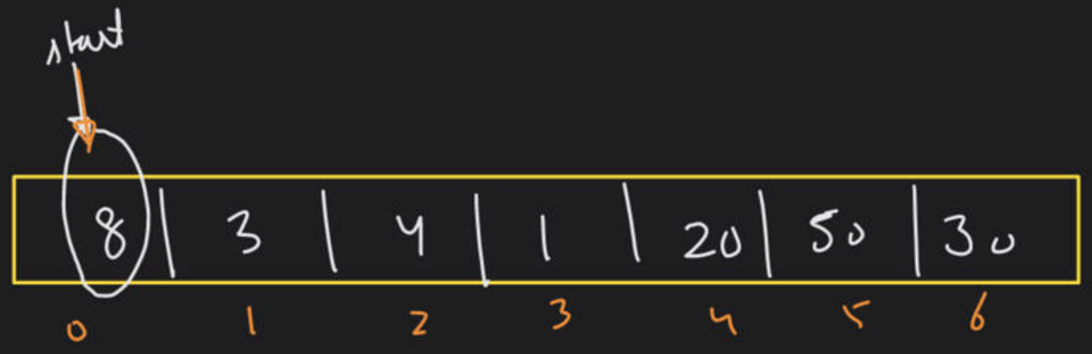
DnC Level-2 [Join Here]

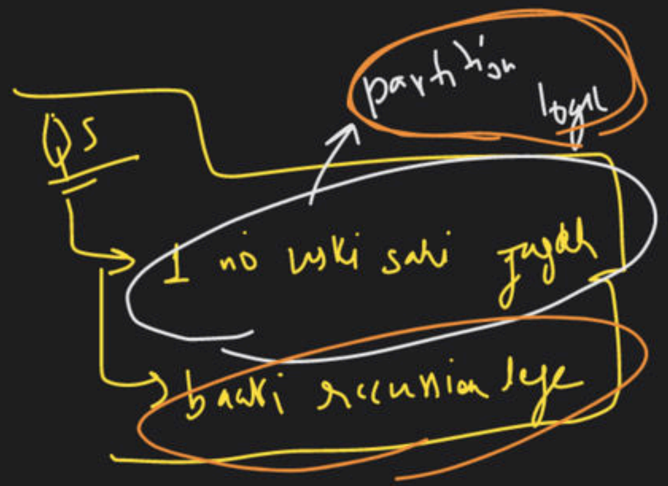
Special class

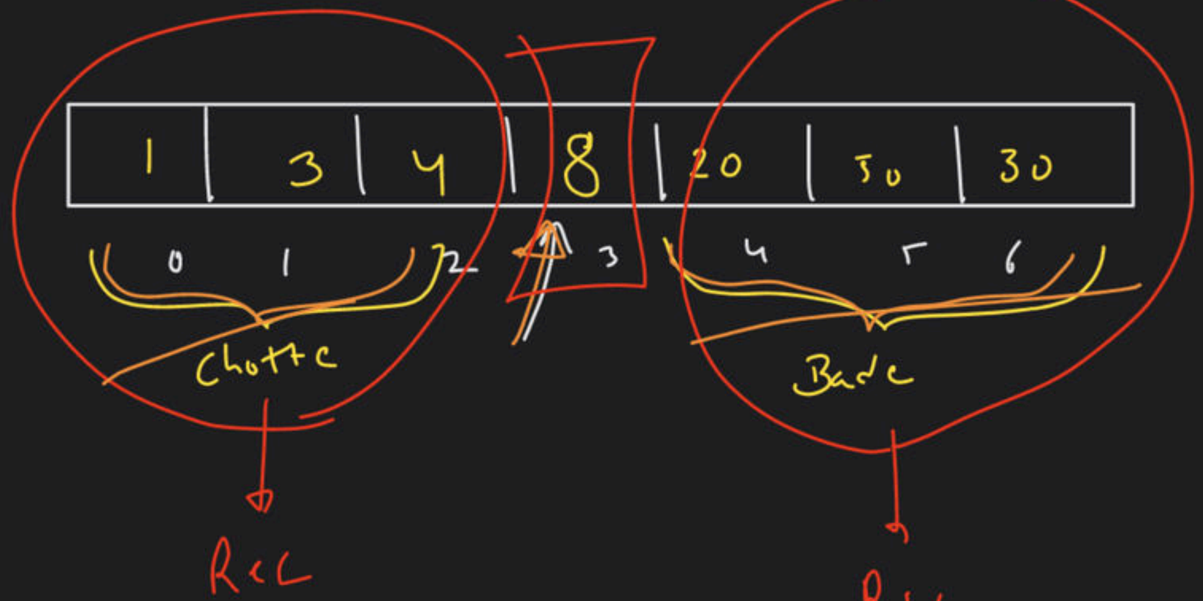
Love Babbar • Mar 19, 2023

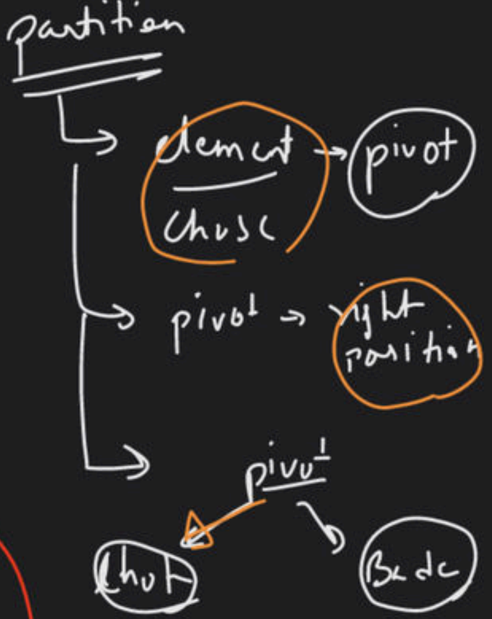
Quick Sort

i/p → array →

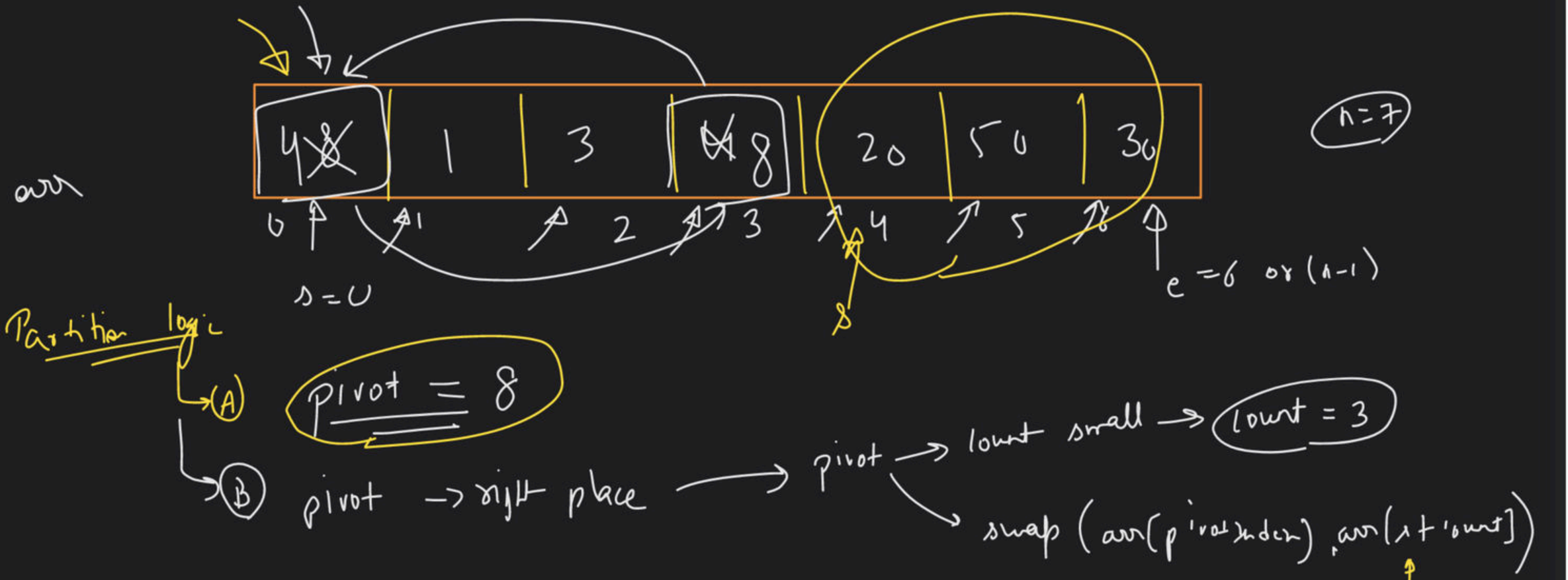


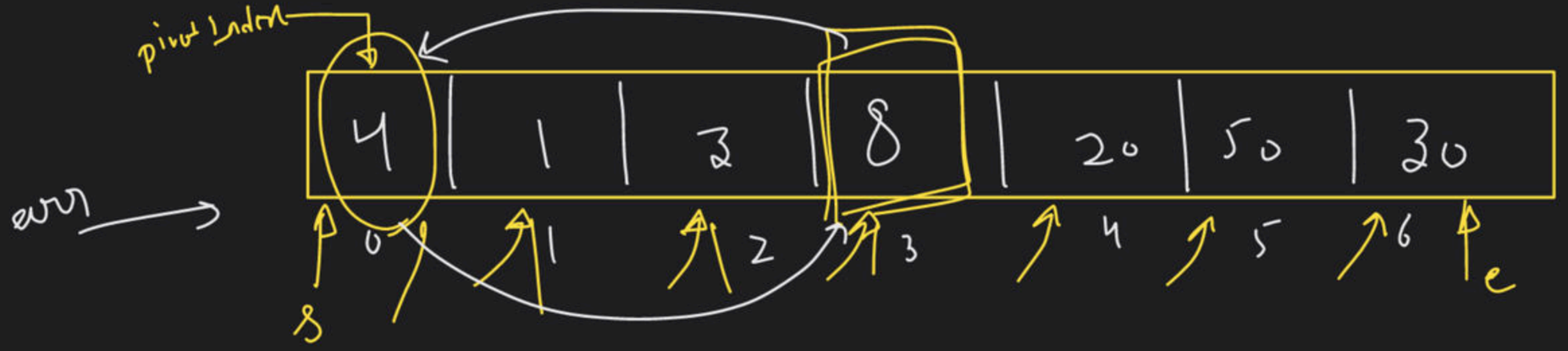






QS → Partition logic
→ Recursive calls


arr
4 8
1
3
8
20
5 6
30
n = 7
0
1
2
3
4
5
6
s = 0
e = 6 or (n-1)
Partition logic
(A)
pivot = 8
(B)
pivot -> right place
pivot -> lount small -> lount = 3
swap (arr[pivotIndex], arr[lt + lount])



Partition logic

(A) pivot choose => pivotIndex = s

(B) pivot -> सही Position पर place karo

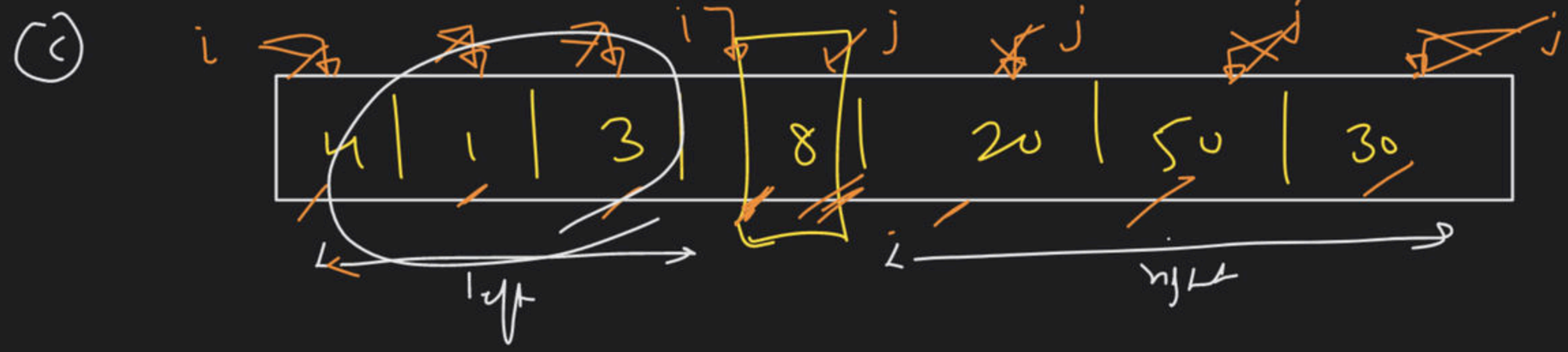
(A) count का < pivot Element

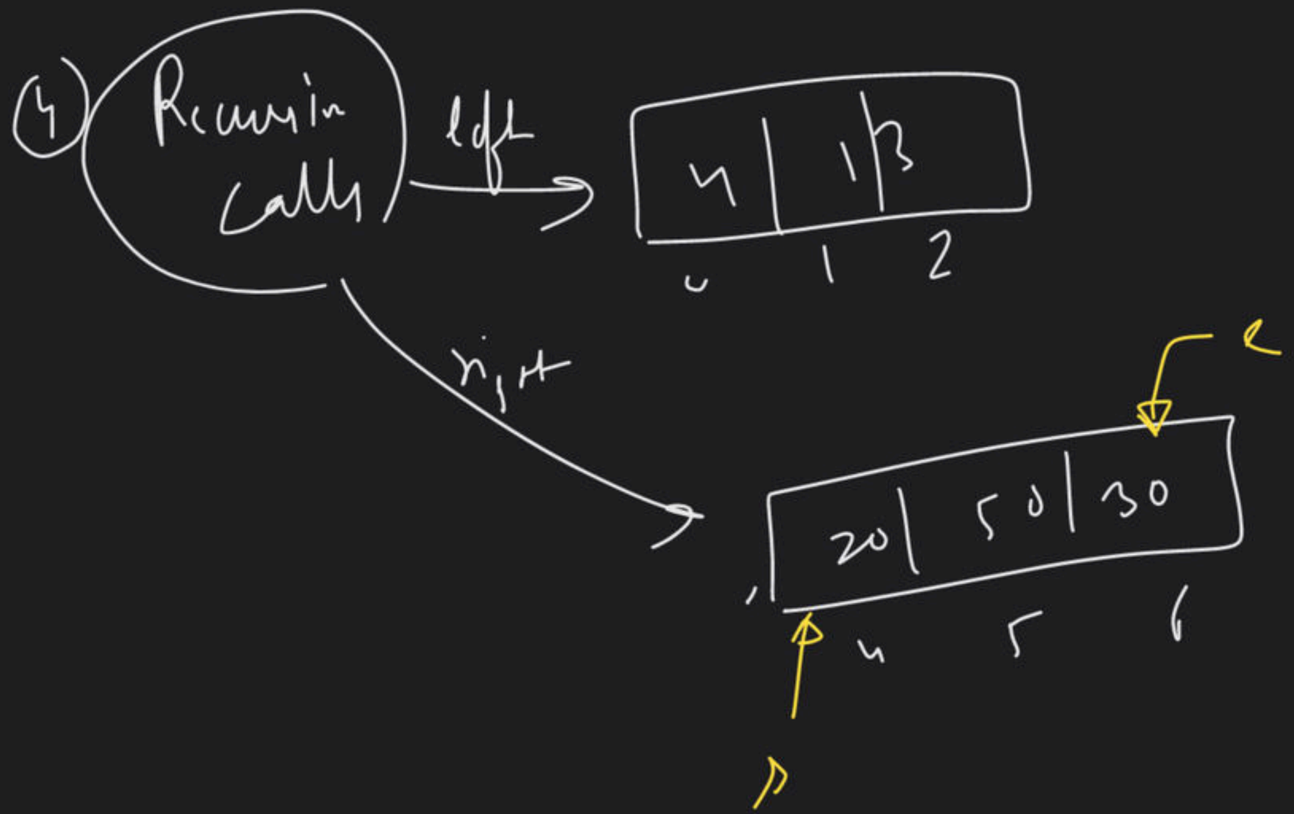
count = 0 1 2 (3) →

8 को 4th element = element hona chahiye

(B) swap (arr[pivotIndex], arr[s+count])

pivot Element


(c)
i
i
j
j
j
j
4
1
3
8
20
50
30
left
right


(4)
Recursive calls
left
4
1
3
0
1
2
right
r
20
50
30
4
5
6
p

Q.S

2 min Break

Partitioning Logic

↳ for place pivot element at its right position →

Recursive logic

sort

sort

Rec

Rec

P.L

(A) Pivot choose

(B) Pivot → right position place

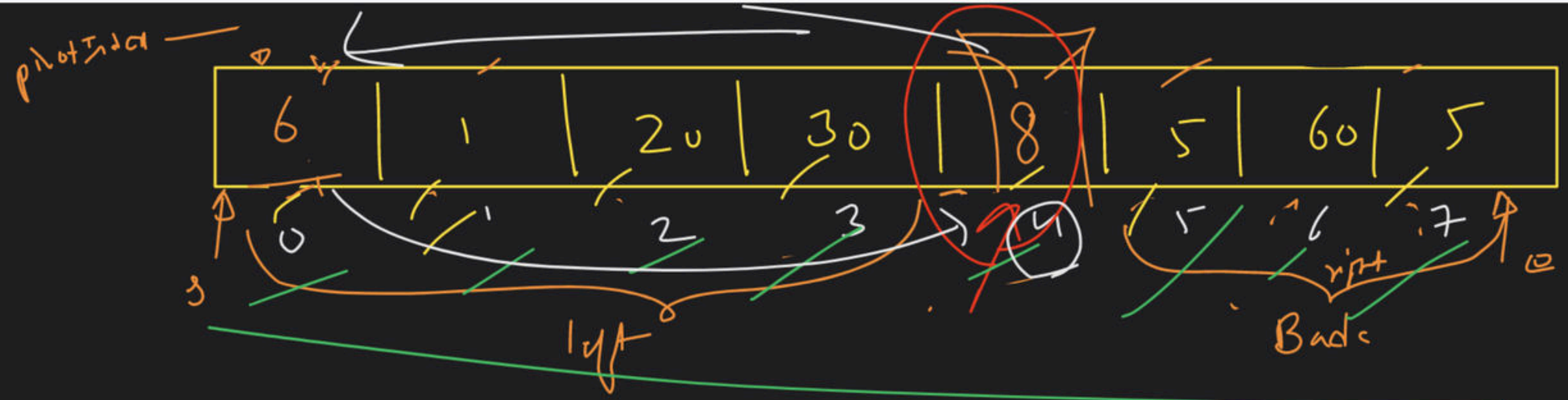
(C)

we are making two

left → choose <= pivot

right → Bade > pivot

(D) return pivotIndex



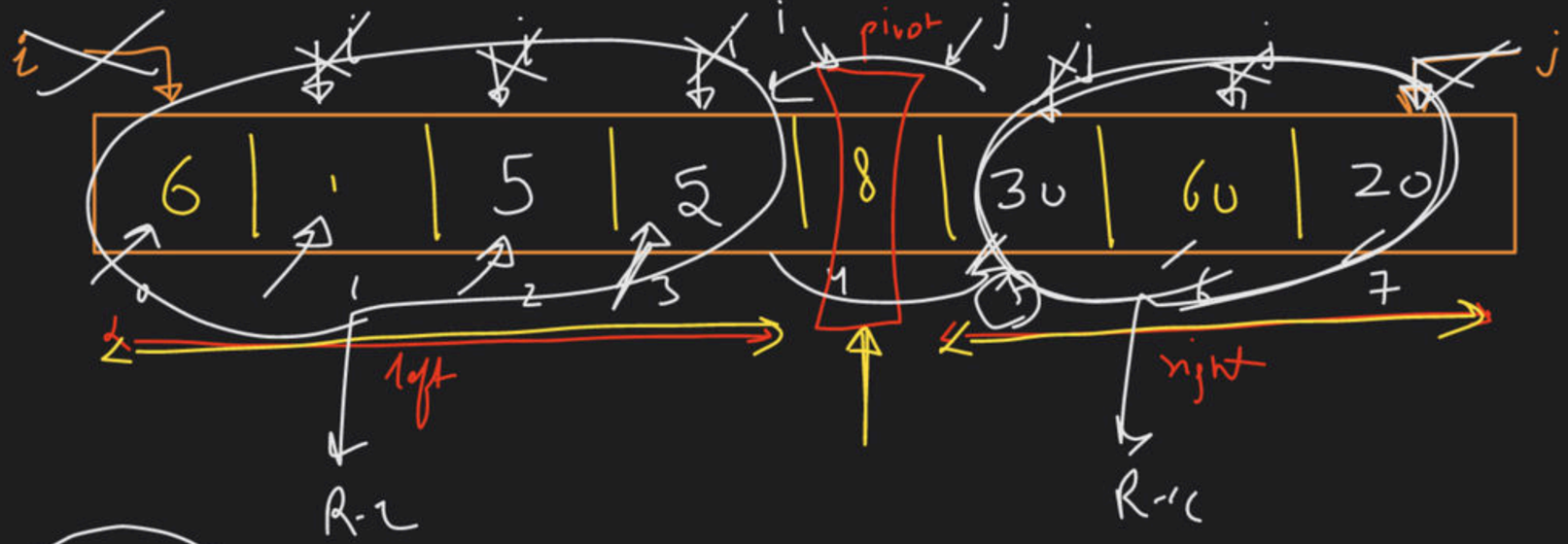
P.1

(A) pivotIndex = s = 0

(B) count <= pivotElement → count = 4

rightIndex = s + count
= 0 + 4 = 4

(C)



wrong element

↓

left part

↓

element > pivot

right part

↓

element < pivot

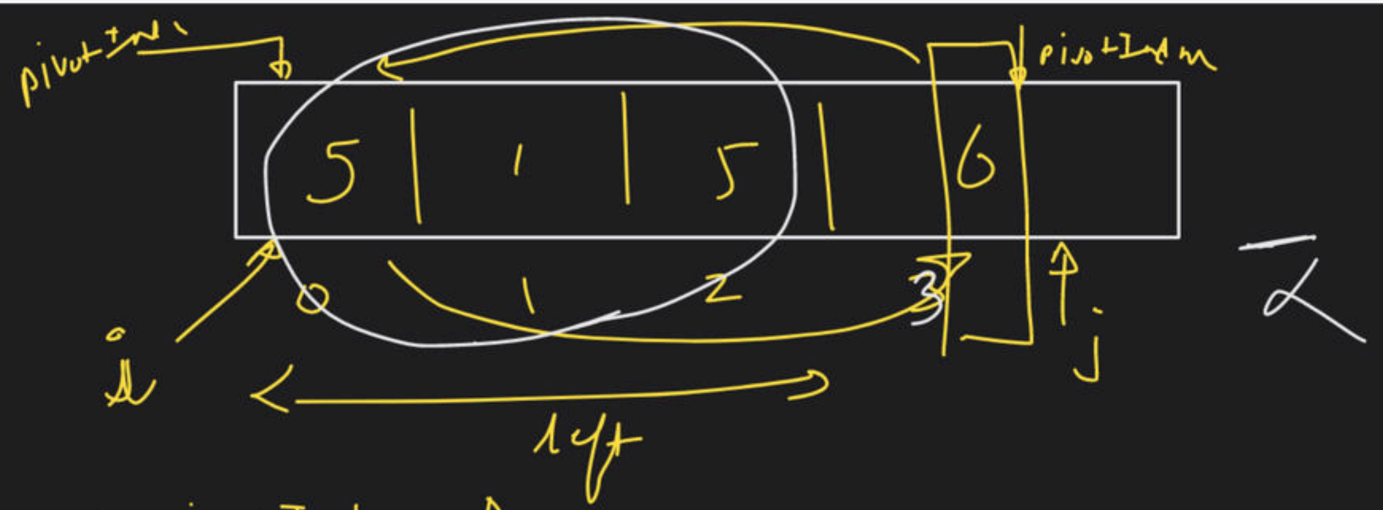
i = 2
j = 7

→ swap(arr[i], arr[j])

i = 3
j = 5

→ swap(arr[3], arr[5])

i = j = pivotIndex → Run out



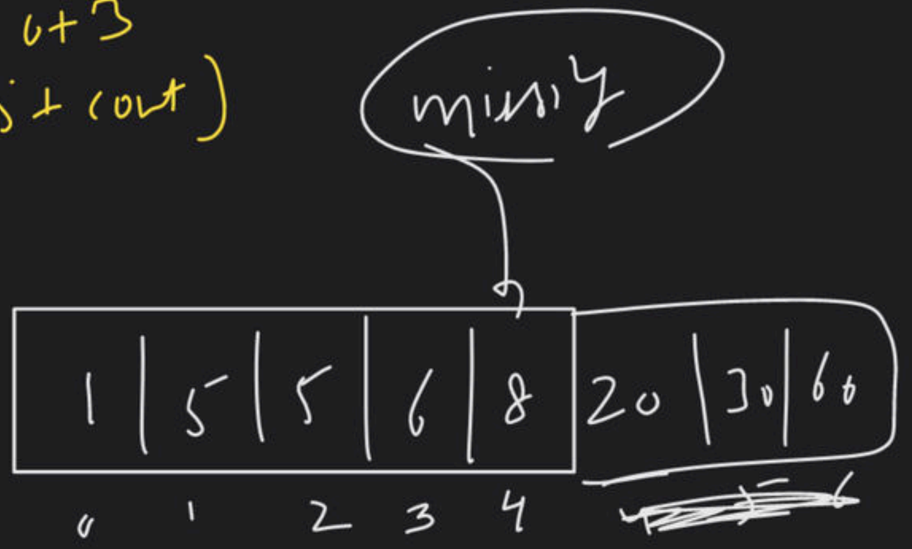
P.L

① pivotIndex = 0

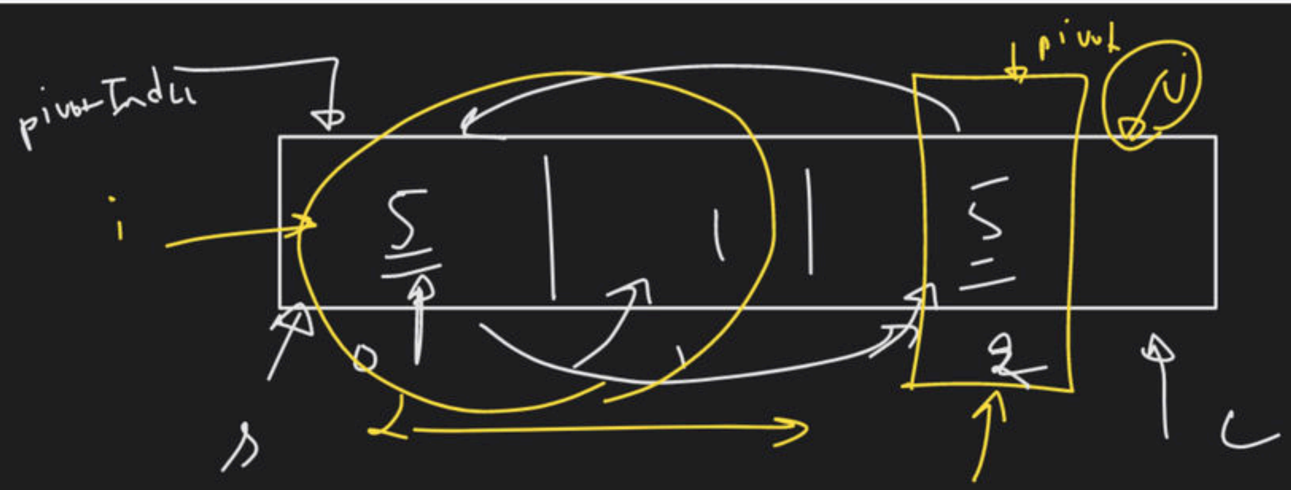
② count = 3

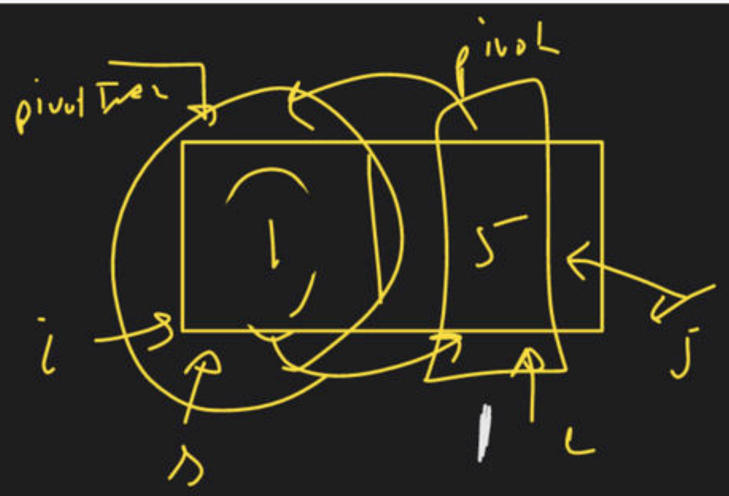
swap (arr[pivotIndex], arr[i+count])

③ ✗



R.C





① pivotIndex = 0

② count = 2

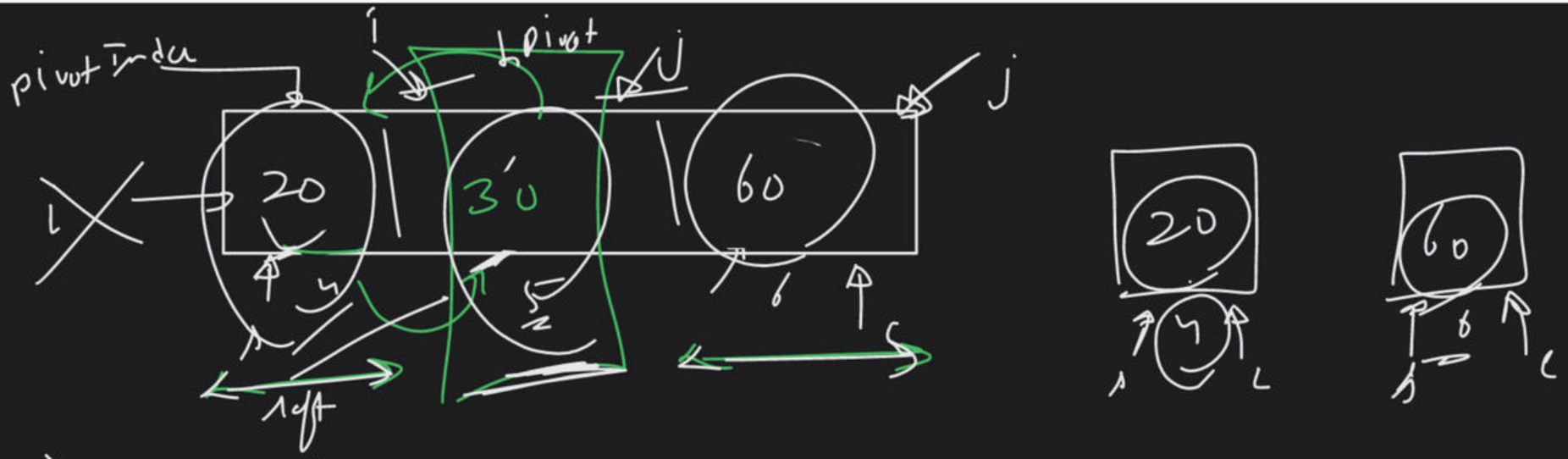
③ ✗

④

① pivotIndex = 1

② count = 1

③ ✗



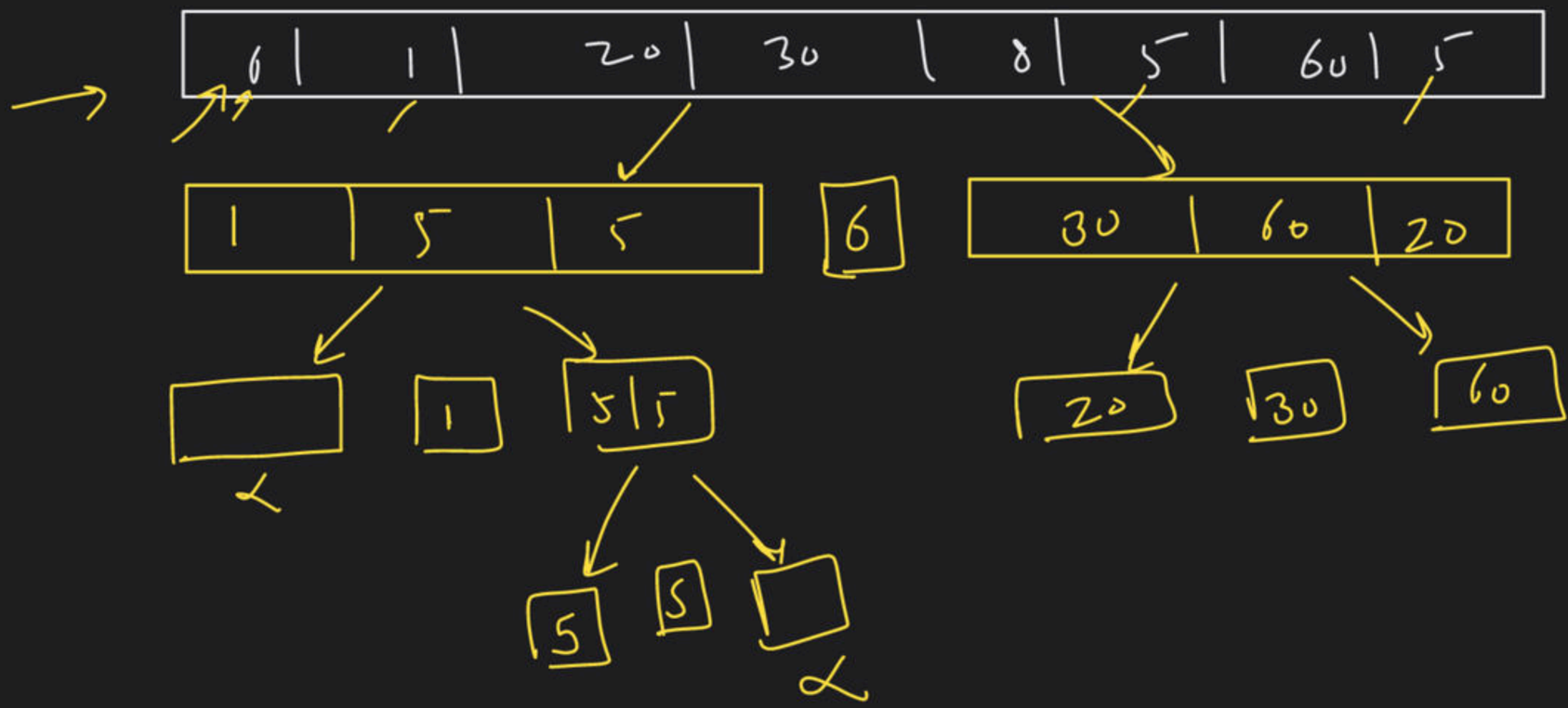
(A) pivotIndex = 4

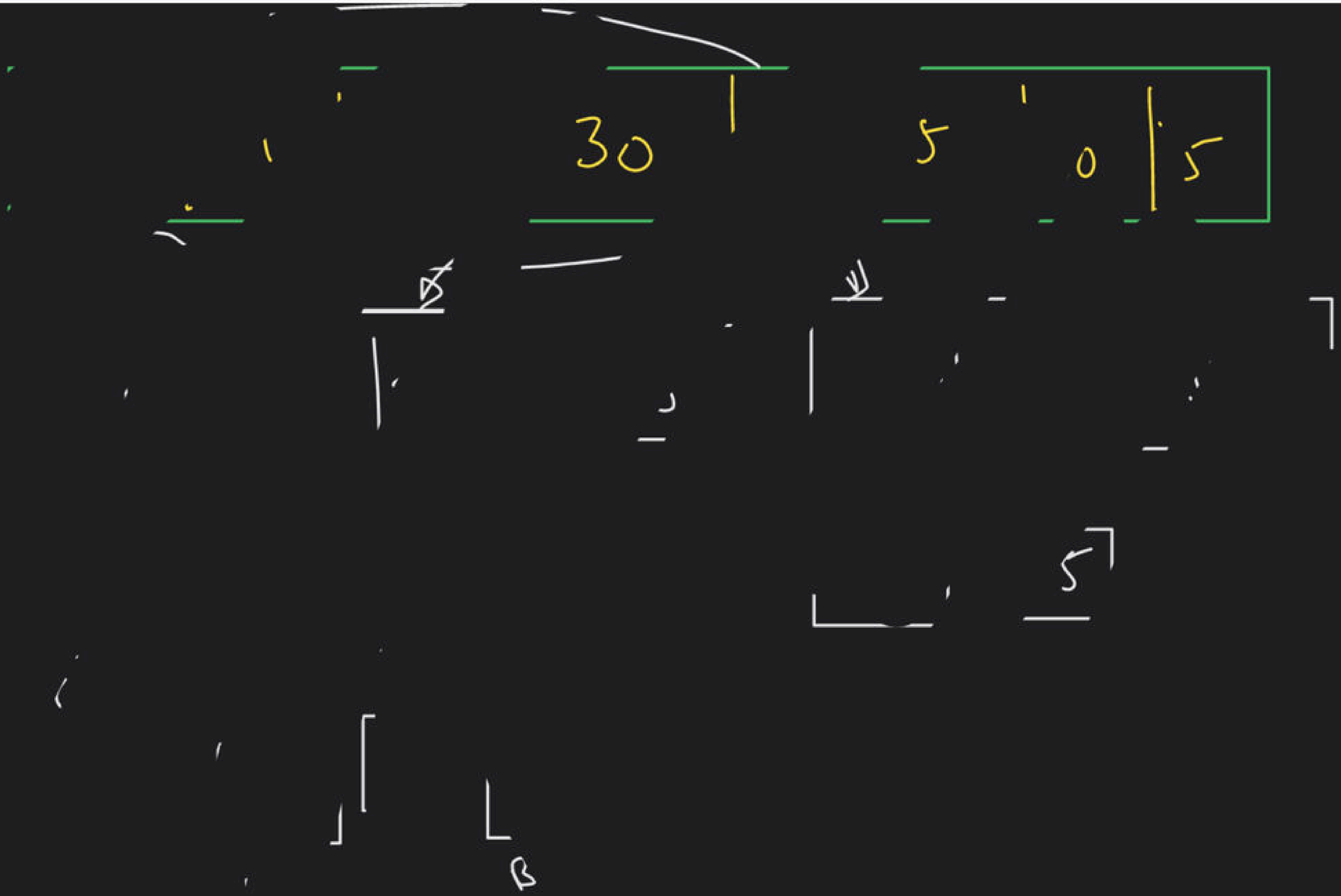
(2) count = 1 → rightIndex = 4 + 1 = 5

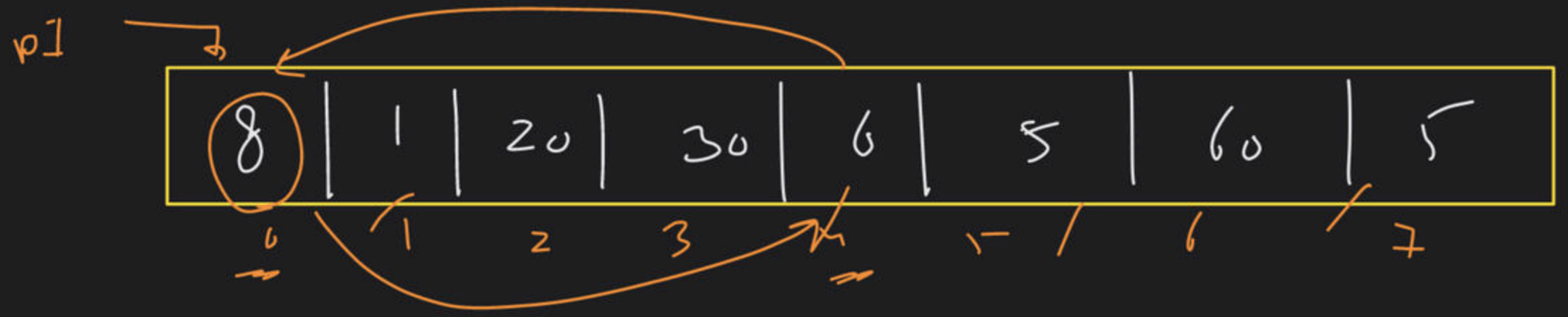
→ swap(arr(pivotIndex), arr(rightIndex));

(3) i = 4, j = 6 → swap(arr[4], arr[6])

T.C →



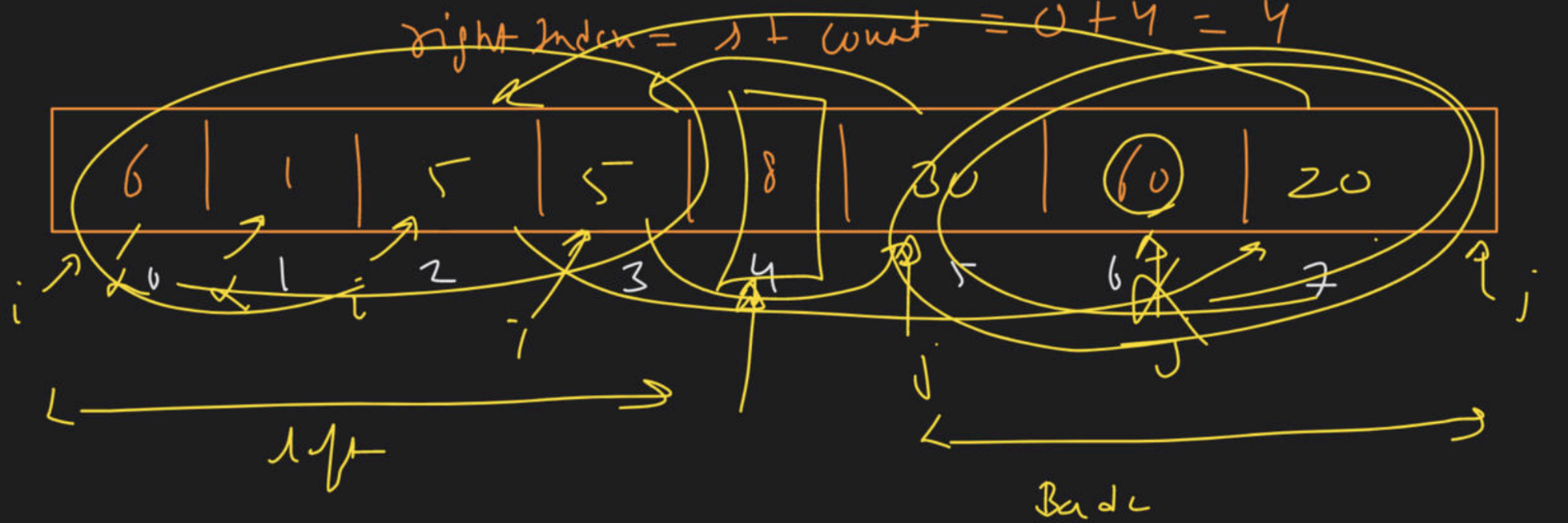

30
5
0
5
5
B



(A)

(i) pI = 0

(ii) right pos → count = 4



$a = \log n$

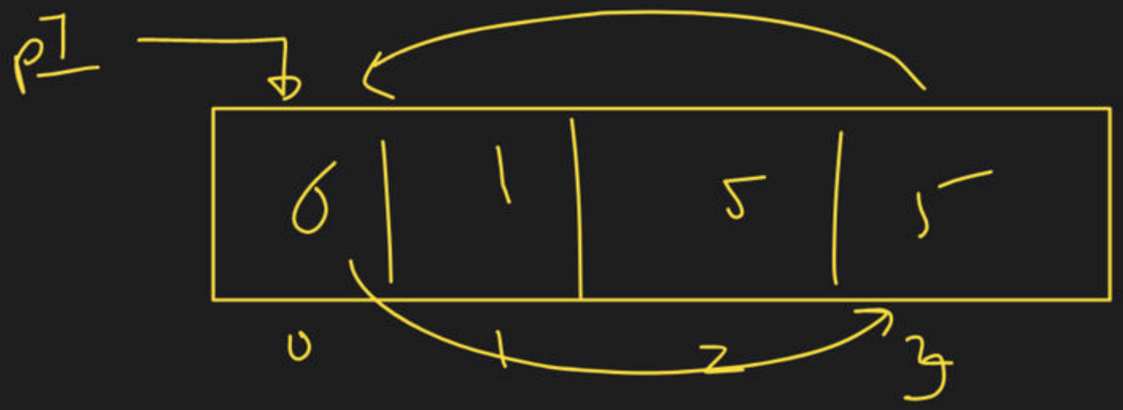
sorted — — ?

partition logic → $O(2n)$

$O(n)$

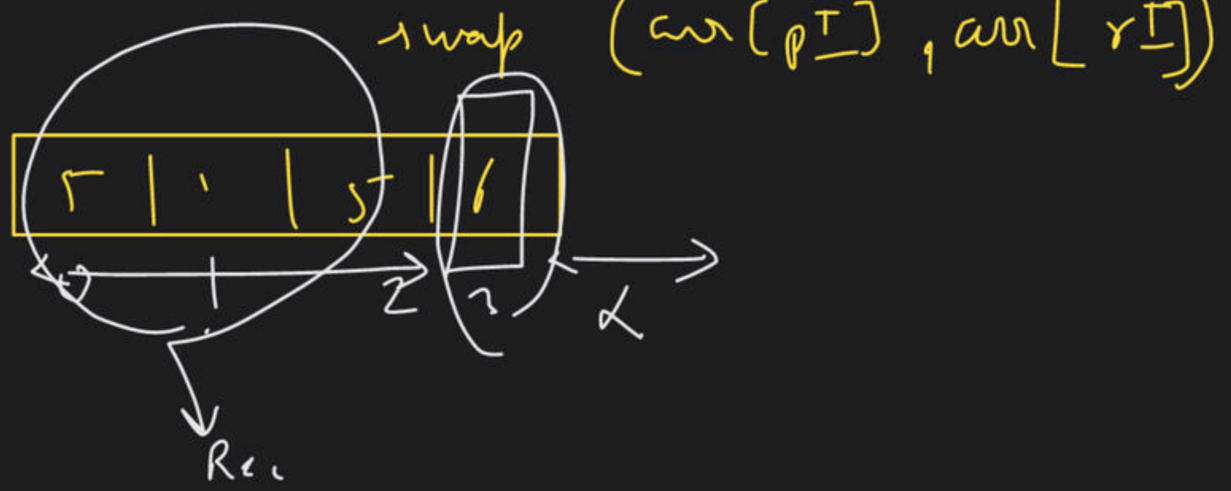
T.C → ? → $O(n \log n)$ → How

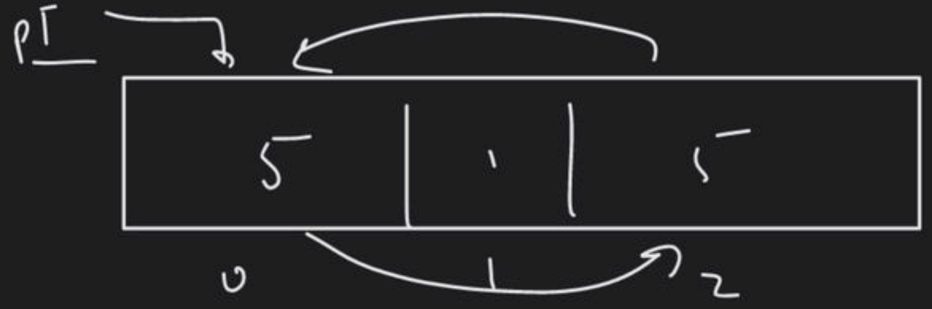
W.C → Reverse sorted → $O(n^2)$



① $p[I] = 0$

② count = 3 $\rightarrow r[I] = 0+3 = 3$

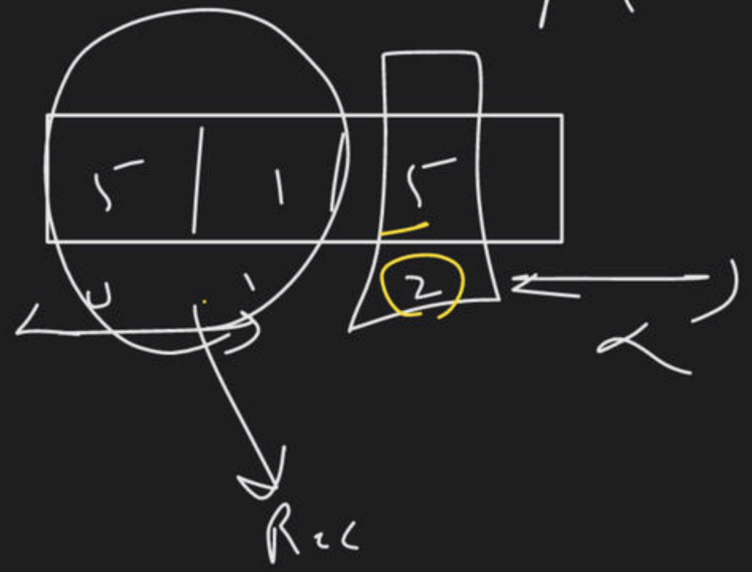


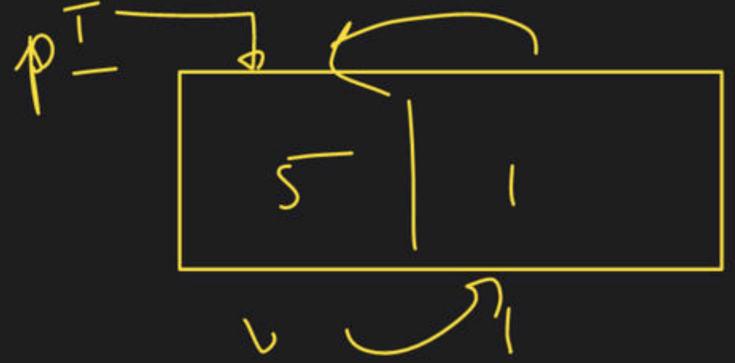


(2) $pZ = 0$

(3) $count = 2 \longrightarrow nT = 1 + count = 0 + 2 = 2$

swap (arr[0], arr[2])

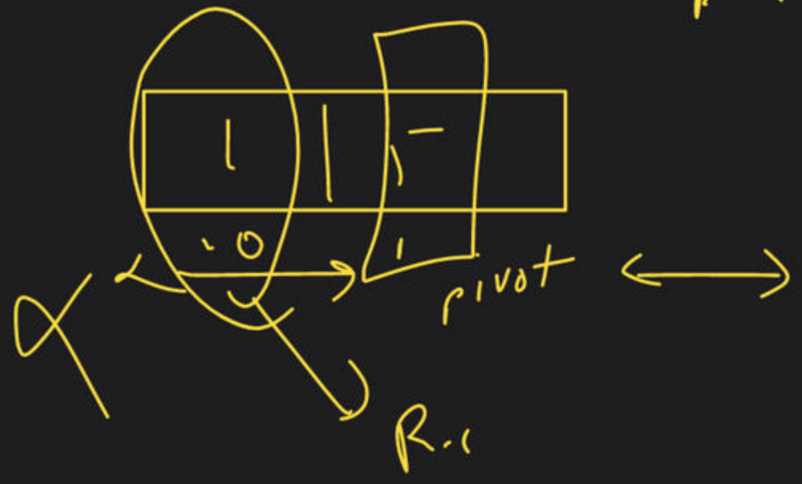


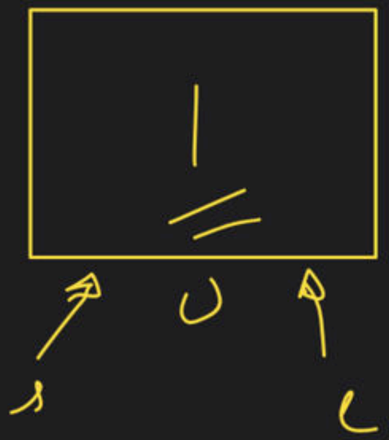


(1) $p = 0$

(2) count = 1 → x → st + count = 0 + 1

swap (arr[p], arr[x])

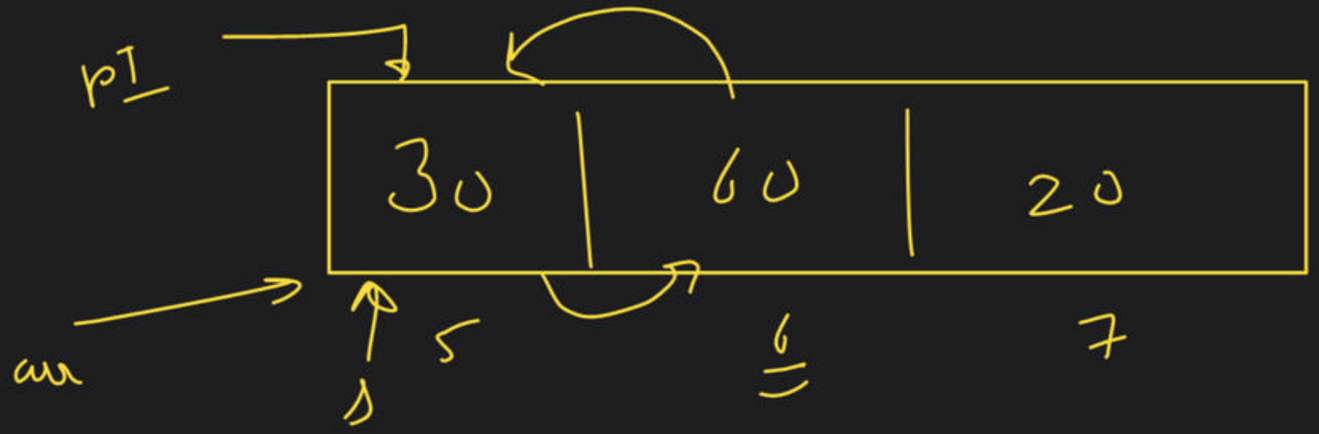




→ single Element

already sorted

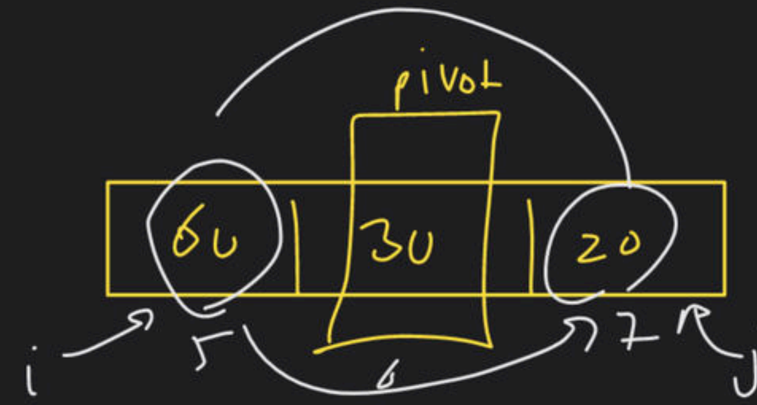
↳ return



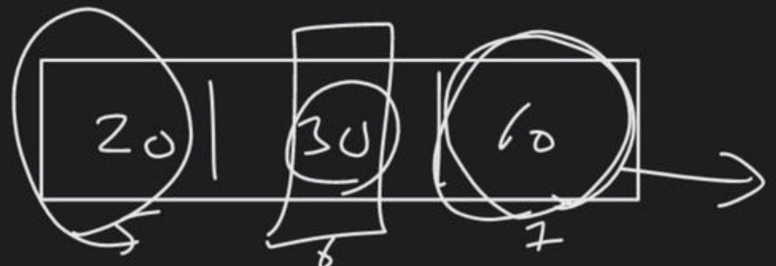
① $pI = 5$

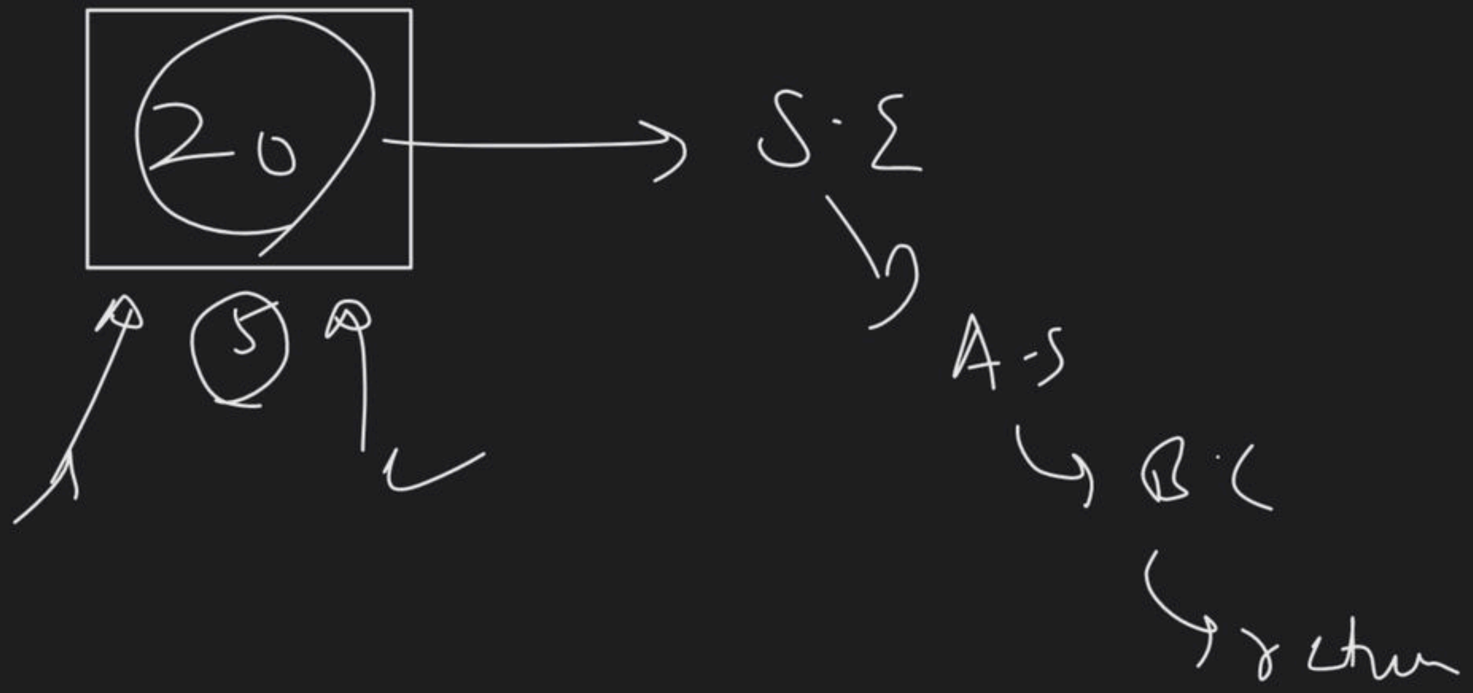
② $count = 1 \rightarrow rI = s + count = 5 + 1 = 6$

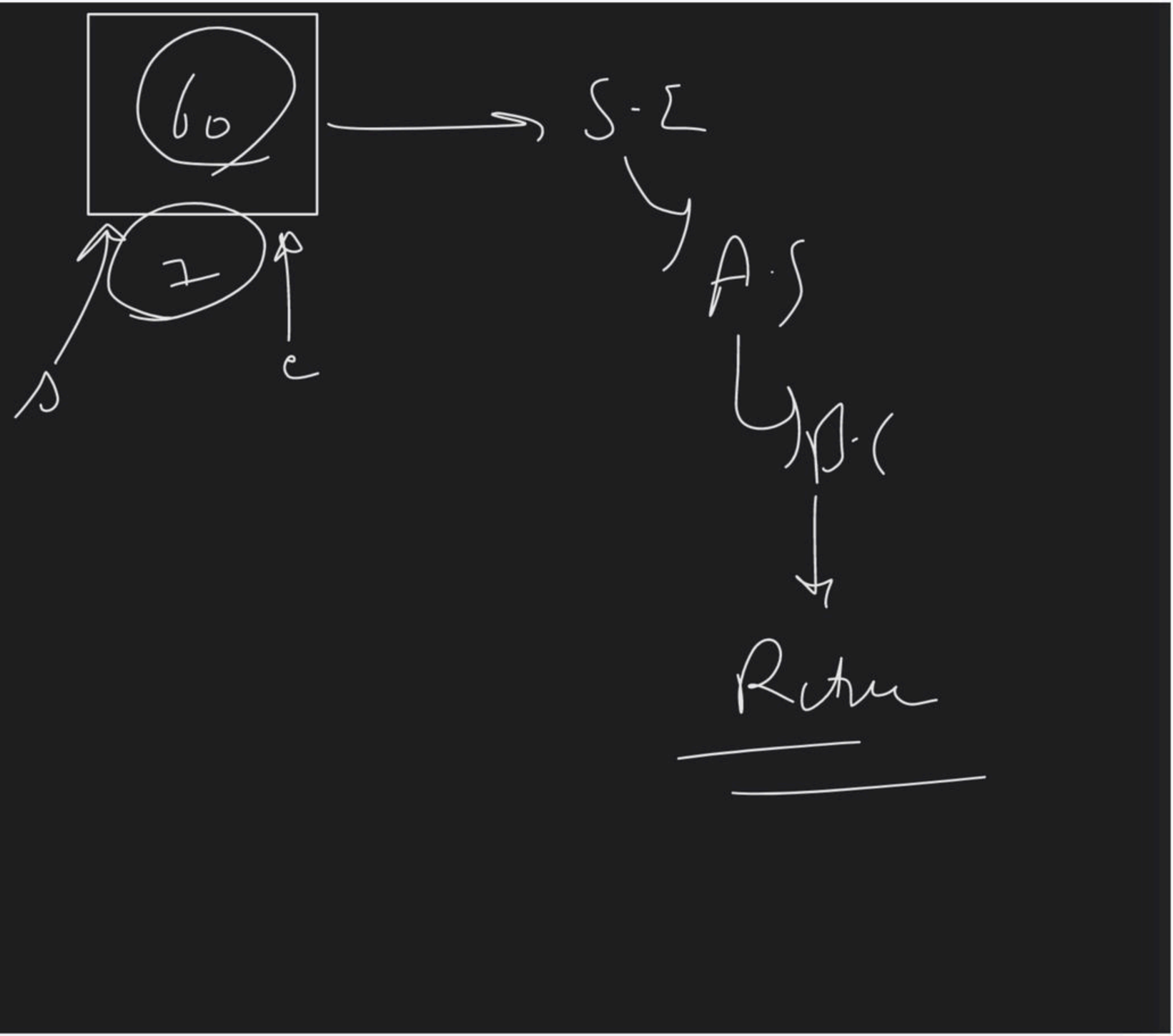
swap (arr[5], arr[6])

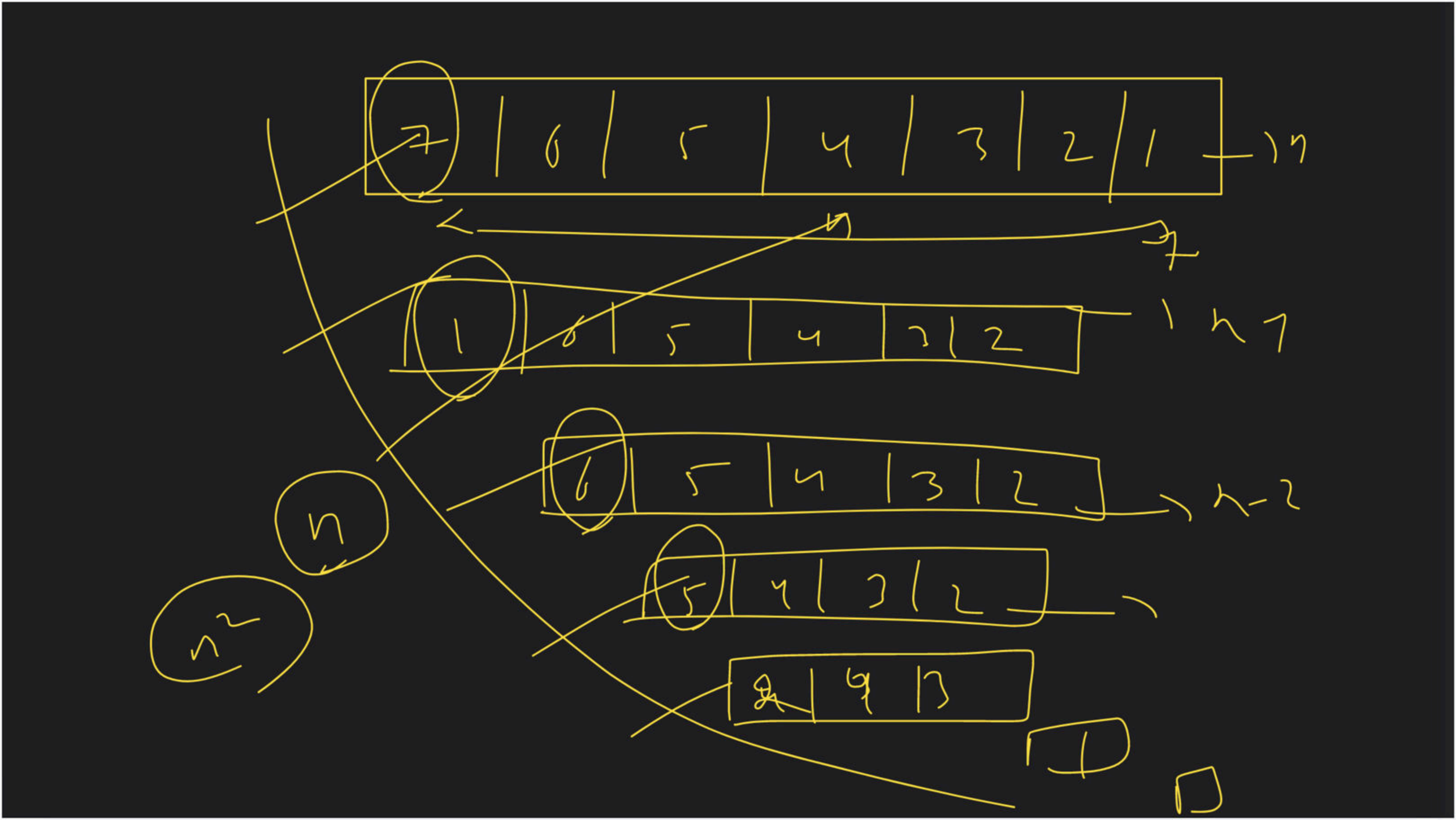


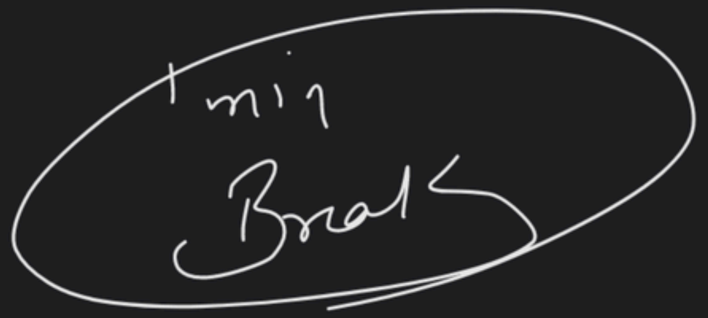
$i = 5$
$j = 7$ → swap (arr[i], arr[j])




20
S·Σ
A·S
B·(
return

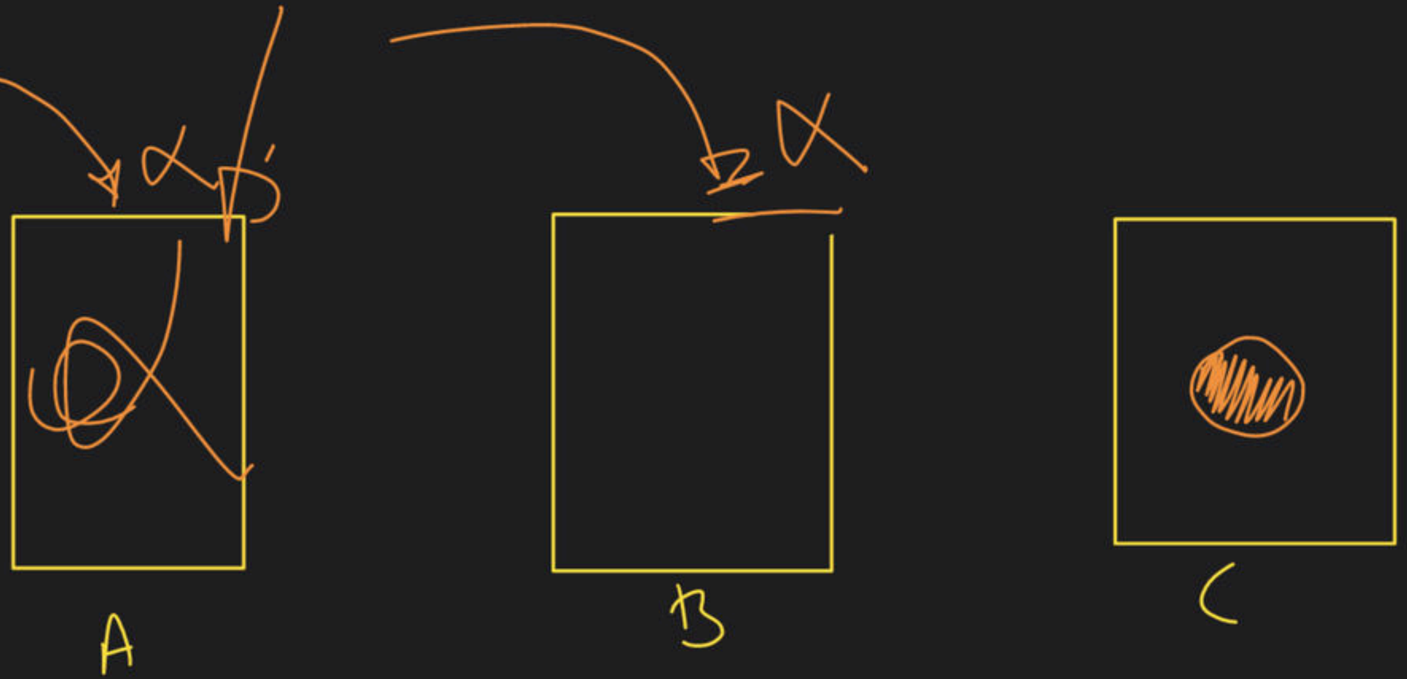

60
S-E
7
S
e
A.S
Return


7 | 6 | 5 | 4 | 3 | 2 | 1
1 | 6 | 5 | 4 | 3 | 2
6 | 5 | 4 | 3 | 2
5 | 4 | 3 | 2
n
n-2
1
0



Back tracking

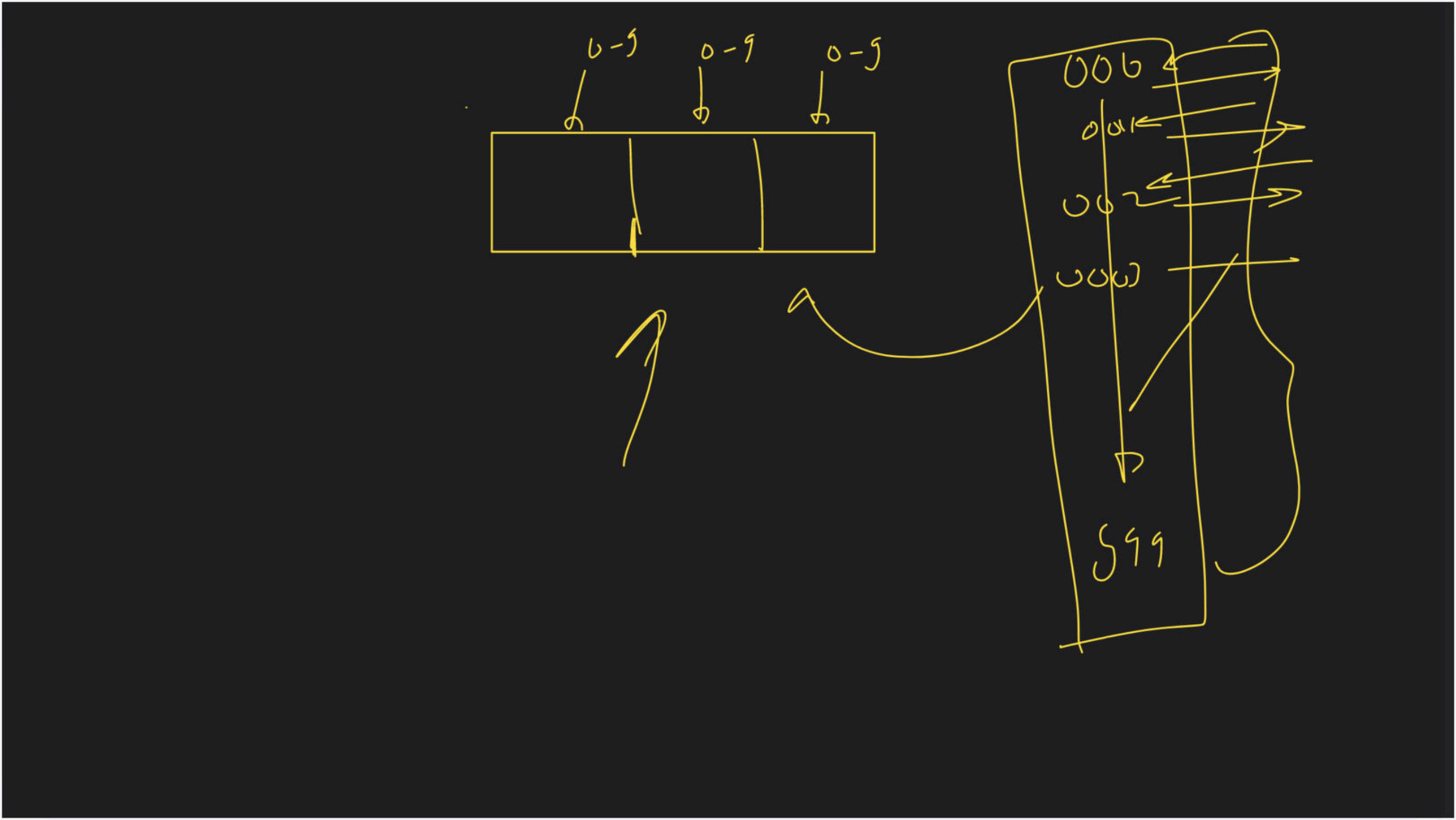
(specific form of Recursion)

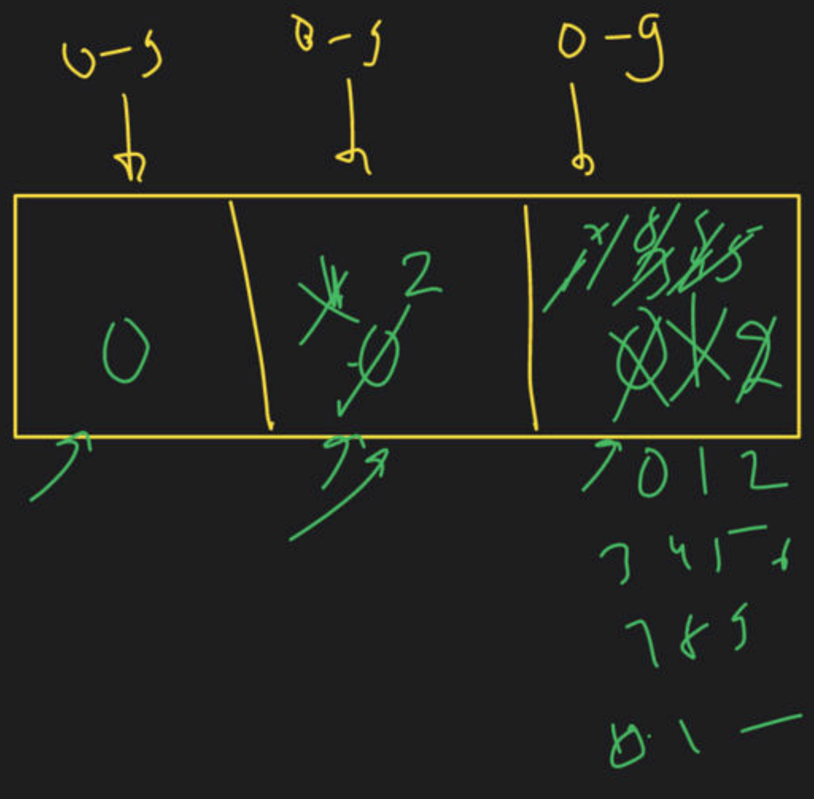


explore all possible solution

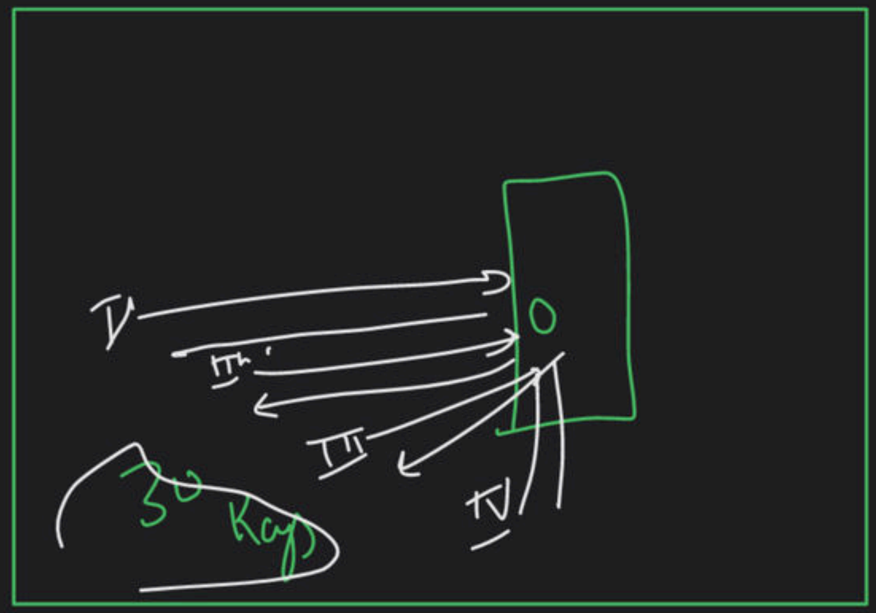
solution → discard → wapas check
nahi karoge

Rat in a maze




0-5
6-7
0-9
0
0 1 2
3 4 5
6 7 8
9

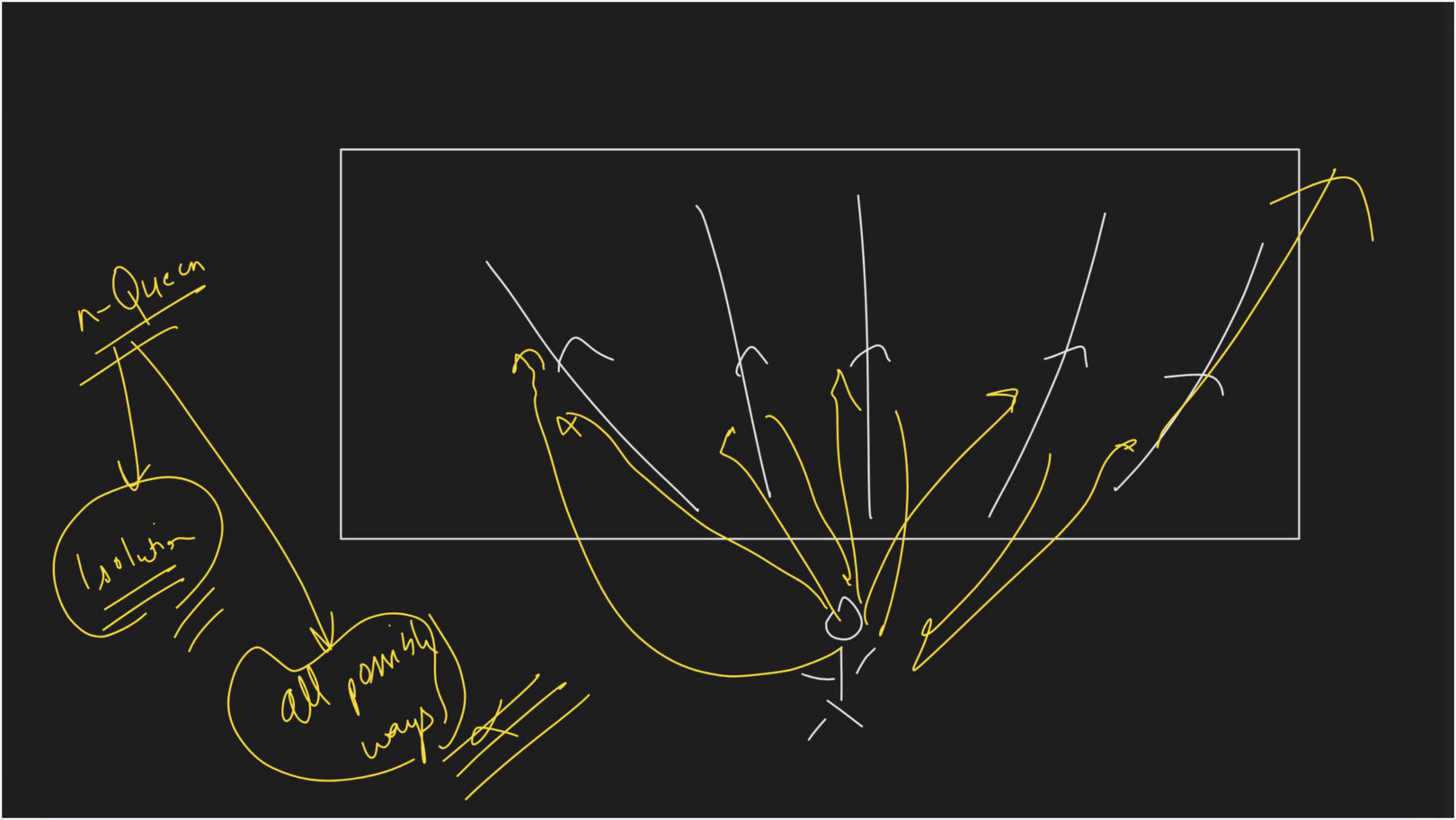
Backtracking



1st → 30th

3 min

Escape Room

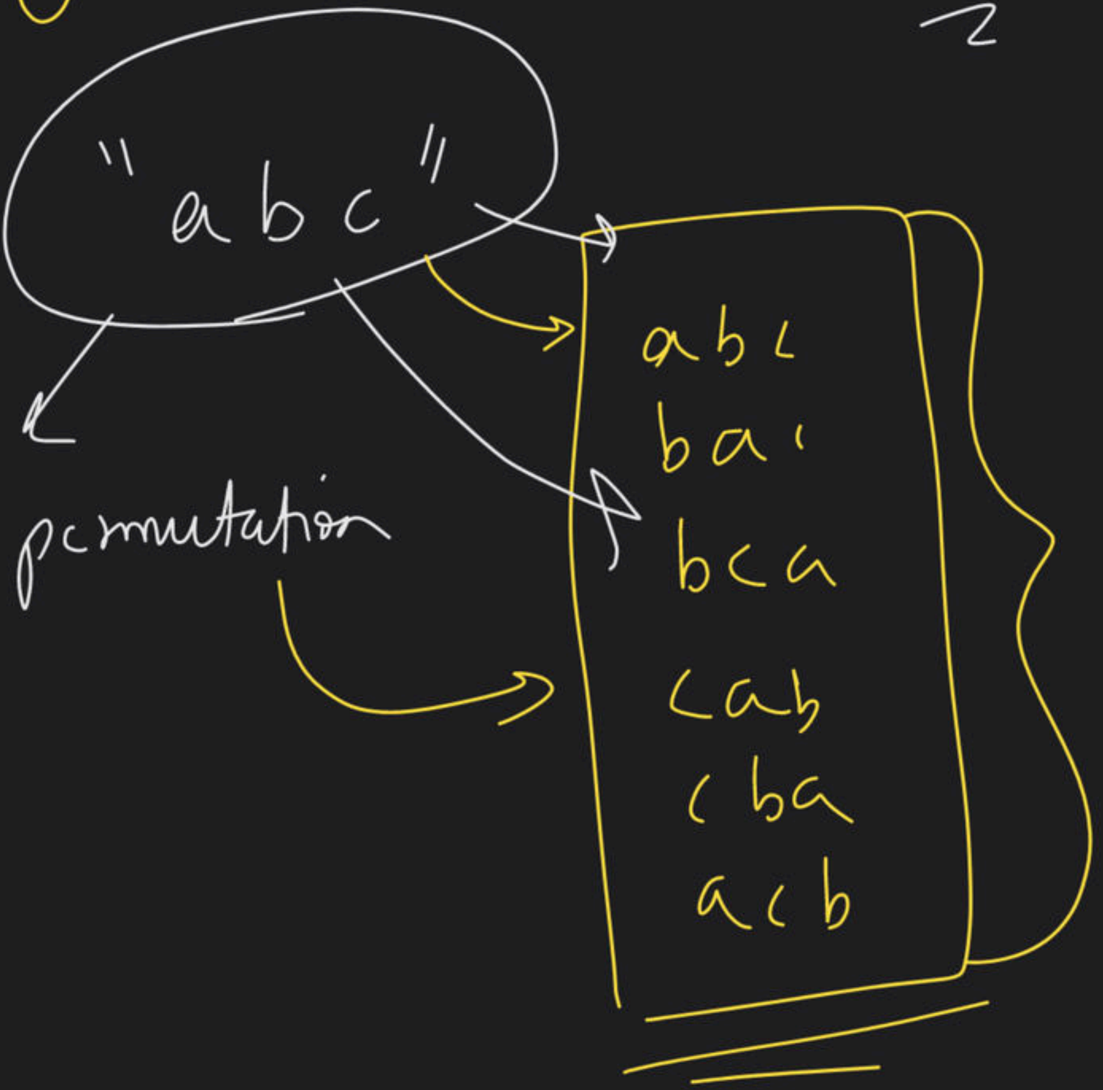

n-Queen
1 solution
all possible ways

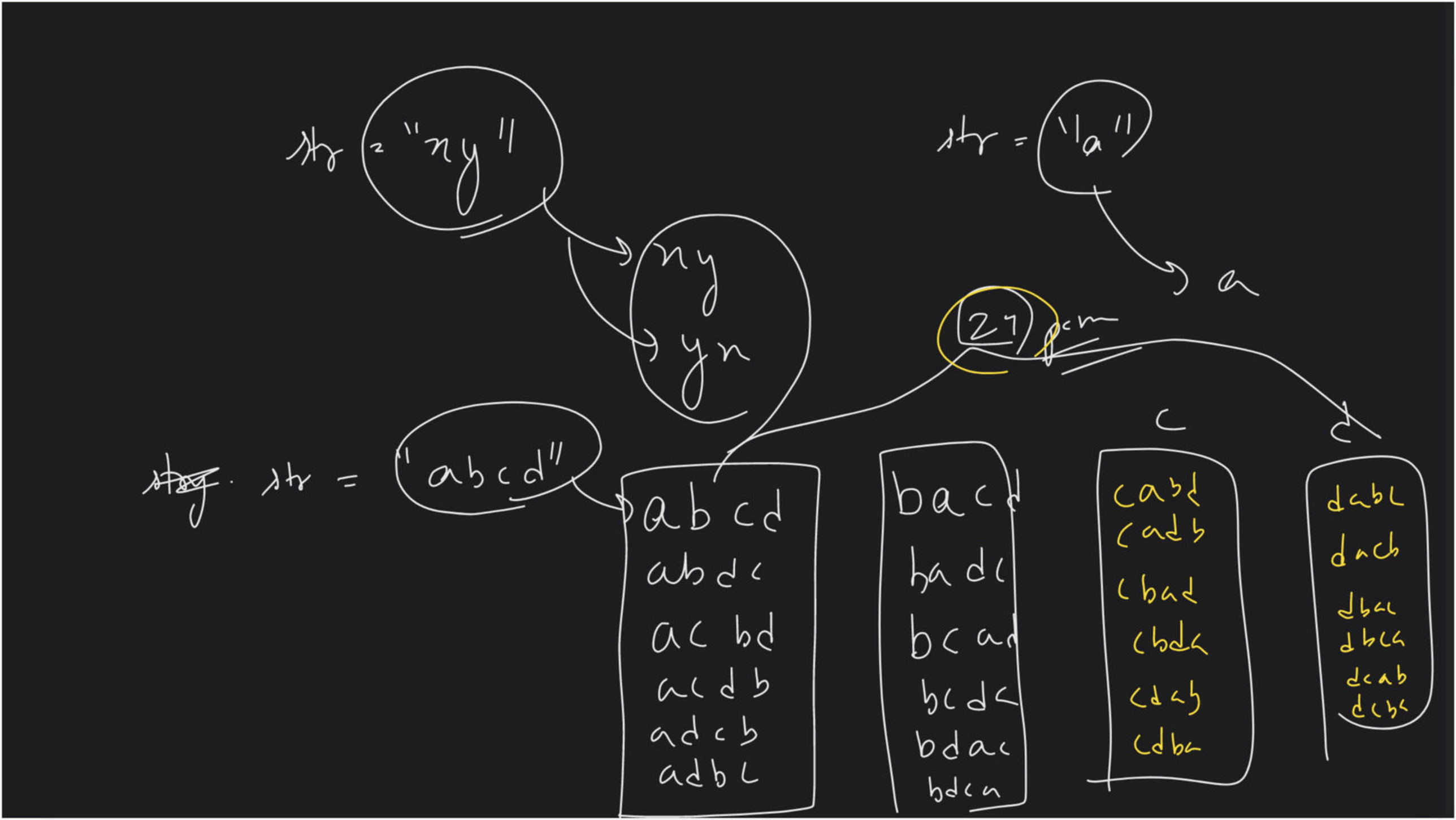
Permutations of String :-

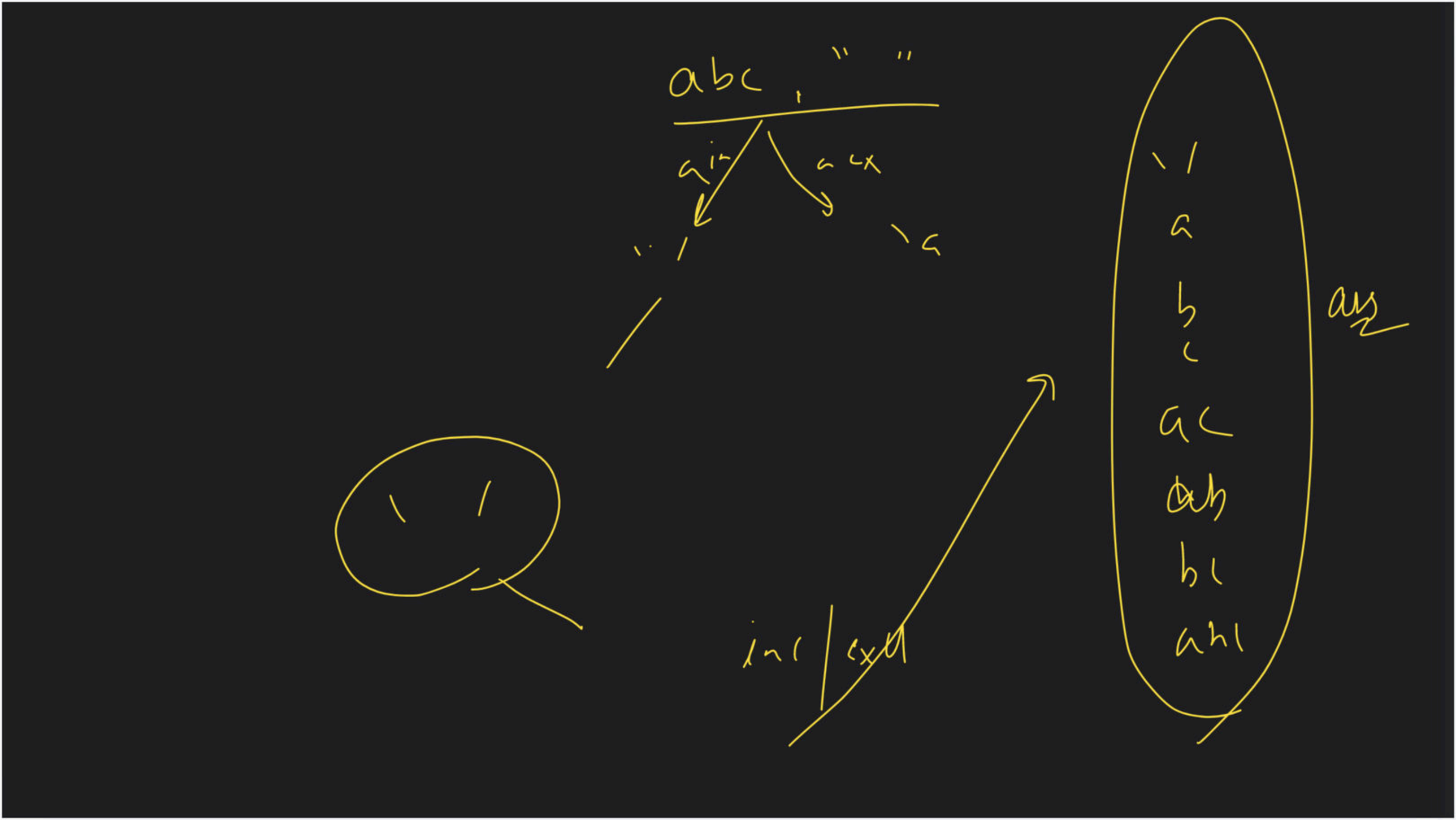
1 min

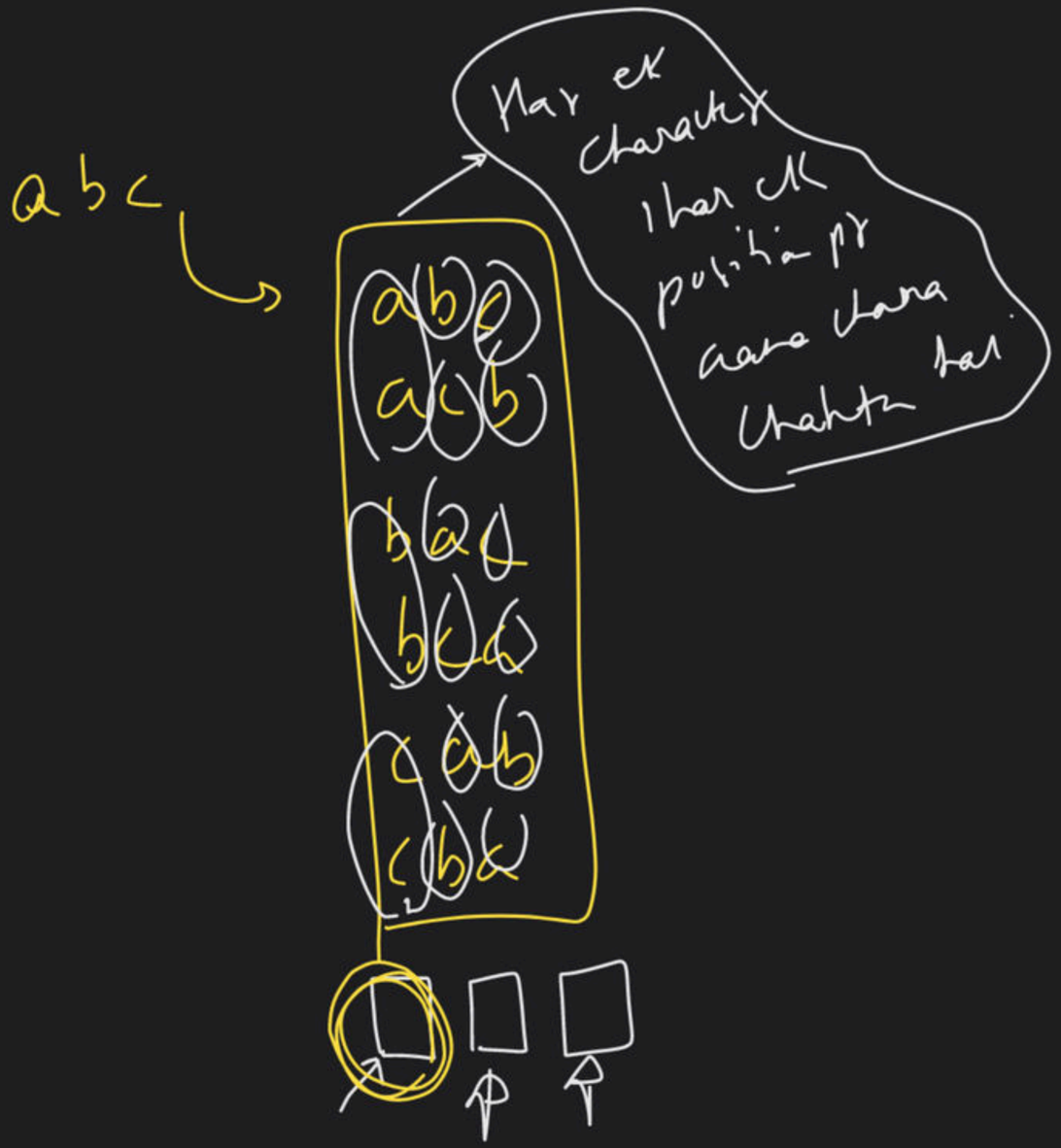
Break

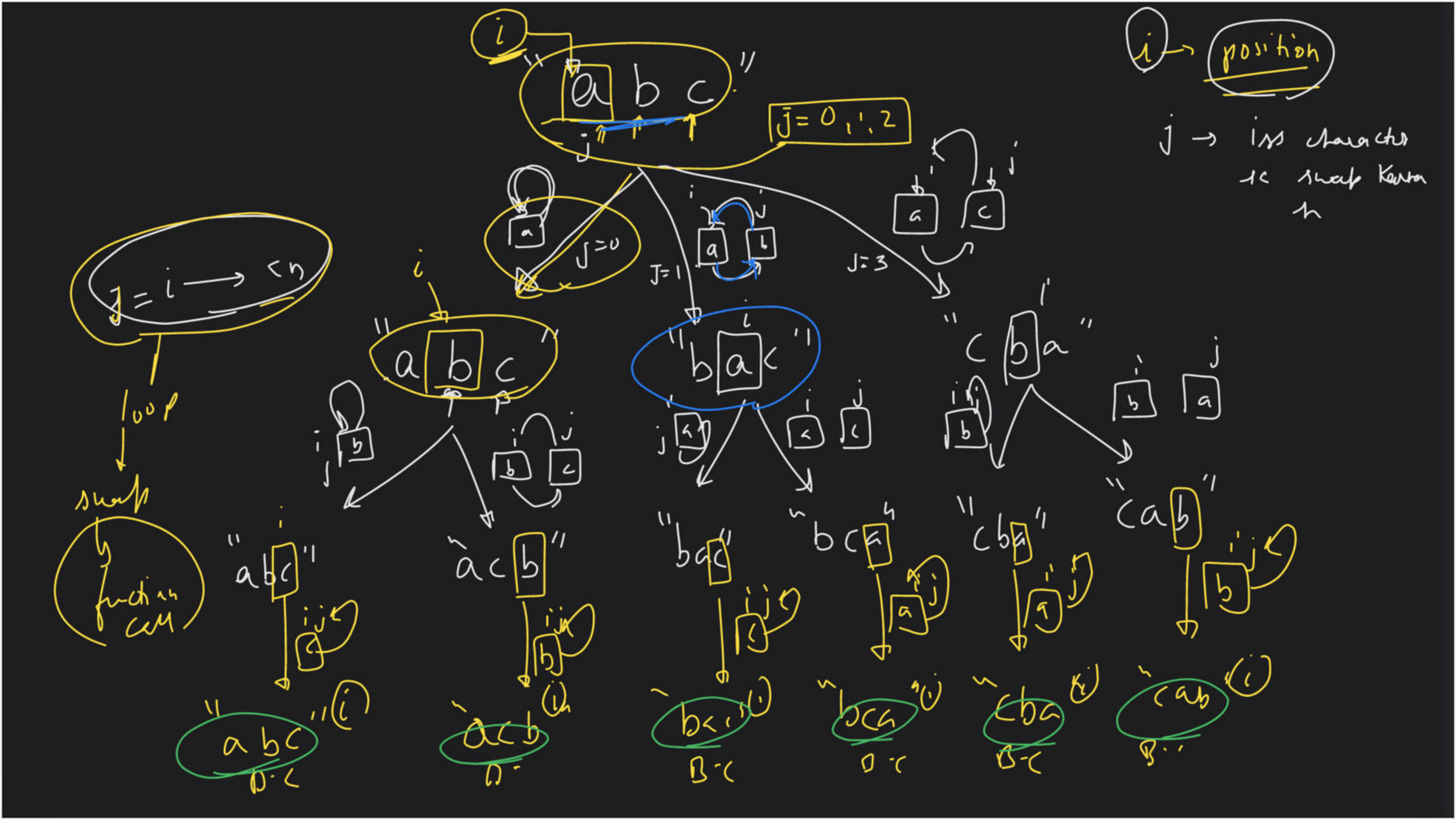
i/p → string str = "abc"

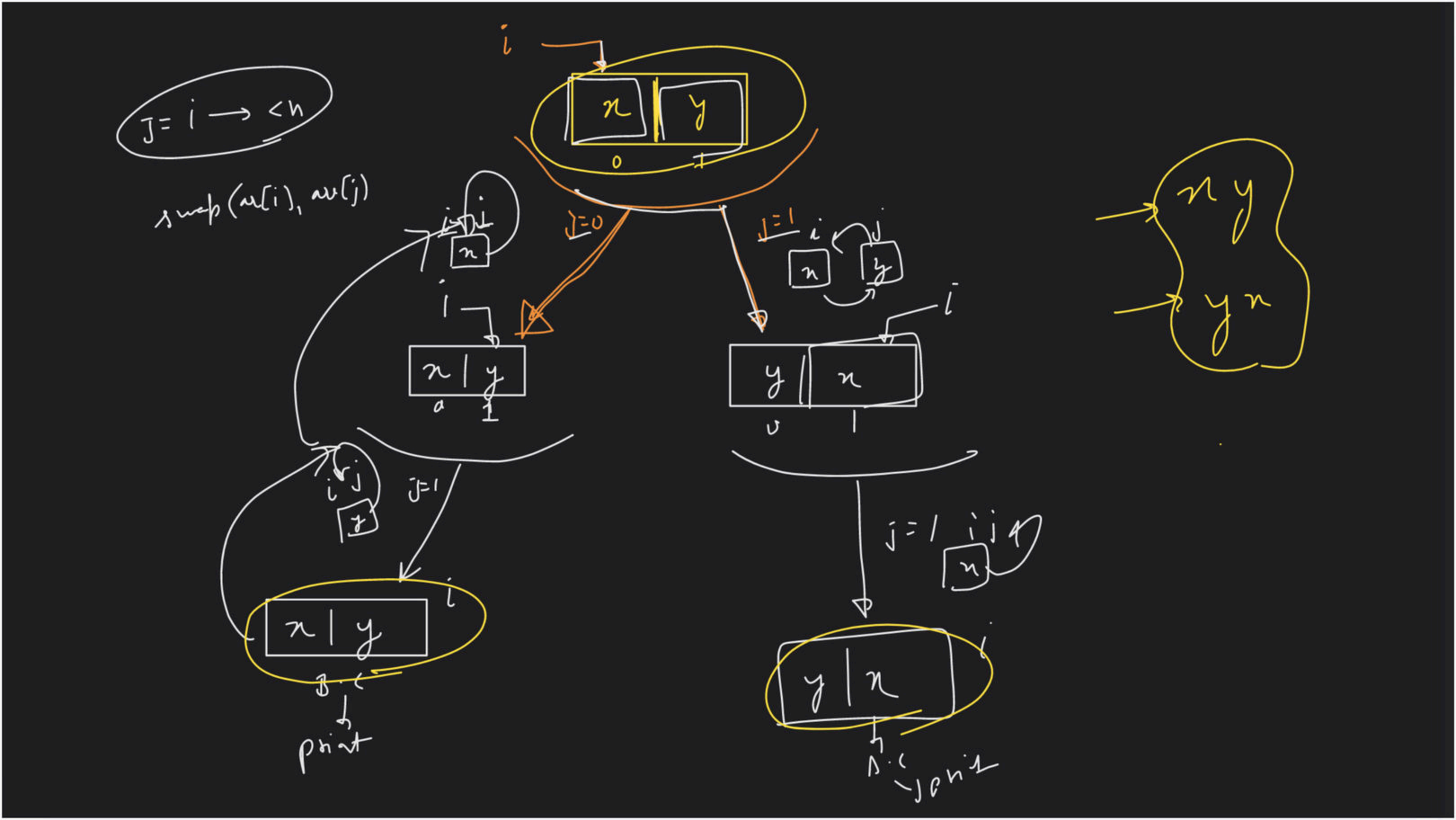


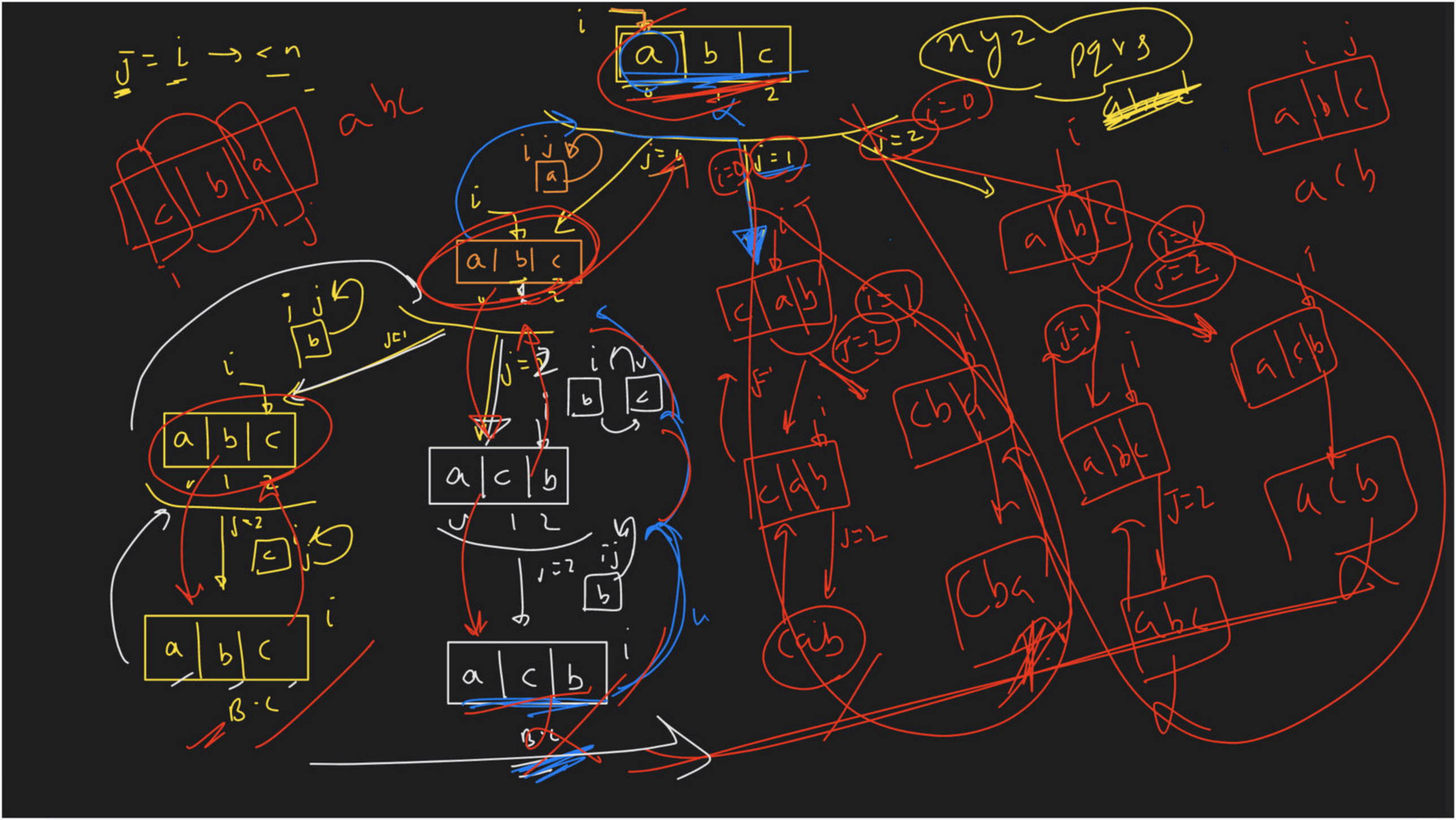

str = "xy"
xy
yx
str = "a"
a
str = "abcd"
abcd
abdc
acbd
acdb
adcb
adbc
bacd
badc
bcad
bcda
bdac
bdca
cabd
cadb
cbad
cbda
cdab
cdba
dabc
dacb
dbac
dbca
dcab
dcba

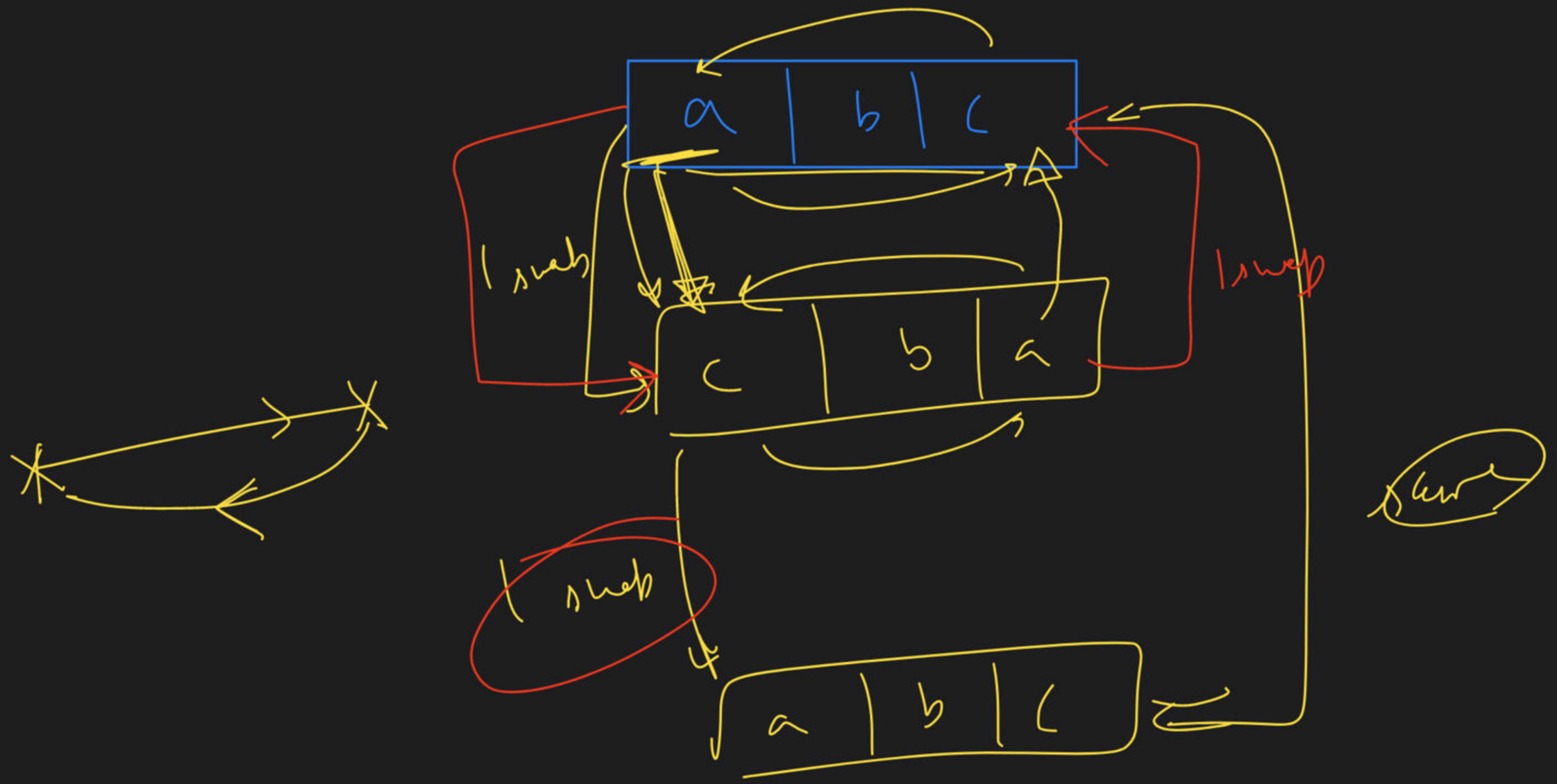

abc
""
ans

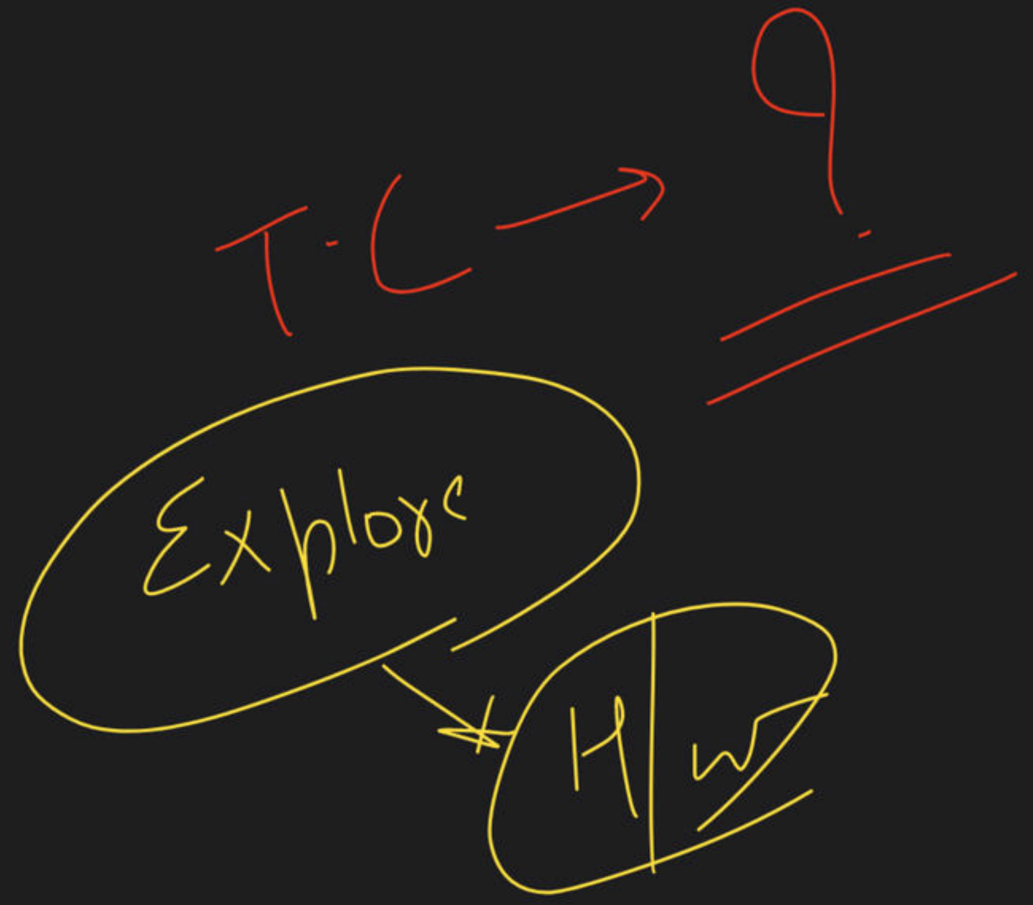

abc
Har ek
character
ka har ek
position pr
alag alag
character hai

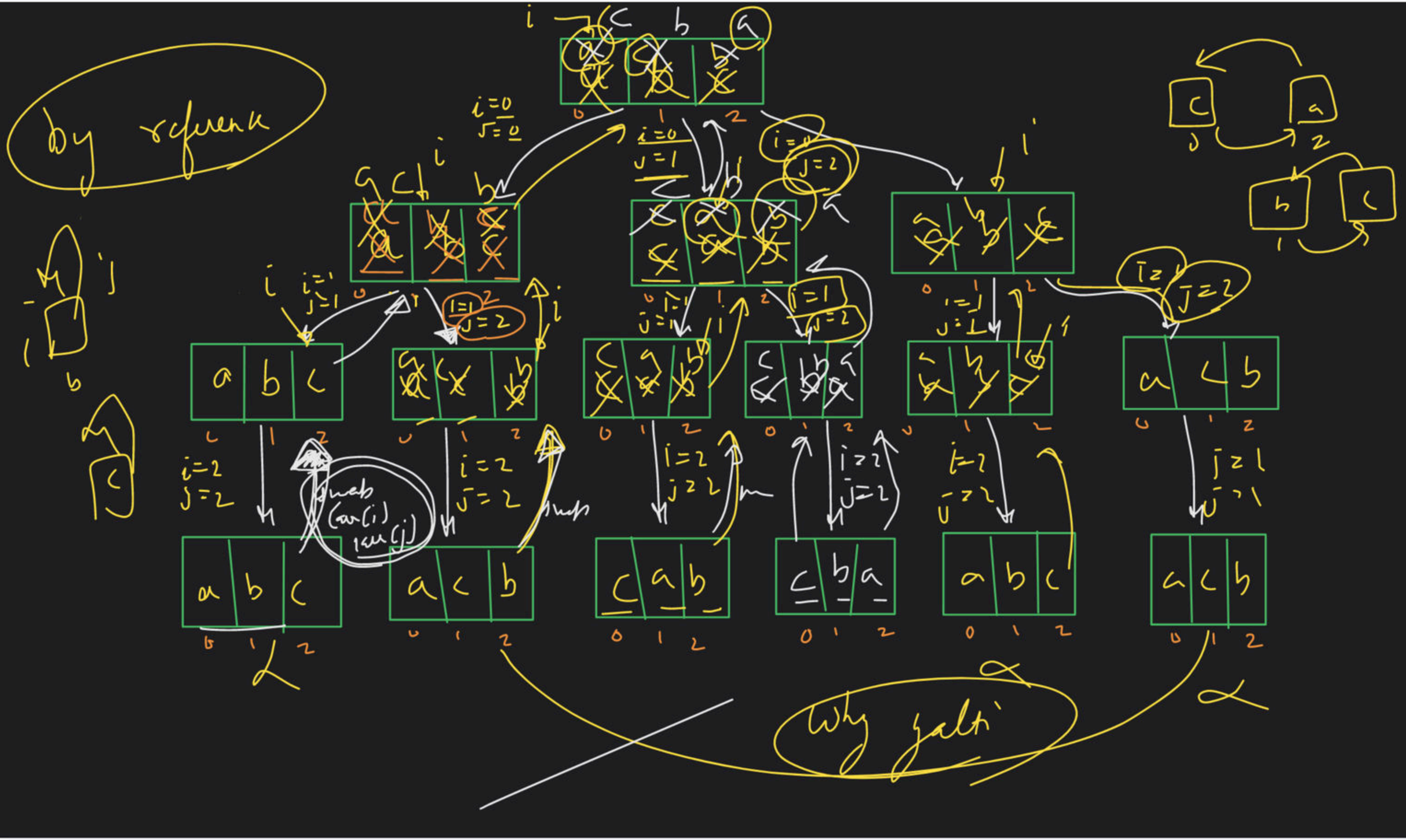

i
a b c
j = 0, 1, 2
i → position
j → iss character ic swap karna h
j = i → n
loop
swap
function call
J=0
J=1
J=3
abc
acb
bac
bca
cba
cab


j = i → <n
swap(arr[i], arr[j])
i
x
y
0
1
j=0
j=1
x | y
y | x
print
join


j = i → < n
a b c
i
j
J=1
J=2
B·C
nyz
pqrs


a
b
c
c
b
a
a
b
c


T·L → ?
Explore
H/W


by reference
swap (arr[i] arr[j])
why galti