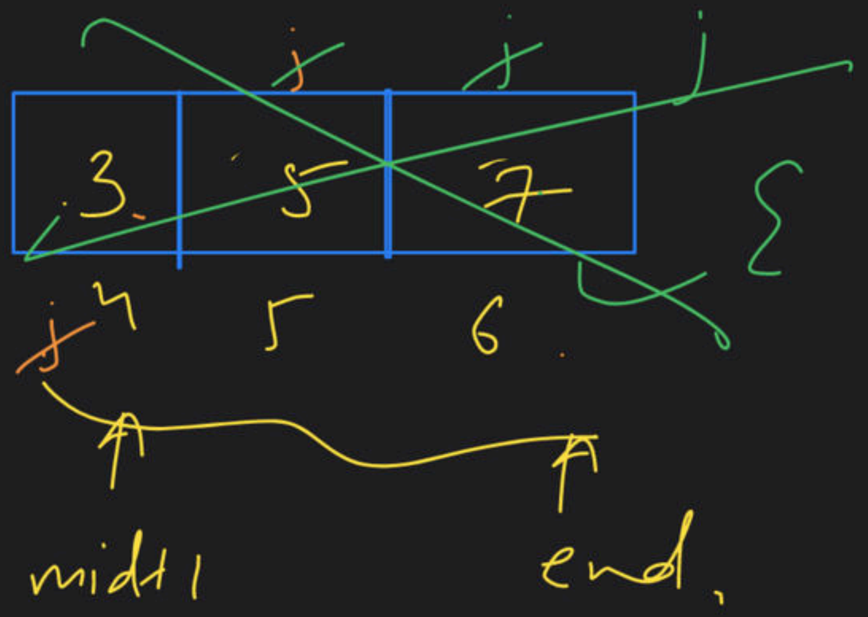

V = temp
mid
middle
e
s
k

2) In-place Merge Sort

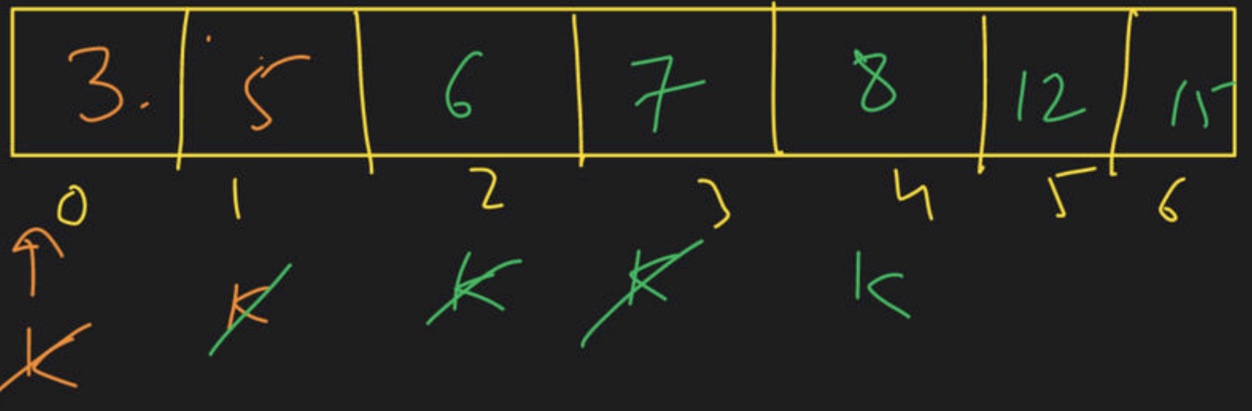
=> Merge Sort → 45

45 → 45 1hr





temp.



temp[k] = V[ ]

temp[0] = V[4]

→ break into left & right half

→ recursive

→ merge 2 sorted array

new

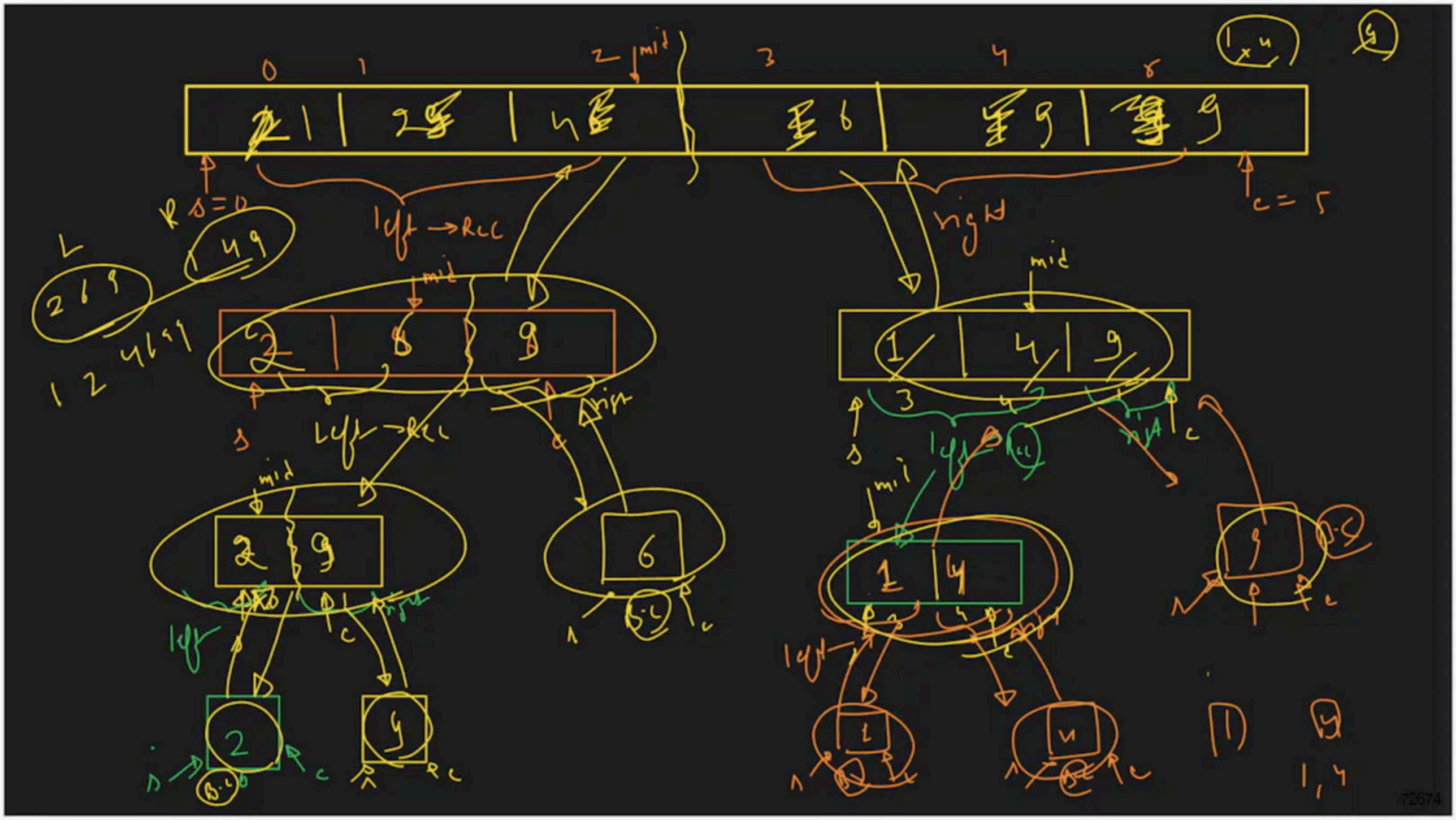
delete costly hote hai

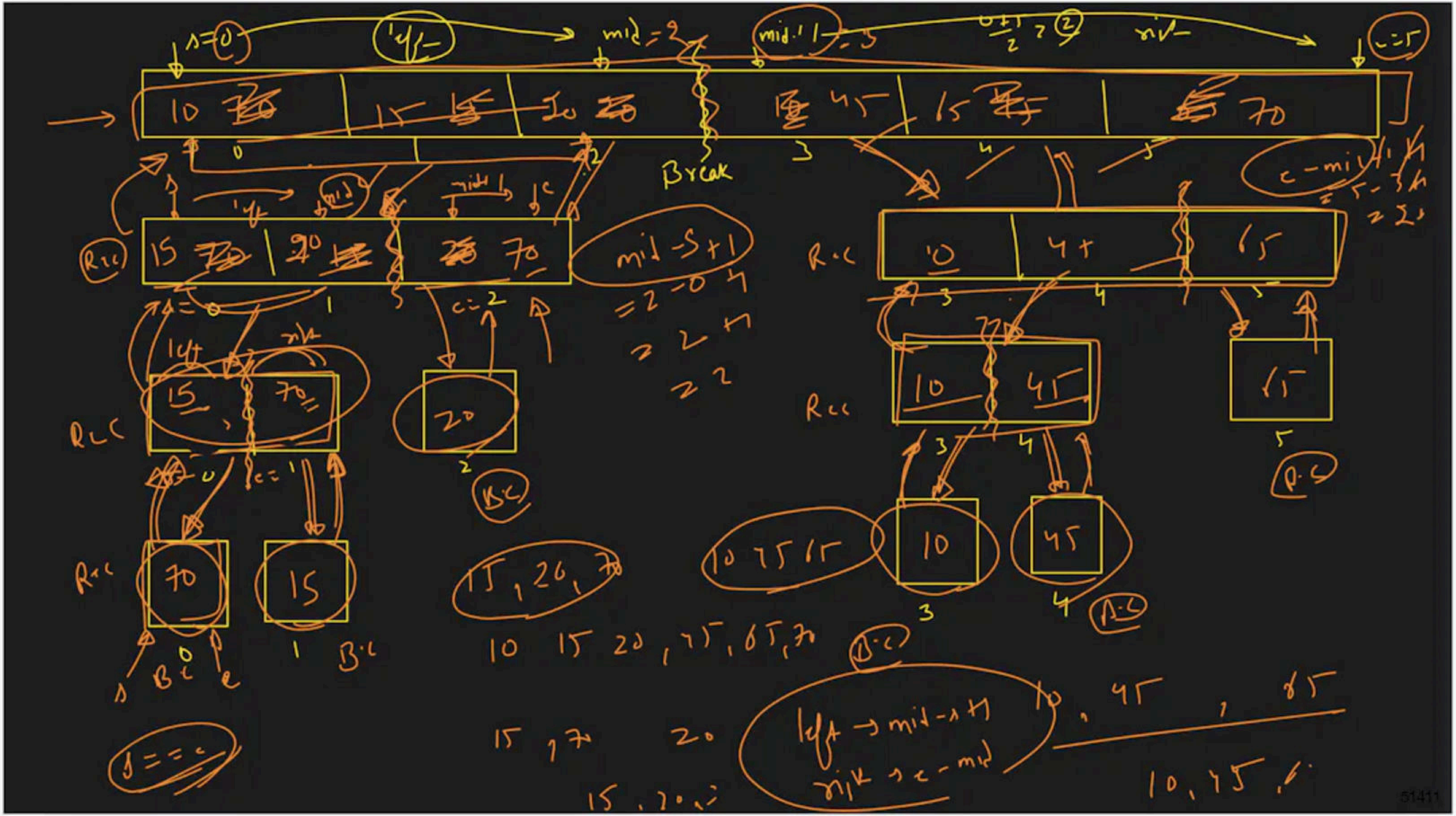
93.1

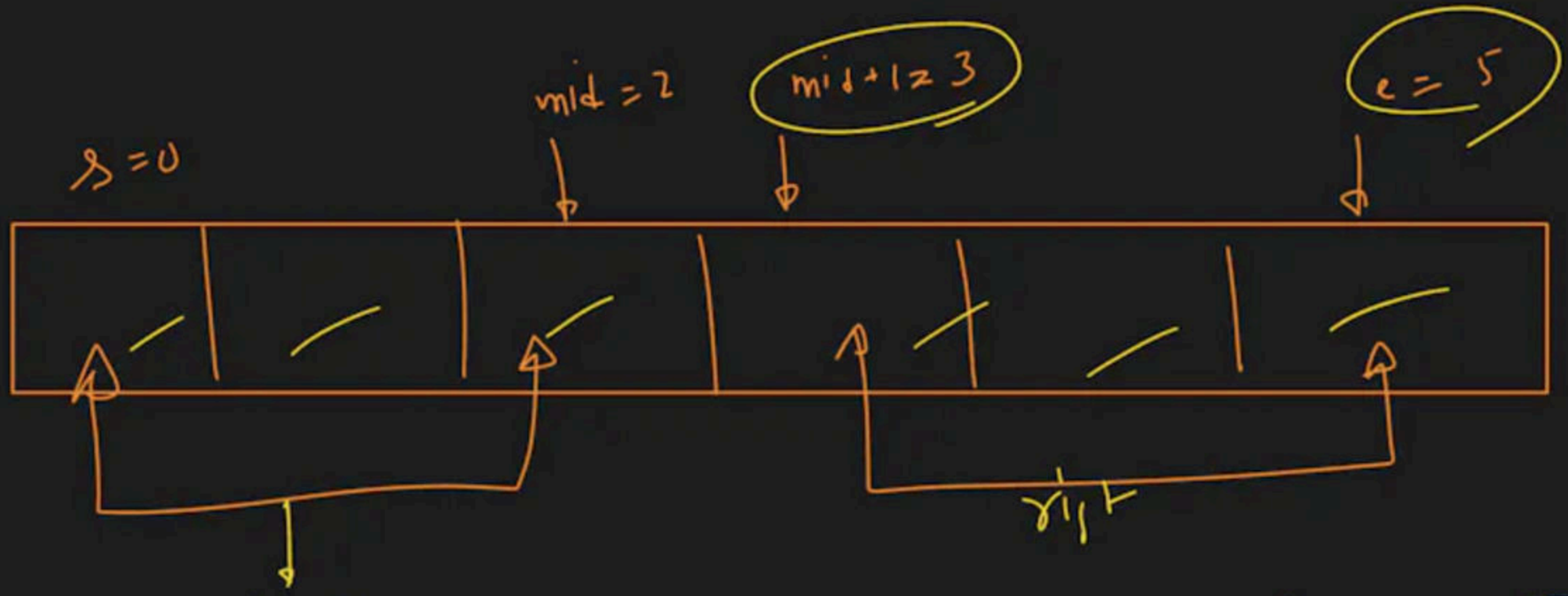
```
mergeSort(arr, s, e)
{
    // B.C
    if(s >= e)
        return;

    int mid = (s+e)/2;

    mergeSort(arr, s, mid);
    mergeSort(arr, mid+1, e);
    merge(arr, s, e, mid);
}
```


mid
left
right
mid
left
right
mid
left
right
mid
left
right
72674


Break
mid - s + 1
= 2 - 0 + 1
left → mid - s + 1
right → e - mid
10 15 20, 45, 65, 70
15, 20, 70
10 45 65
R.C
B.C



len1 = mid - s + 1

= 2 - 0 + 1

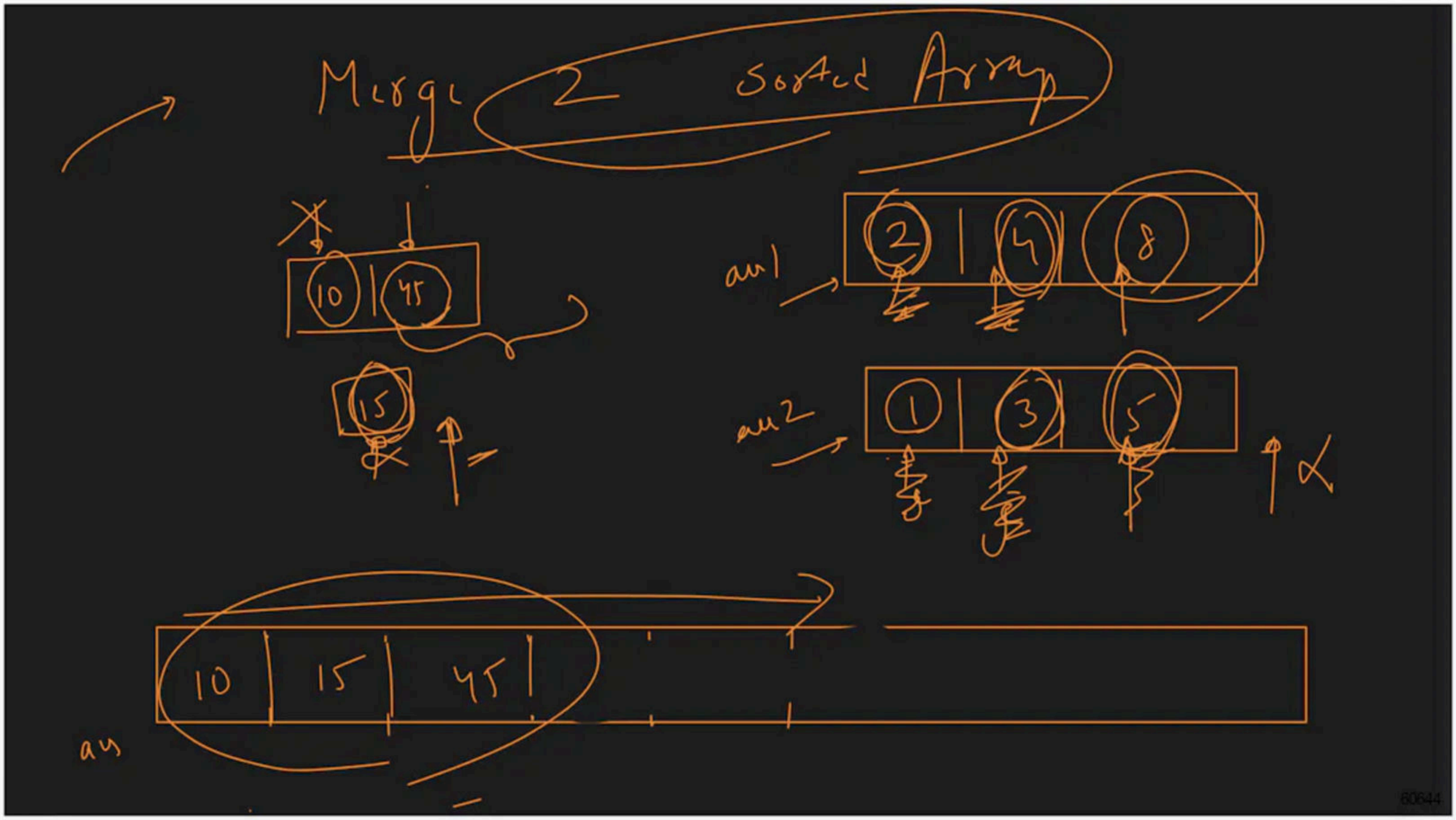
= 3

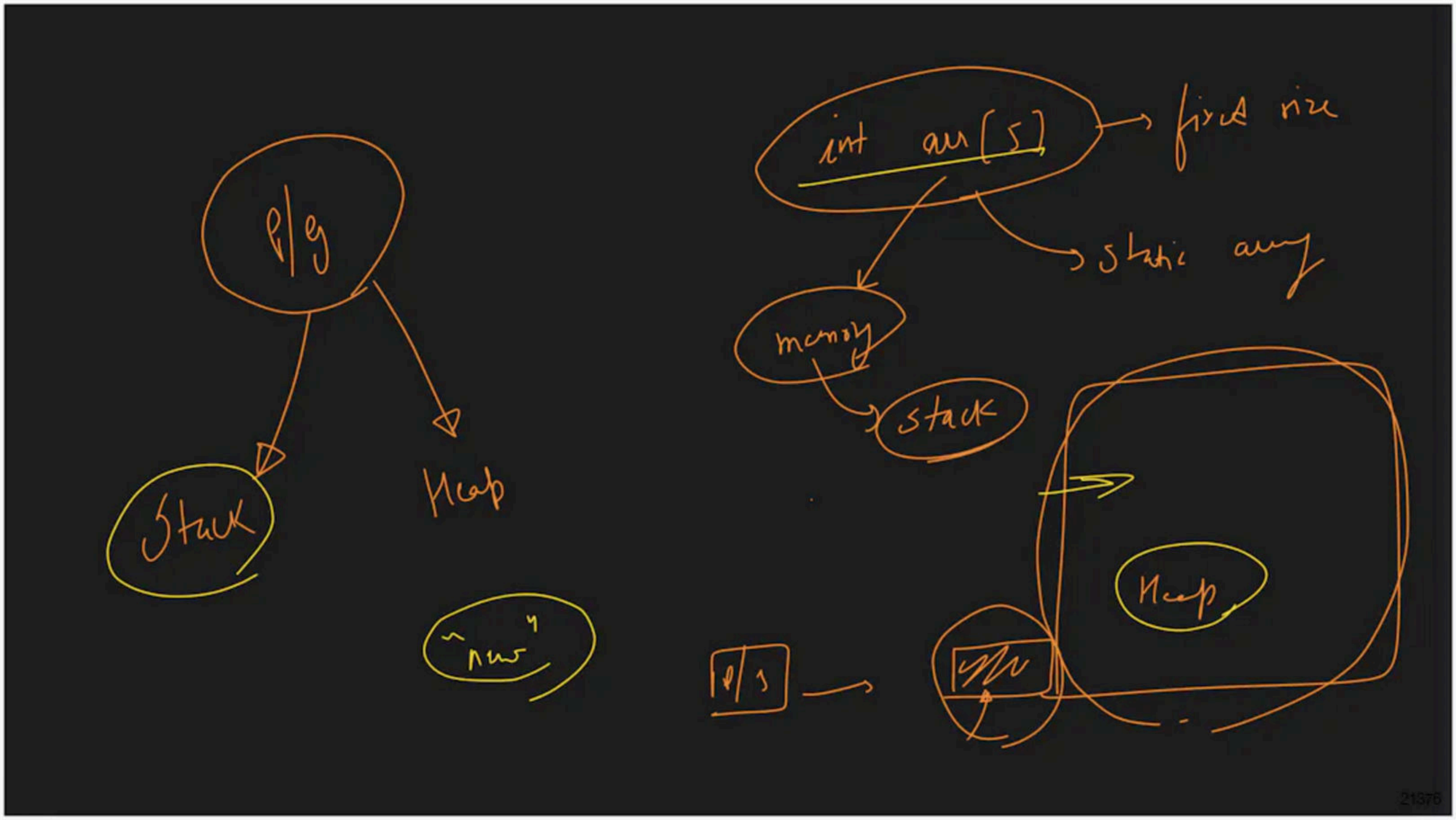
= e - (mid + 1) + 1 = e - mid - 1 + 1

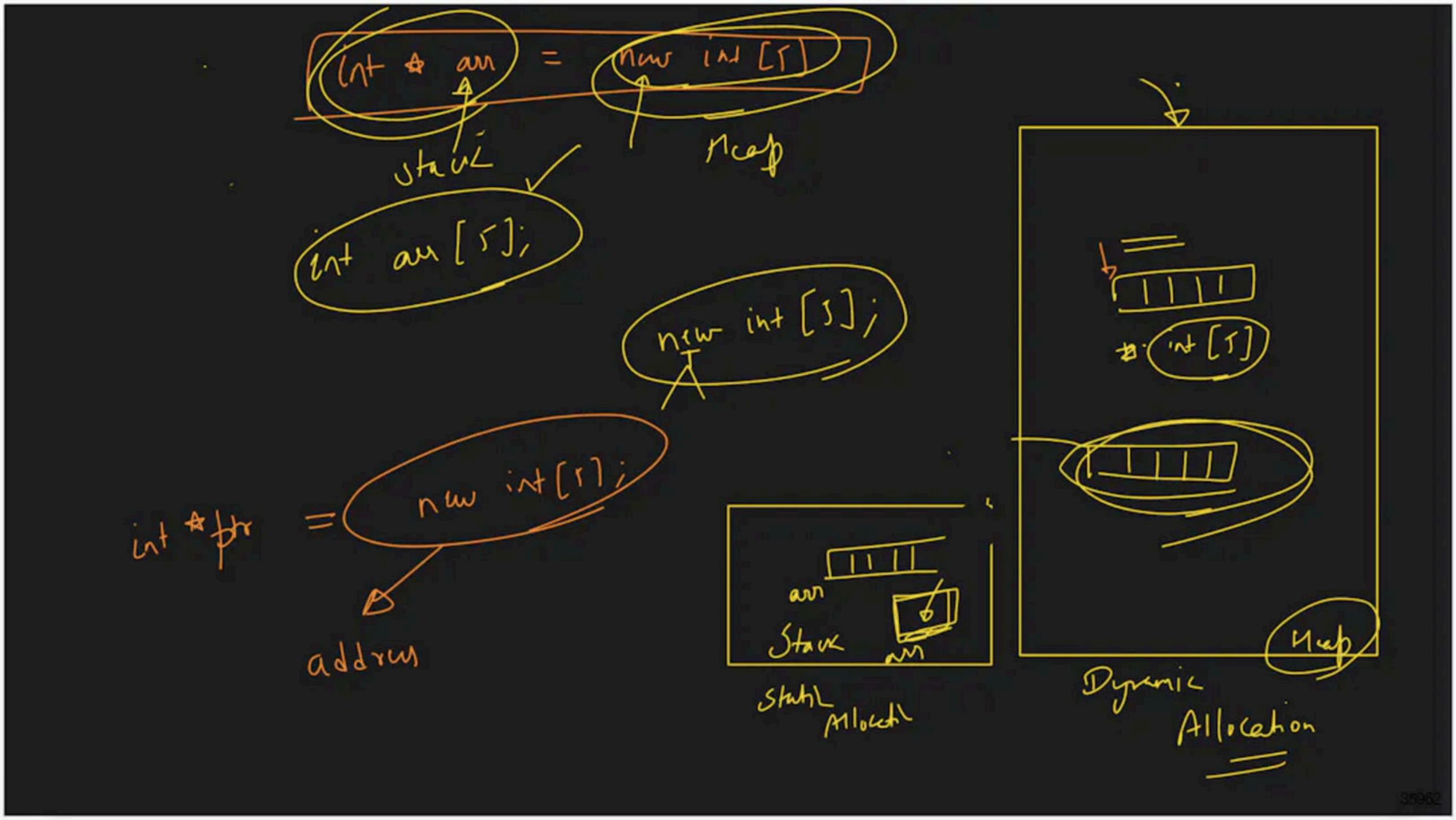
= 5 - 3 + 1

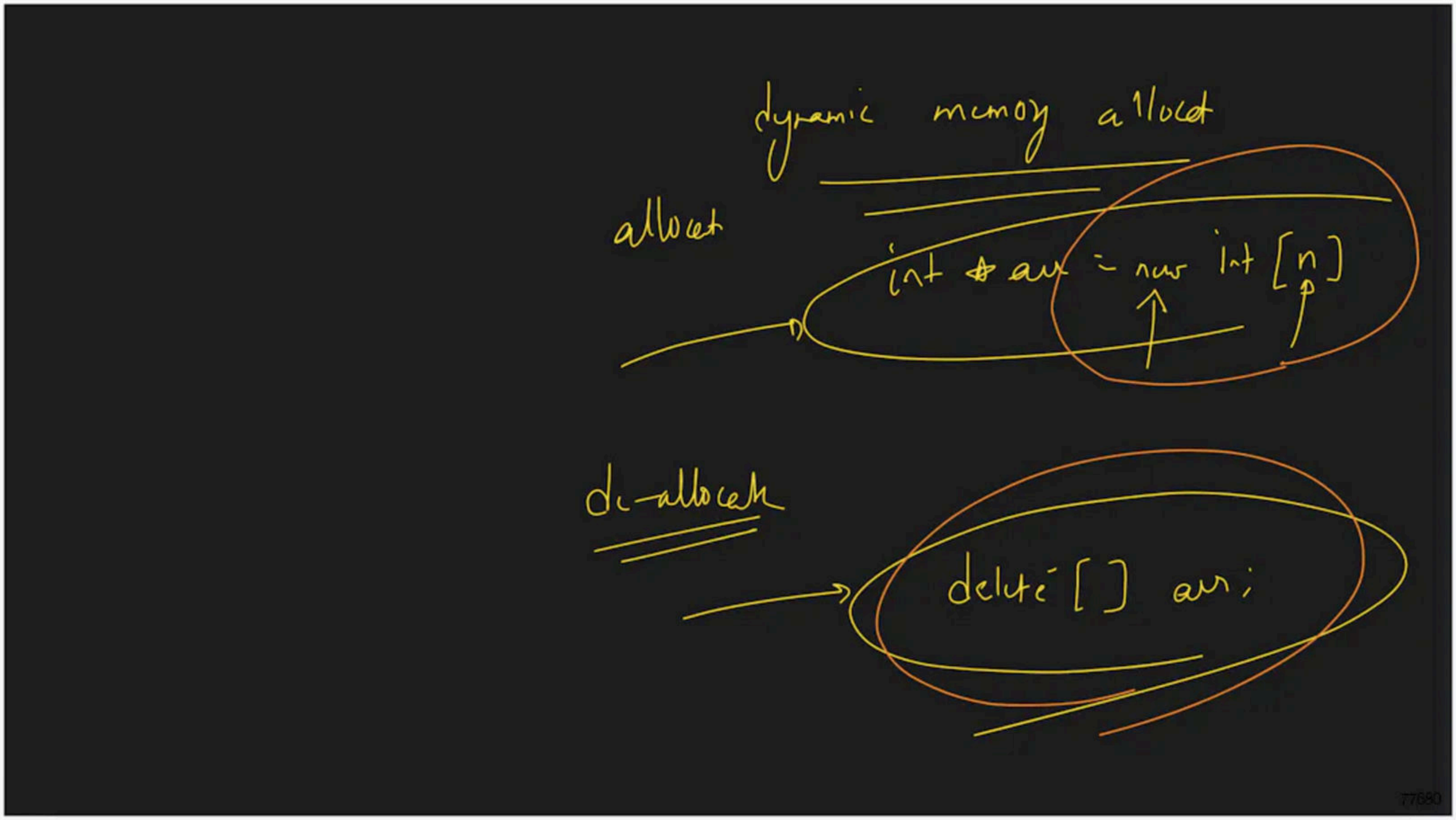
= 2 + 1

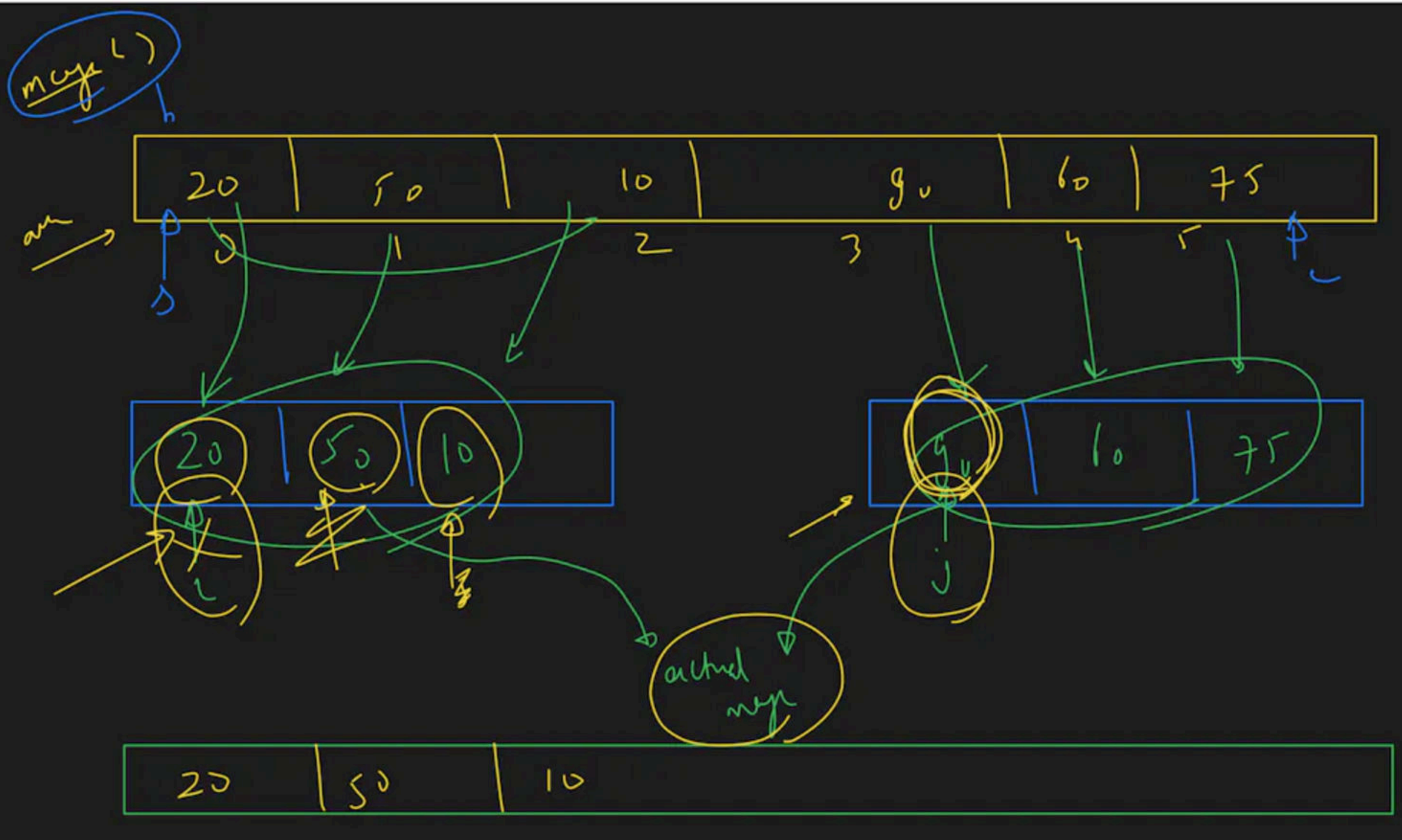
len2

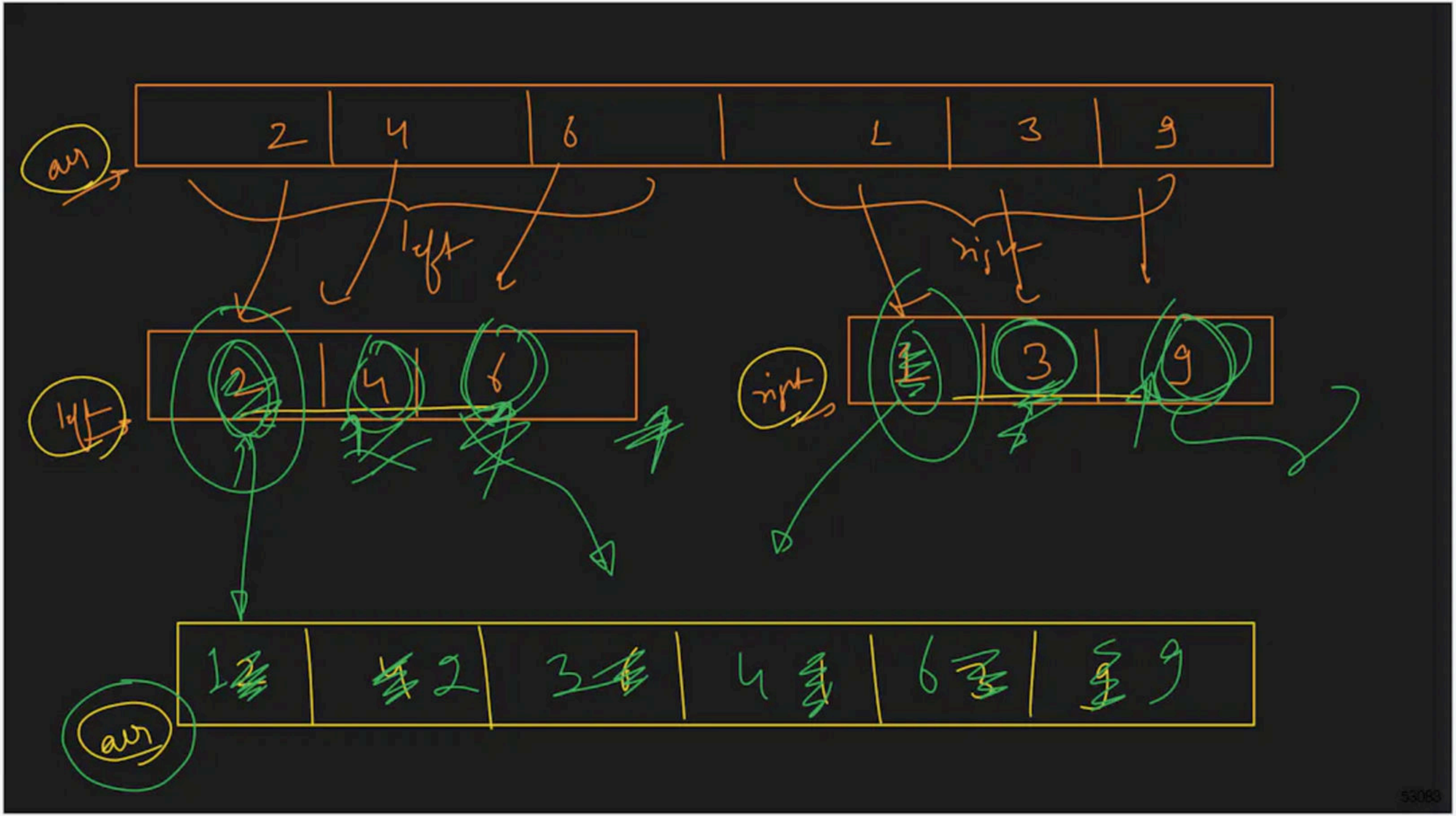

Merge 2 Sorted Arrays
10
45
15
arr1
2
4
8
arr2
1
3
5
arr
10
15
45

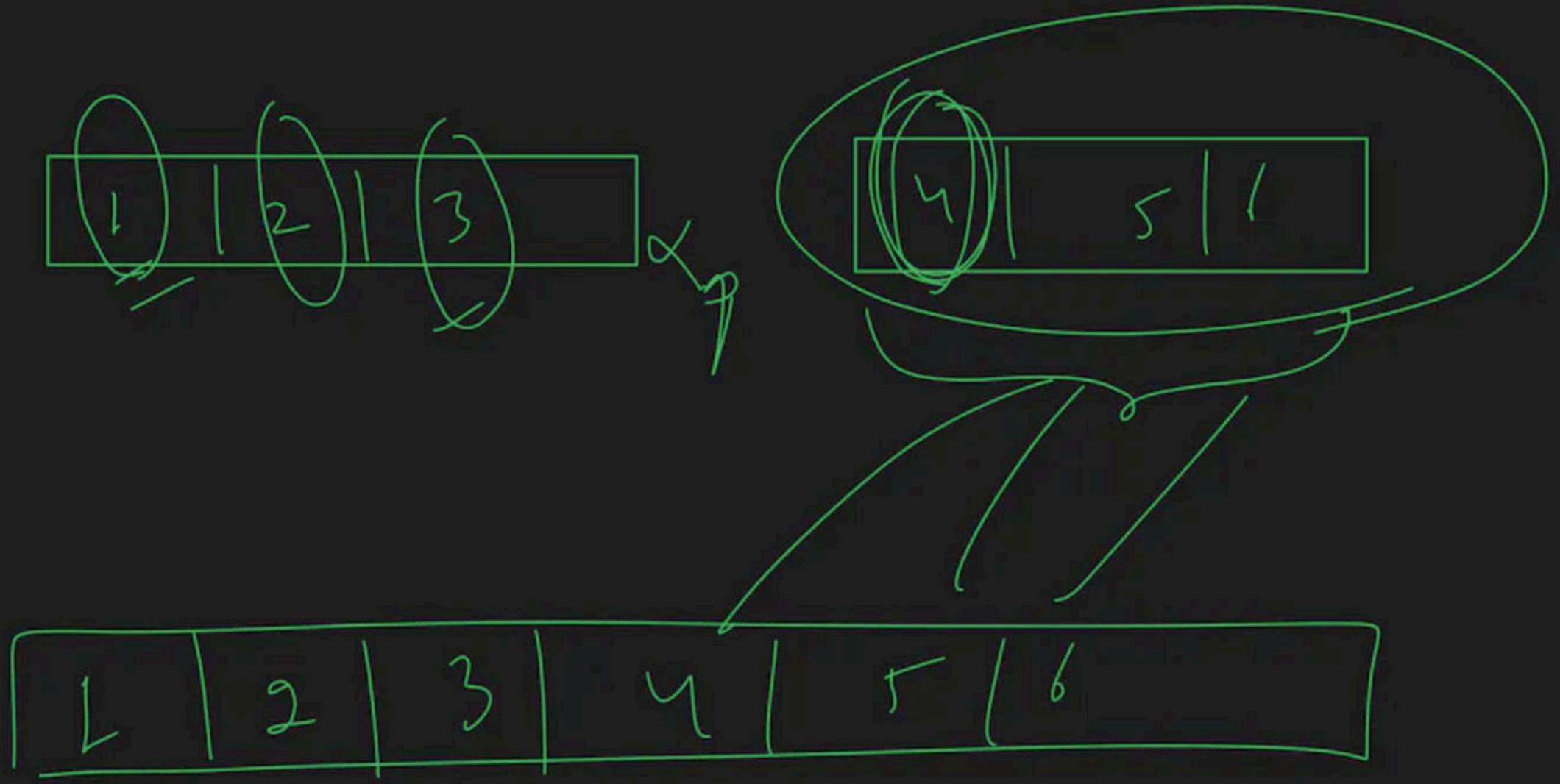

p/g
Stack
Heap
~new~
int arr[5]
fixed size
static array
memory
stack
Heap
p/s


int * arr = new int [5]
stack
Heap
int arr [5];
new int [3];
int *ptr = new int [5];
address
arr
Stack
arr
Static Allocation
int [5]
Heap
Dynamic Allocation


dynamic memory allocat
allocat
int * arr = new int [n]
de-allocath
delete [] arr;


main()
h
20
50
10
90
60
75
arr
0
1
2
3
4
5
5
20
50
10
4
60
75
i
j
actual
array
20
50
10


2
4
6
1
3
9
arr
left
right
left
right
arr


1
2
3
4
5
6

mergeSort (arr , s , e)

{

// B.C

→ Break into L & R

→ Recursic for L & R

→ Merge L & R

}

merge (arr , s , e , mid)

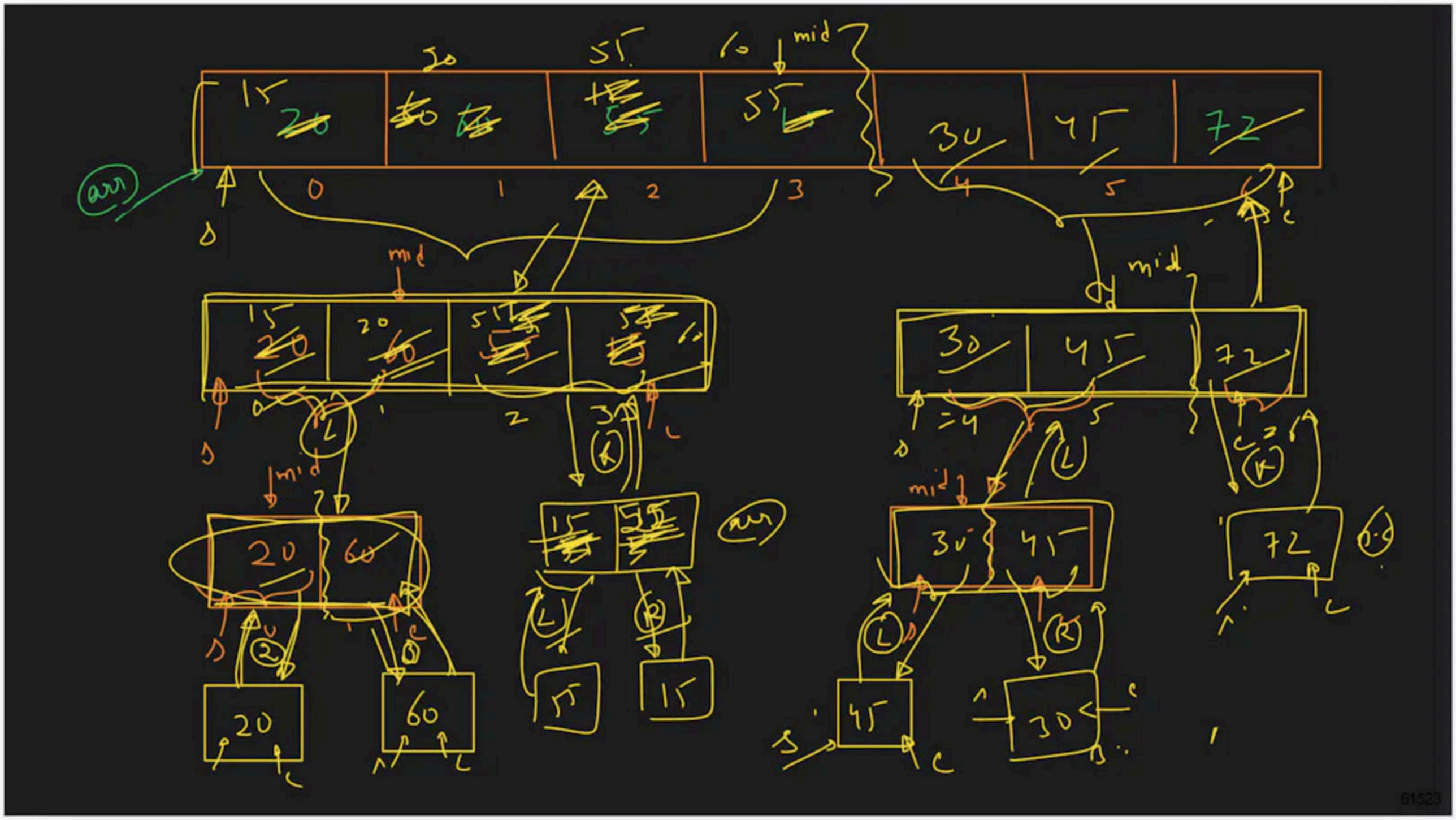
{

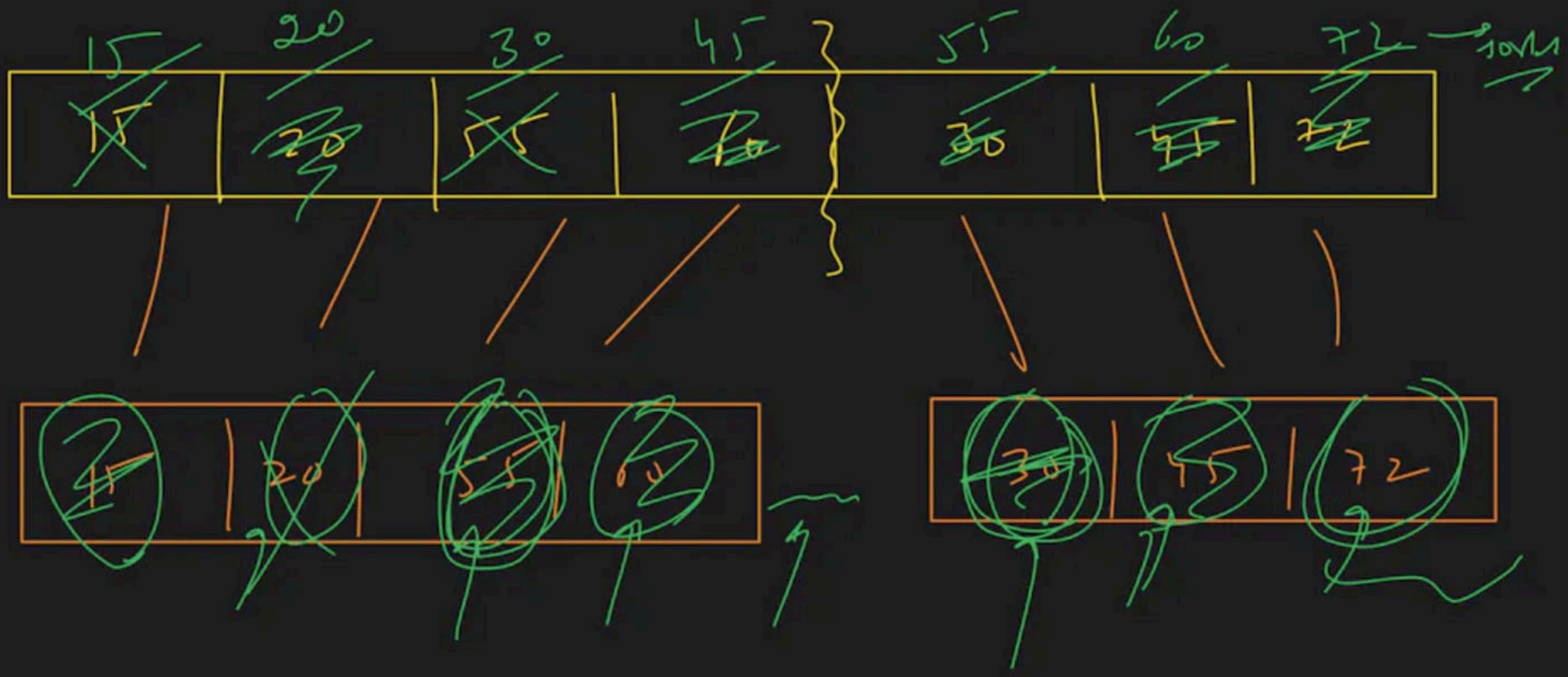
→ Create left & right arrays

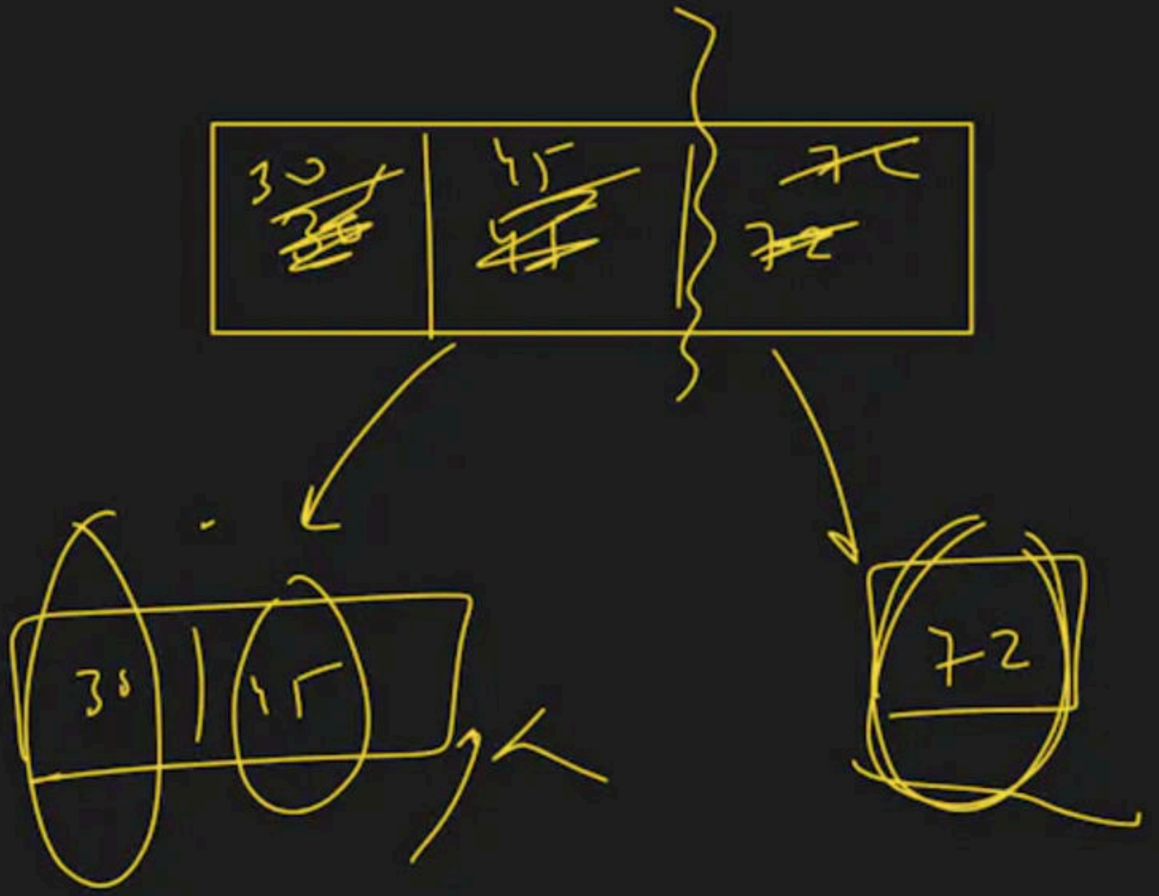
→ copy values from actual array into left & right arr

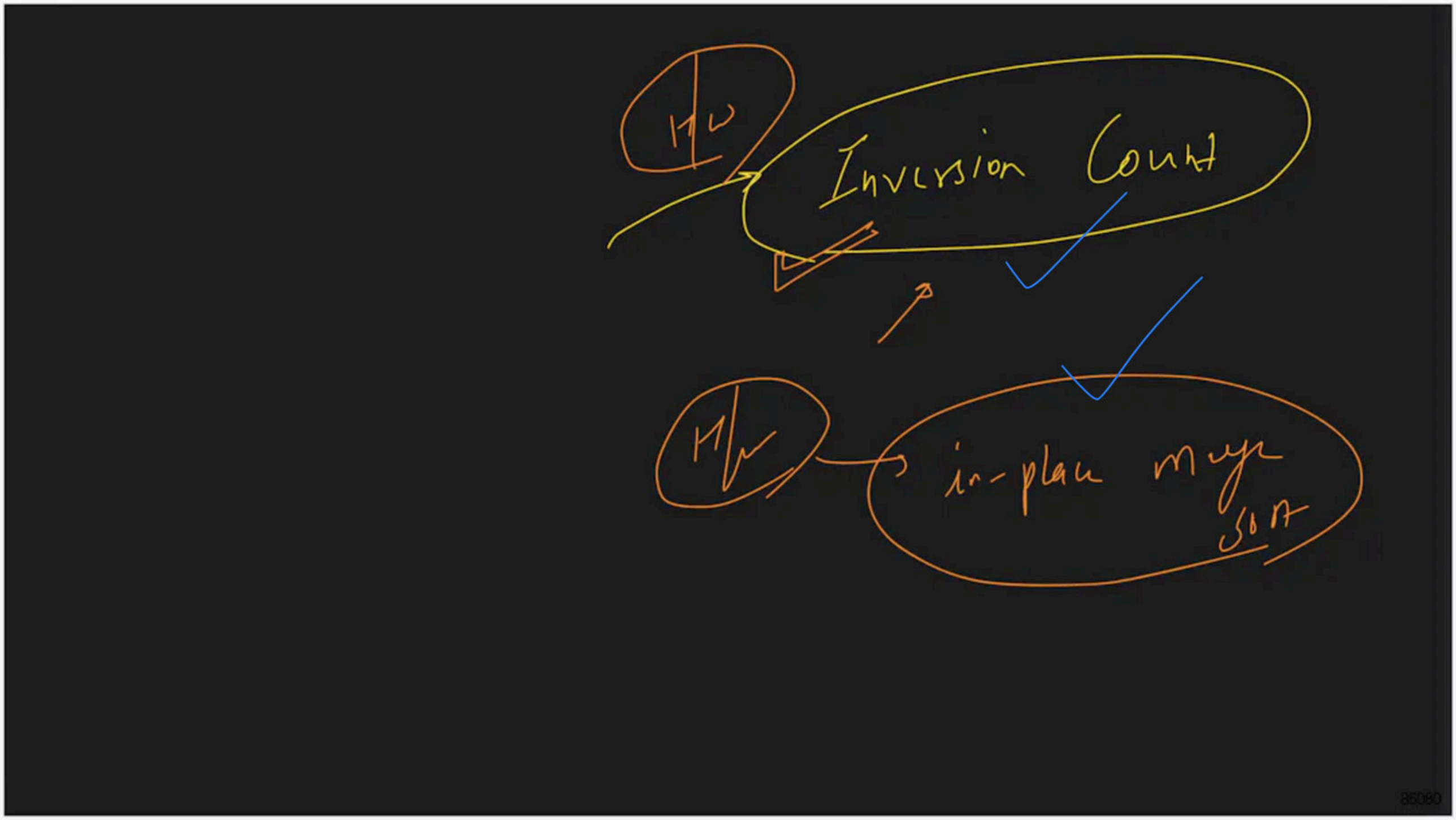
→ actual arr = 2 sorted array (arr 2 pointer)

}

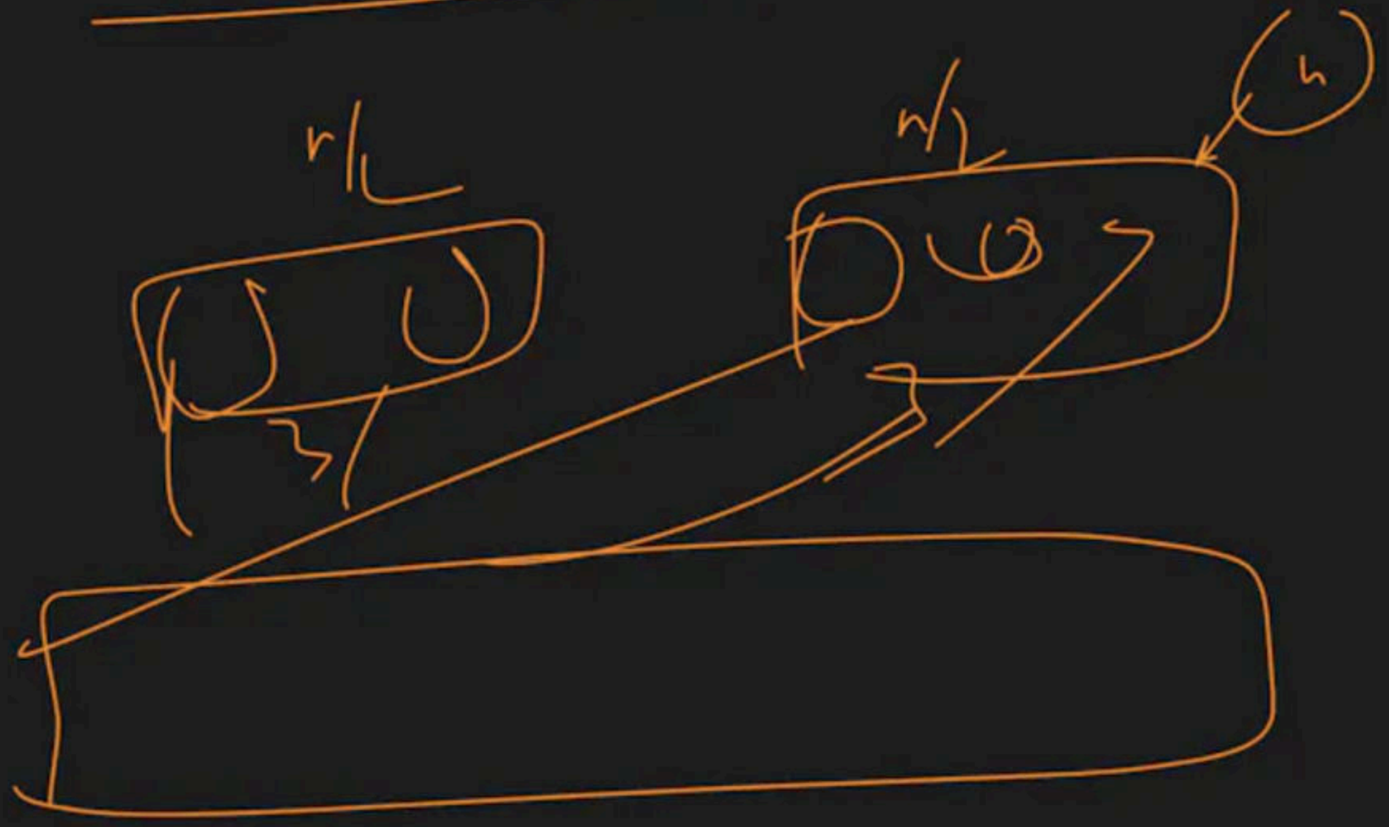

mid
arr
0
1
2
3
4
5
mid
mid
mid
mid
20
60
15
55
30
45
72
15
20
55
60
20
60
15
55
45
30
72


15
20
30
45
55
60
72
10M1


72


HW
Inversion Count
HW
in-place merge sort

T.C → O(n log n)



MS ( )

{

//Dc → K_1

MS(half)

MS(half)

Merge()

}

$$T(n) = K_1 + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + n*K$$

$$T(n) = K_1 + 2T\left(\frac{n}{2}\right) + n*K$$

$$2T\left(\frac{n}{2}\right) = 2K_1 + 4T\left(\frac{n}{4}\right) + \frac{n*K}{2} \times 2$$

$$4T\left(\frac{n}{4}\right) = 4K_1 + 8T\left(\frac{n}{8}\right) + \frac{n*1}{4} \times 4$$

$$T(1) = K_1$$

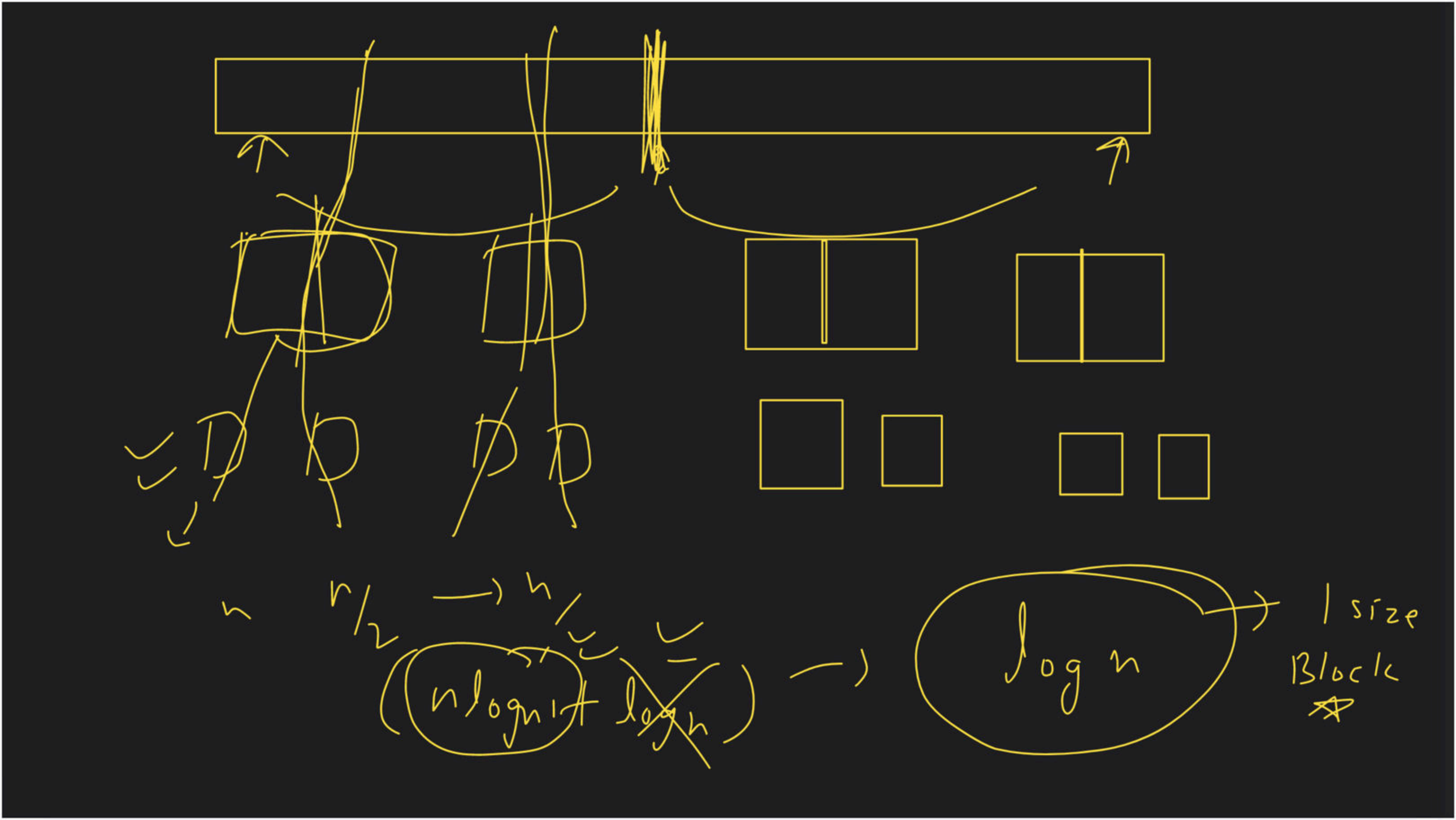
$T(n) =$

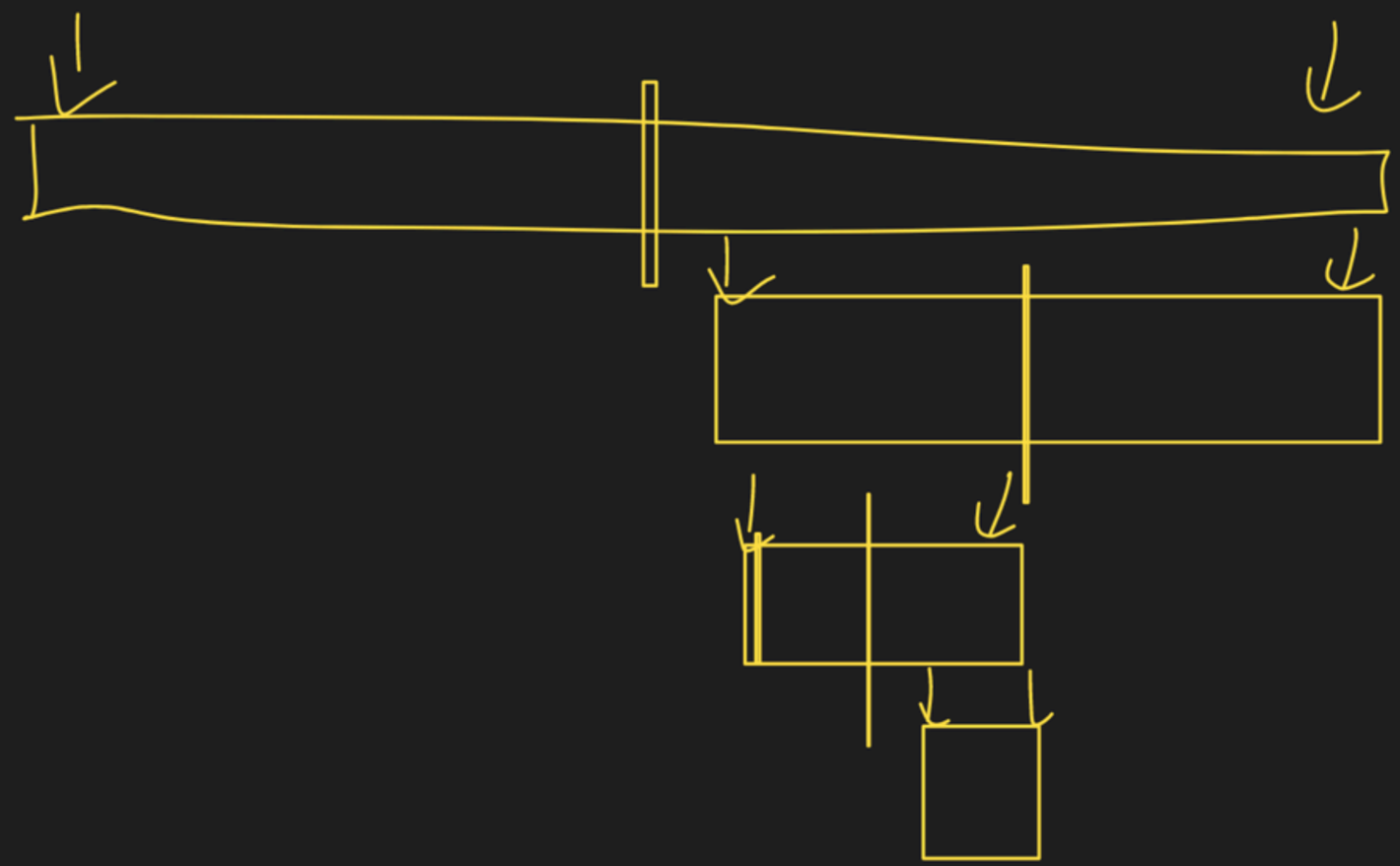
$a * n * K$

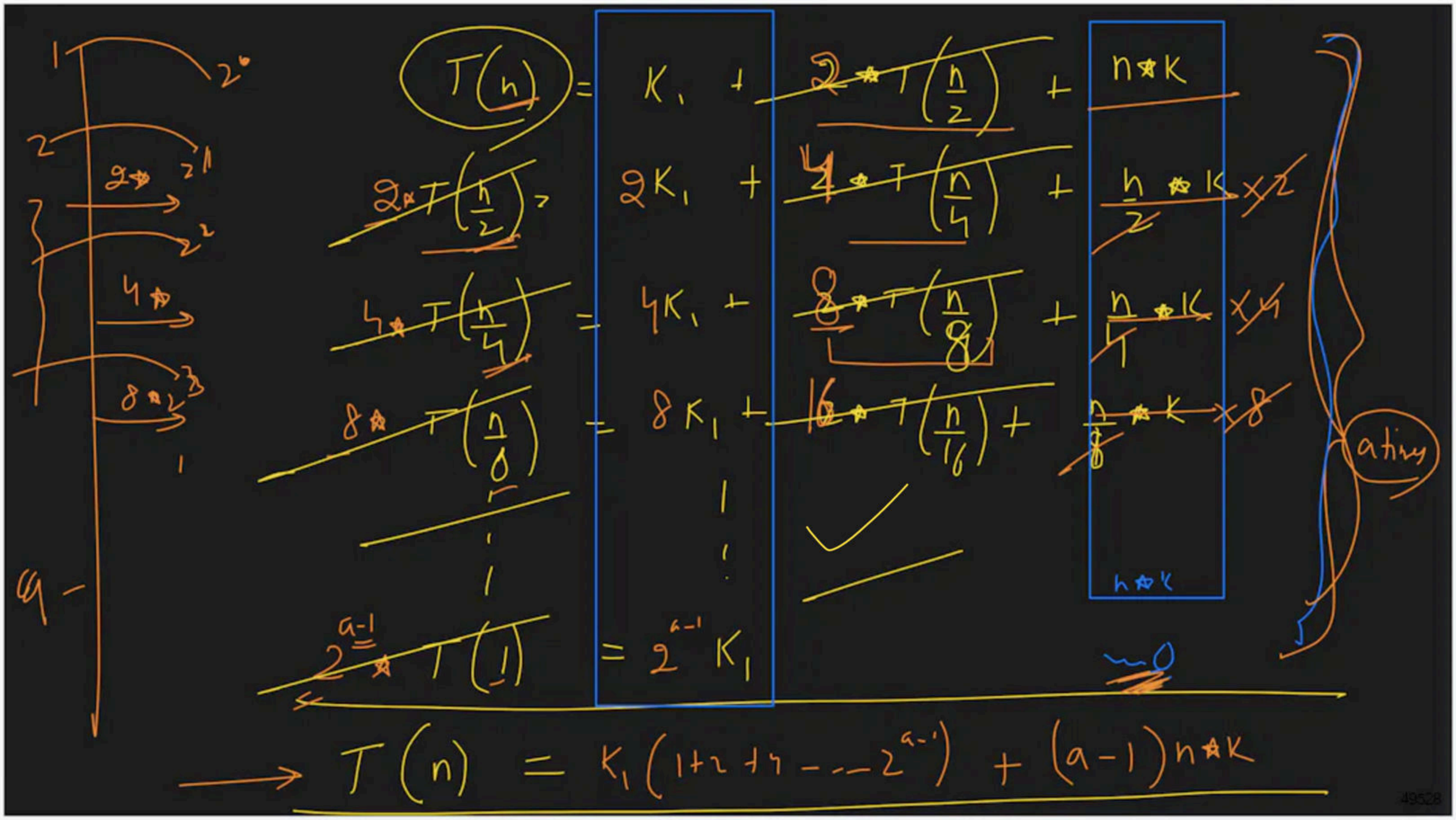
25054

$$T(n) = K_1 + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + n * K$$

$$T(n) = K_1 + 2T\left(\frac{n}{2}\right) + n * K$$


n
n/2 → n/4
(n log n + log n)
log n
1 size
Block




T(n) = K1 + 2*T(n/2) + n*K
2*T(n/2) = 2K1 + 4*T(n/4) + n/2*K
4*T(n/4) = 4K1 + 8*T(n/8) + n/4*K
8*T(n/8) = 8K1 + 16*T(n/16) + n/8*K
2^(a-1) T(1) = 2^(a-1) K1
h*K
~0
T(n) = K1(1+2+4+---2^(a-1)) + (a-1)n*K

$$T(n) = K_1\Big(\underbrace{1+2+4+\cdots+2^{a-1}}_{n}\Big) + (a-1)*n*K$$

$$= nK_1 + a*n*K$$

$$= n(K_1 + a \times K)$$

$$= n \times a \times K$$

$$= n \times a = n \times \log n$$

$$T(n) = K_1\left(\underbrace{1+2+4+\cdots+2^{a-1}}_{g.p}\right) + (a-1)n \star K$$

$$= K_1 \times n + a \star n \star K$$

$$= K_1 n + \log n \times n \times K$$

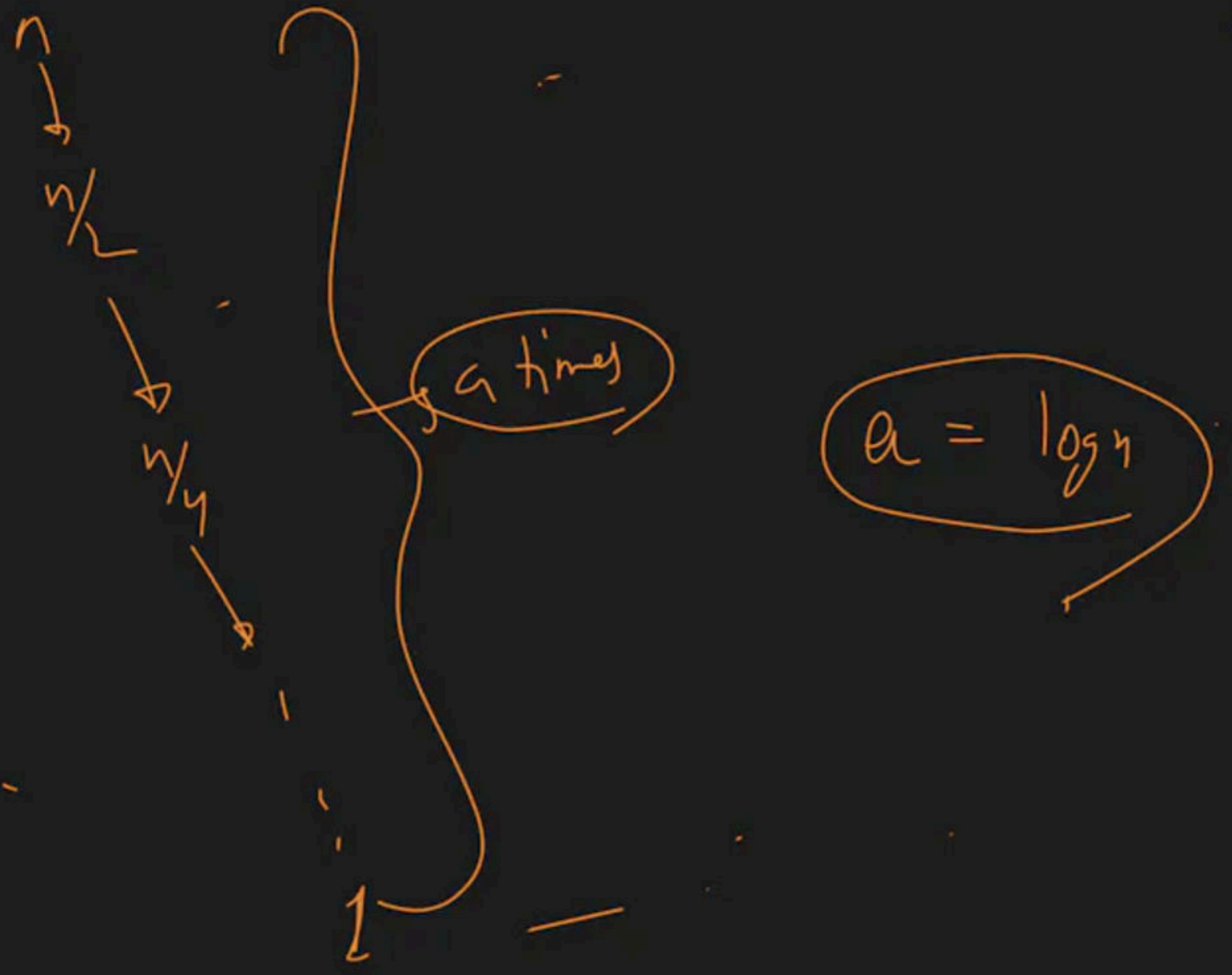
$$= Kn + n\log n \star K = n\log n \star K$$

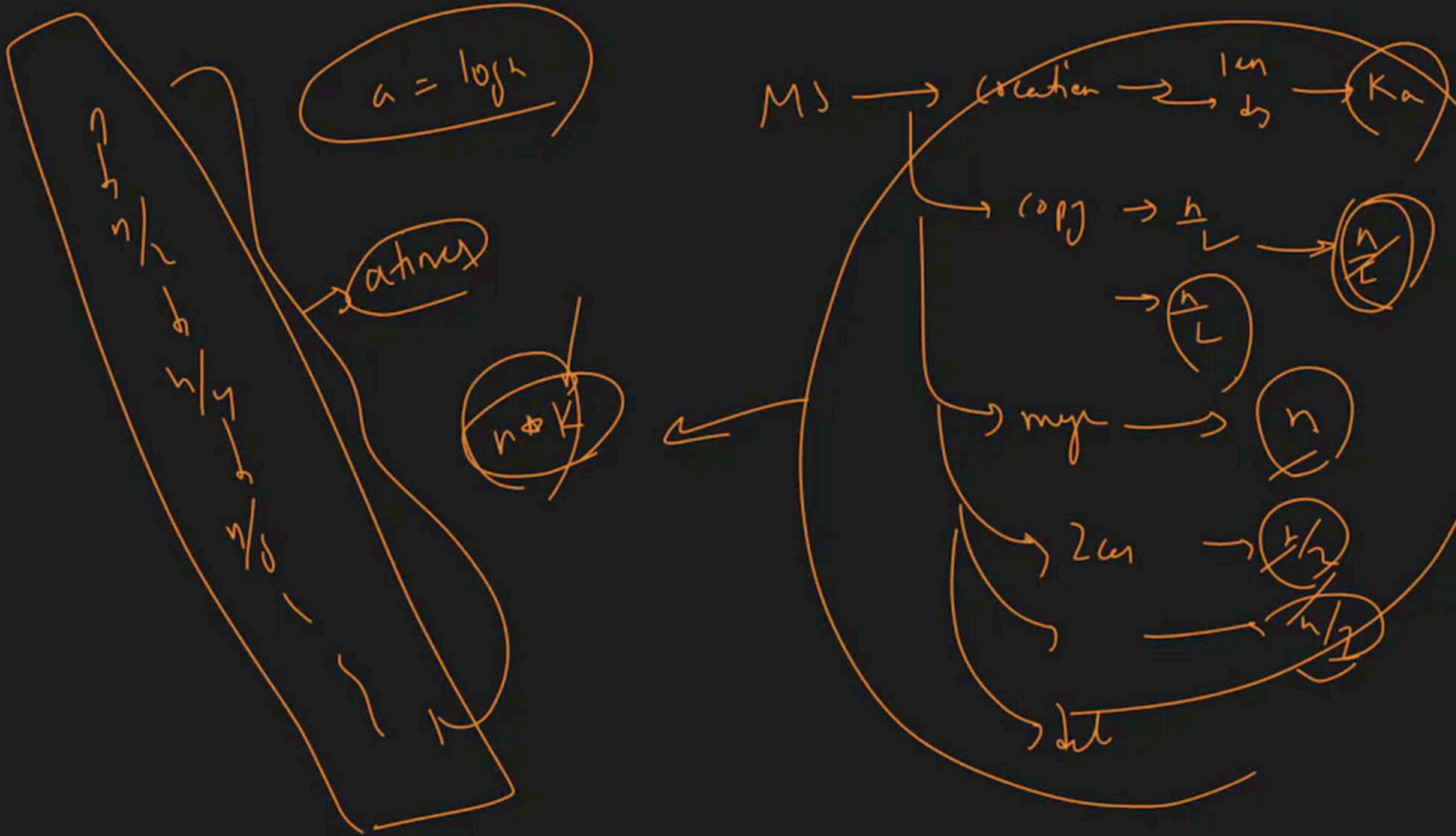
$$= n\log n$$

(a) $\dfrac{(r^n - 1)}{r-1}$

$1 \propto \left(\dfrac{2^a - 1}{2-1}\right)$

$2^a = 2^a = 2^{\log_2 n} = n$

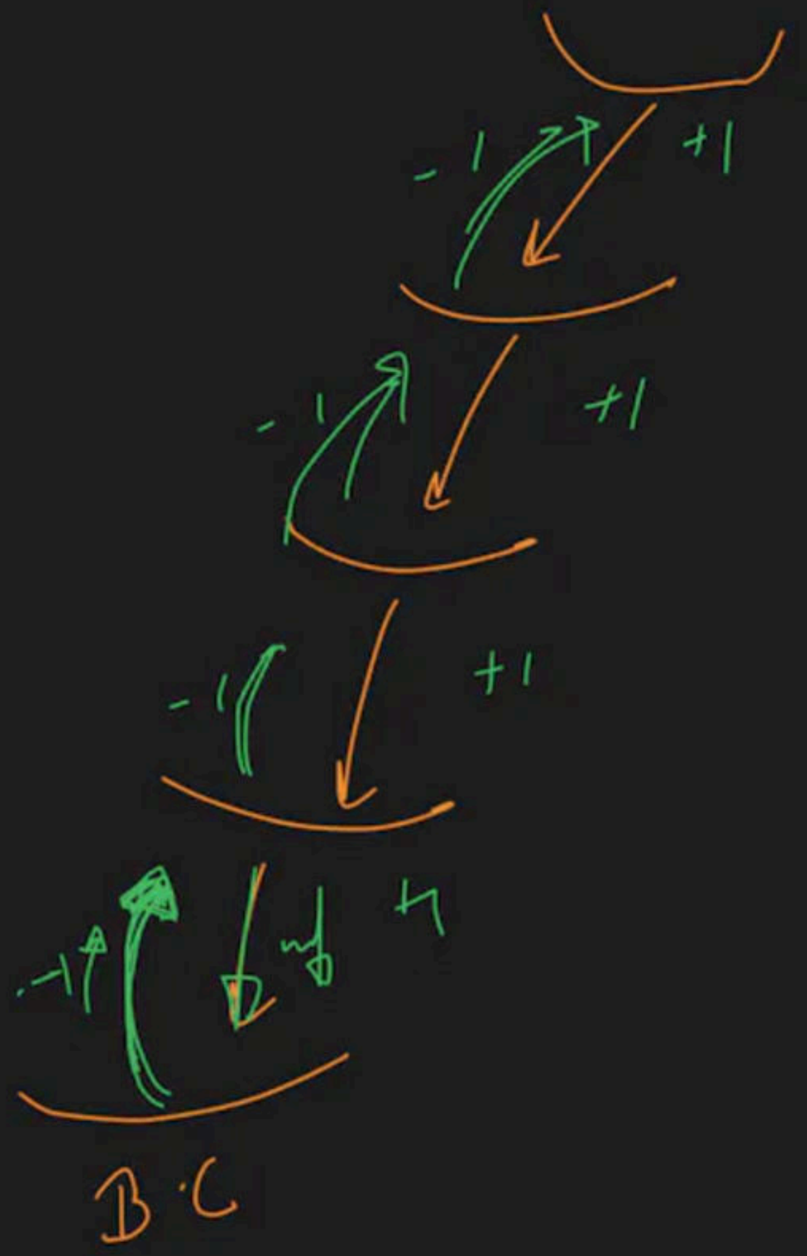

n
n/2
n/4
1
a times
a = log n

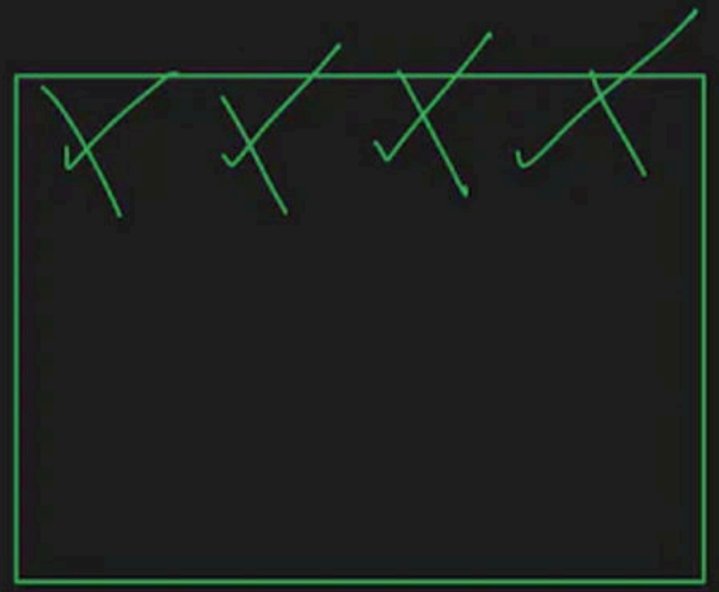

a = logn
n
n/2
n/4
n/8
n*K
MS
copy
n
n/2

DnC && Backtracking Class - 2

Special class

Love Babbar • Oct 20, 2023



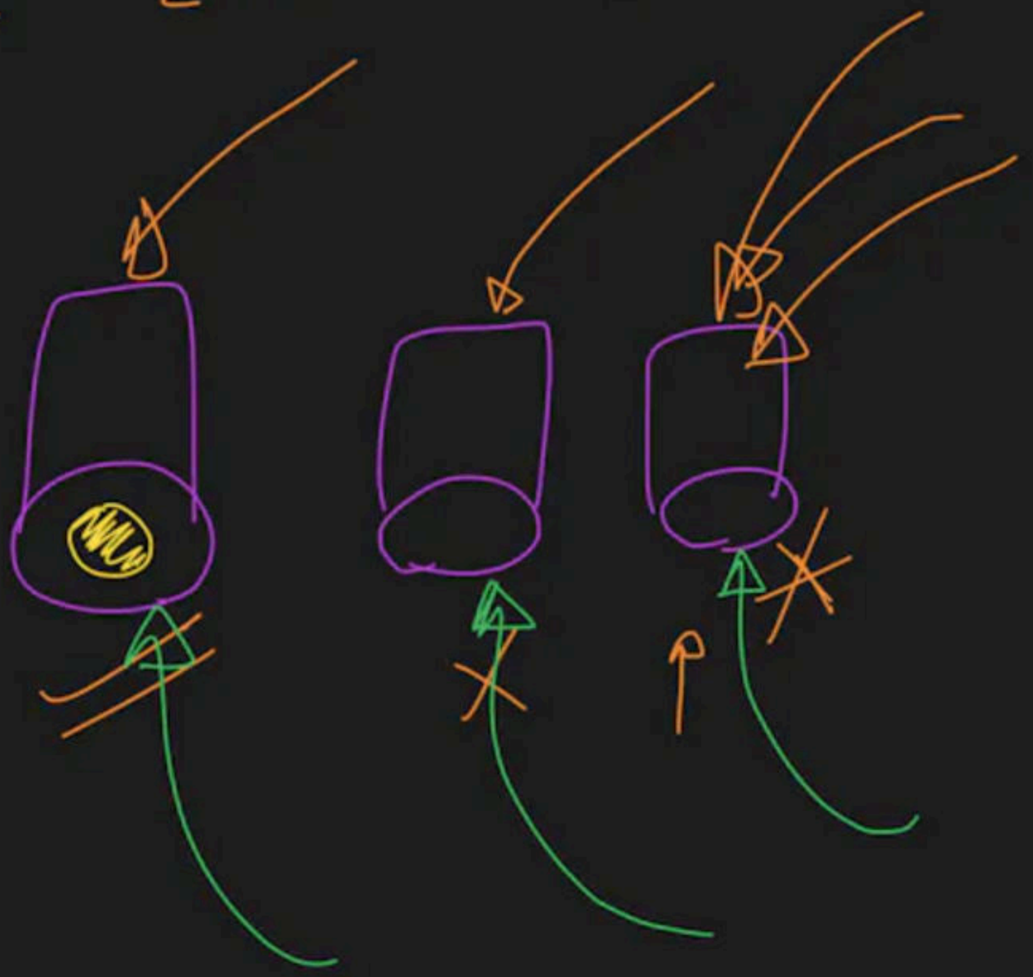


Backtracking :-

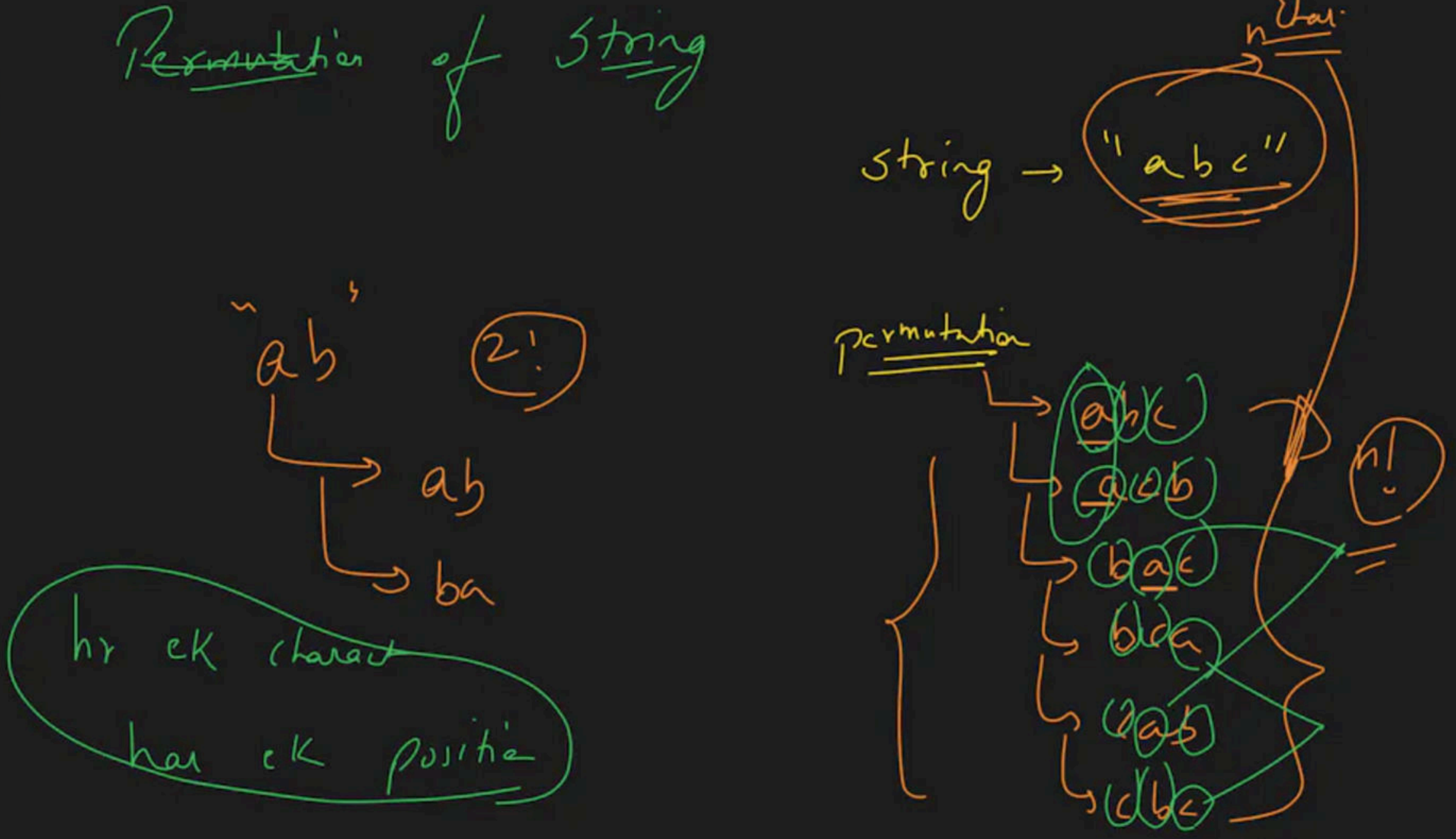
explore all possible sol

explore

dubara explore X



Permutation of String



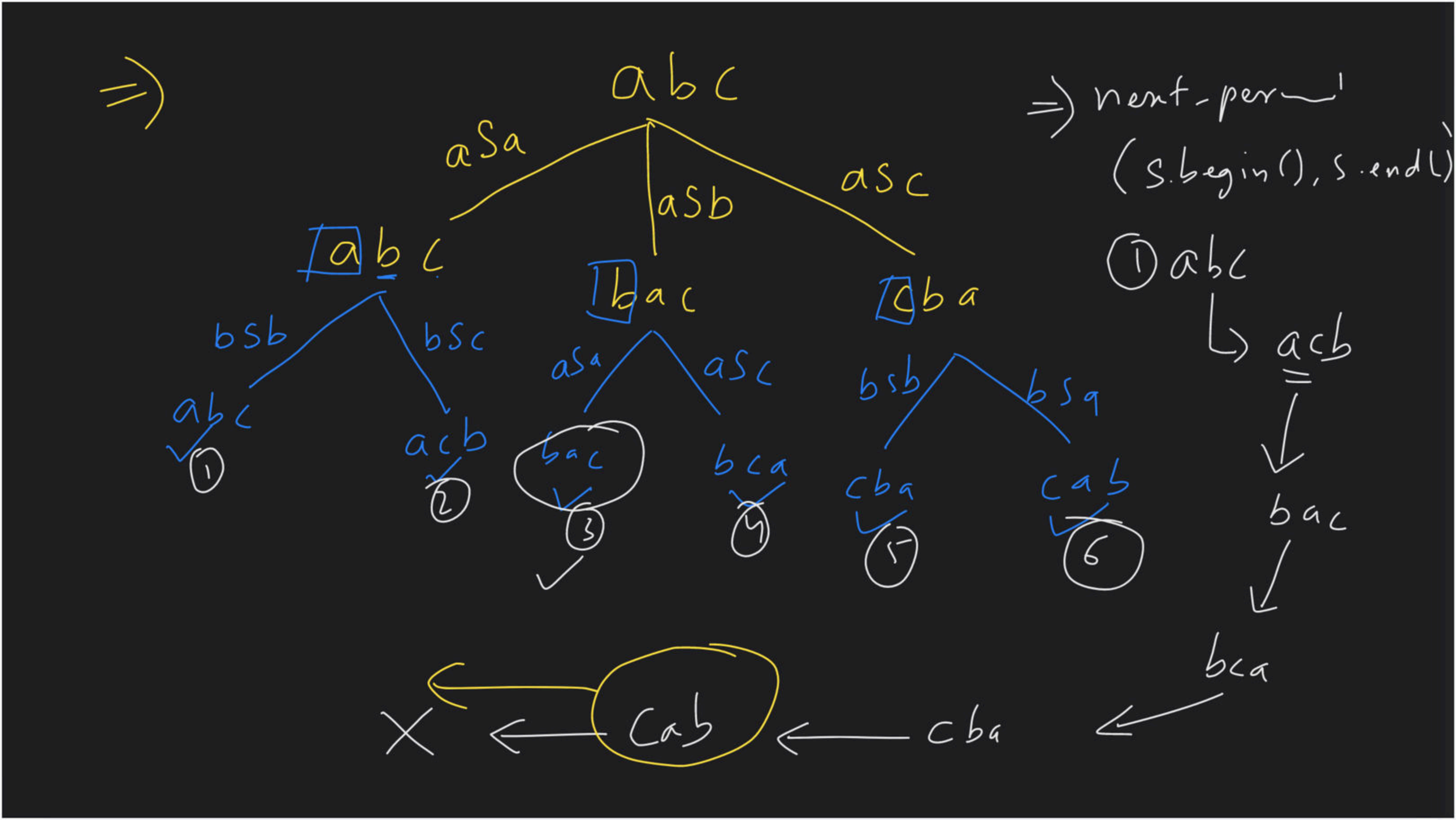
=) Doubt

=) STL =) kyne ki'

next_permutation ( )

abc

=)

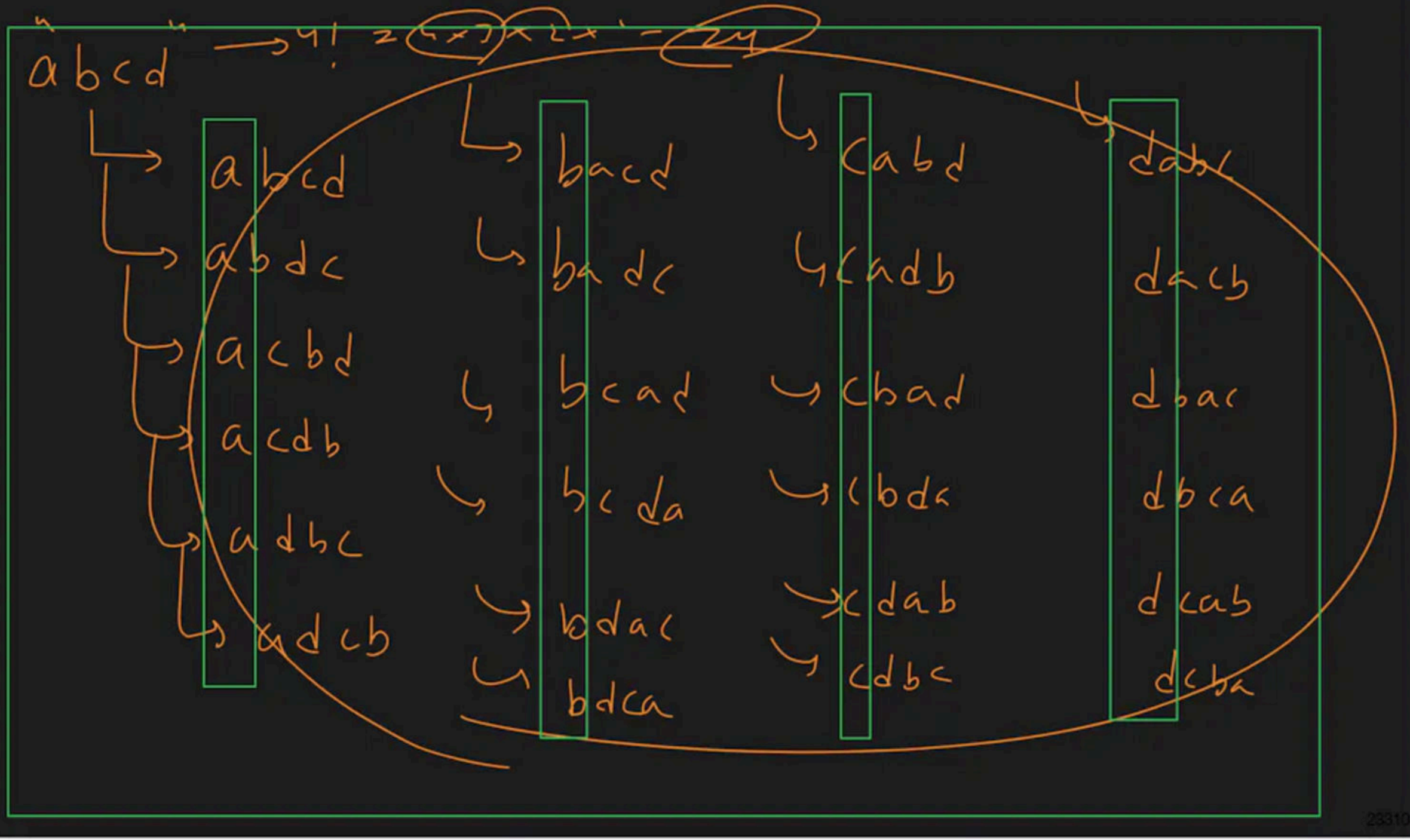

abc
aSa
aSb
aSc
abc
bac
cba
bSb
bSc
aSa
aSc
bSb
bSa
abc
acb
bac
bca
cba
cab
1
2
3
4
5
6
next_permu
(s.begin(), s.end())
① abc
acb
bac
bca
cba
cab
X

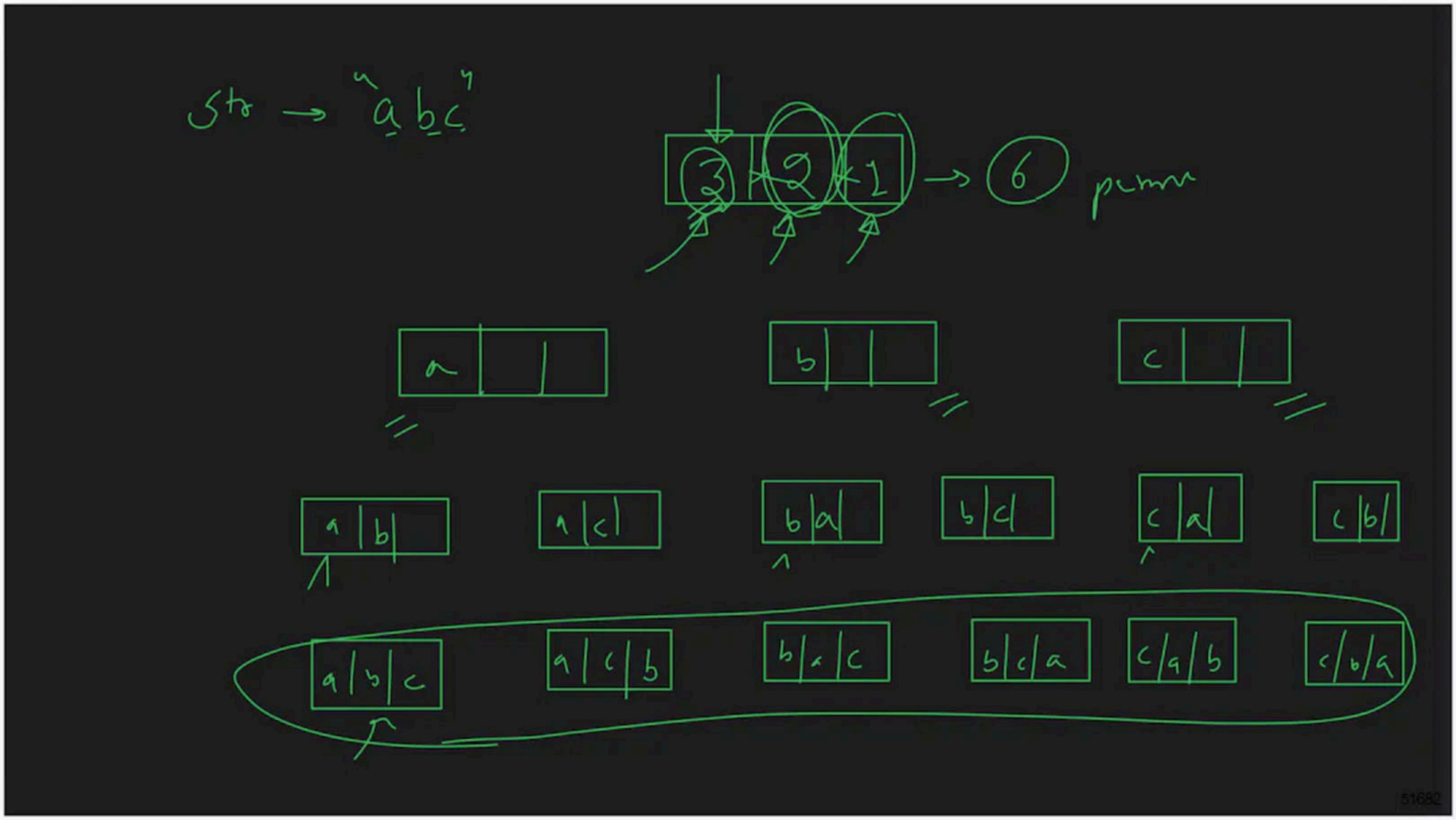
⇒ S = "abc"

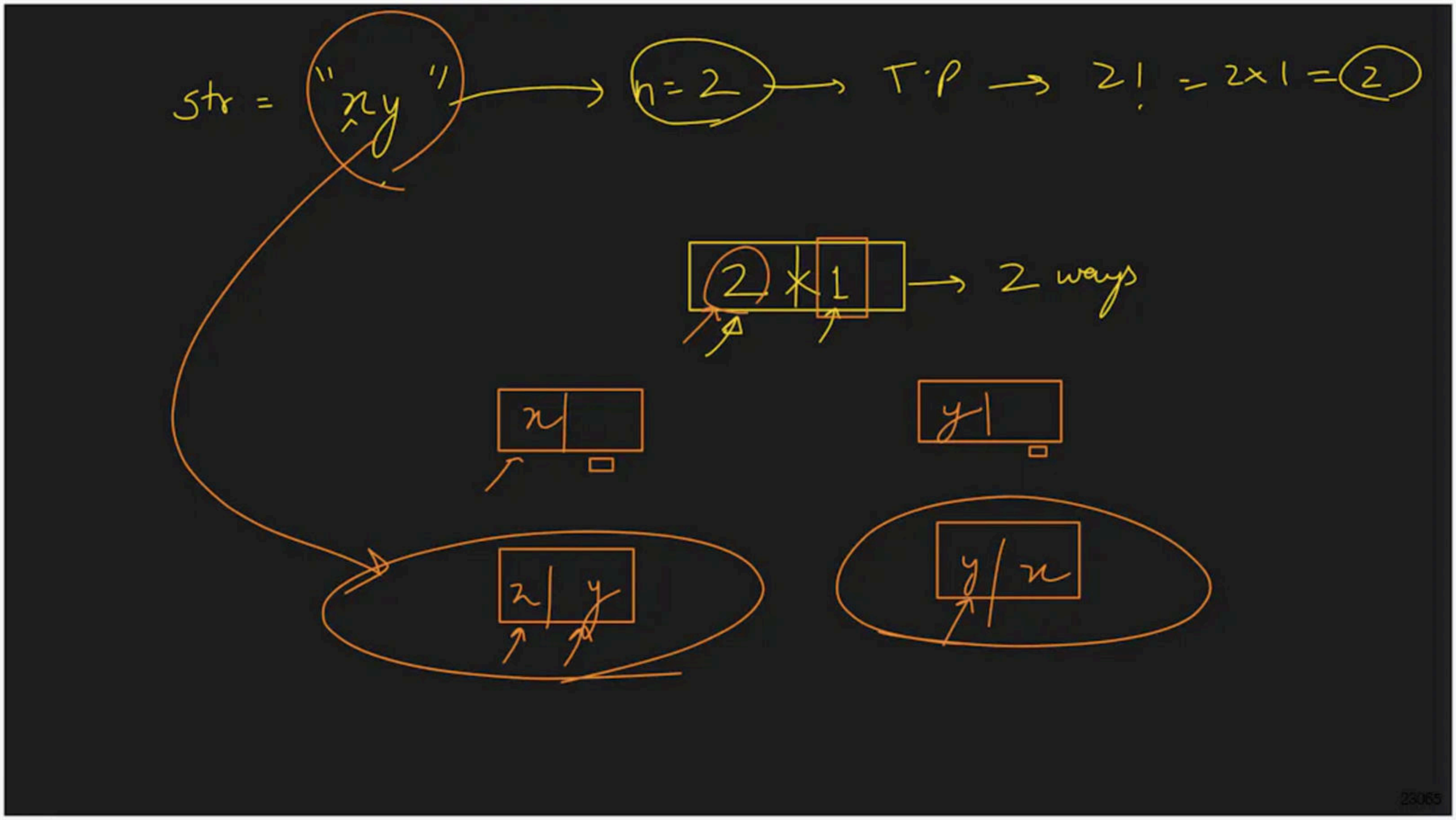
↳ cout << S << endl;

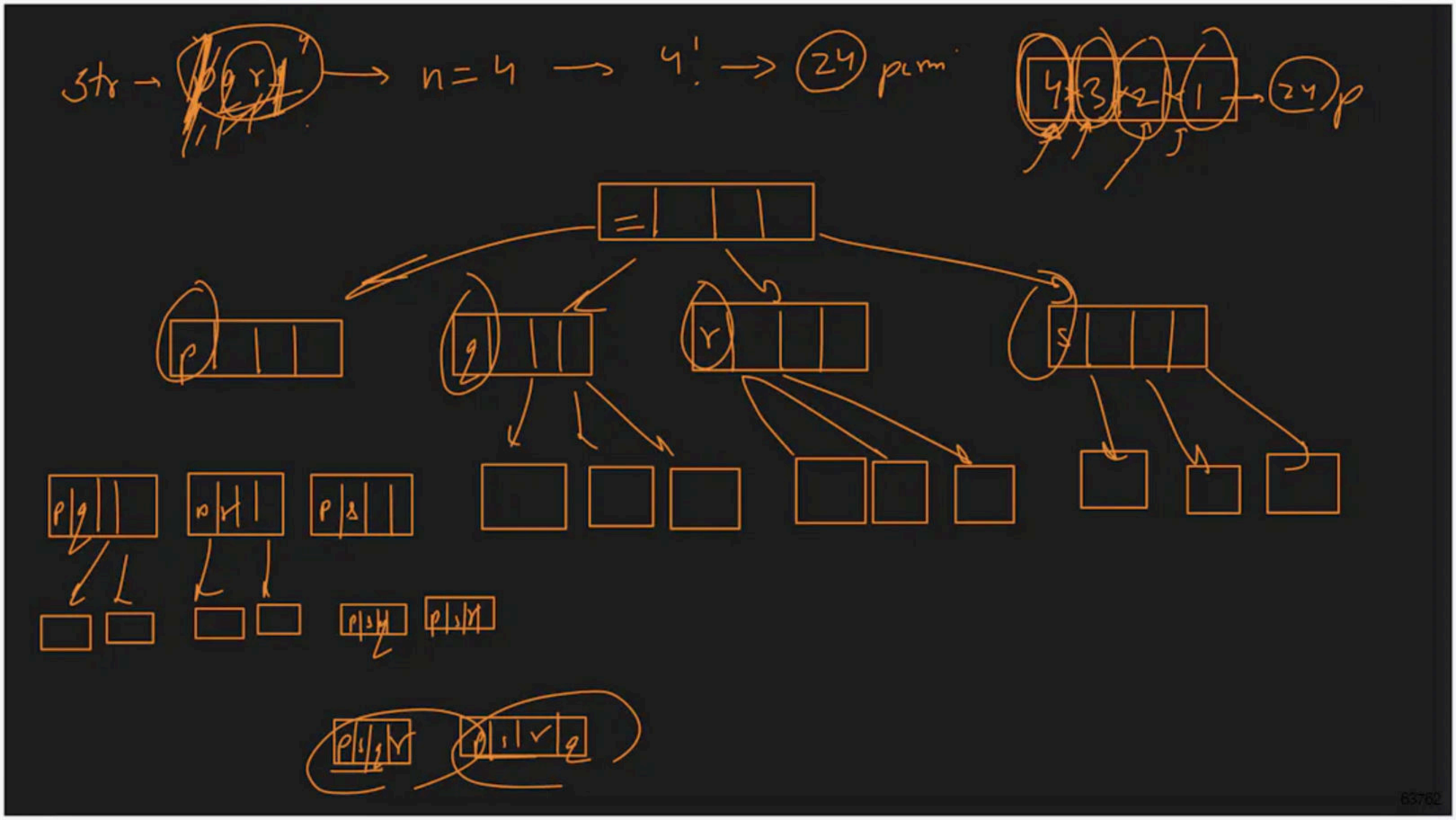
while(next_permut(s.begin, s.end()))
{
cout << s << endl;
}

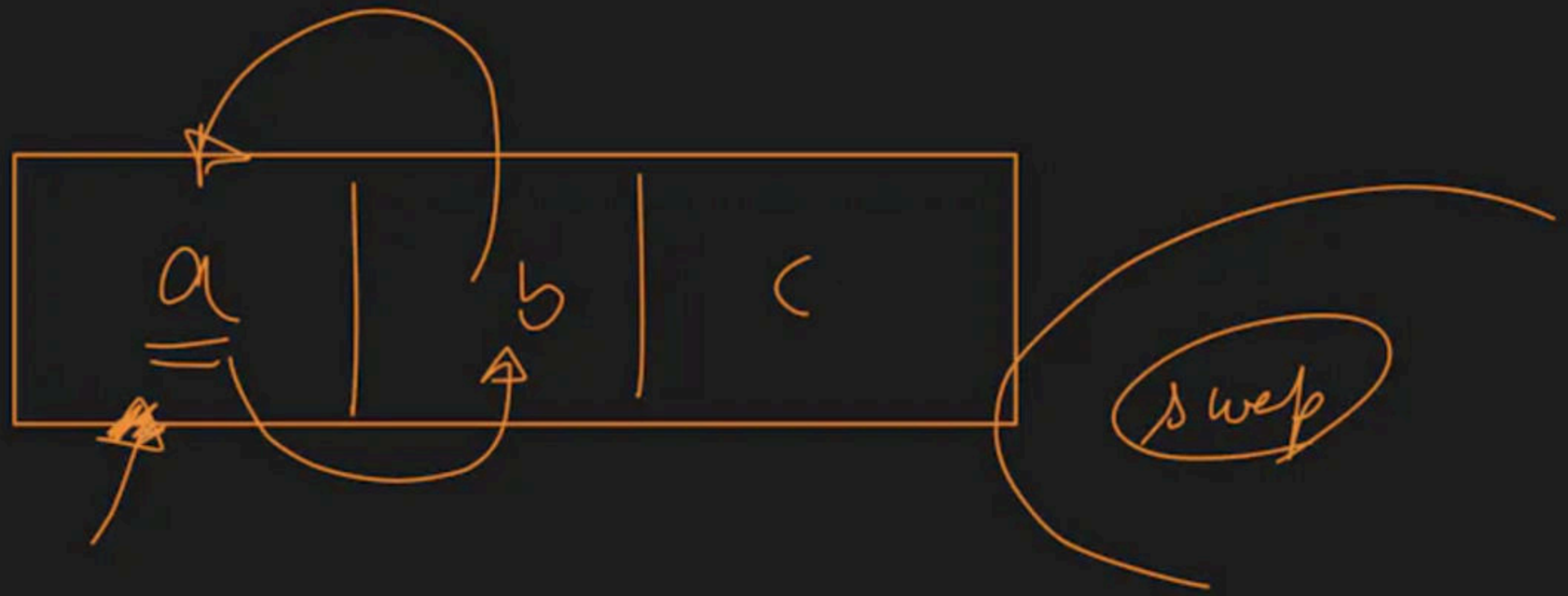
Bahar ☆☆


"abcd" → 4! = 24
abcd
abdc
acbd
acdb
adbc
adcb
bacd
badc
bcad
bcda
bdac
bdca
cabd
cadb
cbad
cbda
cdab
cdba
dabc
dacb
dbac
dbca
dcab
dcba

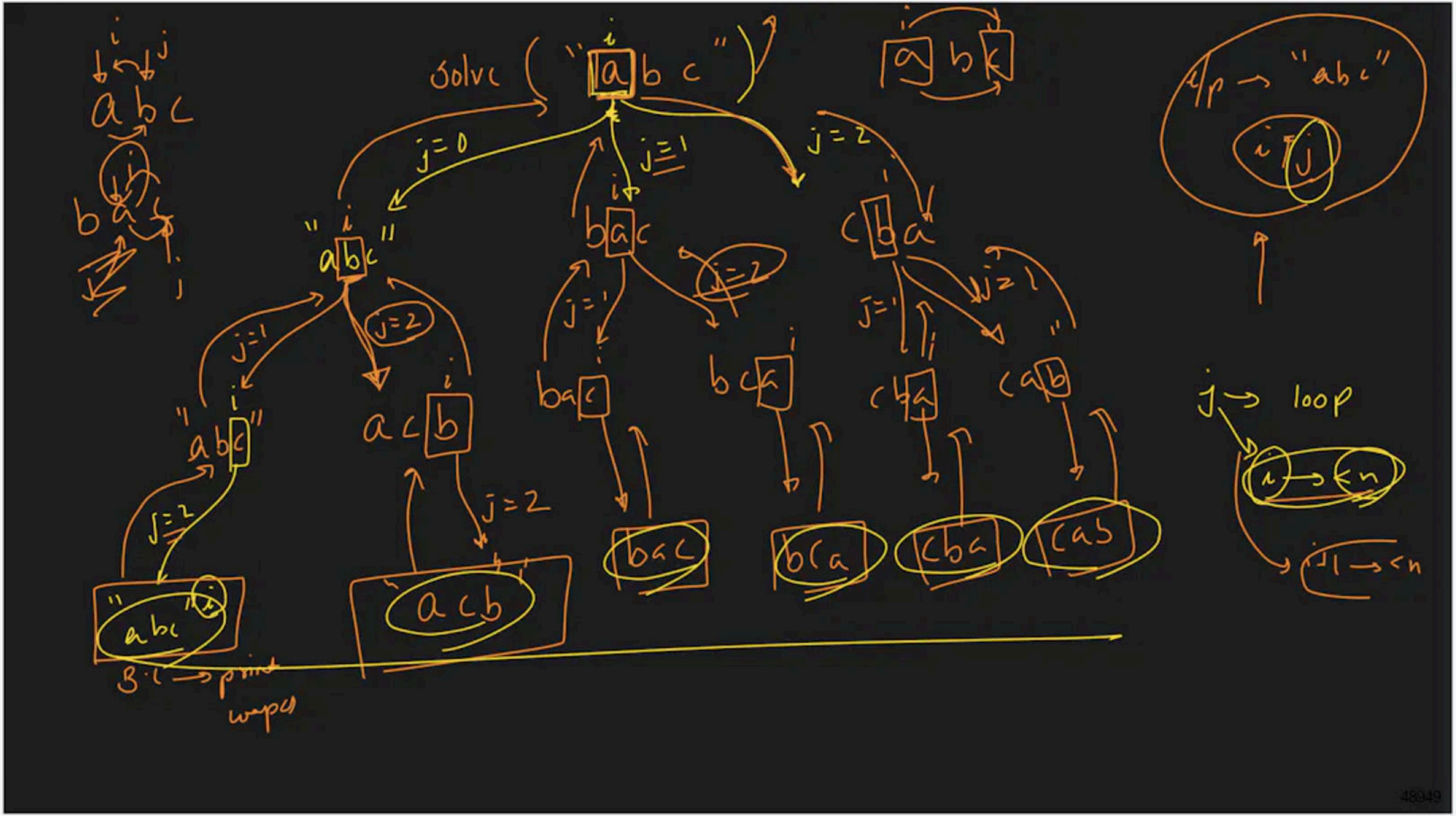

Str → "a b c"
3 2 1
6
perm
a
b
c
a b
a c
b a
b c
c a
c b
a b c
a c b
b a c
b c a
c a b
c b a


str = "xy"
n = 2
T.P → 2! = 2 × 1 = 2
2 * 1
→ 2 ways
x
y
x | y
y | x

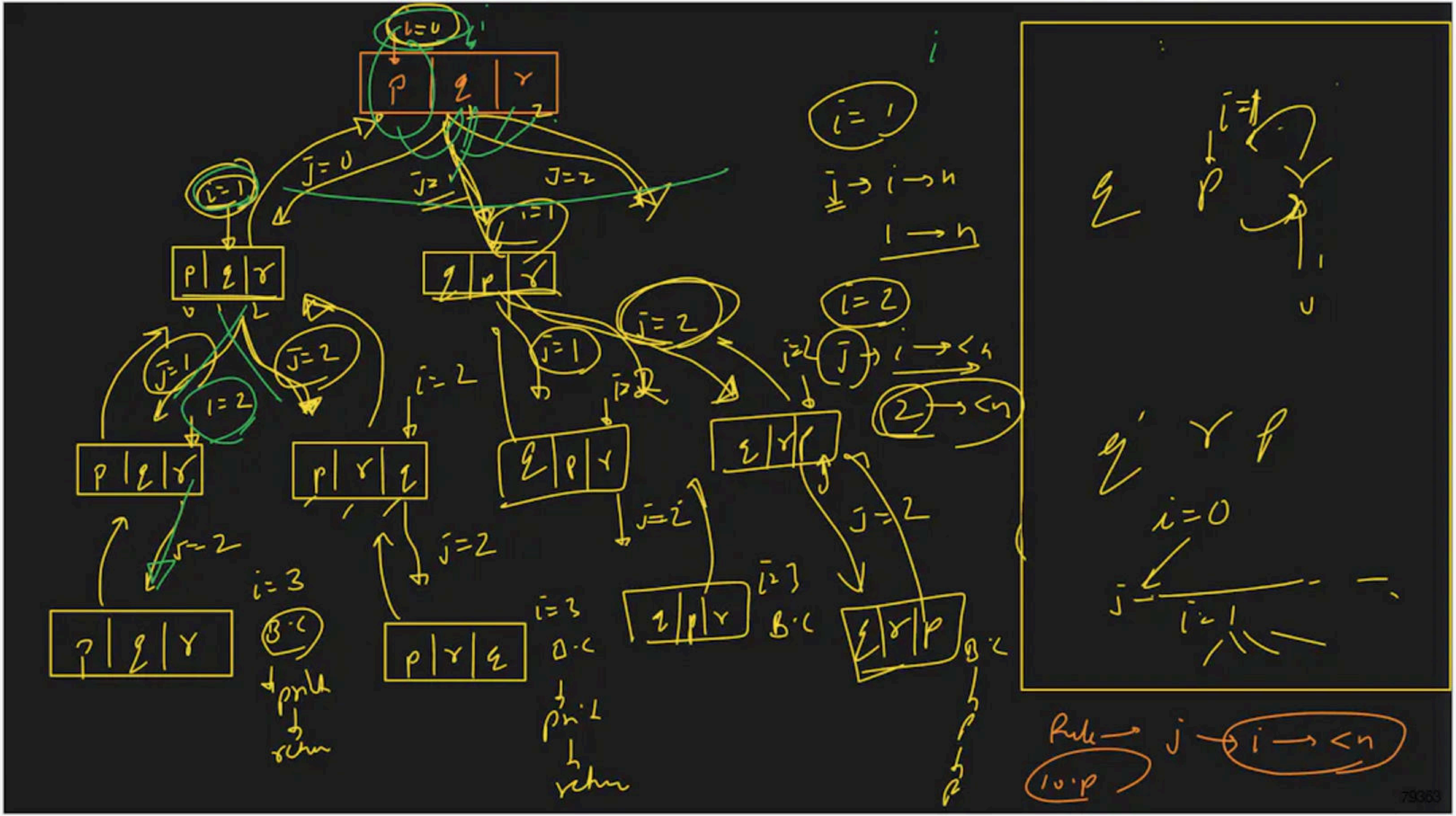

Str
n=4
4!
24
perm
4
3
2
1
24
p


a
b
c
swap

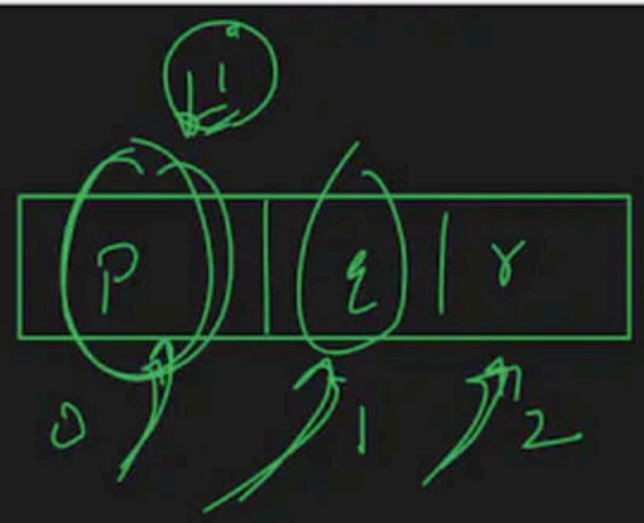

b
a
c

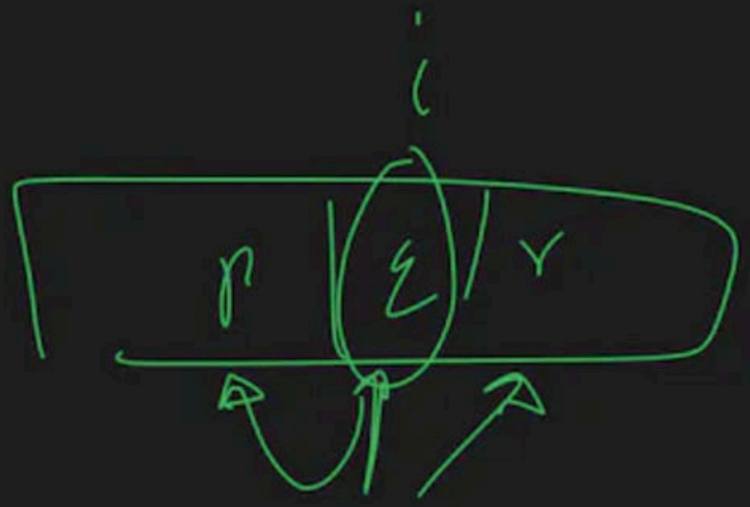


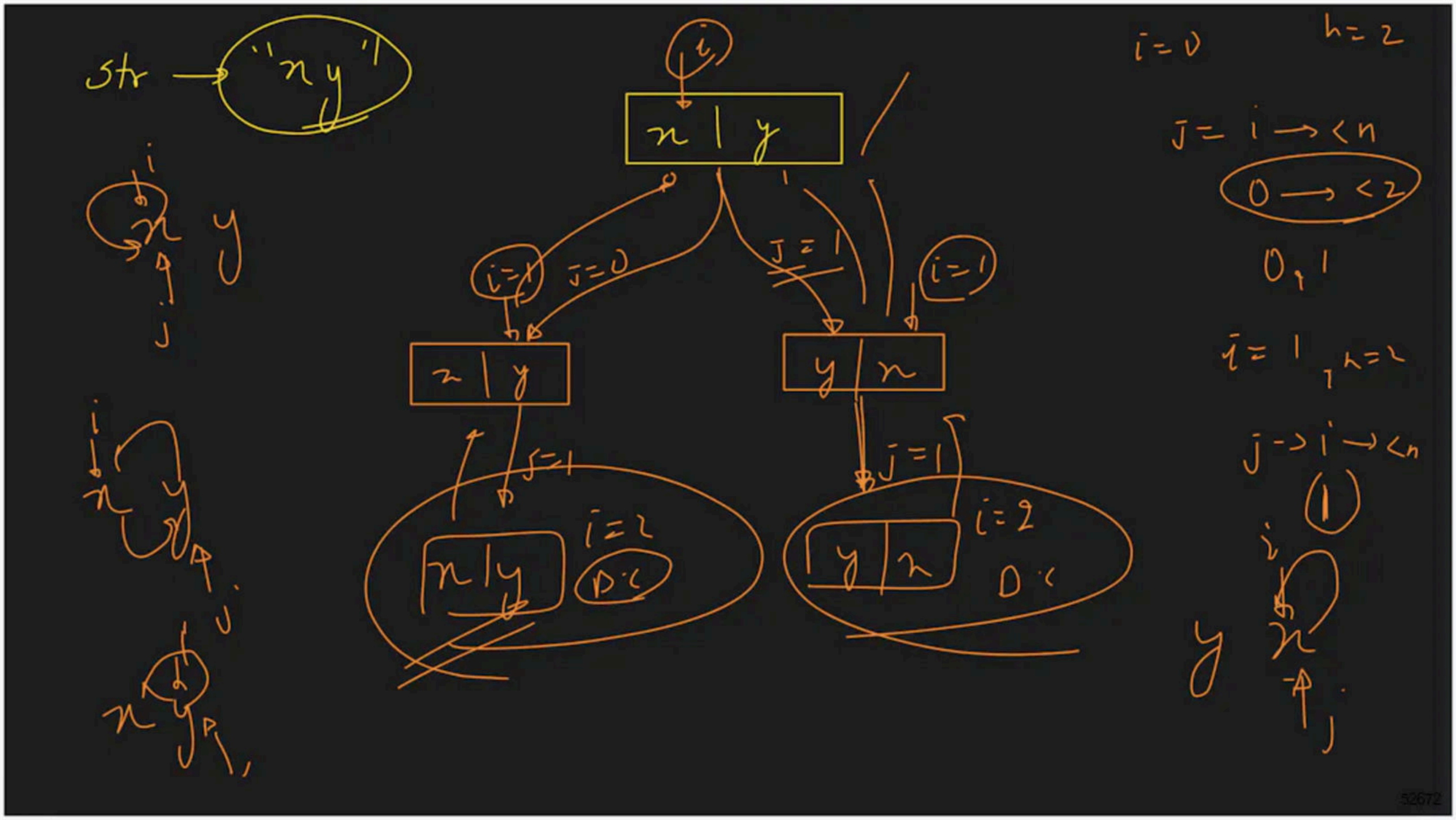
solve ("abc")
j=0
j=1
j=2
abc
bac
cba
acb
bca
cab
ip → "abc"
j → loop
i → <n
i+1 → <n
B.C → print

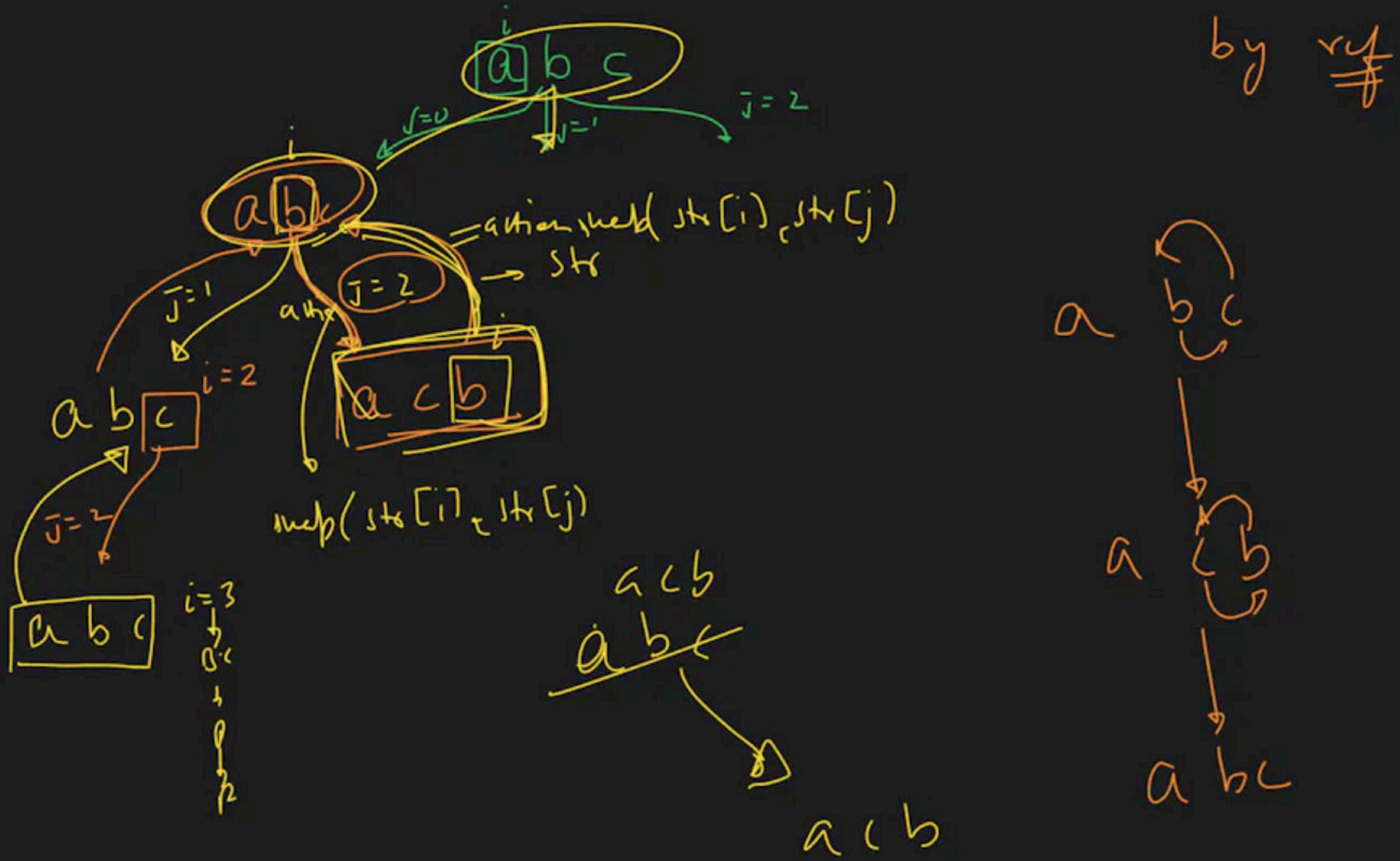


i = 0
p | q | r
j = 0
j = 1
j = 2
i = 1
i = 2
i = 3
B·C
print
return
i → i → n
1 → n
i → < n
2 → < n
Rule → j → i → < n

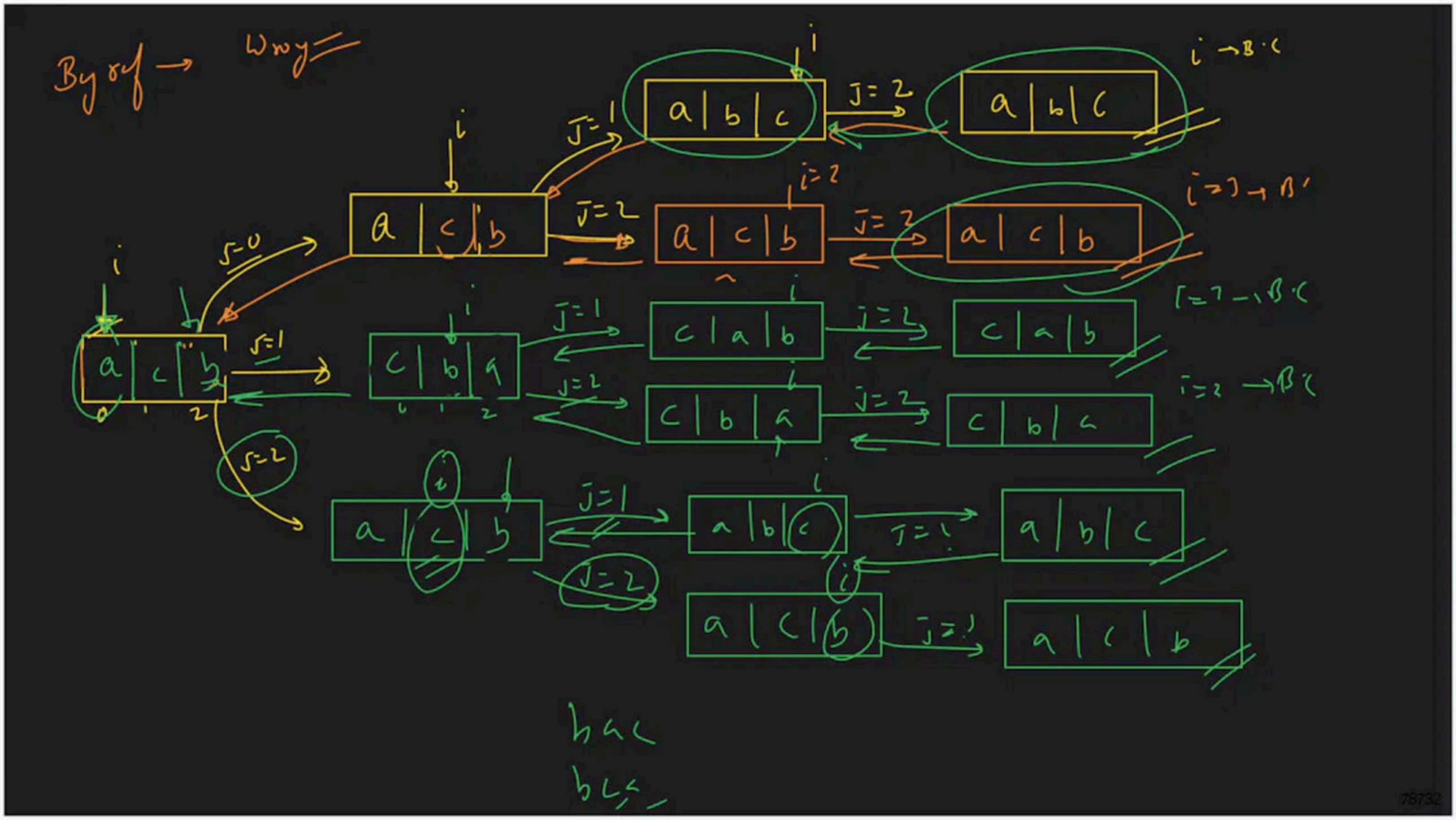

p
ع
γ
0
1
2

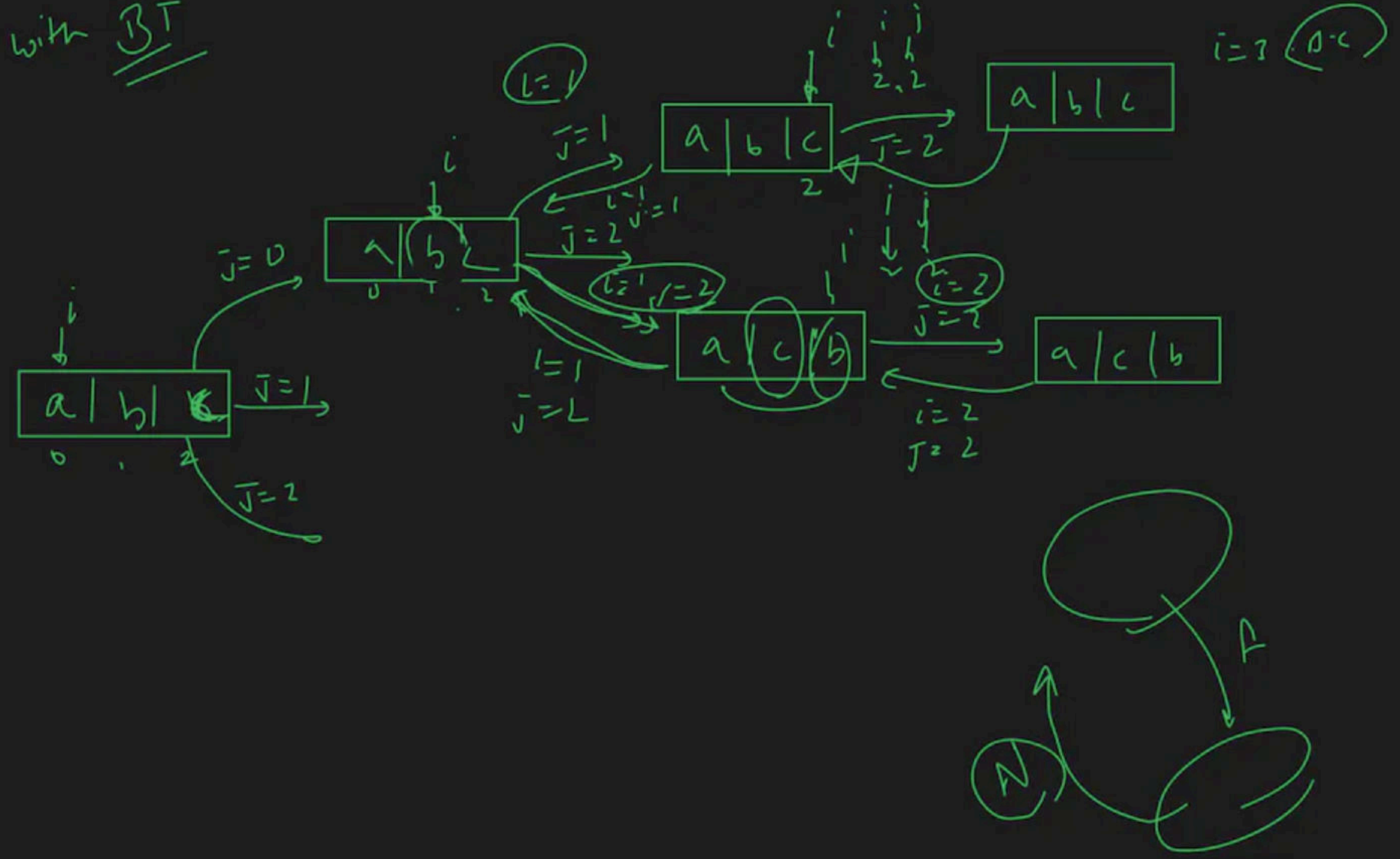

p
ع
γ


str
"xy"
i
x | y
i=0
j=0
j=1
i=1
z | y
y | x
j=1
i=2
x | y
y | x
i=0
h=2
j = i → <n
0 → <2
0, 1
i=1
h=2
j → i → <n
1
i
x
y
j



a b c
j = 2
j = 1
j = 2
str
a c b
i = 2
a b c
j = 2
a b c
i = 3
a c b
a b c
a c b
a b c
a c b
a b c
by


By def →
Way =
a | b | c
a | c | b
c | a | b
c | b | a
i
j = 1
j = 2
i → B · C
i = 1 → B'
i = 2 → B' C
i = 2 → B C
s = 0
s = 1
s = 2
bac
bca

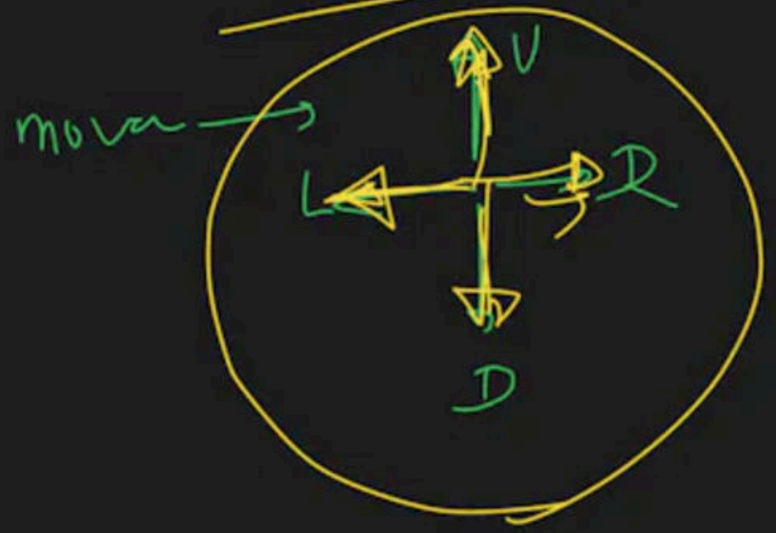

with BT
a | b | c
j = 0
j = 1
j = 2
i = 1
a | c | b
i = 2
j = 2
i = 3
f
N

Rat in a Maze

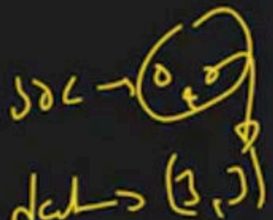
print all possible ways

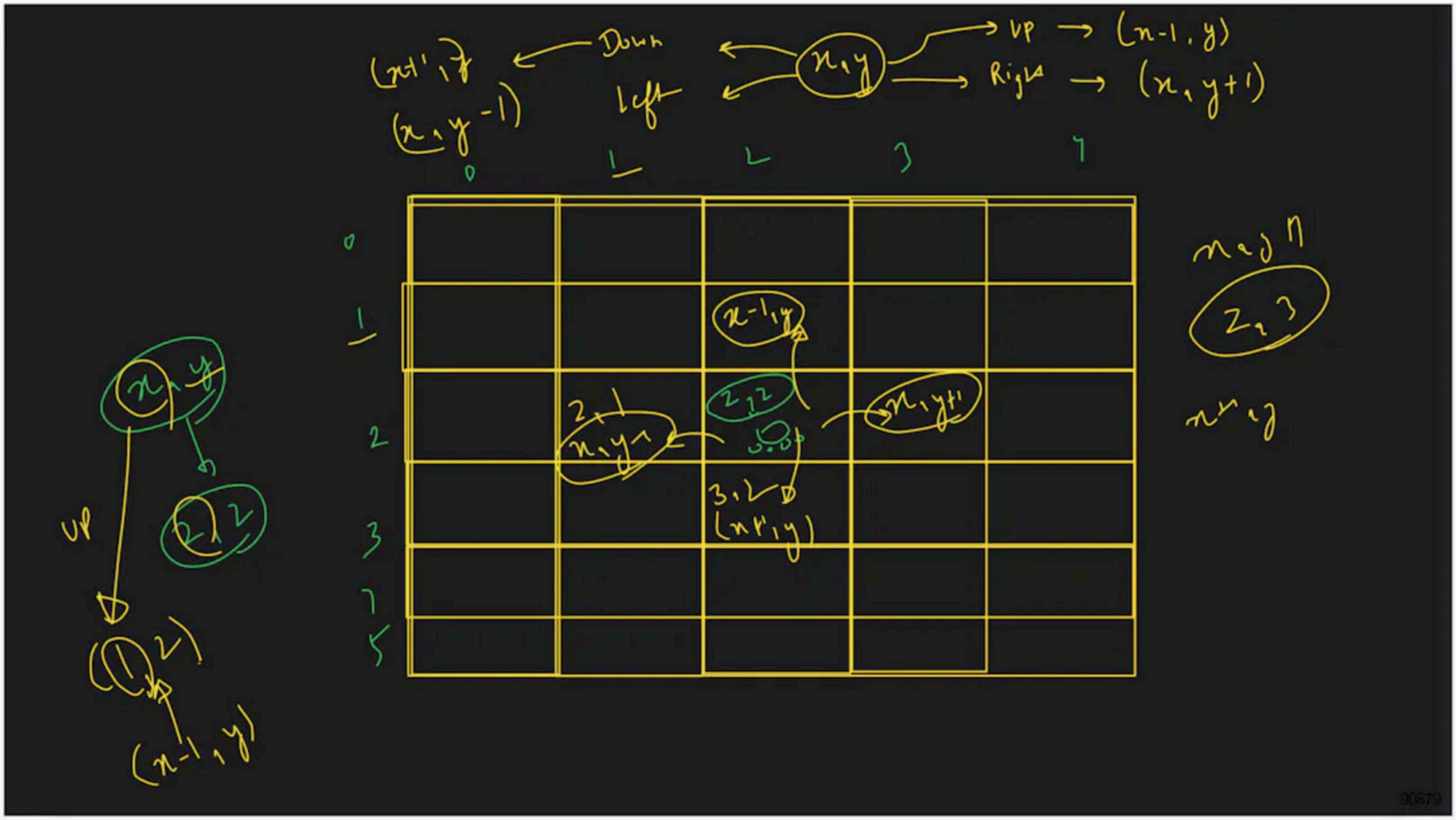
0 → closed path

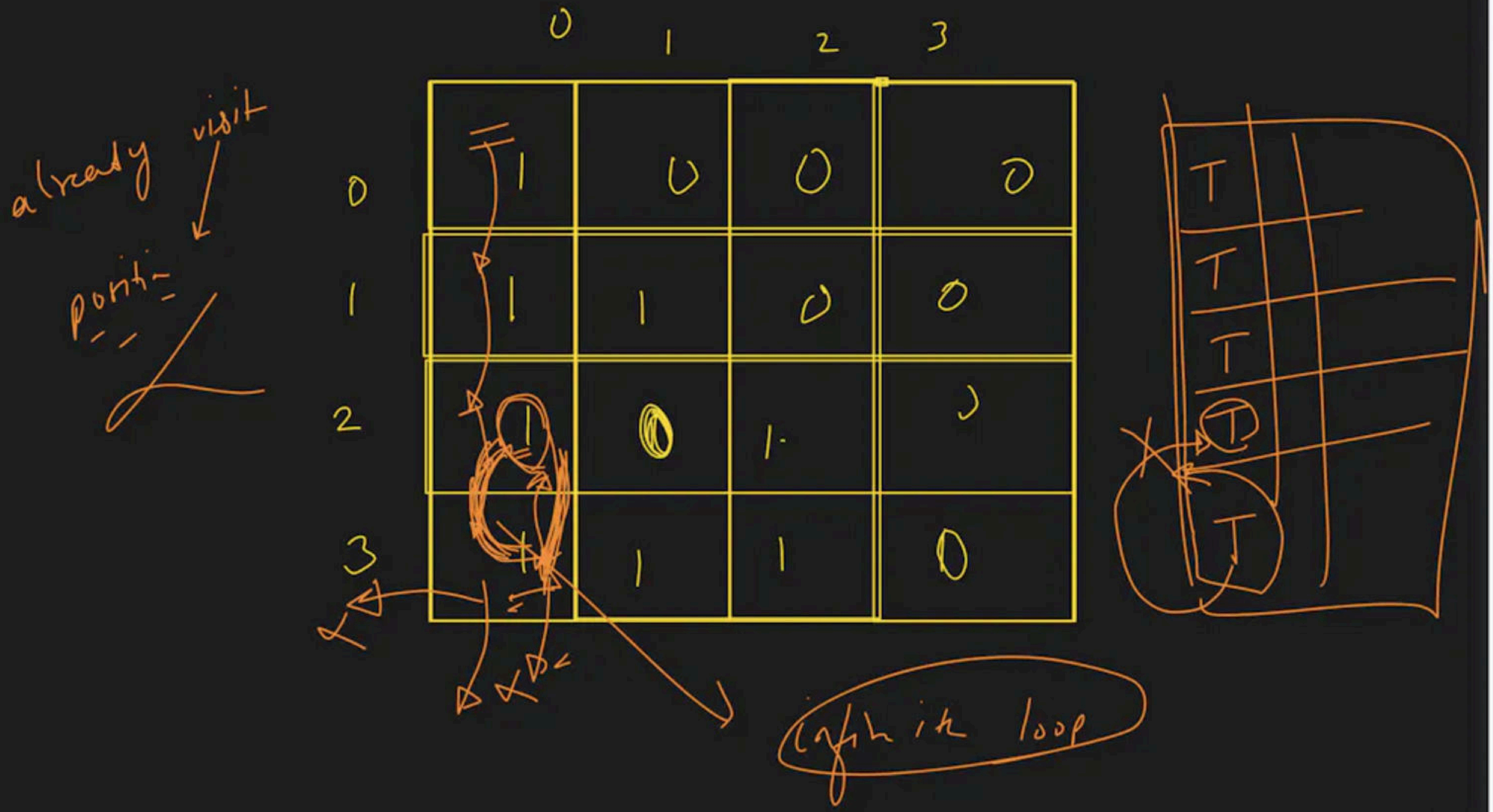
1 → Open path

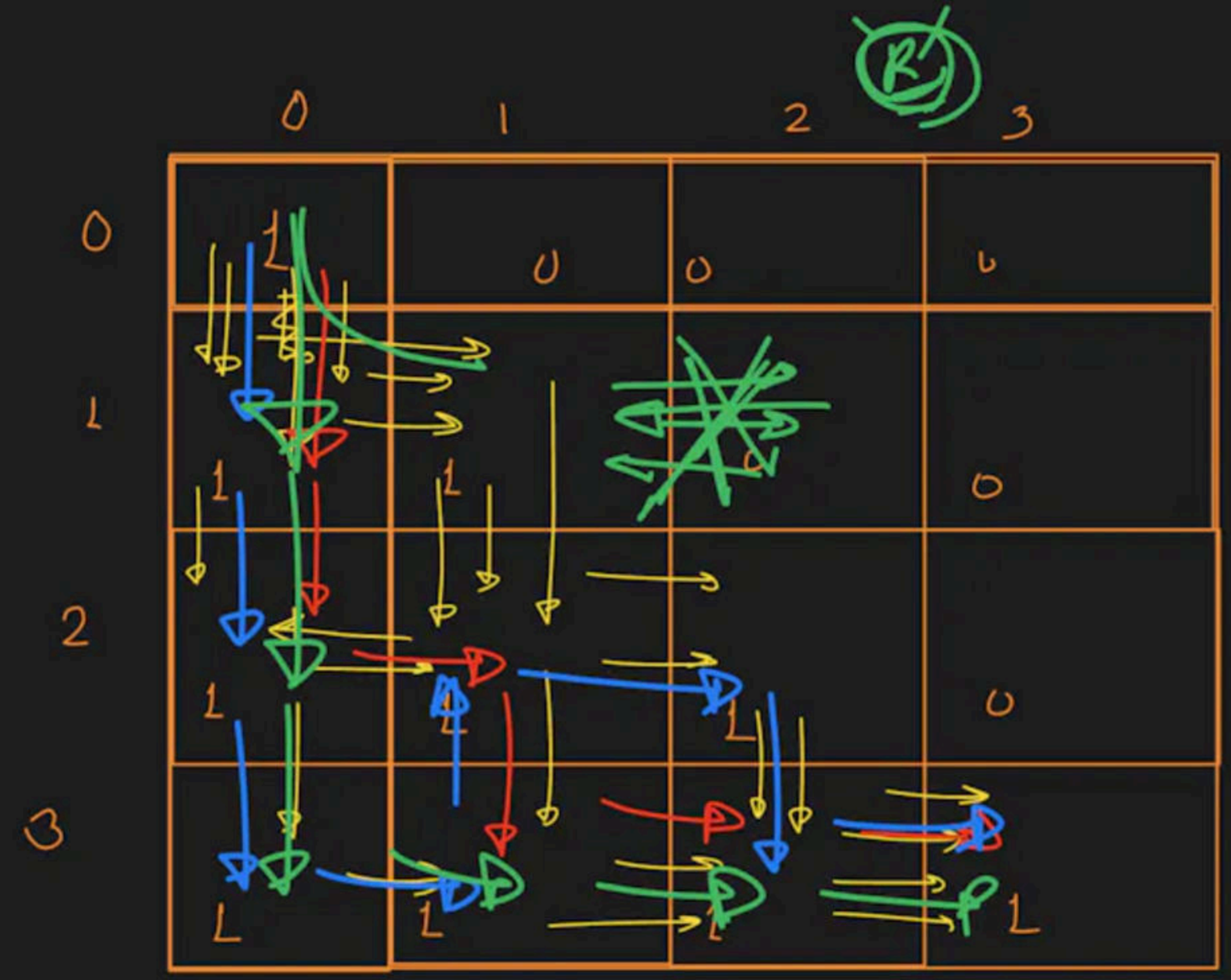


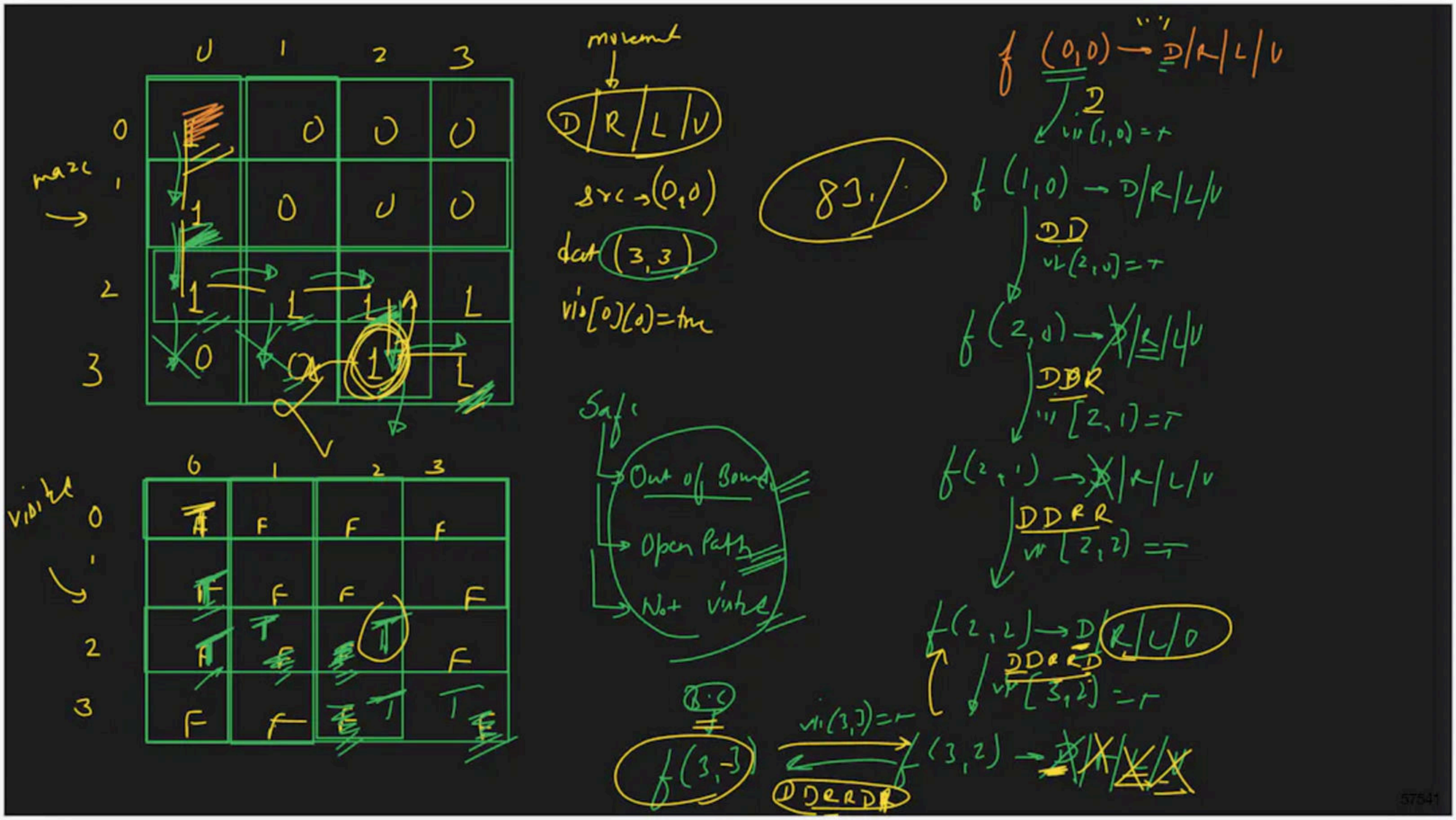
	0	1	2	3
0	Sundar Rat 1	1	0	0
1	1	1	0	1
2	1	1	1	0
3	1	1	1	Sundar (1,1)

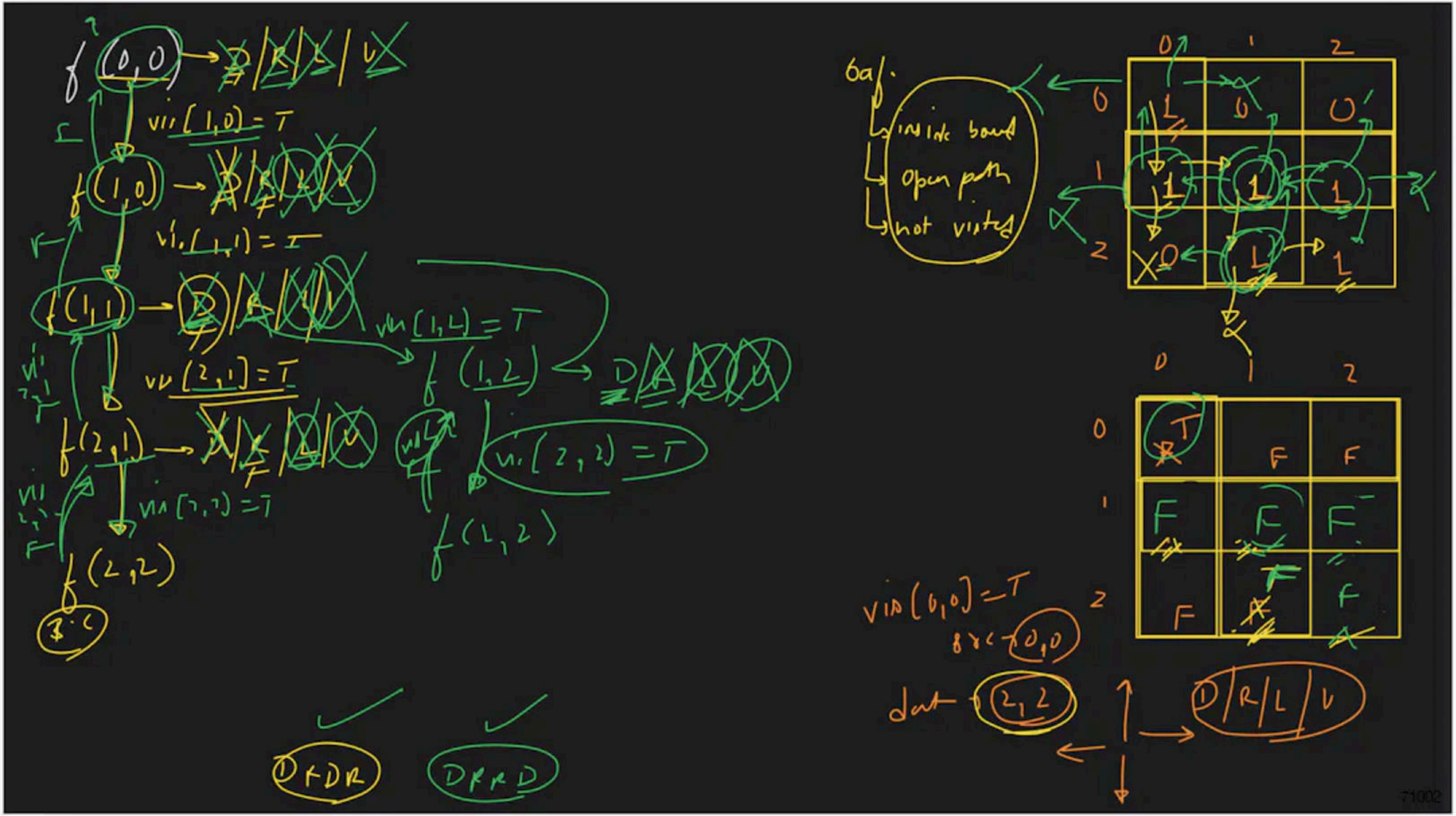


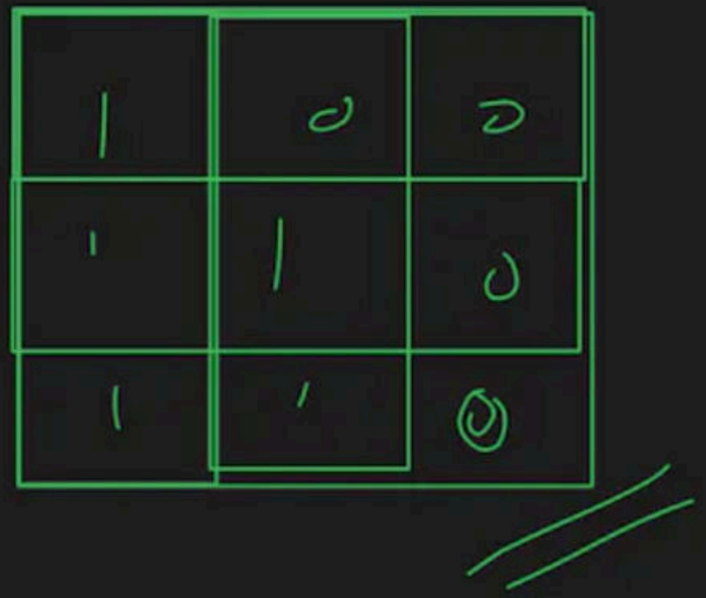

Down
Left
Up → (x-1, y)
Right → (x, y+1)
0
1
2
3
4
0
1
2
3
4
5
Up

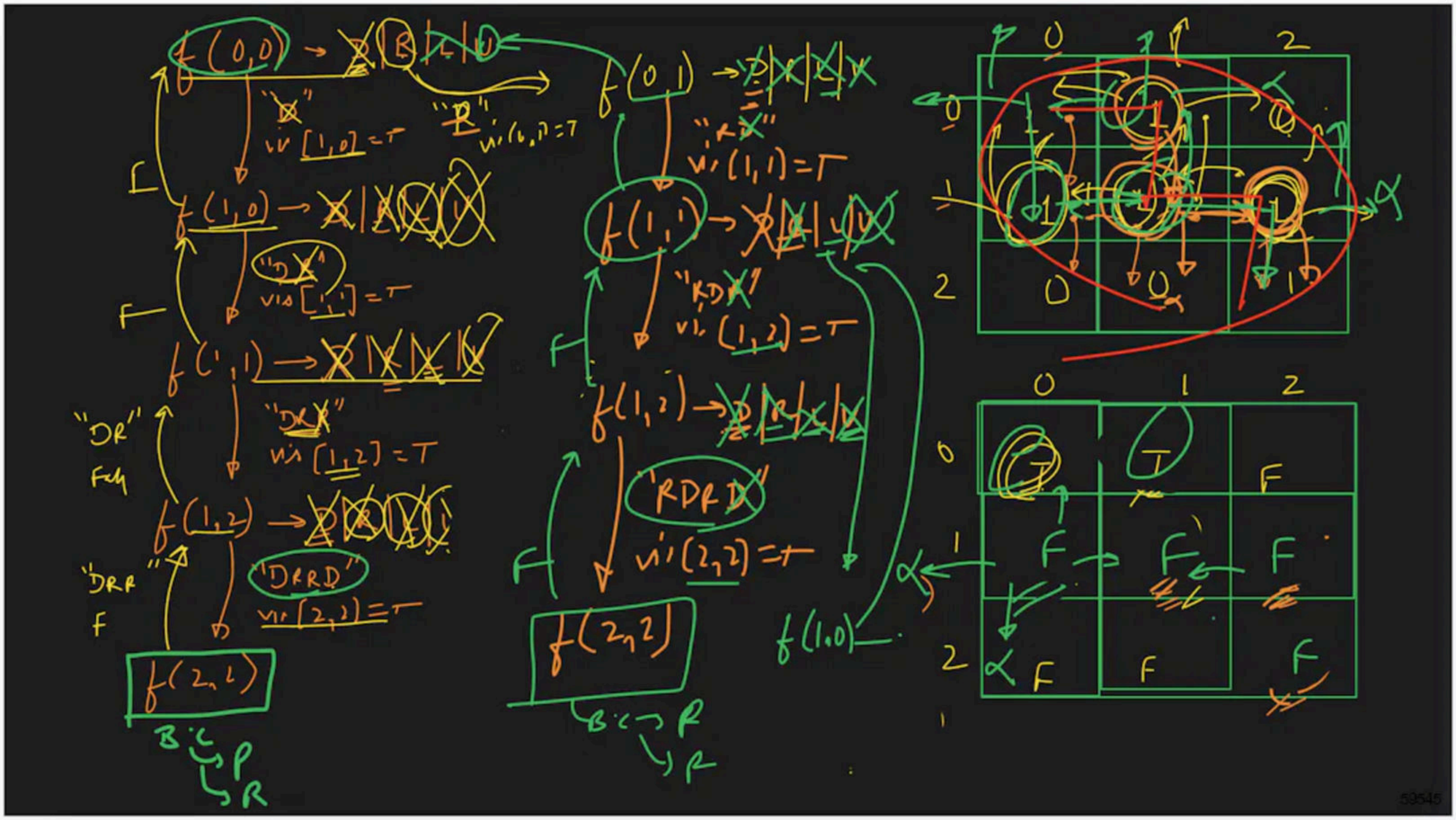

already visit
infinite loop

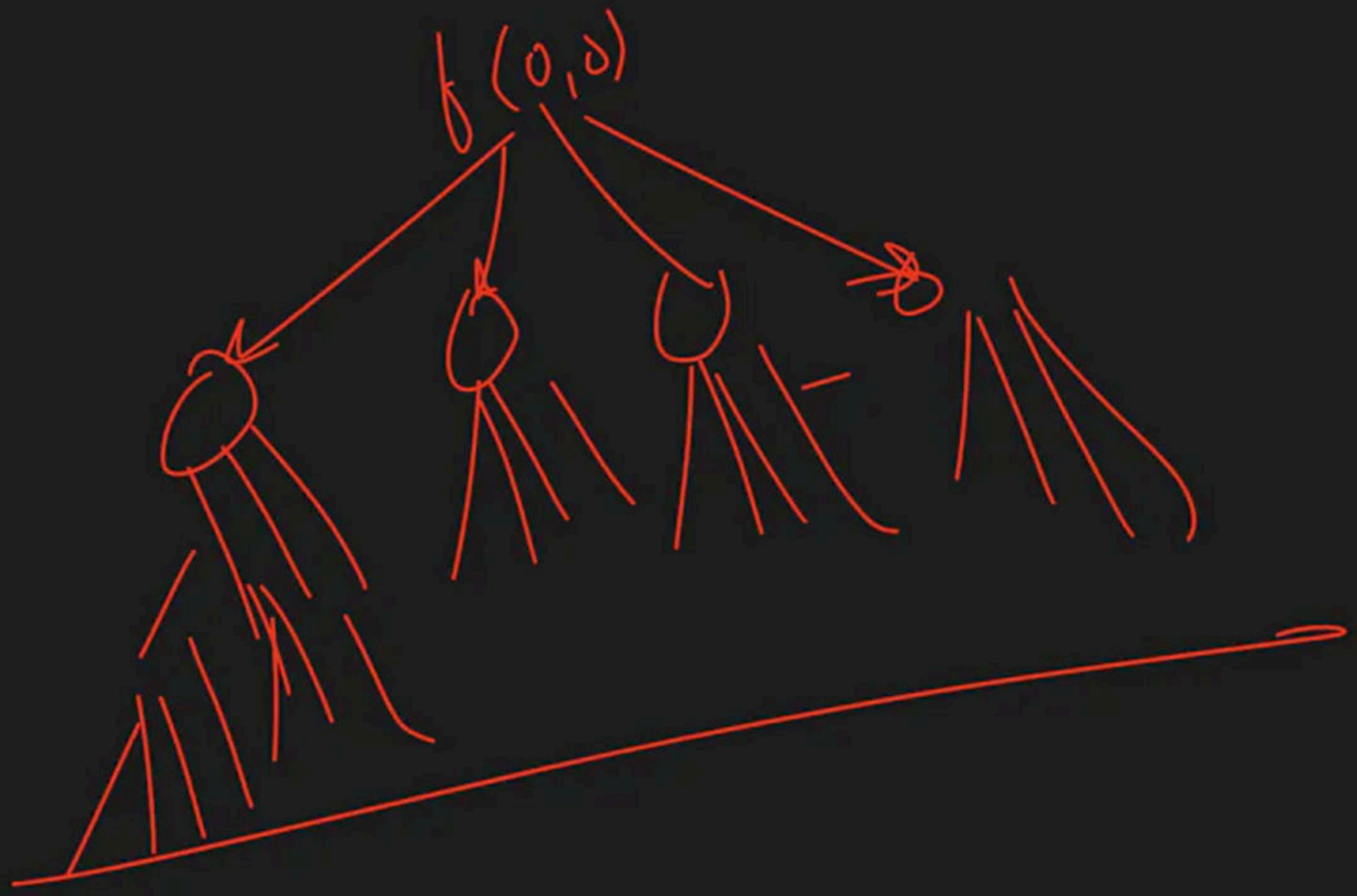

0
1
2
3
0
1
2
3


maze
visited
movement
D R L U
src → (0,0)
dest (3,3)
vis[0][0] = true
Safe
Out of Bound
Open Path
Not visited
83.
f(0,0) → D/R/L/U
D
vis[1,0] = T
f(1,0) → D/R/L/U
DD
vis[2,0] = T
f(2,0) → X/R/L/U
DDR
vis[2,1] = T
f(2,1) → X/R/L/U
DDRR
vis[2,2] = T
f(2,2) → D/R/L/U
DDRRD
vis[3,2] = T
(3,2) → X/X/X/X
f(3,3)
vis(3,3) = T
DDRRDR
57541


f(0,0)
vis(1,0) = T
f(1,0)
vis(1,1) = T
f(1,1)
vis(1,2) = T
f(1,2)
vis(2,1) = T
f(2,1)
vis(2,2) = T
f(2,2)
Goal
inside bound
open path
not visited
vis(0,0) = T
src (0,0)
dst (2,2)
D/R/L/U
DRDR
DRRD




f(0,0)
f(0,1)
f(1,0)
f(1,1)
f(1,2)
f(2,1)
f(2,2)
f(1,0)
"D"
"R"
"DR"
"DRR"
"RDR"
"RDRD"
vis[1,0]=T
vis[0,1]=T
vis[1,1]=T
vis[1,2]=T
vis[2,1]=T
vis[2,2]=T
F
B↔L
B↔R
L↔R
0
1
2
T
F
α



=> Debugging

2 min Break

⇒ Vector of digit to number

⇒ Pynamic mem allocation

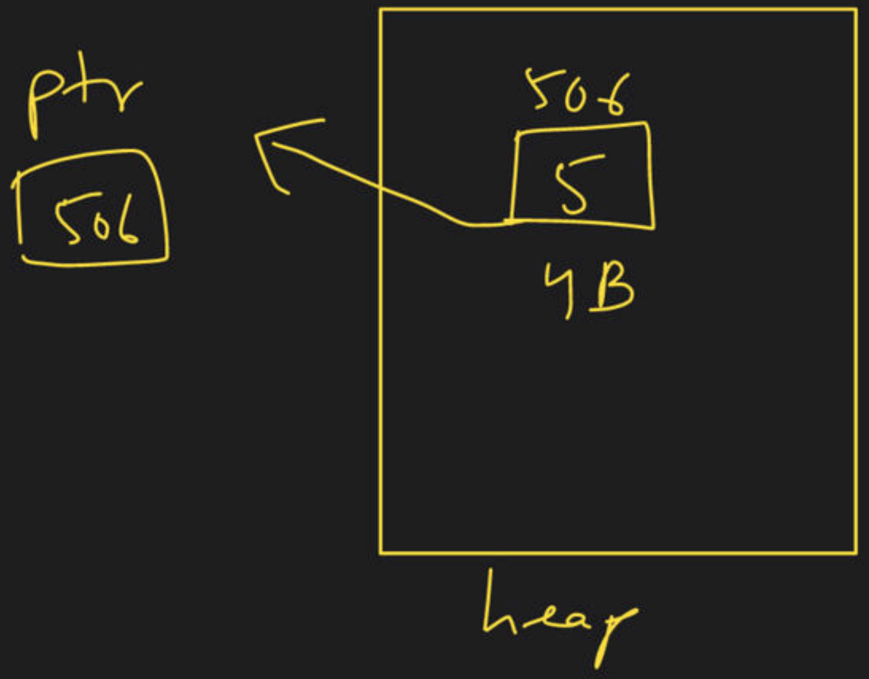
4 1 → w/o new
- ① integer → allocate heap

5 2 w/o new keyword
- ② array → heap

6 3 " " "
- ③ 2D array → heap.

⇒ int *ptr = new int(5);

address int*



Void* ptr = malloc(4)

int* intptr = (int*) ptr;

int *ptr = malloc(4);

int *ptr = (int*) malloc(4);

*intptr = 5;

① allocate 1 int

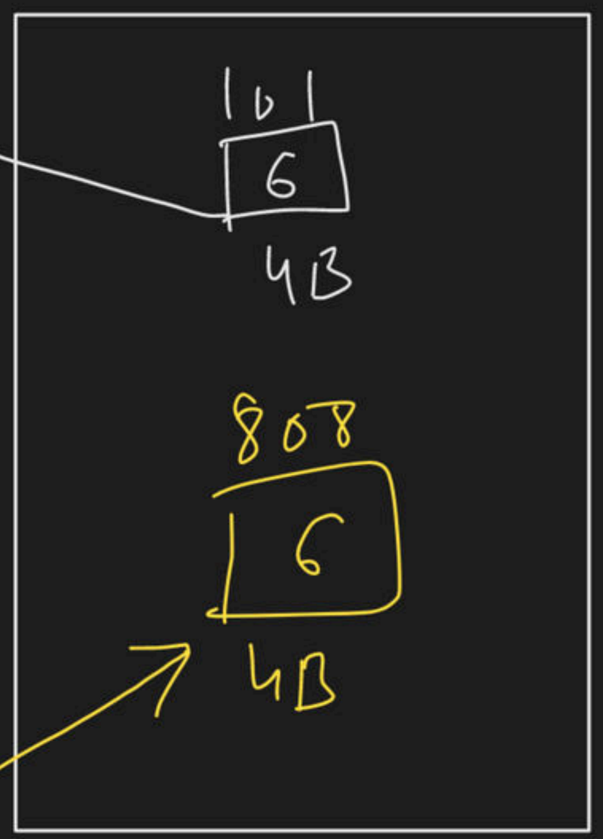
① `int *ptr = new int(6);`

ptr → [101] → *ptr = 6

stack 606

② `malloc(no. of Bytes)` ⇒ `int *ptr = (int*) malloc(4);`

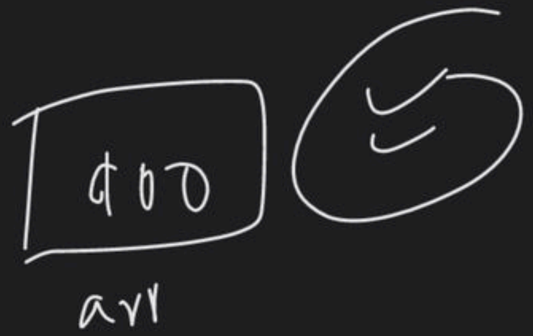
`*ptr = 6;`



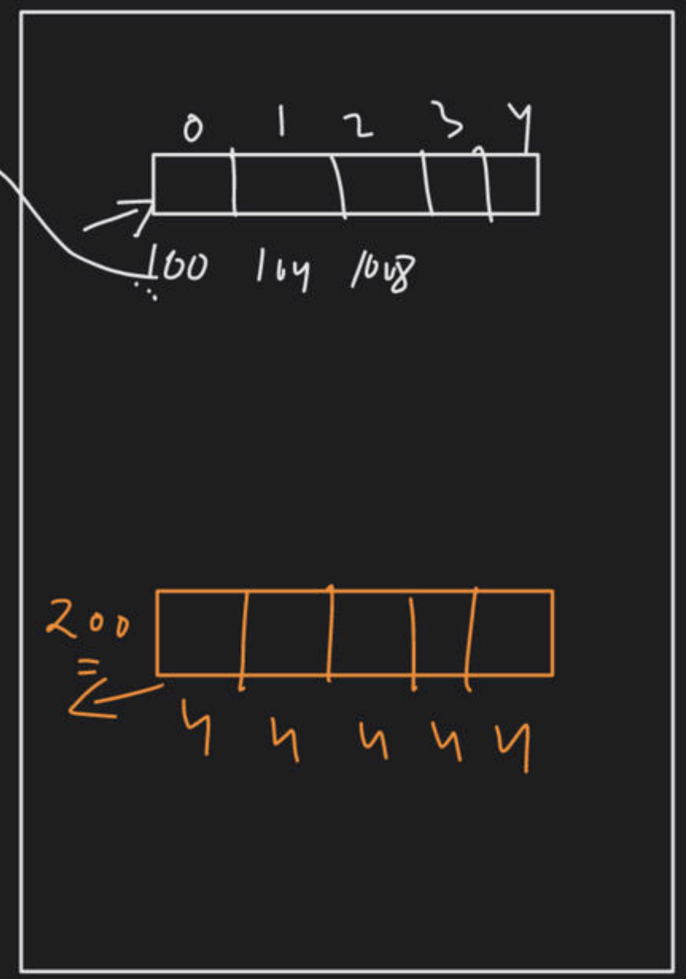
1D allocation

5 length array.

① `int *arr = new int[5];`



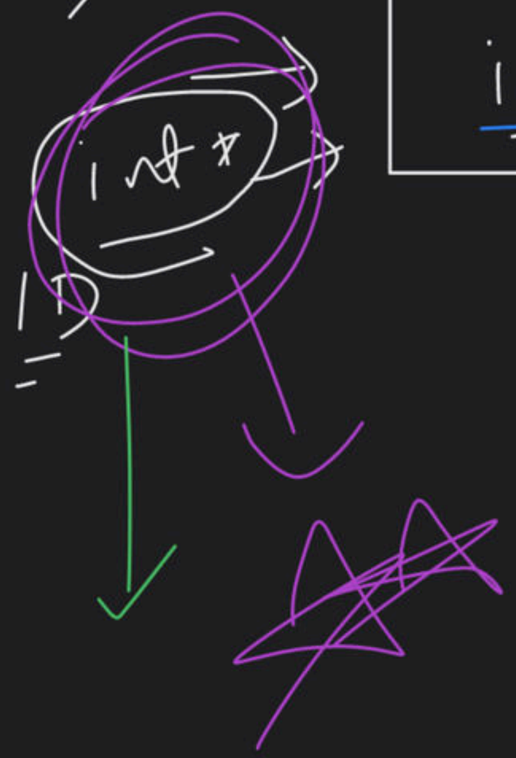
1D array allocation

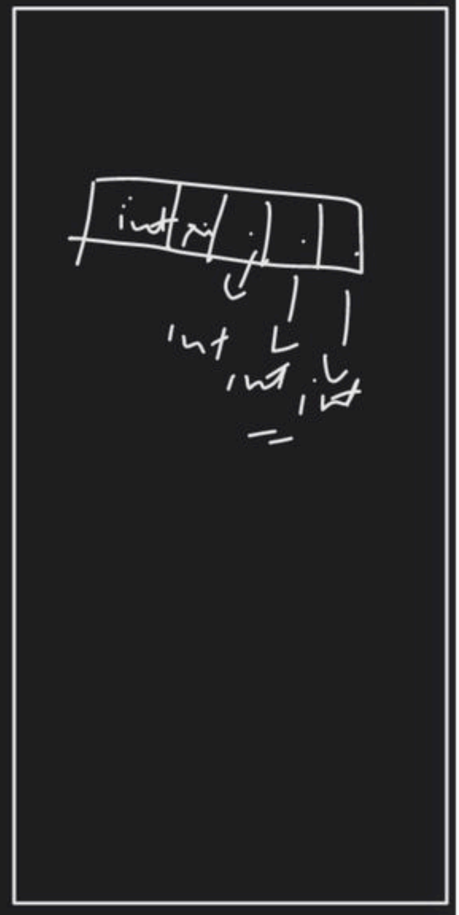


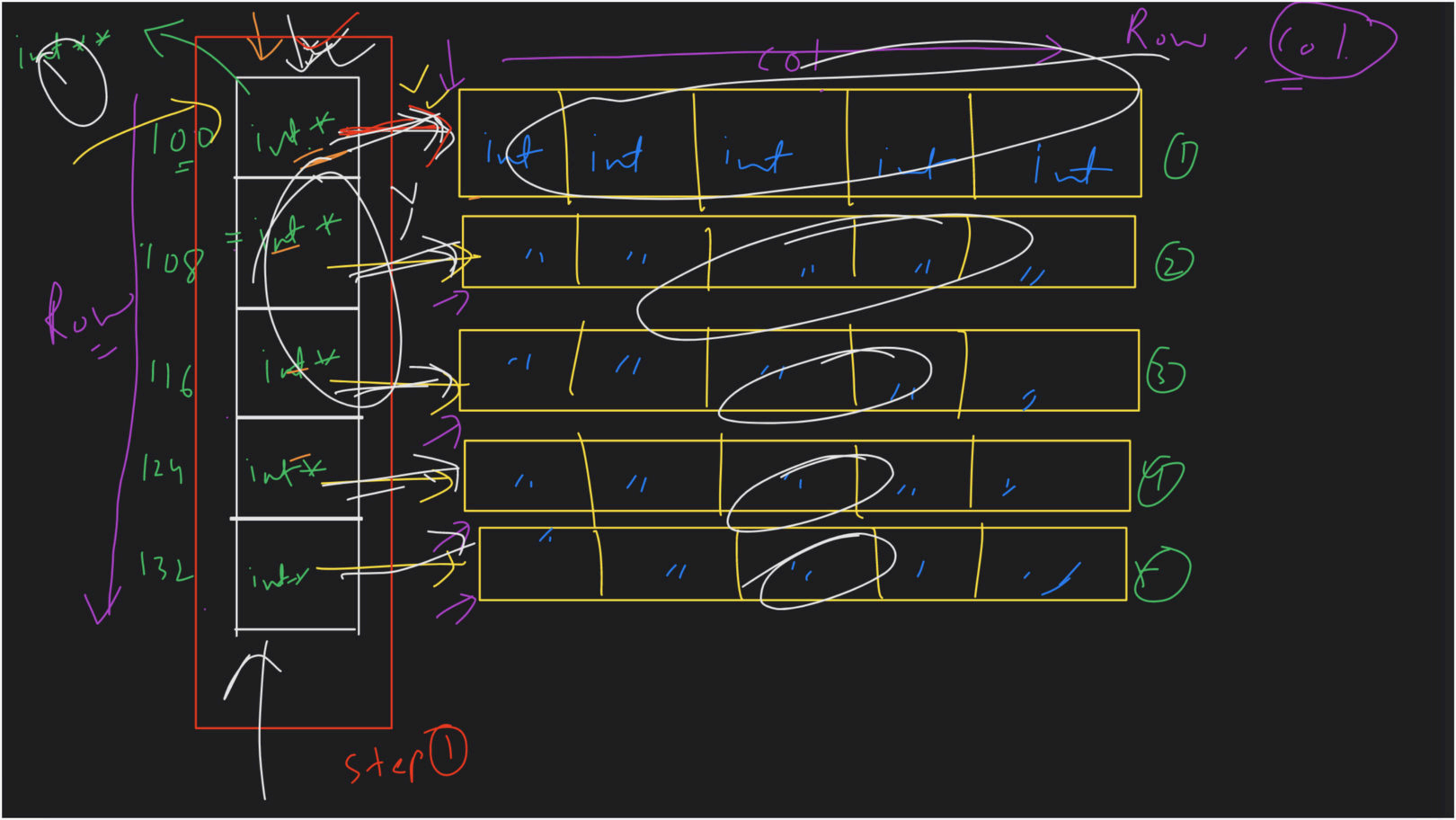
② `int *arr = (int*) malloc(20);`

2D heap allocation → Bohot saari 1D array

=>





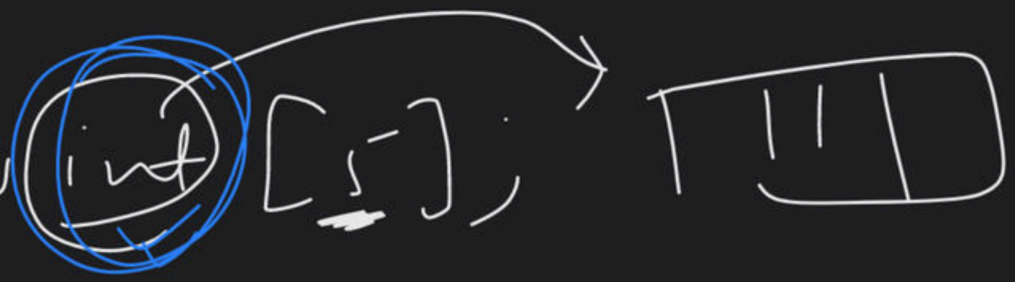

int**
100
108
116
124
132
int*
int*
int*
int*
int*
Row
Col
int int int int int
step①

2D allocation → 2 step

⇒ ① int* type ki array allocate Rows

⇒

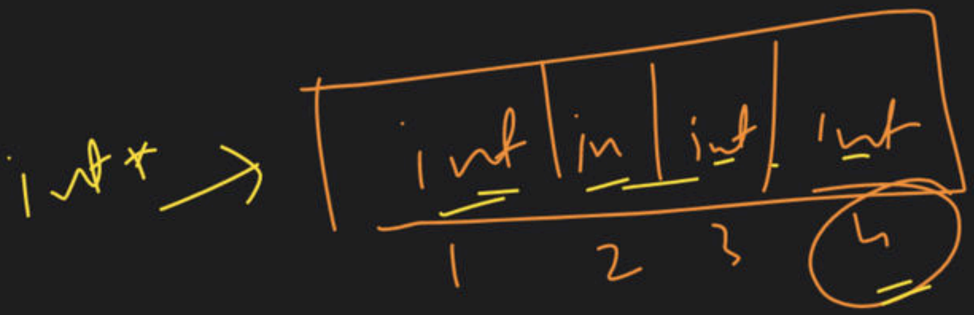
```
int * ptr = new int [5];
```



```
int ** ptr = new int* [Rows]
```

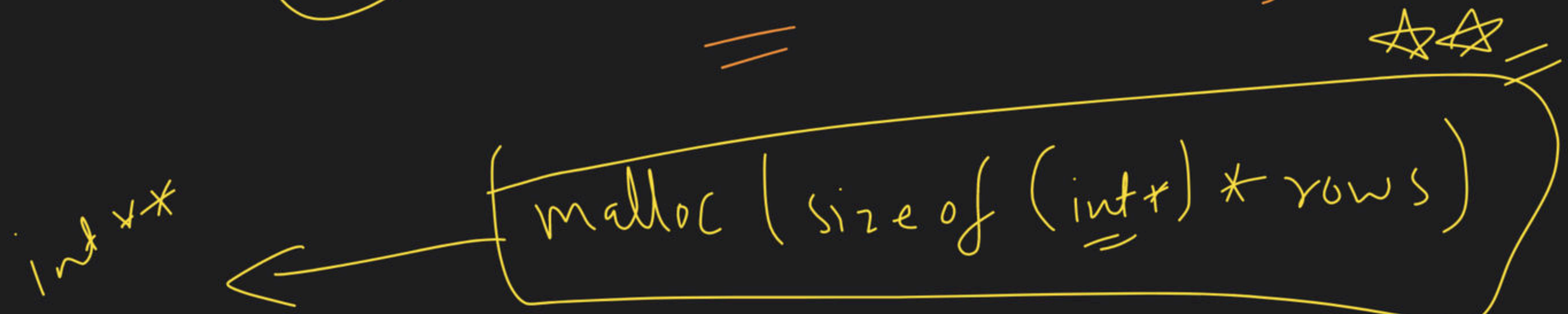
② Loop till all the rows allocated.

=) Malloc 2D



malloc (sizeof(int) * 4)

=



⇒ Next ⇒ ⇒

Mon → 9 → 11
Tues 9 - 11
Wed 9 - 11

OOPs Live class

Lab today

⇒ Code

① 2D allocation
② Merge Sort