Doubt Class With Lakshay Bhaiya

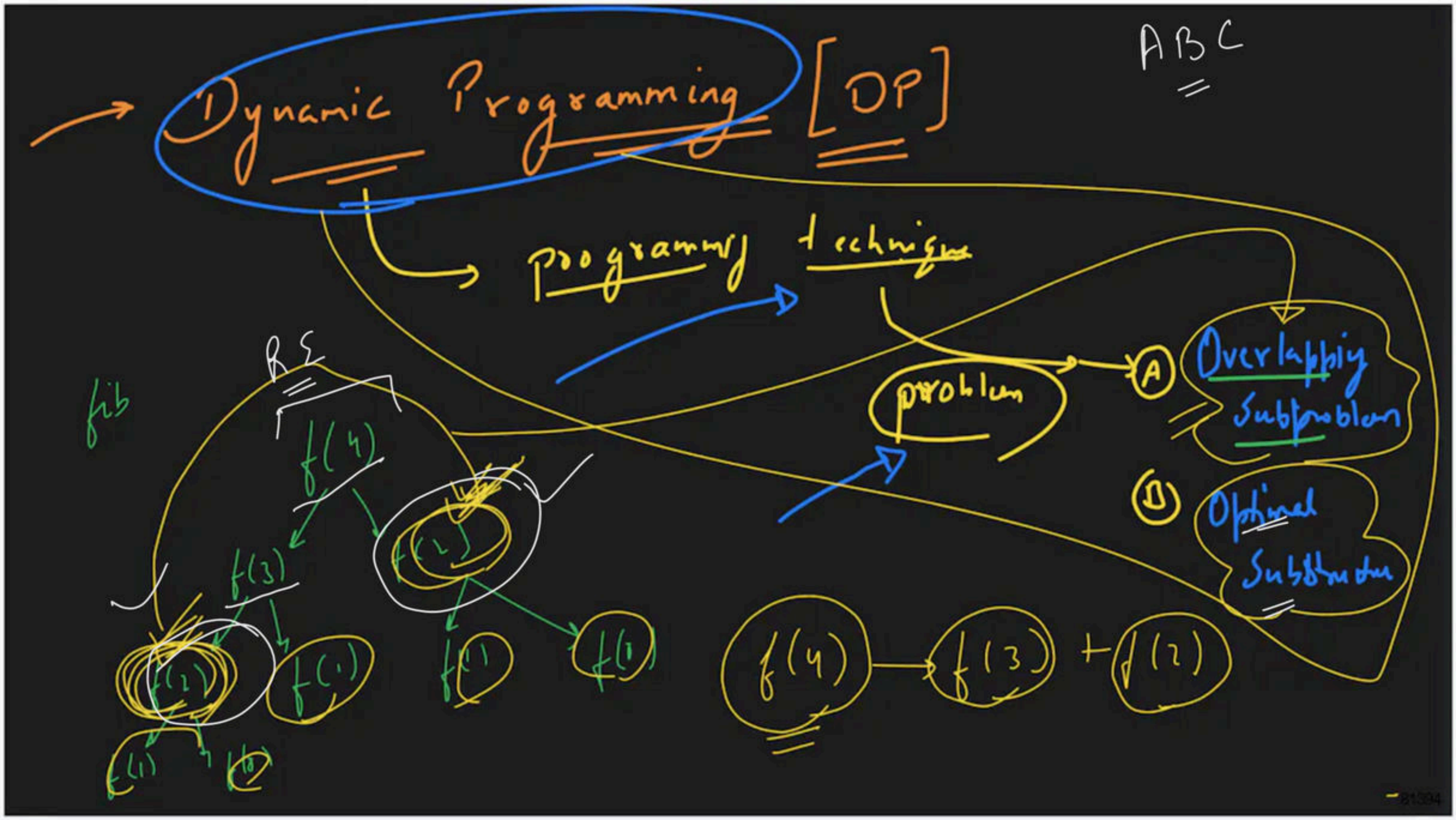
Special class

Love Babbar • Jan 21, 2024

Dynamic Programming Class-1

Special class

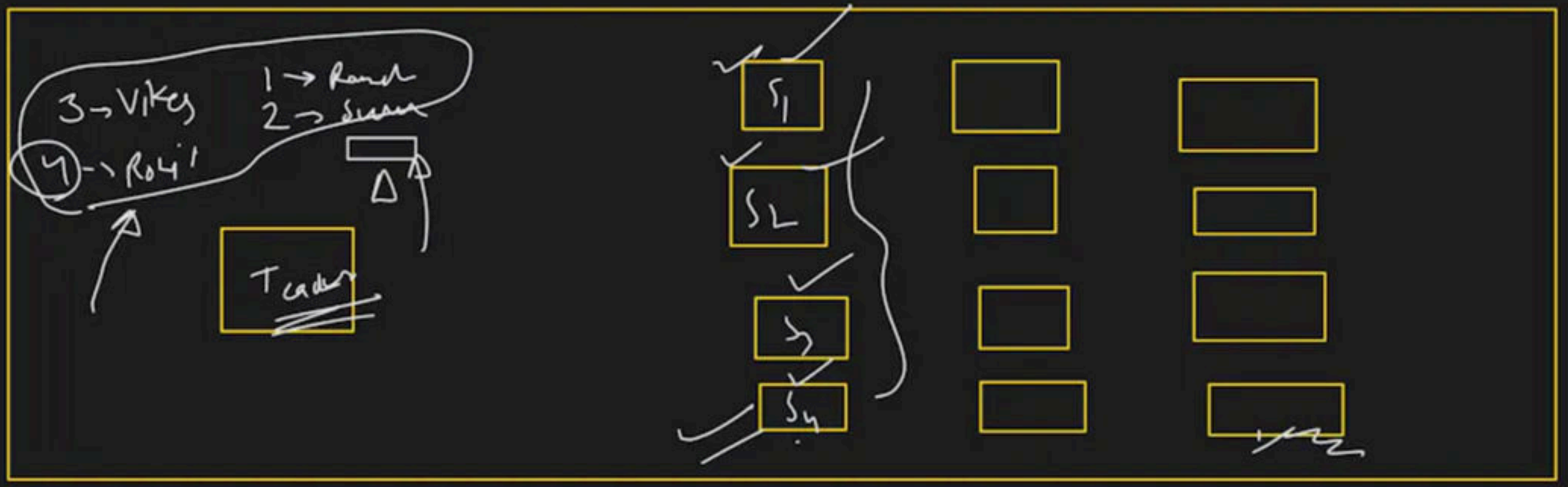
Love Babbar • Jan 5, 2024

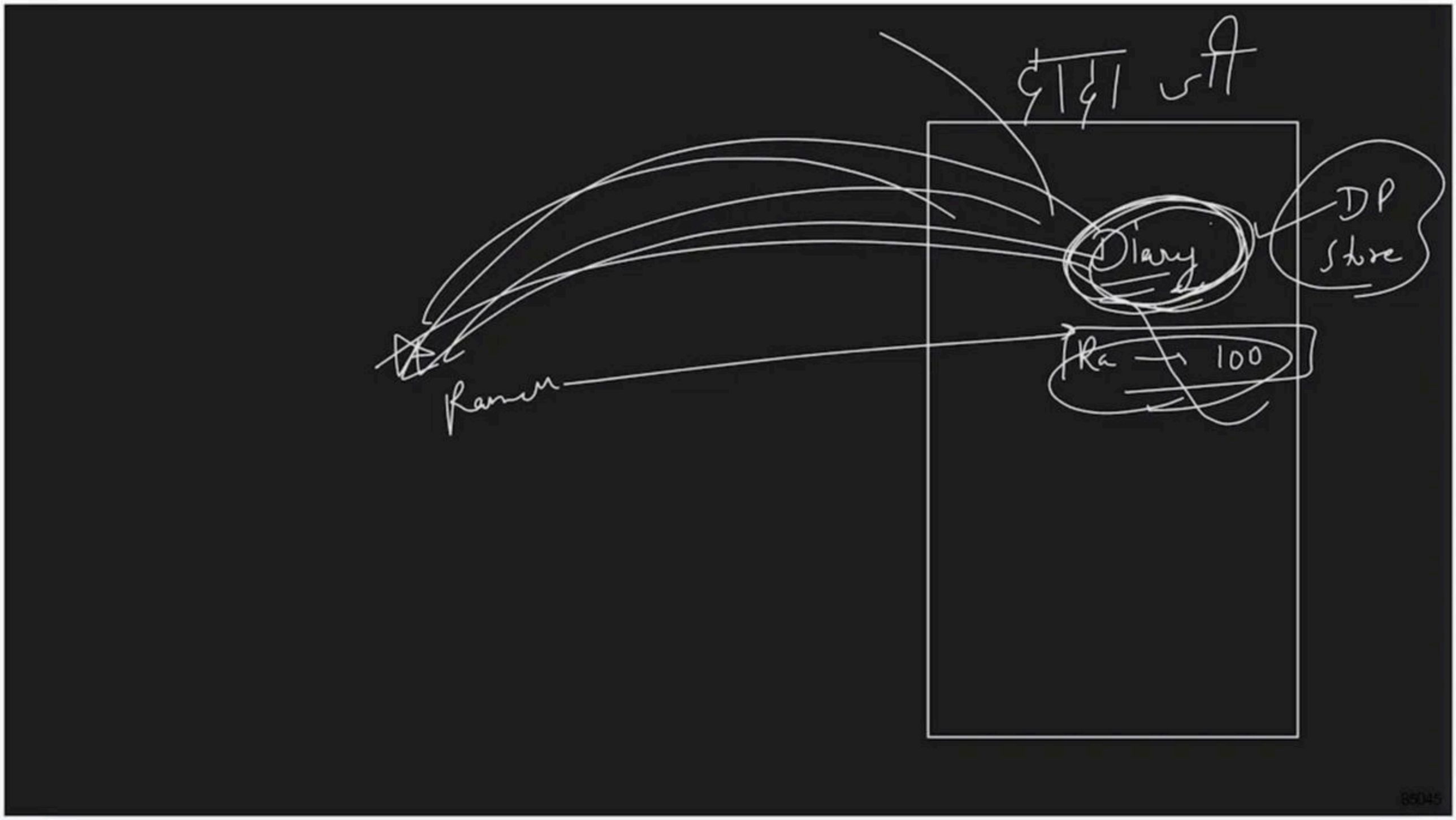

ABC
Dynamic Programming [DP]
Programming technique
problem
A
Overlapping Subproblem
B
Optimal Substructure
fib
RE
f(4)
f(3)
f(2)
f(1)
f(0)
f(4) → f(3) + f(2)

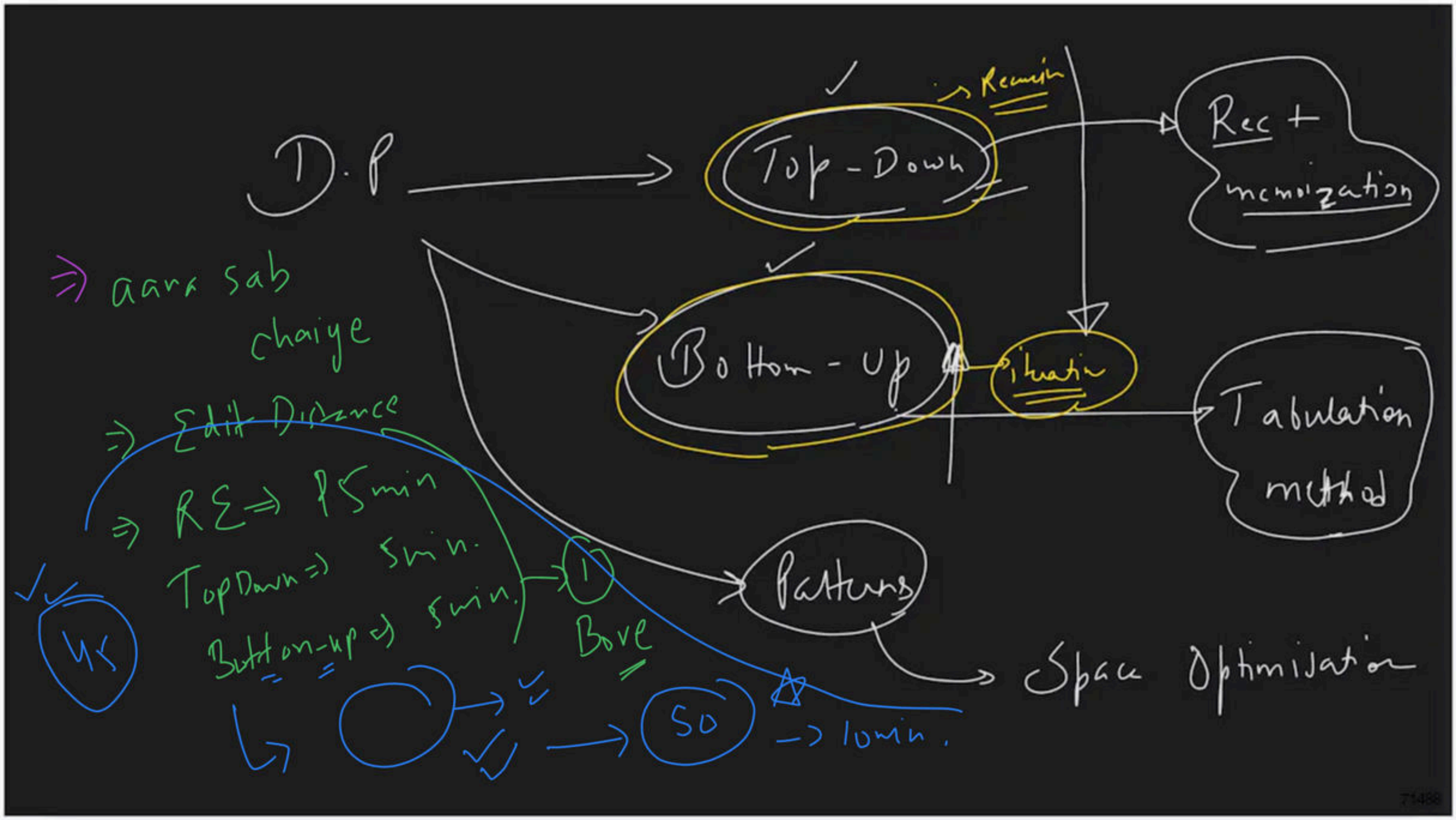
D.P → ?

→ solution

→ same problem → base base solve karke




Diary
DP
Store
Ra → 100
Ram


D.P
Top-Down
Recursion
Rec + memoization
Bottom-up
Situation
Tabulation method
Patterns
Space Optimization
aaya sab chaiye
Edit Distance
RE
PS
min
TopDown
Bottom-up
Bore
50
10 min.

→ Fibonacci no :-

Algorithm → Rec

0, 1, 1, 2, 3, 5, 8, 13, 21 - - - - - -

R.R →

Recursion

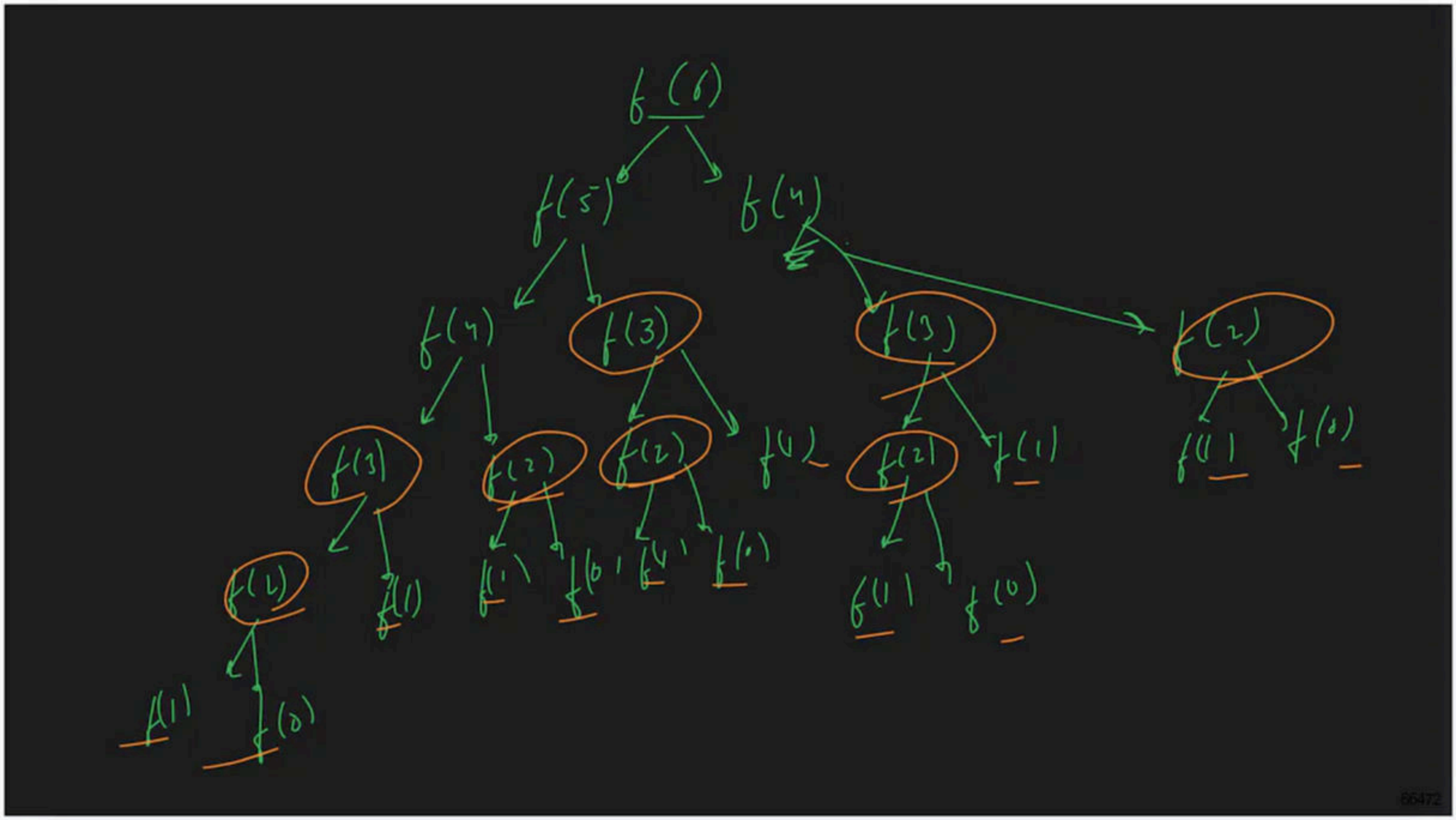
$$f(n) = f(n-1) + f(n-2)$$

n^{th} fibonacci no = $(n-1)^{th}$ fib no + $(n-2)^{th}$ fib no

let $n = 6$

$$f(6) = f(5) + f(4)$$

6^{th} fib no = 5^{th} fib no + 4^{th} fibonacci


f(6)
f(5)
f(4)
f(4)
f(3)
f(3)
f(2)
f(3)
f(2)
f(2)
f(1)
f(2)
f(1)
f(1)
f(0)
f(1)
f(0)
f(1)
f(0)
f(1)
f(0)
f(1)
f(0)
f(1)
f(0)

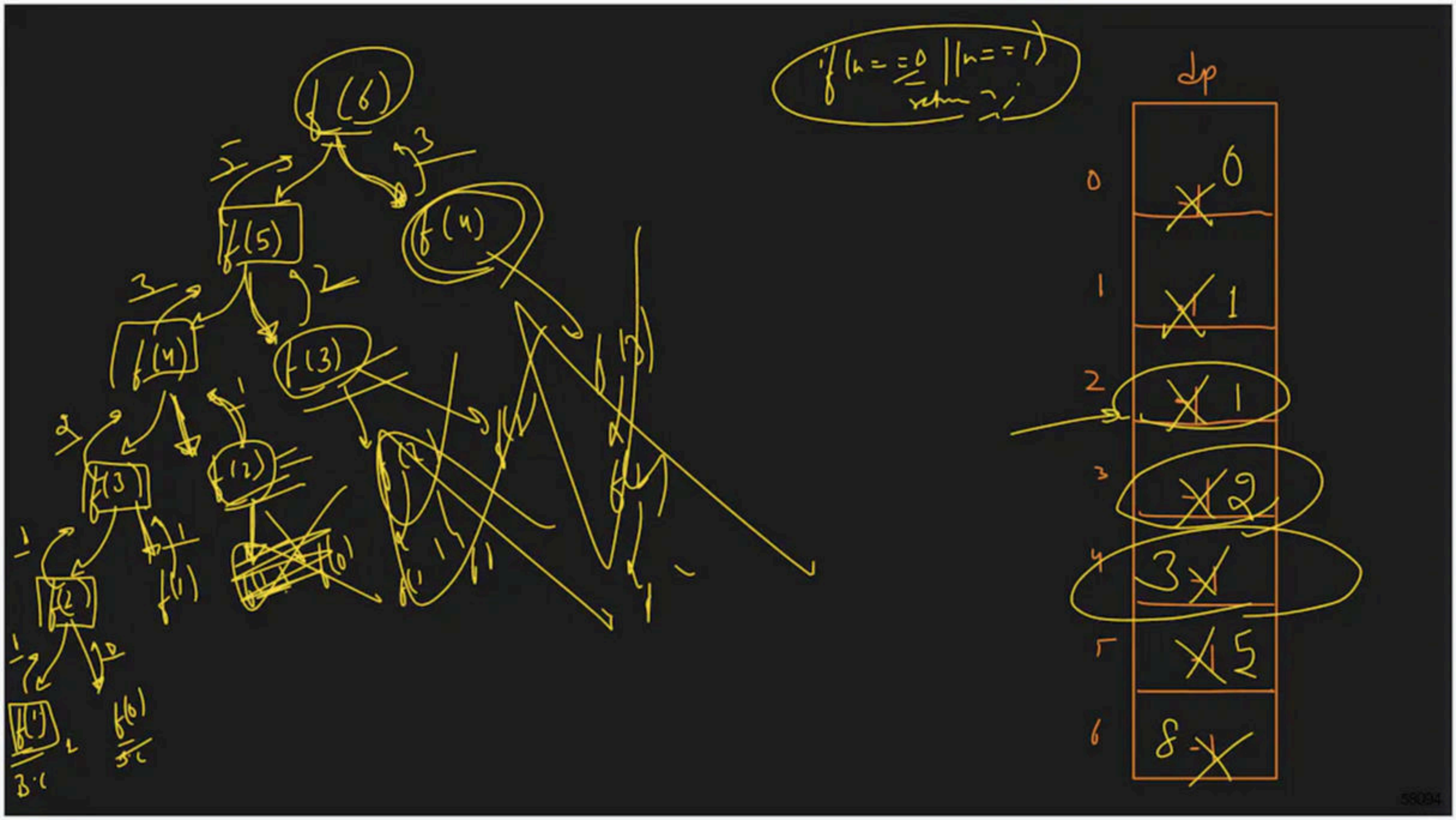
Main

① Create a dp array

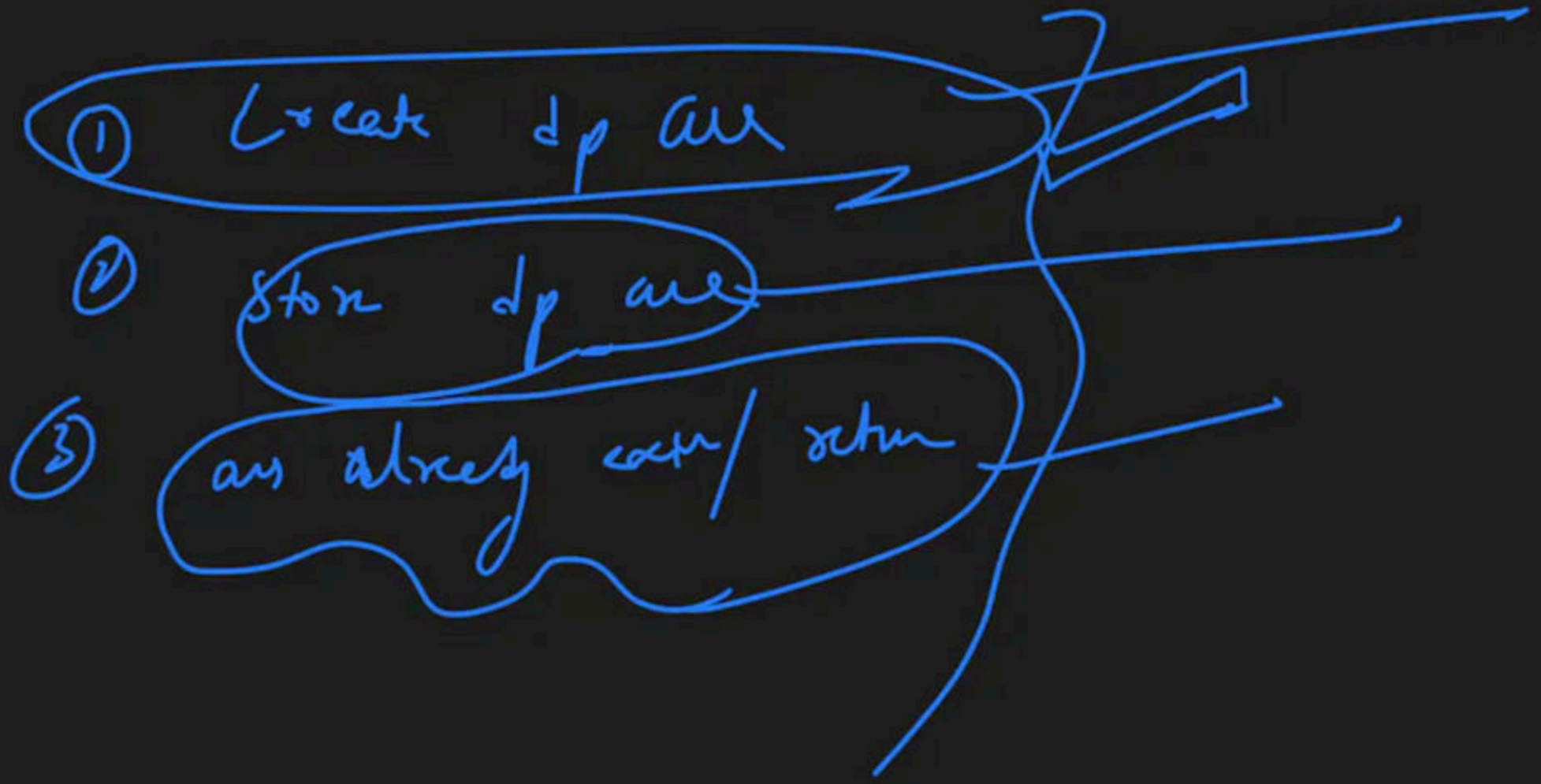
② store/return ans in dp array

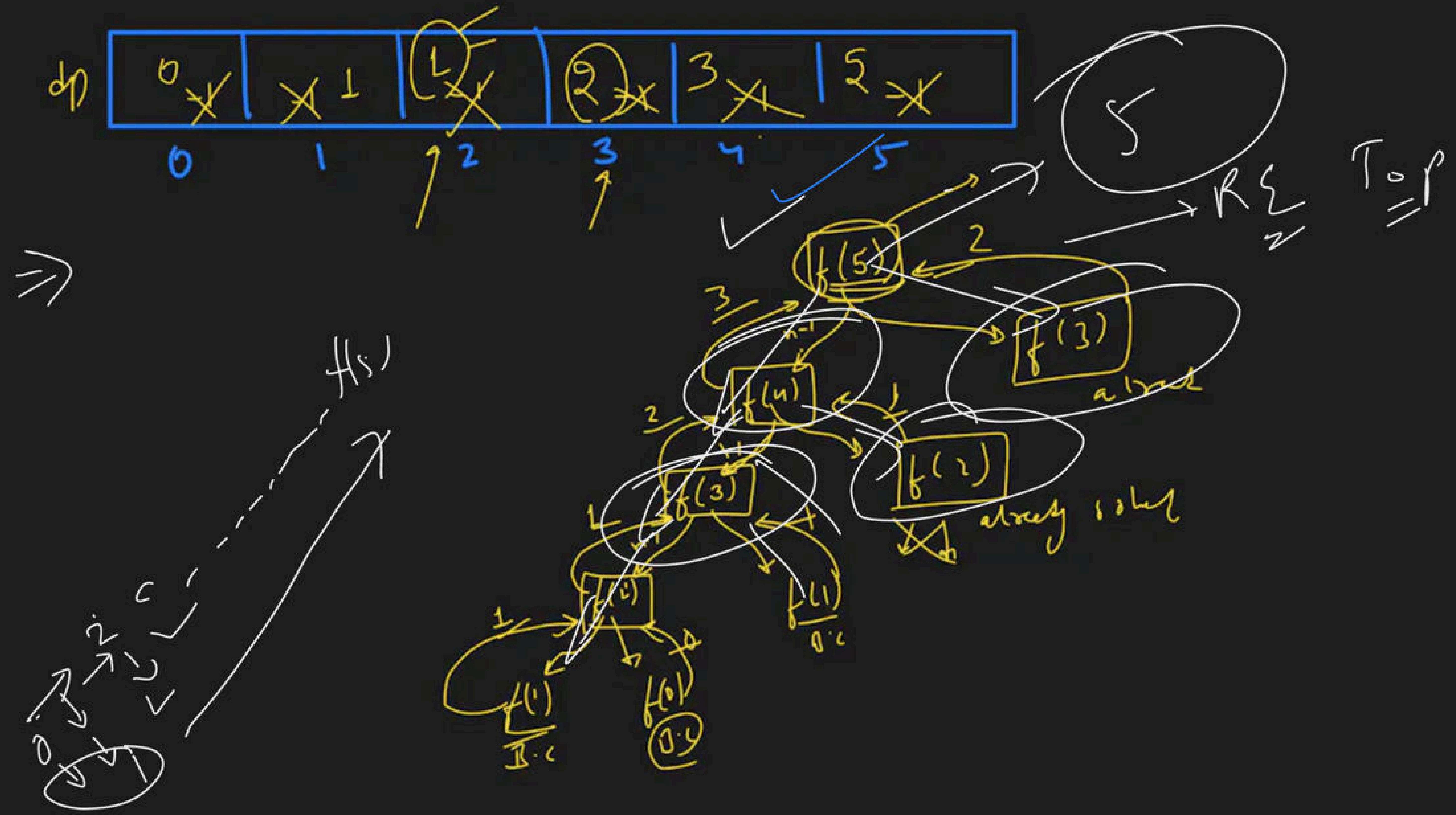
③ if ans already exists in dp array, then return

Step


if (n == 0 || n == 1)
return n;
dp
0
1
2
3
4
5
6
f(6)
f(5)
f(4)
f(4)
f(3)
f(3)
f(2)
f(2)
f(1)
f(1)
f(0)

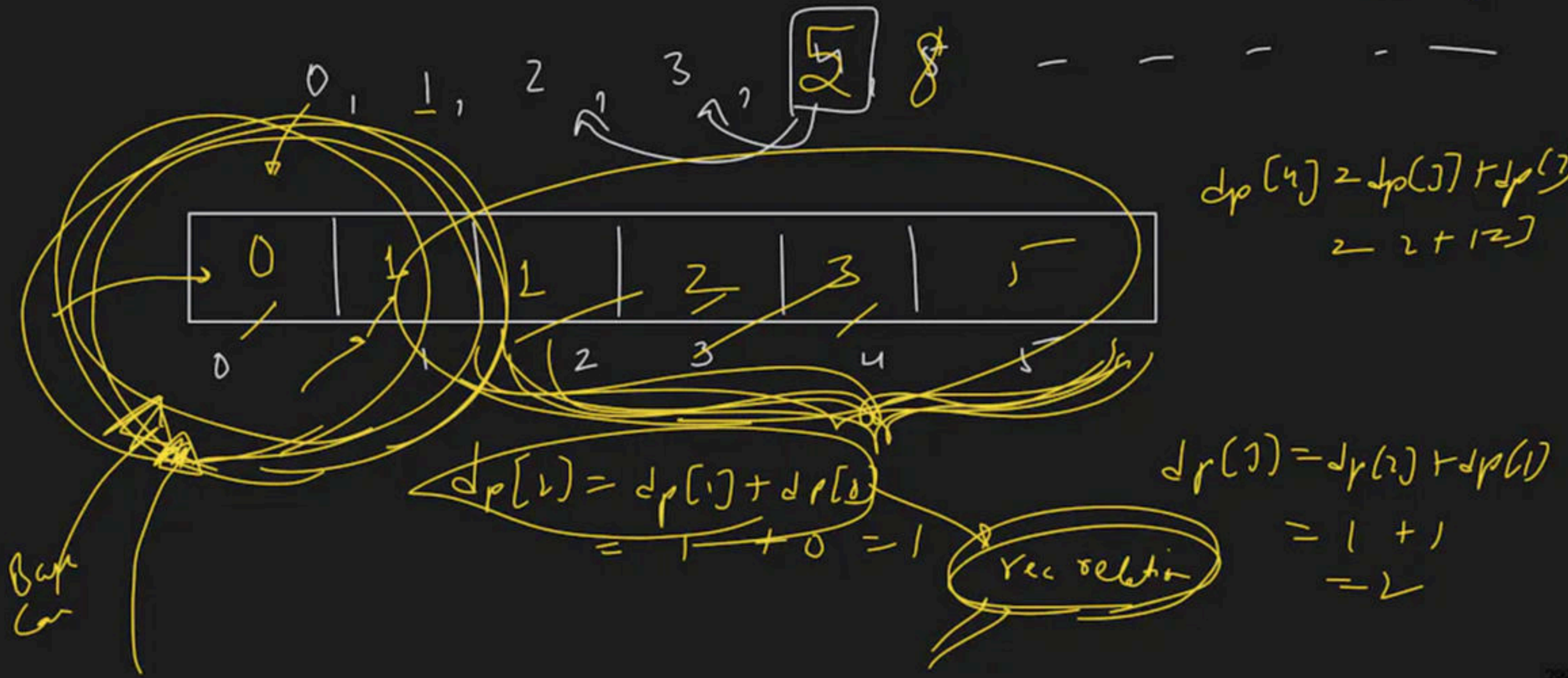

Memoized
plain Rec
dp
0
1
2
3
4
-1


1
Locate dp arr
2
Store dp arr
3
arr already calc/ return



dp
0 1 2 3 4 5
f(5)
f(4)
f(3)
f(2)
f(1)
f(0)
already solved
RE
Top

→ Tabulation method (Bottom-up)



Tabulation

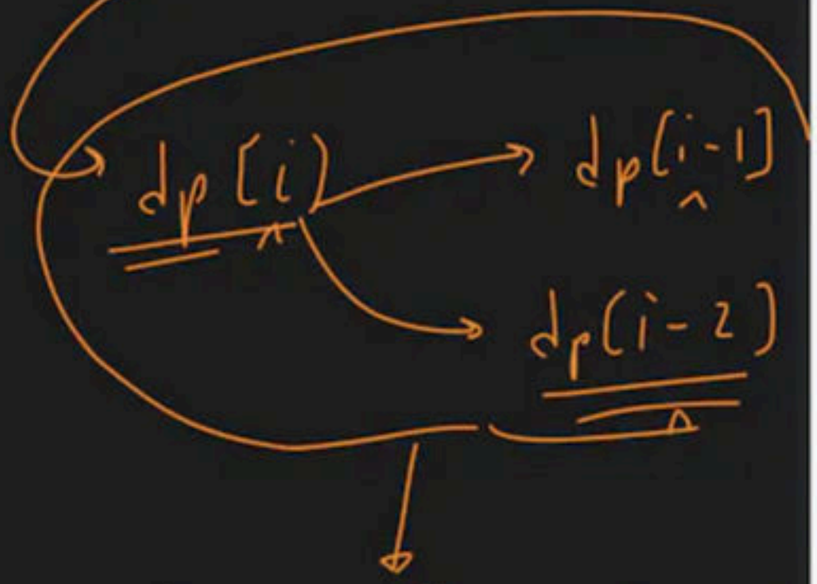
1. → create dp array
2. → Analyse -> Base case → & fill dp array accordingly
3. → fill remaining dp array using logic/formula of recursive relation

Space Optimisation :-

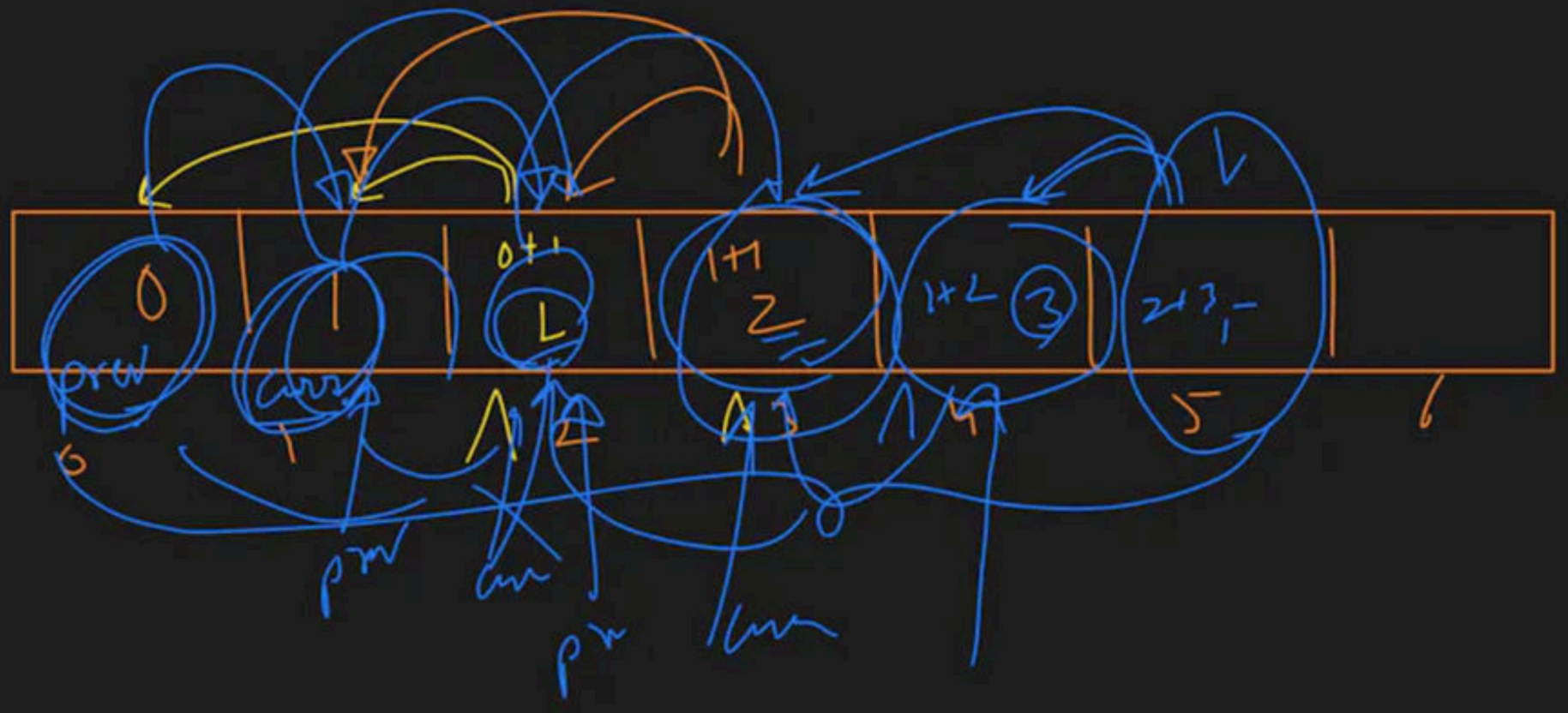
int solve

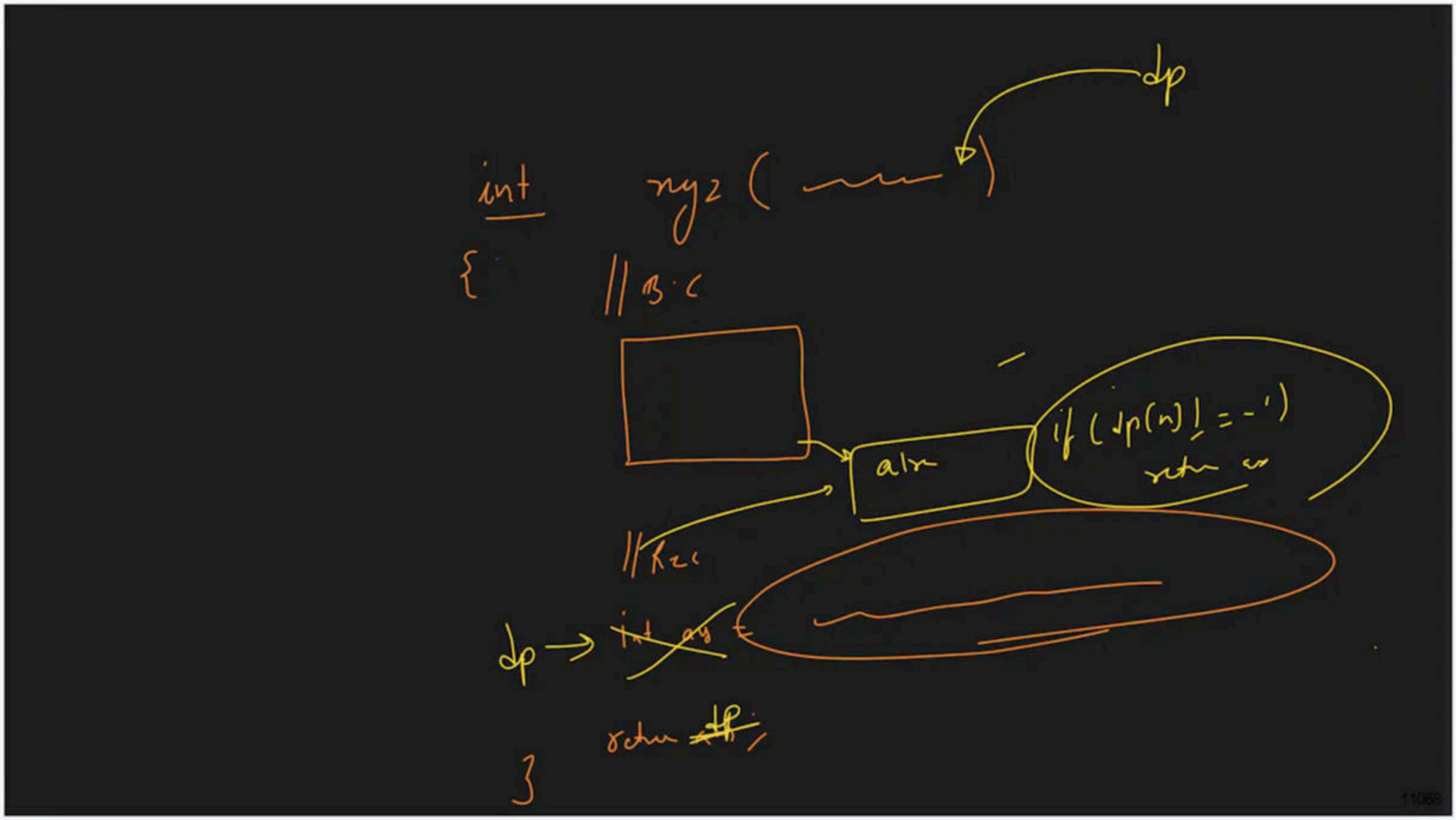
```
int solveUsingTab(int n)
{
    vector<int> dp(n+1, -1);

    dp[0] = 0;
    dp[1] = 1;

    for(int i=2; i<=n; i++)
    {
        dp[i] = dp[i-1] + dp[i-2];
    }
    return dp[n];
}
```

Space Optimization



Kya koi pattern bnaa hai

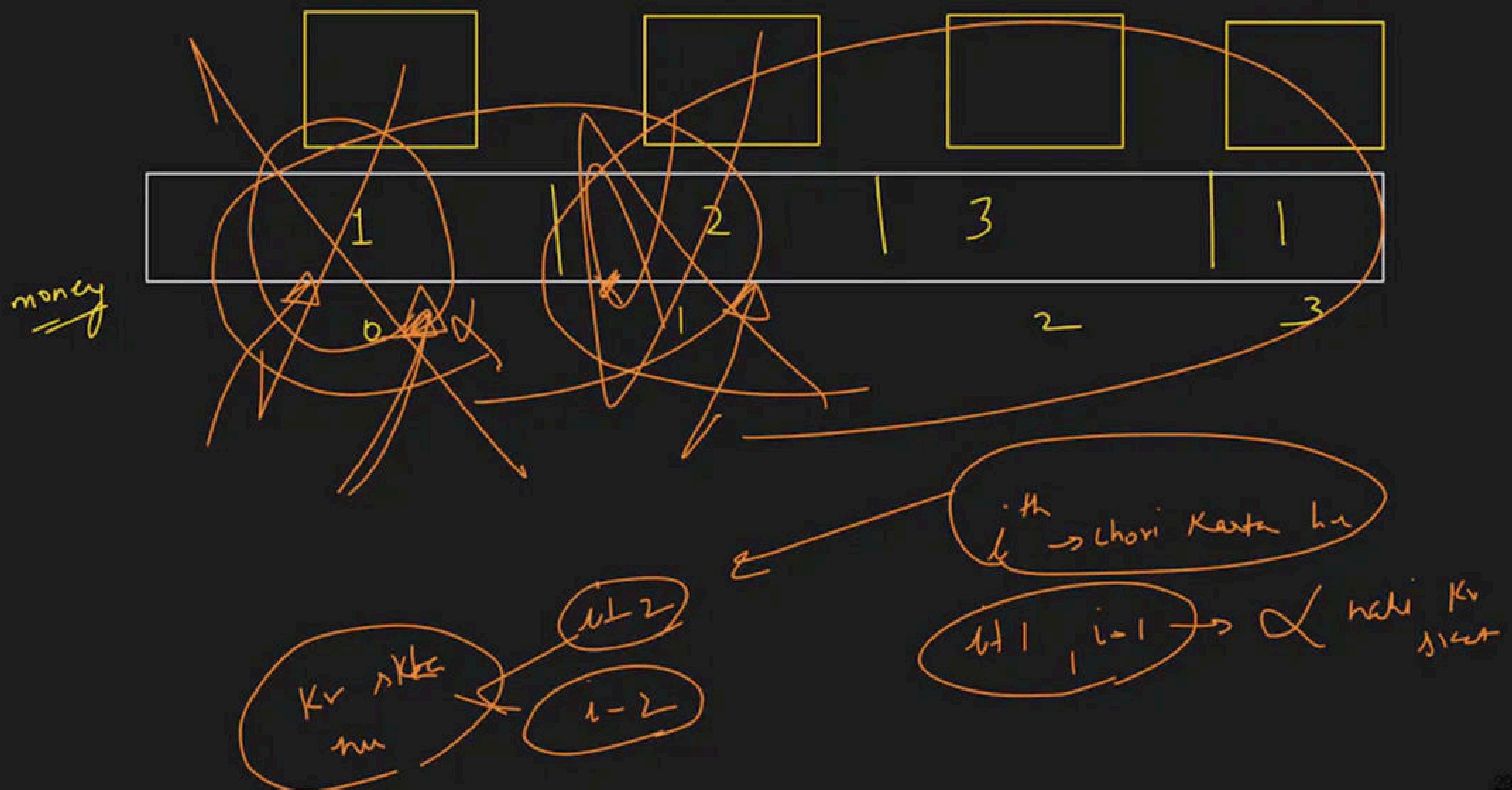


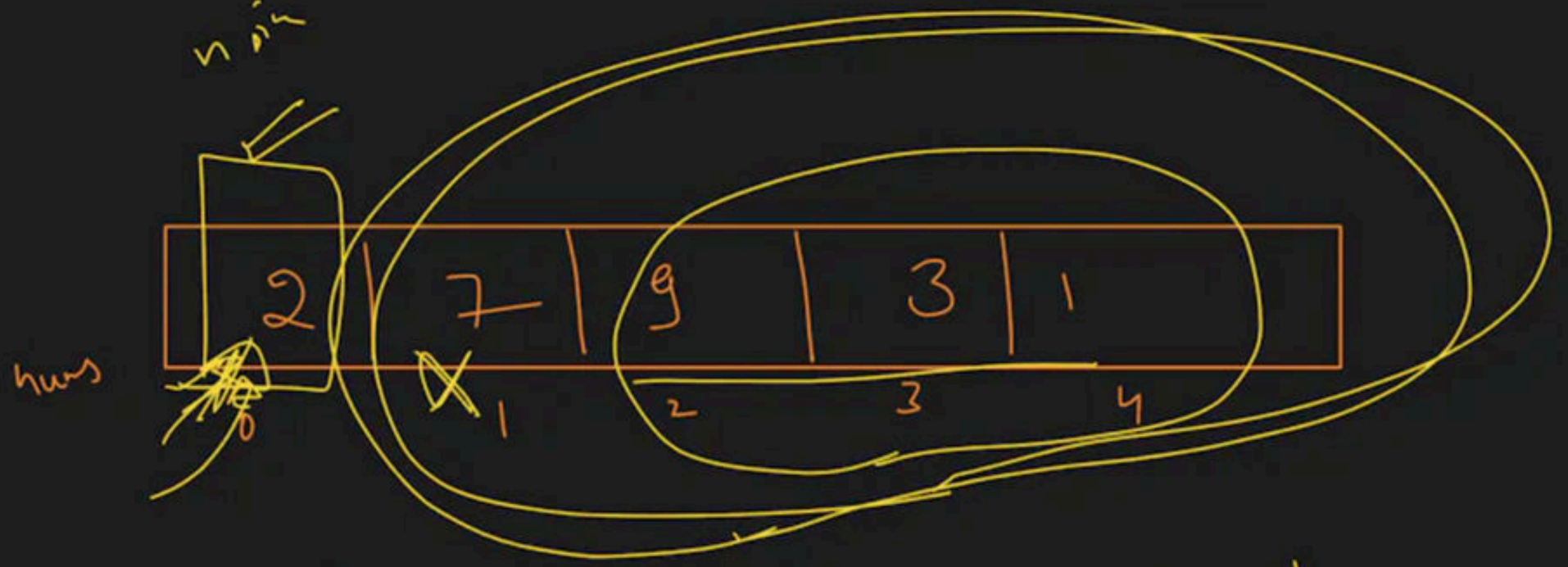

dp
int xyz ( )
{
// B.C
alx
if (dp(n) != -1)
return
// Rec
dp →
return
}

Dynamic Programming Class-2

Special class

Love Babbar • Jan 8, 2024


1
2
3
1
money
0
1
2
3
ith → chori karta hu
i+1, i-1 → nahi kr sakta
kr sakta hu
i+2
i-2



case → include → $2 + f(n-2) \longrightarrow$ ans1

case → exclude → $0 + f(n-1) \longrightarrow$ ans2

max → find Ans

i=0

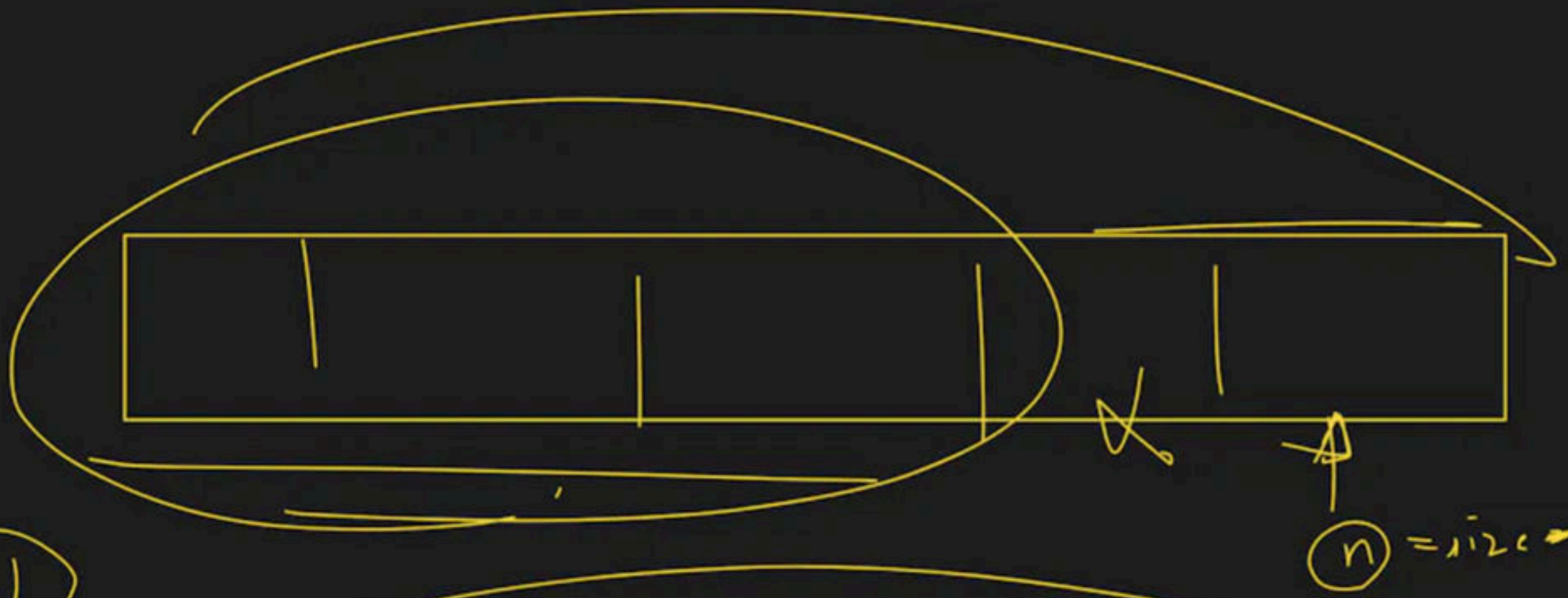
i+1

i+2

L → R

int include = nums[i] + solve(i+2)

int ~~output~~ exclude = 0 + solve(i+1)

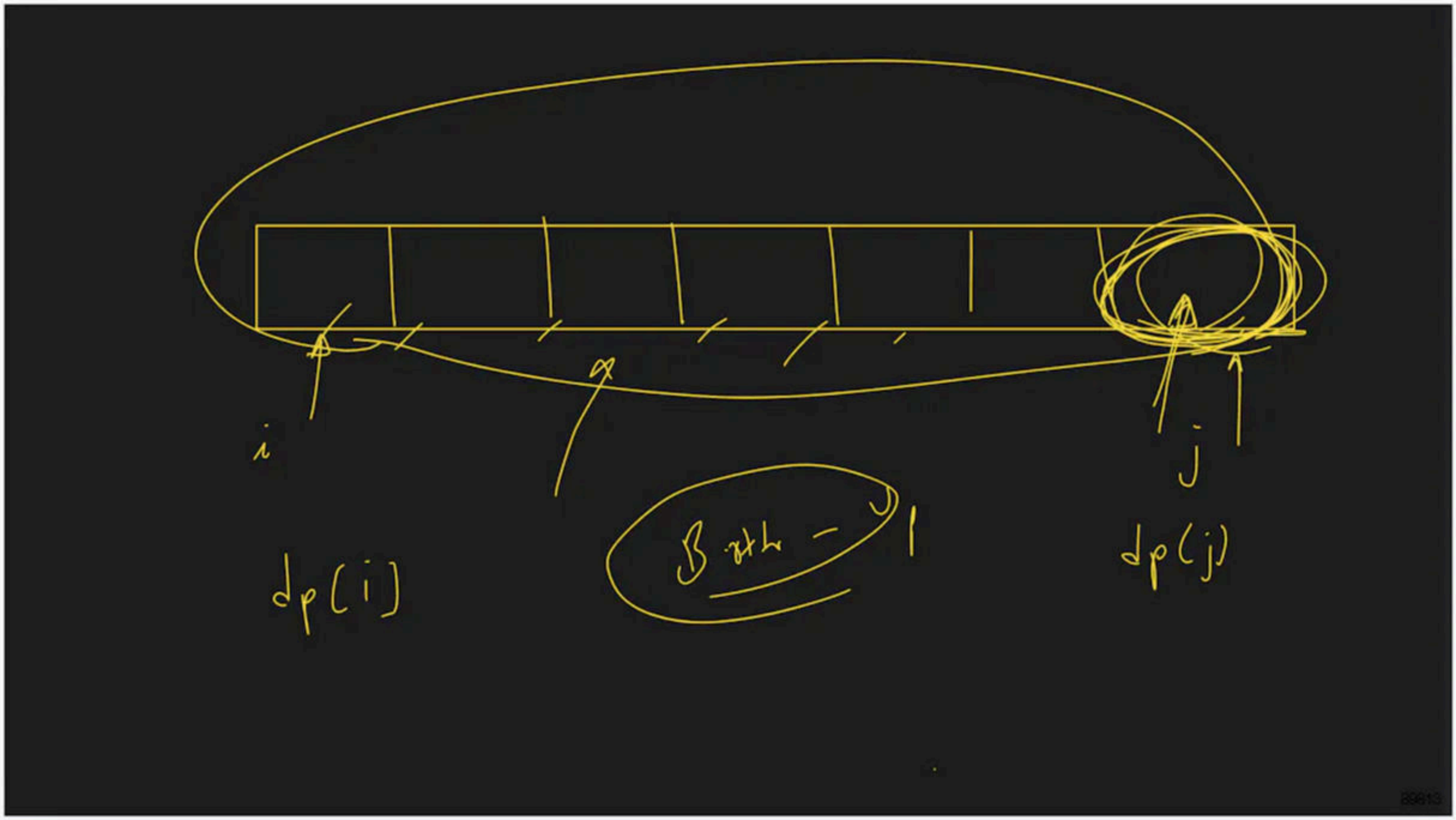


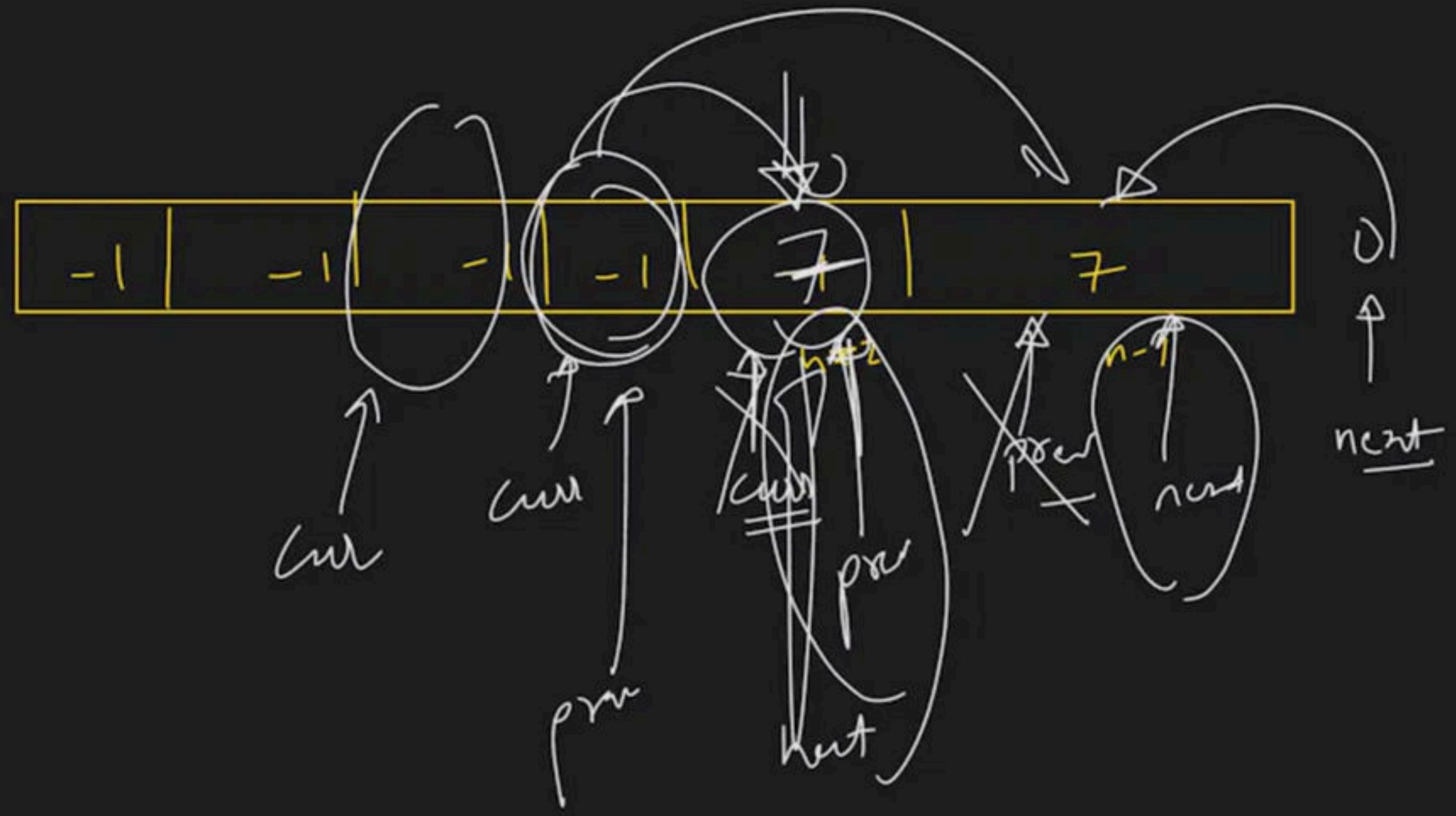
R → L

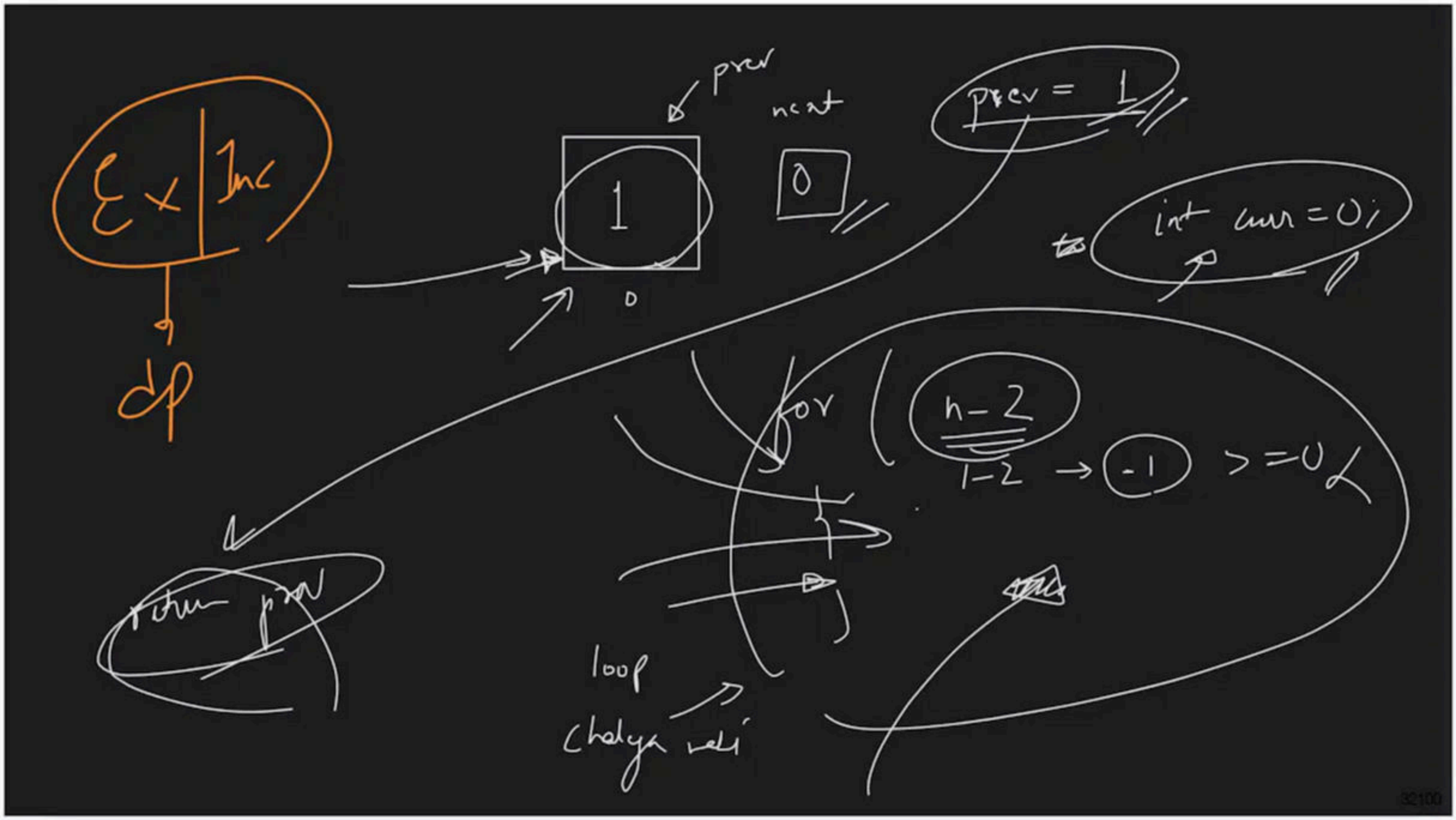
L → R

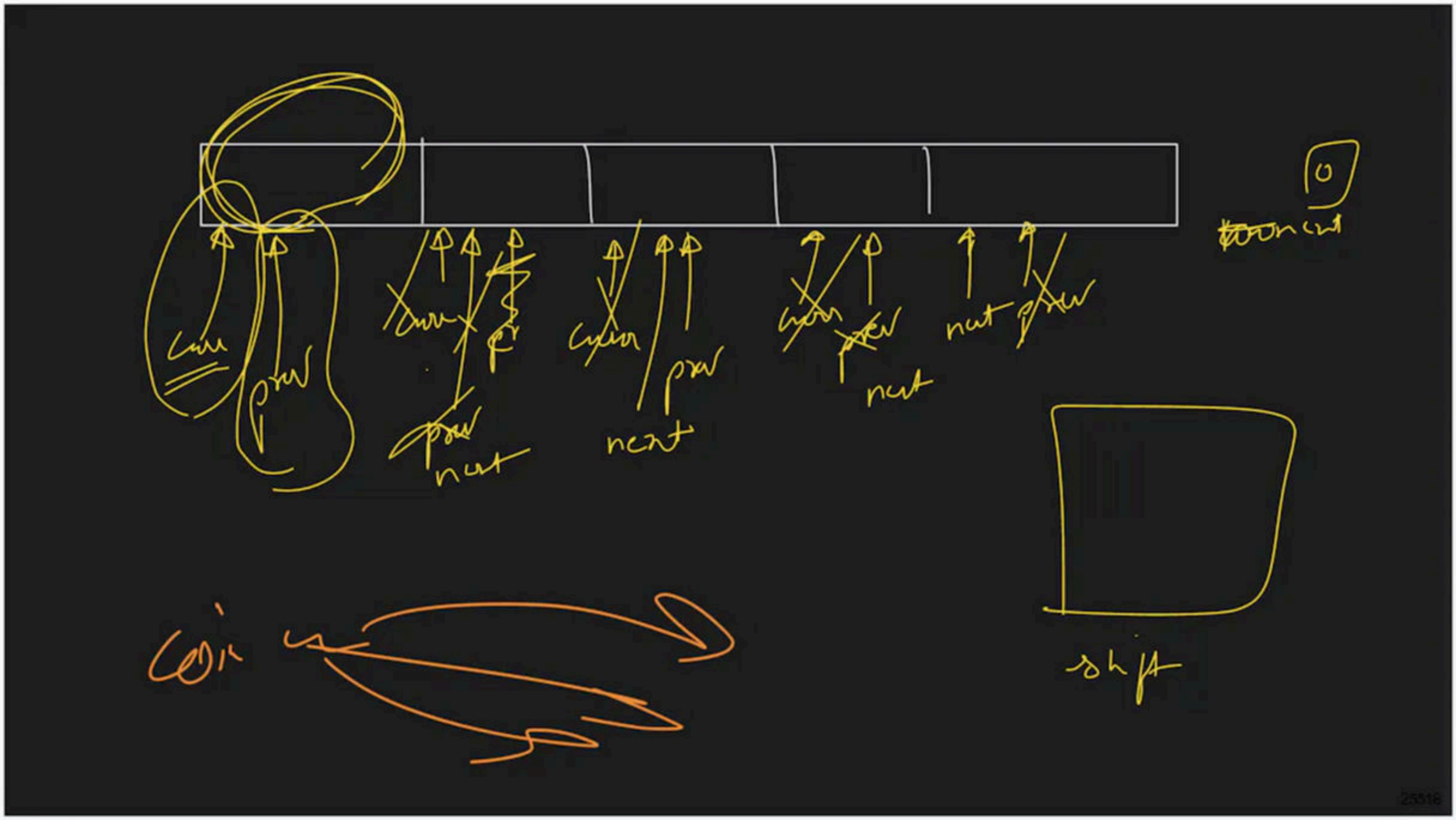
include = num(n) + solve(n-2)

exclude = 0 + solve(n-1)


i
j
dp(i)
dp(j)

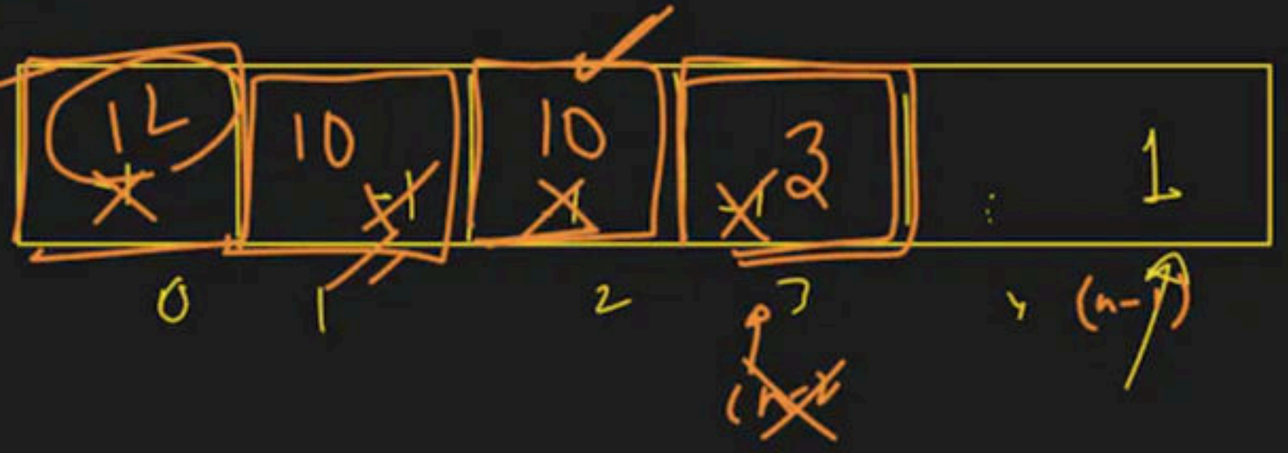

-1
-1
-1
-1
7
7
curr
curr
prev
curr
prev
next
prev
curr
next


Ex | Inc
dp
prev
next
1
0
0
Prev = 1
int curr = 0;
to
for
n-2
1-2
-1
>=0
return prev
loop
Chalya rahi


curr
prev
curr
prev
next
curr
prev
next
curr
prev
next
curr
prev
next
0
shift

ind → 10 –
ex → 10/

ind = 2+10 = 12
ex = 10

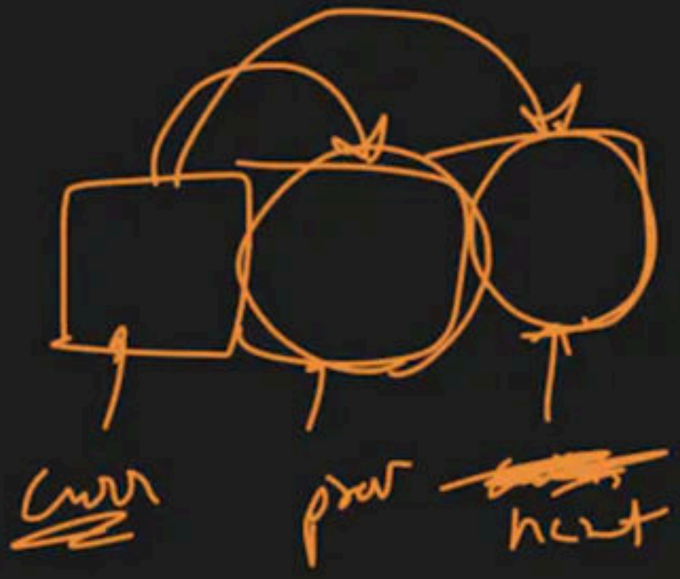


include = 3+0
= 3

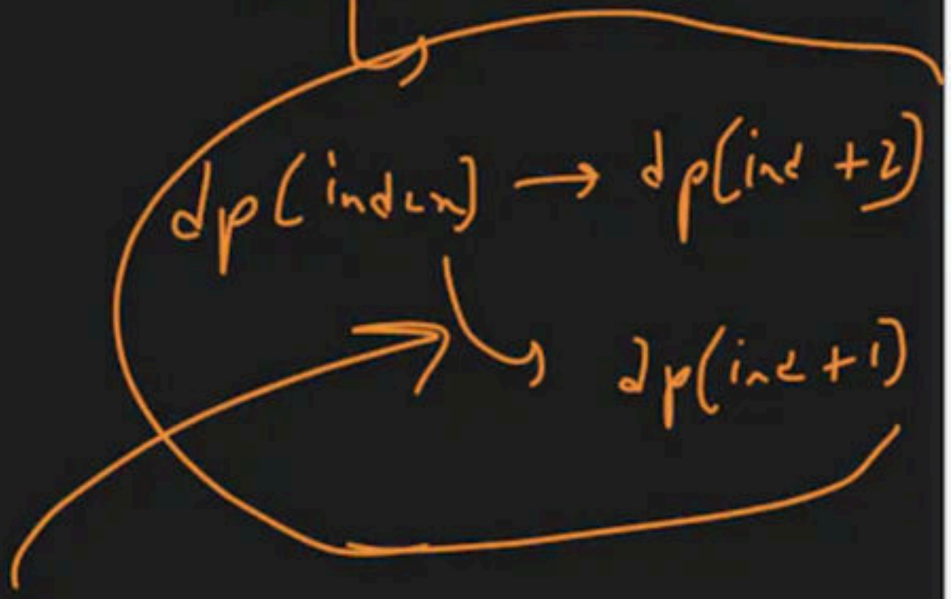
ex → 2

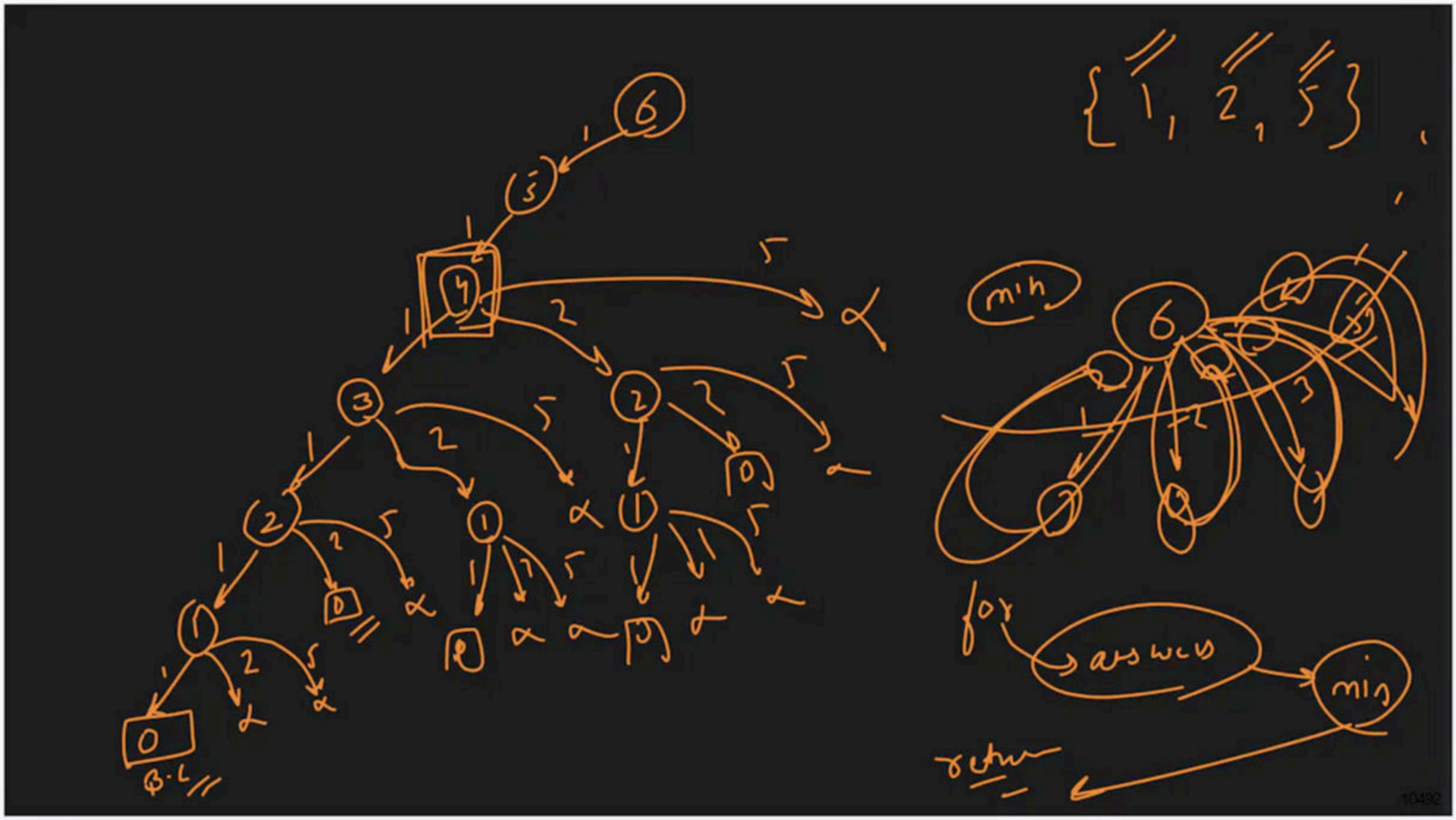
include 10
ex - 3

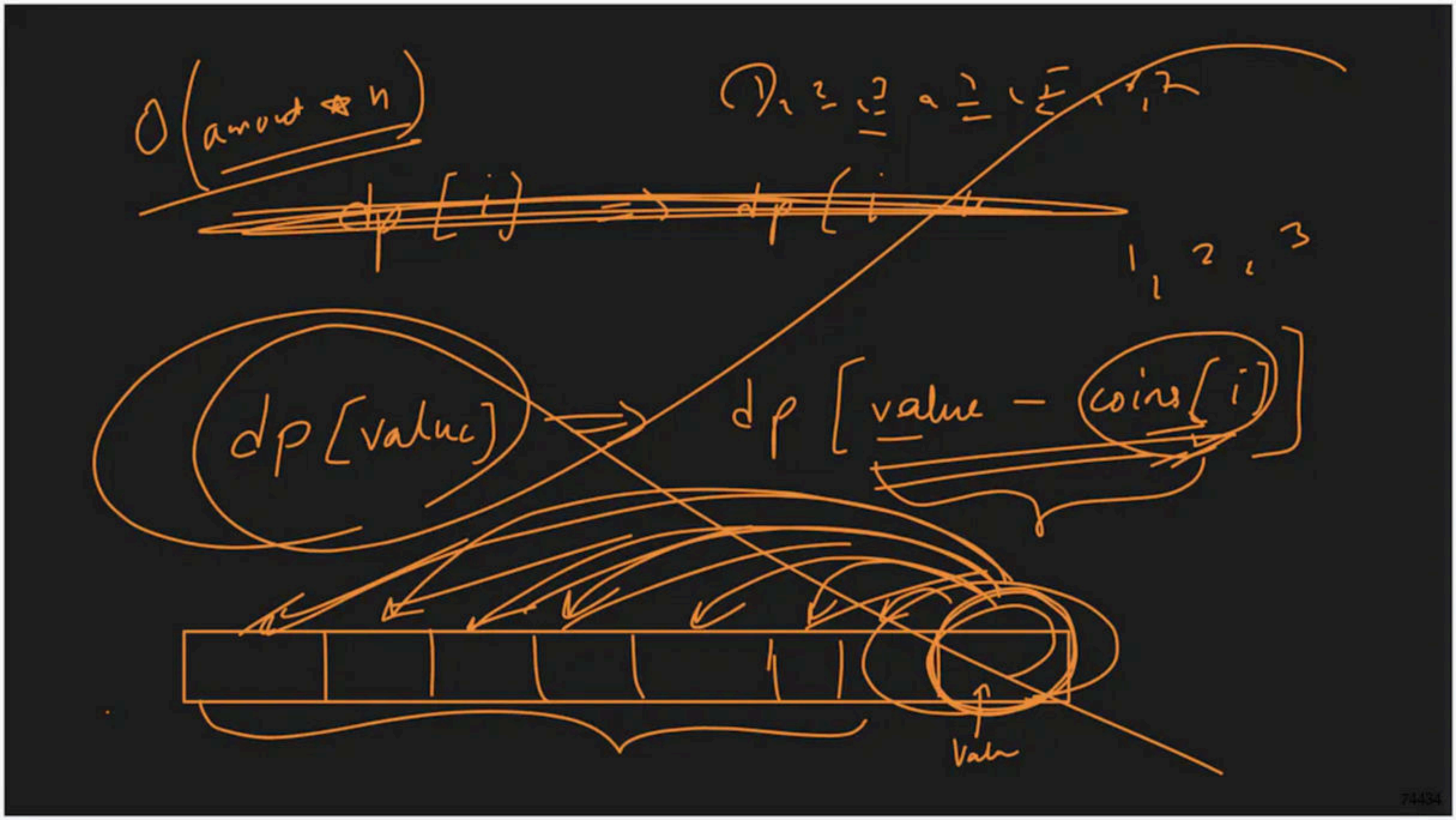
for (n-2 ⟶ 0

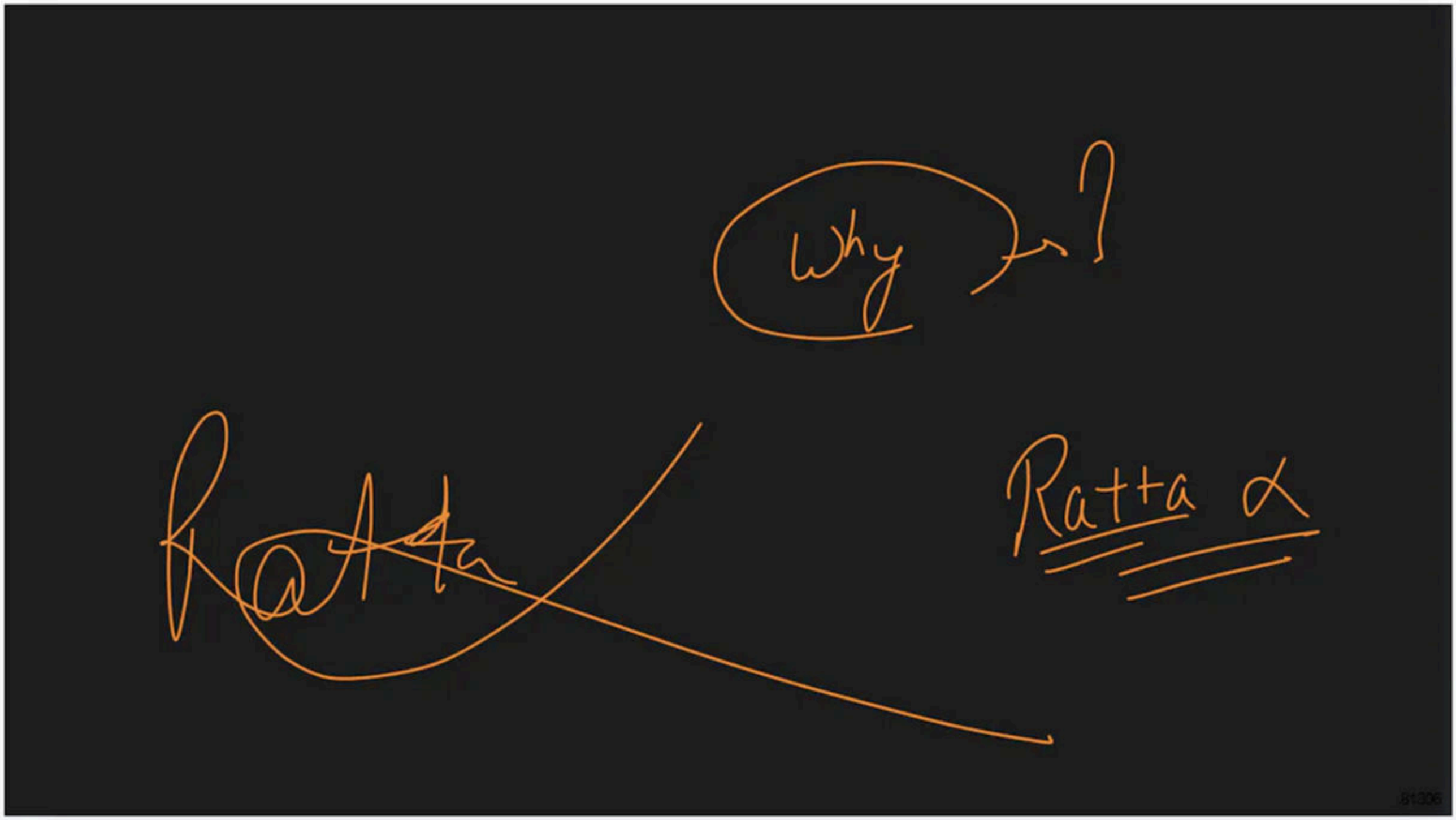


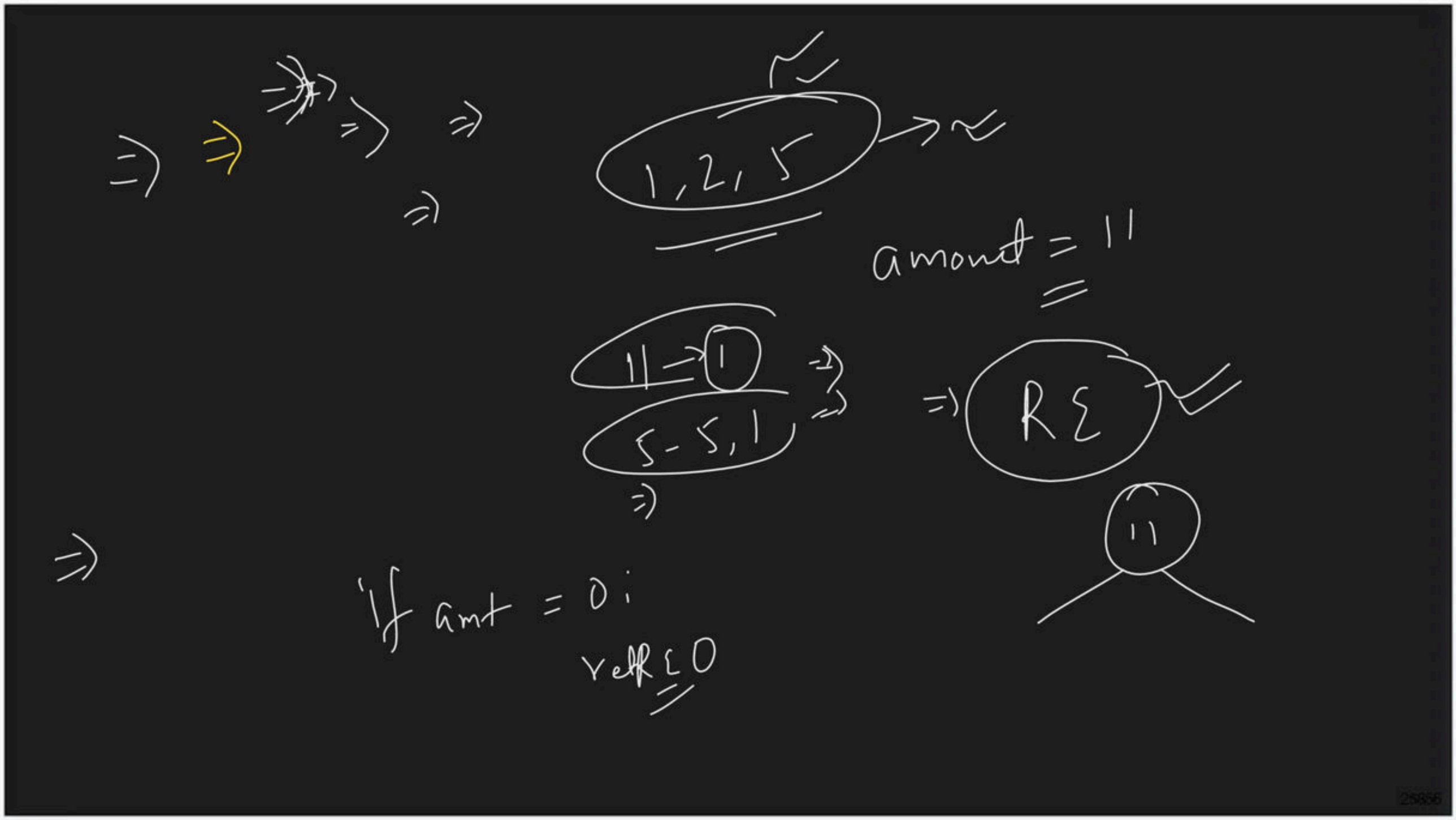
Code → Tabulation




min
for
return


O(amount * n)
dp[i]
1, 2, 3
dp[value]
dp[value - coins[i]]
Value


Why ?
Ratta
Ratta


1, 2, 5
amount = 11
11 → 1
5 - 5, 1
RE
11
if amt = 0;
retR = 0

Dynamic Programming Class-3

Special class

Love Babbar • Jan 10, 2024

→ Painting fence

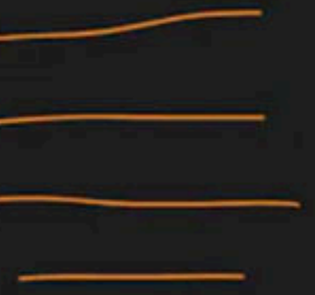
n → post, K → colors

Condt → no two adjacent posts have the same color

n = 1
K = 3 → R, G, B

R | G | B → ans = 3

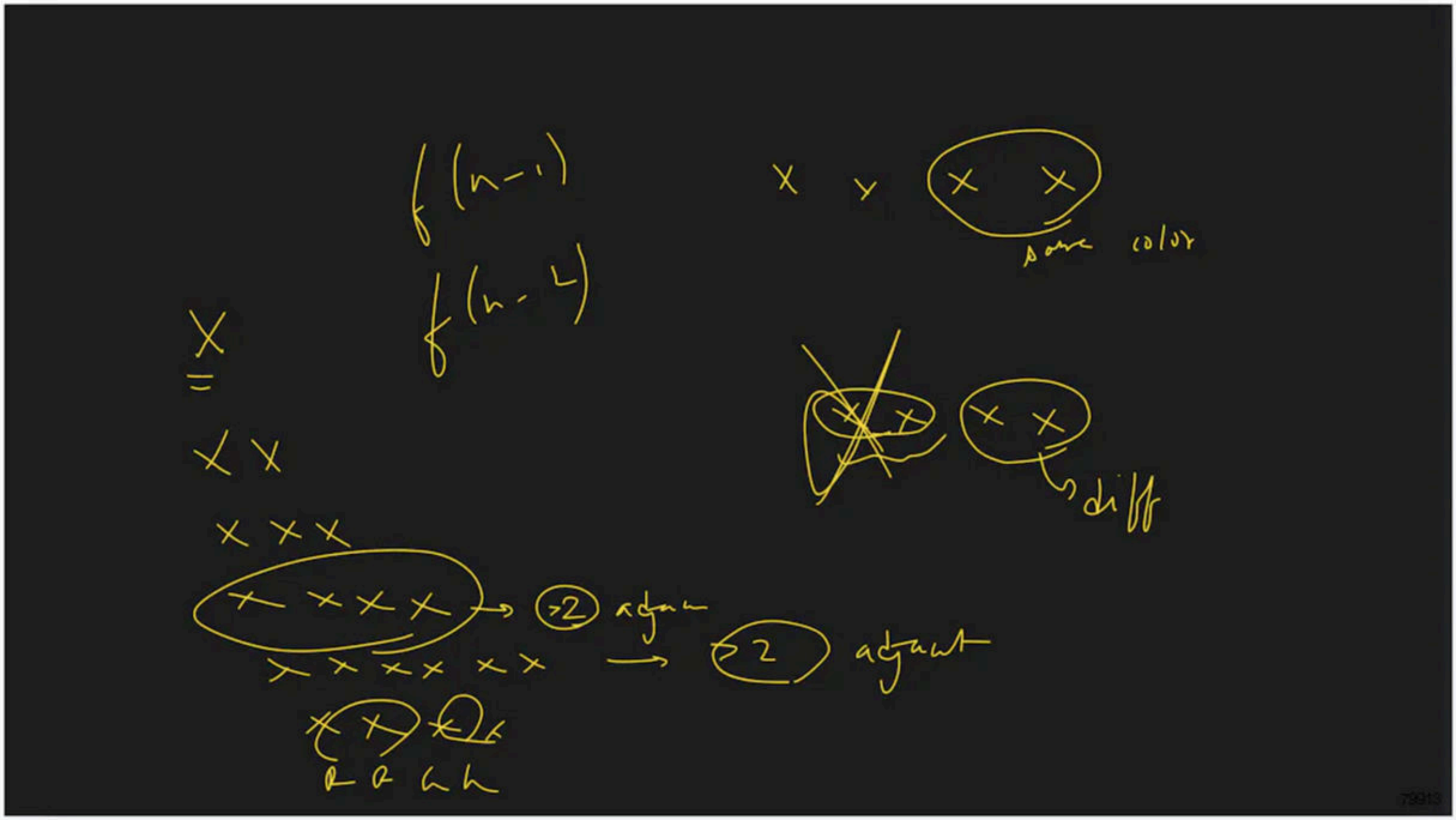
n = 3 →

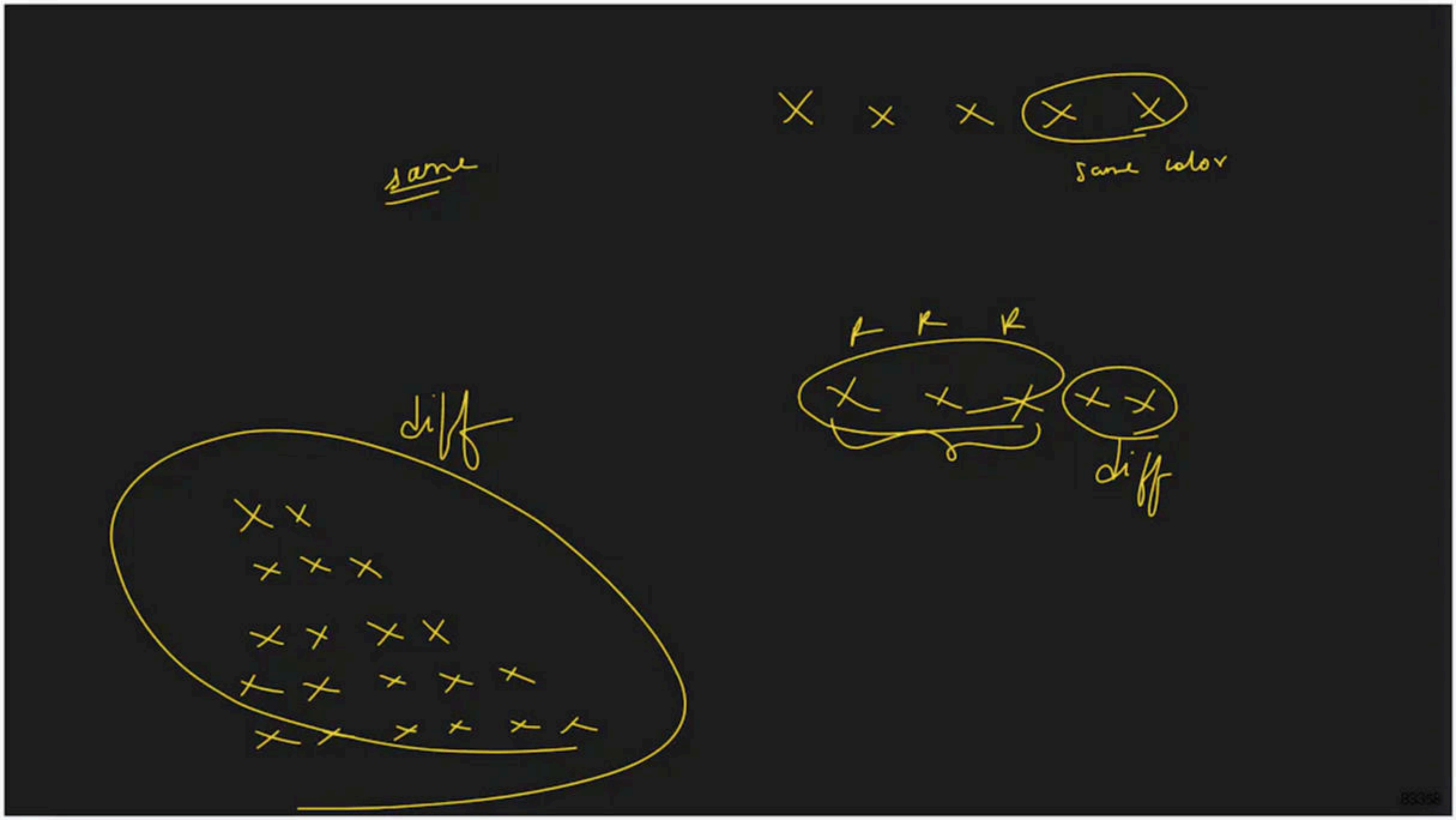


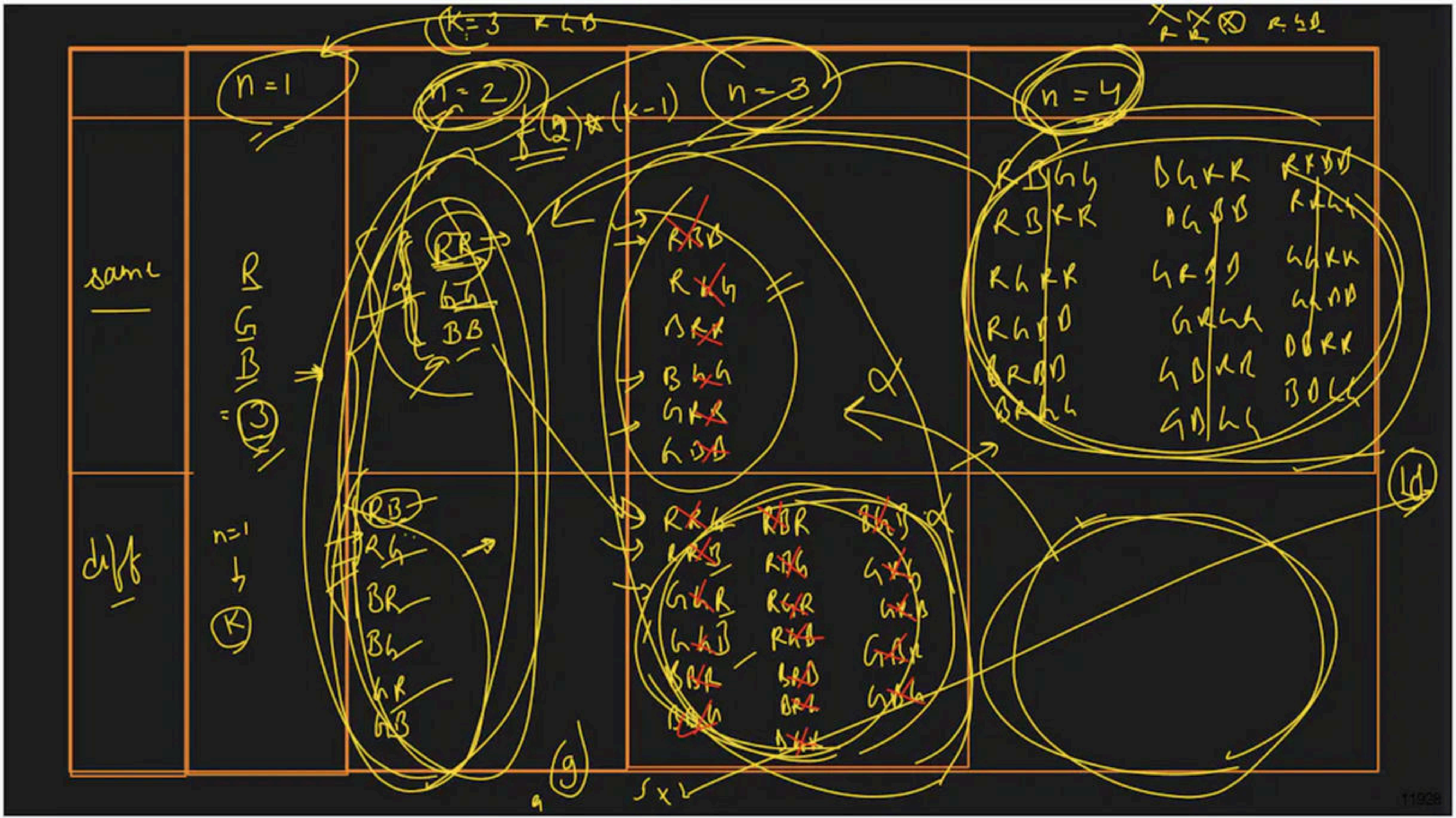
n = 2 →

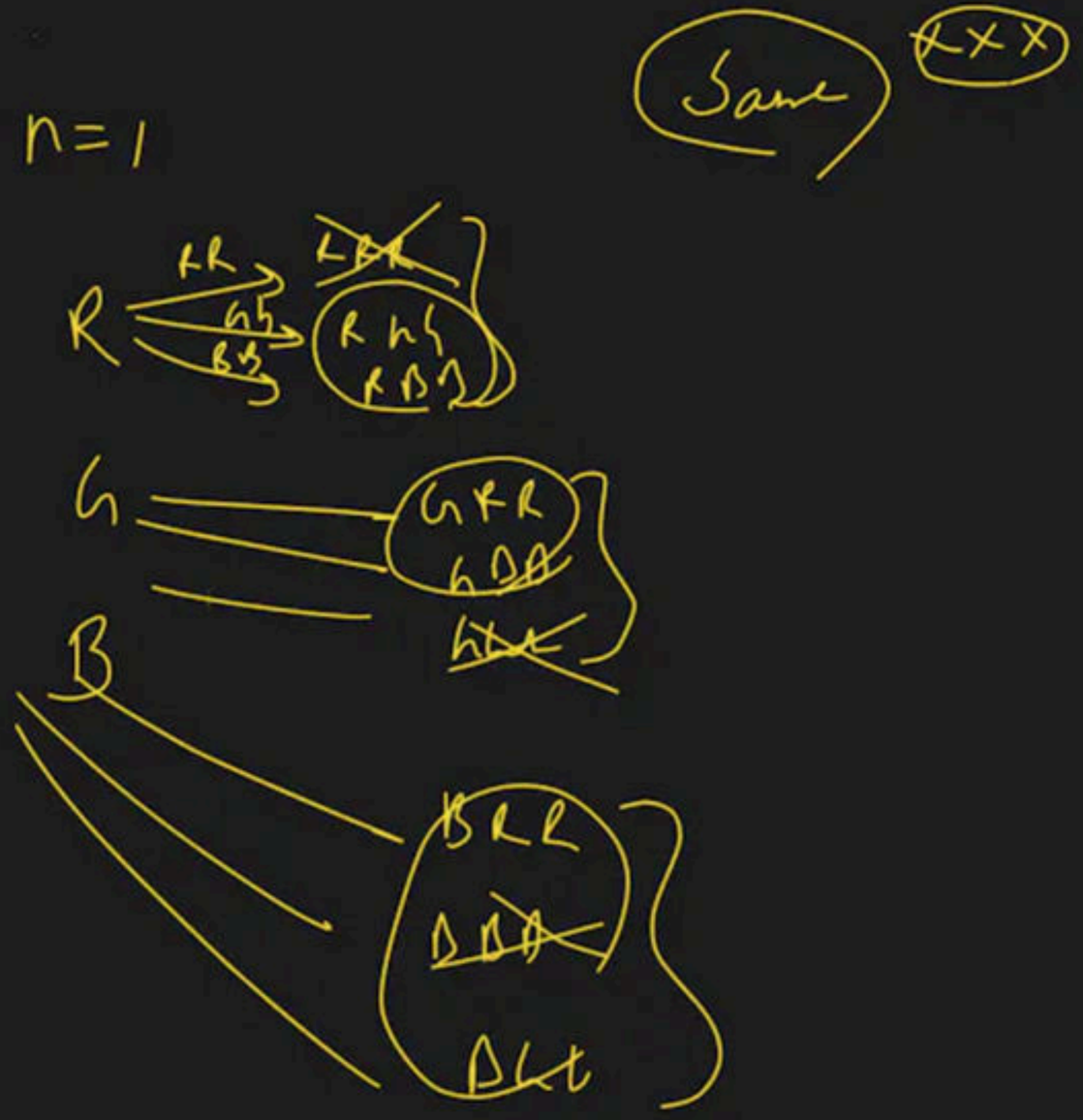
RR	GG	BB
RG	GR	BR
RB	GB	BG

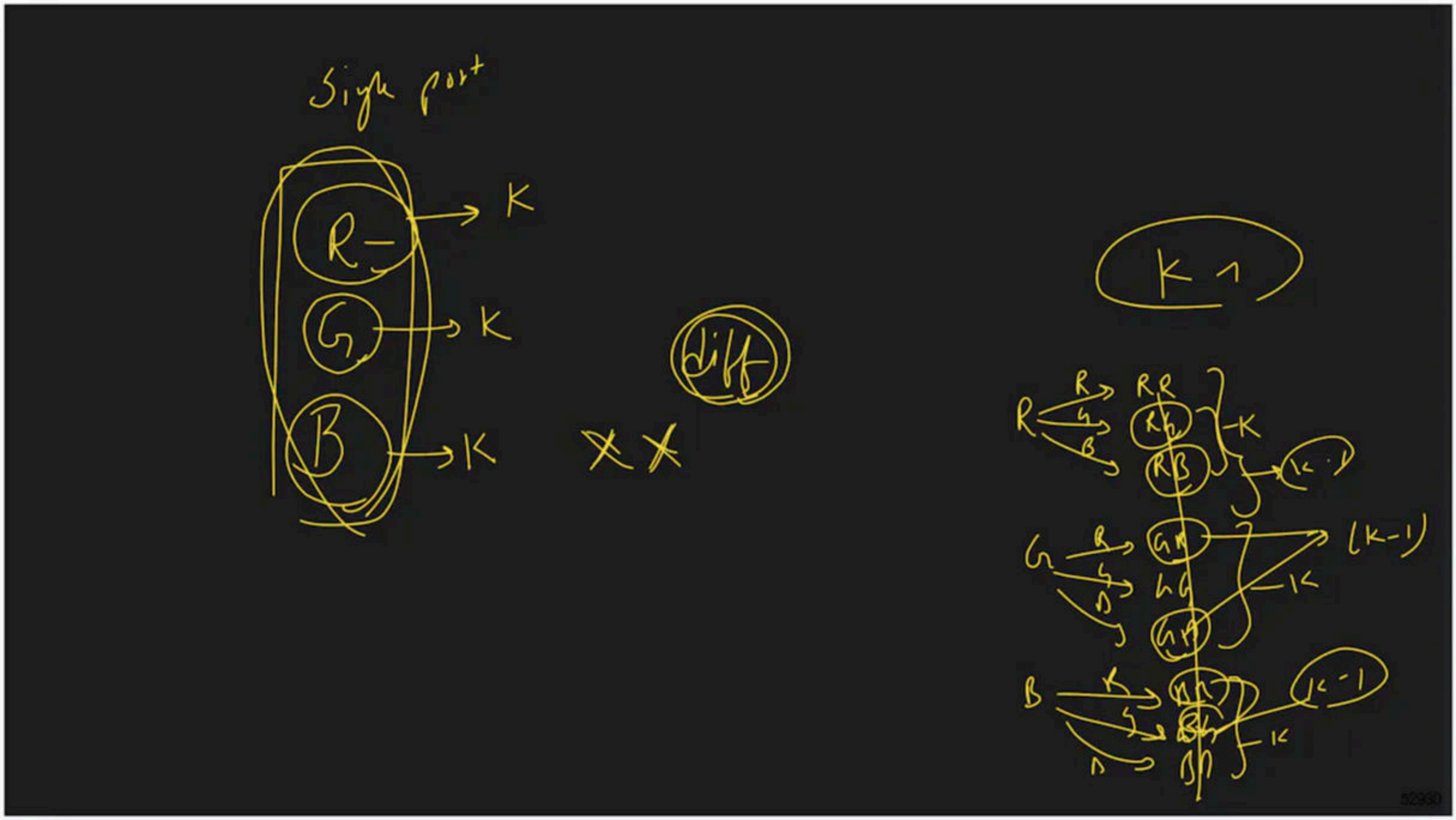
→ ans = 9


f(n-1)
f(n-2)
X
=
XX
XXX
>2 adjacent
>2 adjacent
same color
diff
Rough


same
same color
diff
diff


K=3
n=1
n=2
n=3
n=4
same
diff
R
G
B
3
K
RR
GG
BB
RB
RG
BR
BG
GR
GB
(k-1)
5x2
11938

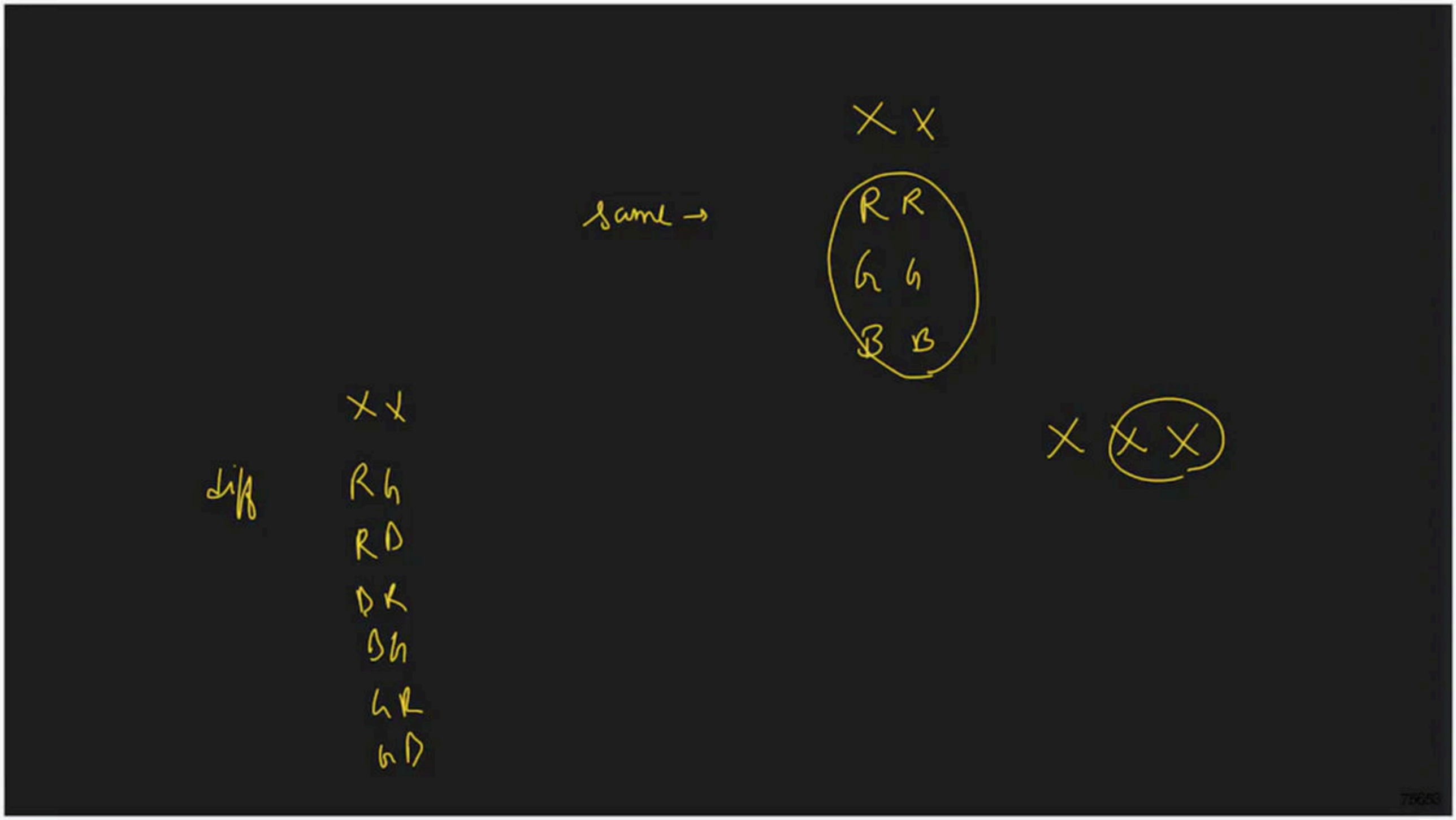

Same
n=1
R
G
B

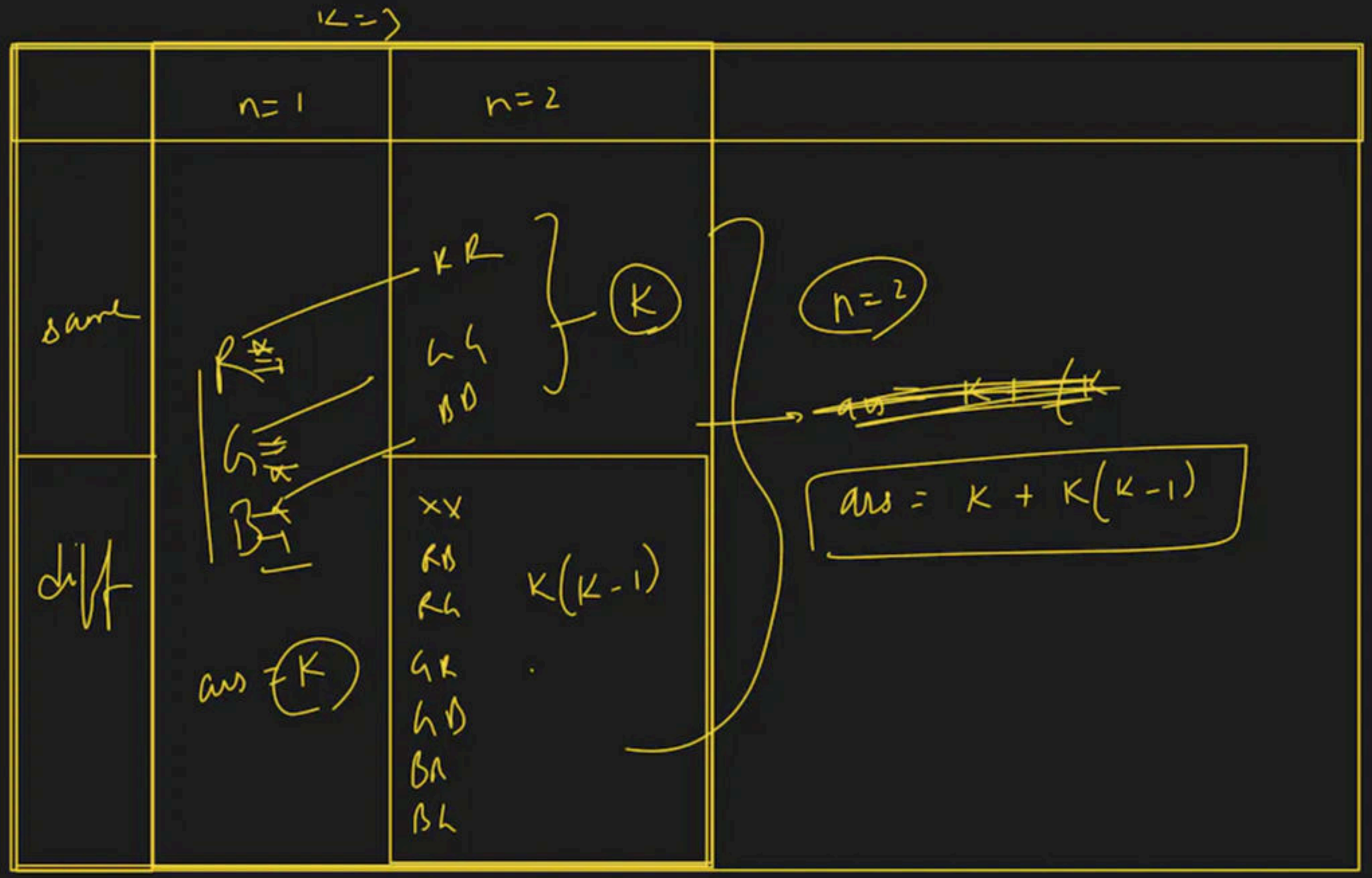

Sign post
R
G
B
K
K
K
diff
K
R
G
B
RR
RB
K
K-1
K-1
K
K-1
K

$$f(n) = \text{same}_n + \text{diff}_n$$

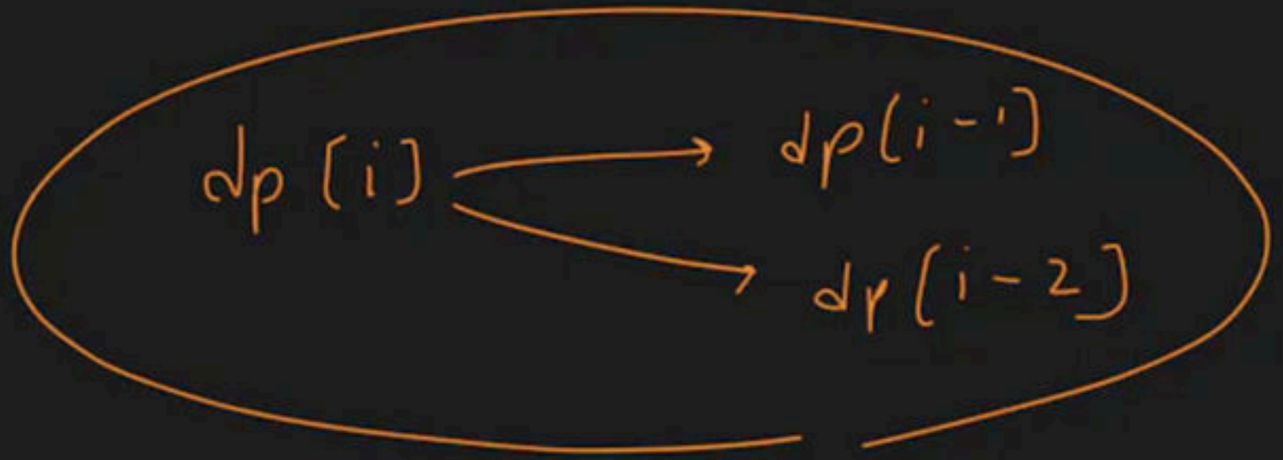
$$= \left[f(n-2)\right](k-1) + \left[f(n-1)\right](k-1)$$

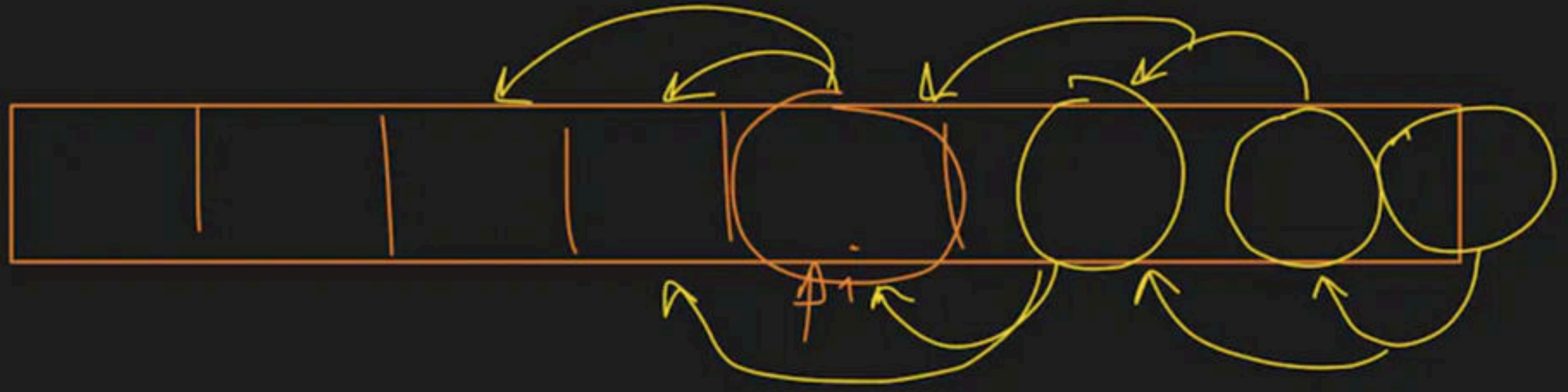
$$f(n) = (k-1)\left[f(n-1) + f(n-2)\right]$$

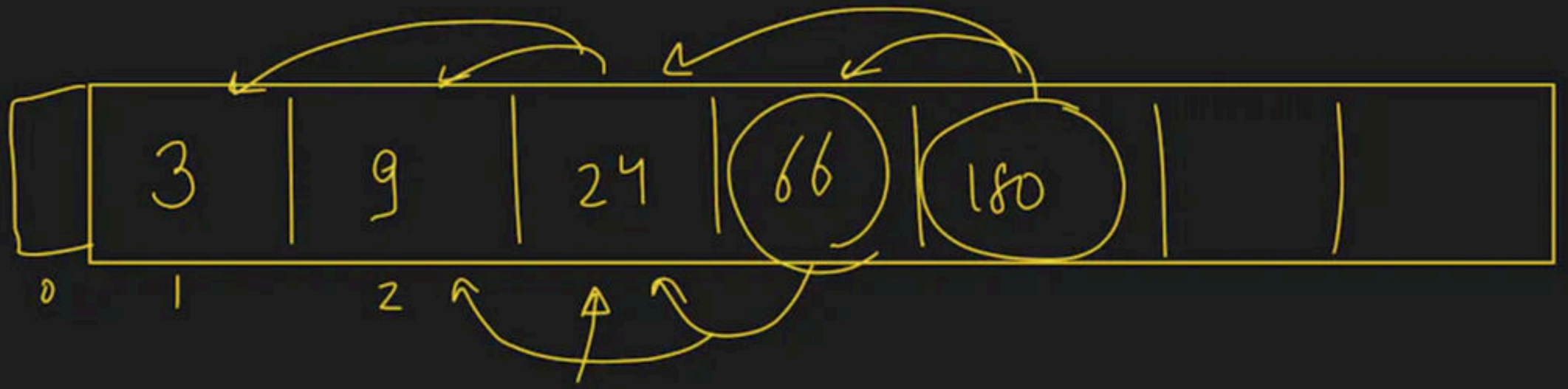

X X
same →
R R
G G
B B
X X
X X X
diff
R G
R D
D K
D G
G R
G D

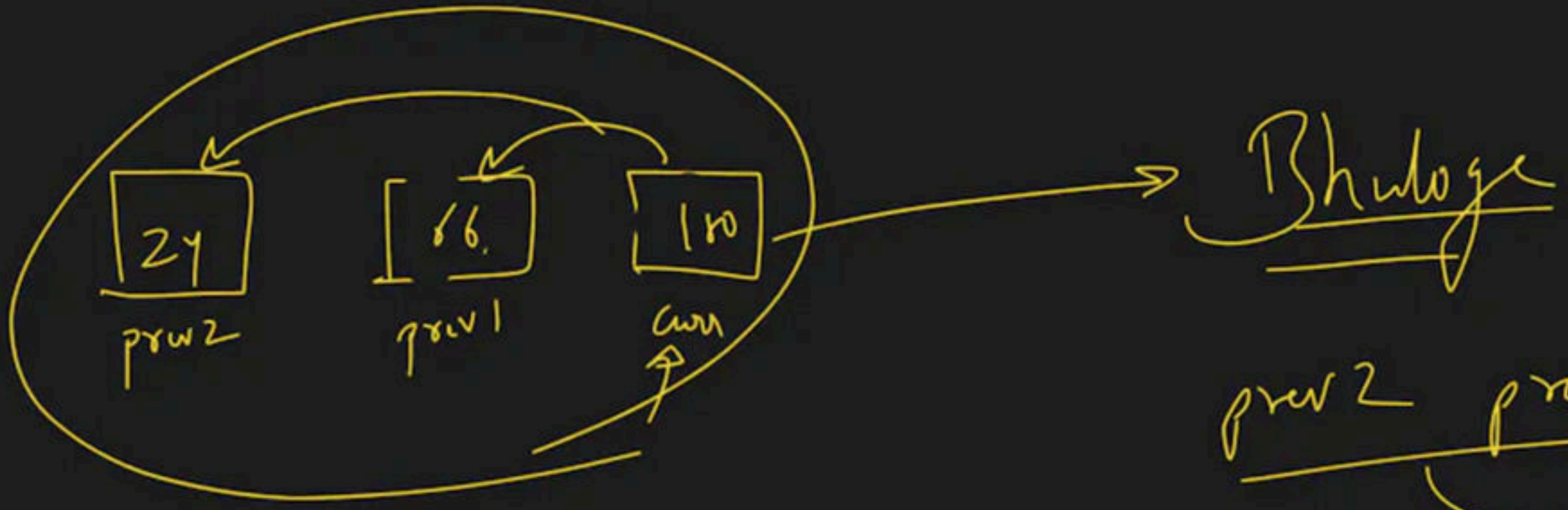

k=>
n=1
n=2
same
diff
KR
GG
BB
K
R
G
B
XX
RB
RG
GR
GB
BR
BG
K(K-1)
ans = K
n=2
ans = K + K(K-1)

Space Optimisation

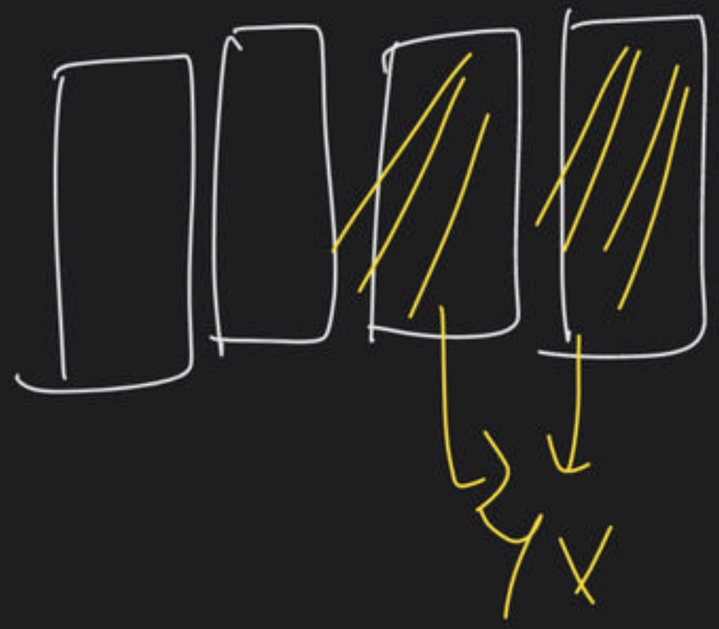





3
9
24
66
180
0
1
2


24
66
180
prev2
prev1
curr
Bhuloge
prev2
prev
shift

⇒Q⇒ n post

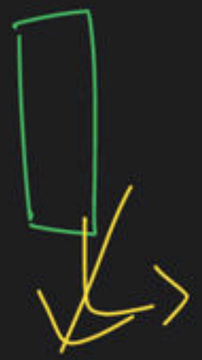


K Colors

Bucket

RR RR

n = 1



k = 3

k = 1

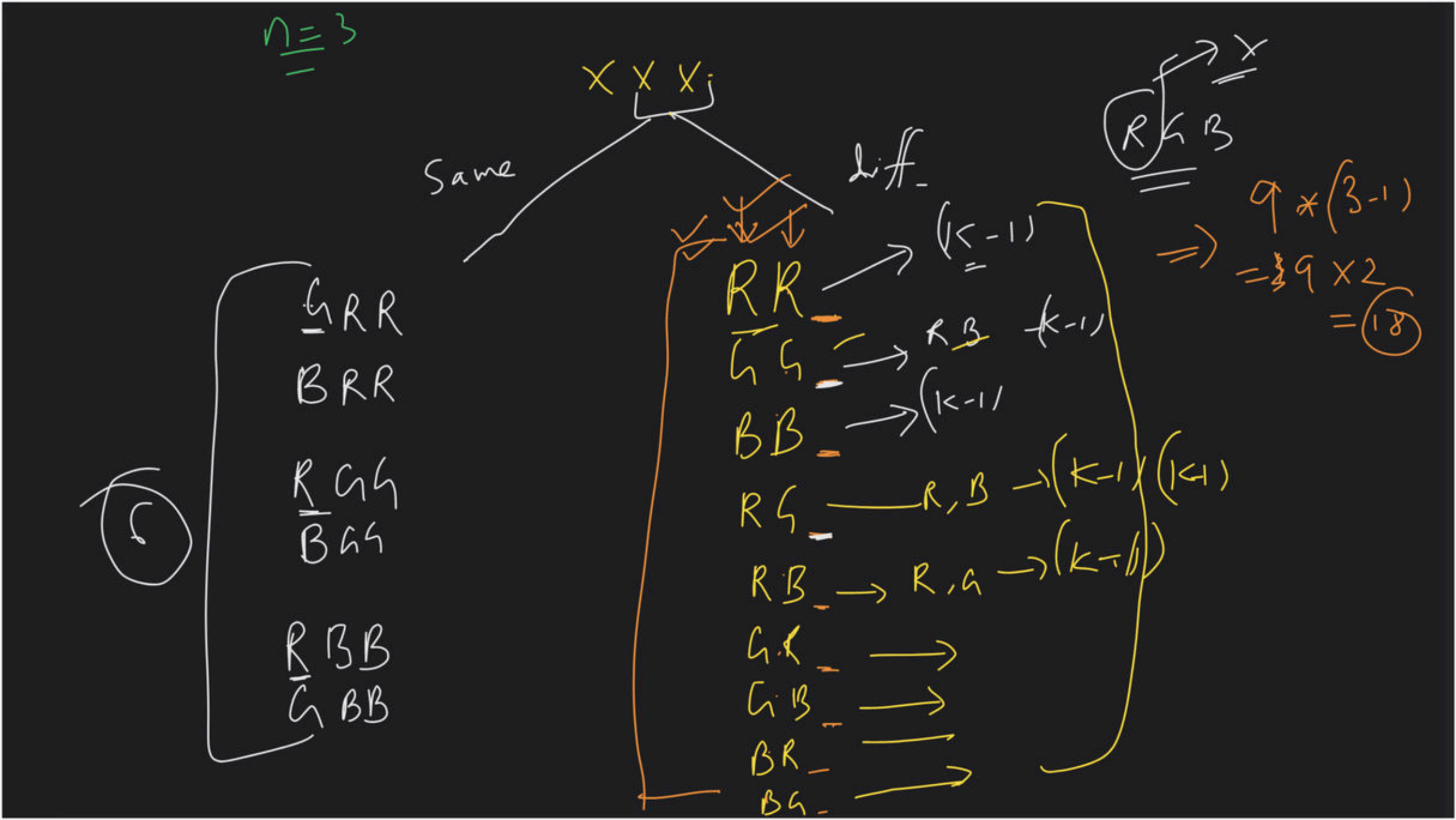
k = k

h = 2

K =>

Lastice fine	n = 1	n = 2	n = 3	n = 4
Same	K	RR GG BB — (3) => K	G RK B RR R GG B GG R BB G BB (6) $k \cdot (k-1)$ $= f(n-2) * (k-1)$	
diff		RG RB GR GB BR BG => (6) $\leq 3 \cdot (3-1)$ $k \cdot (k-1)$ $= \ldots$	(18) $f(2) * (k-1)$ $= f(n-1) * (k-1)$	

$k + k(k-1) = (9)$


n = 3
X X X
Same
Diff
G R R
B R R
R G G
B G G
R B B
G B B
R R → (k-1)
G G → R B (k-1)
B B → (k-1)
R G → R, B → (k-1)(k-1)
R B → R, G → (k-1)
G R
G B
B R
B G
R G B
X
9 * (3-1)
= 9 × 2
= 18

$f(n) = f(n-1) * (k-1) + f(n-2) * (k-1)$

diff

same

0/1 Knapsack Problem

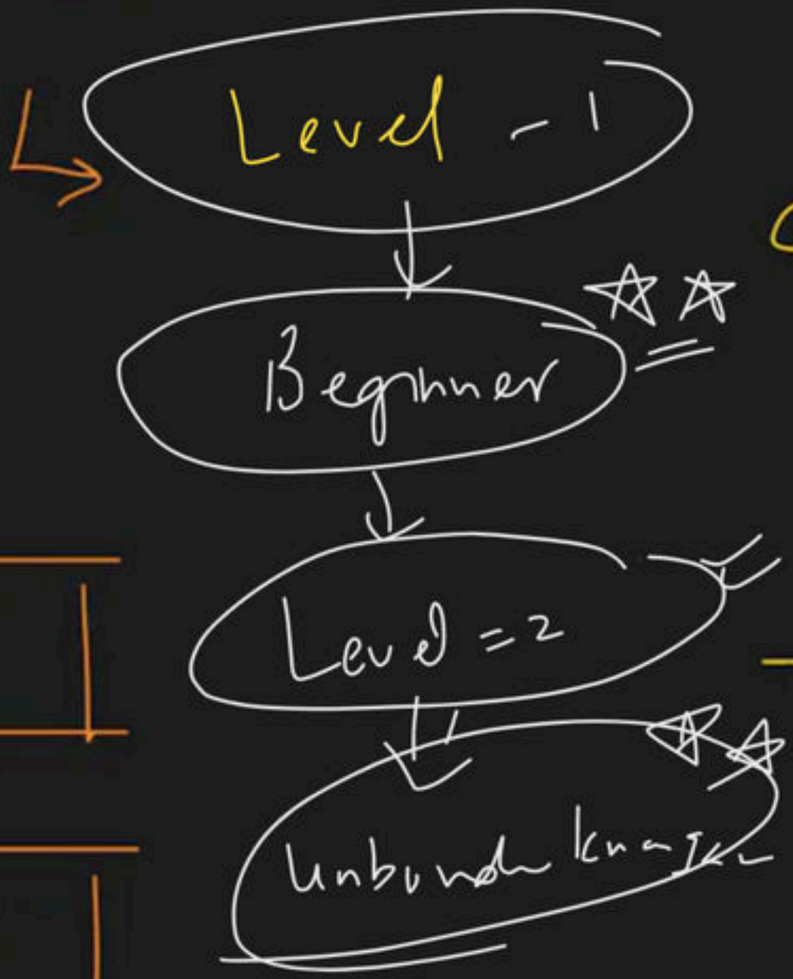
220

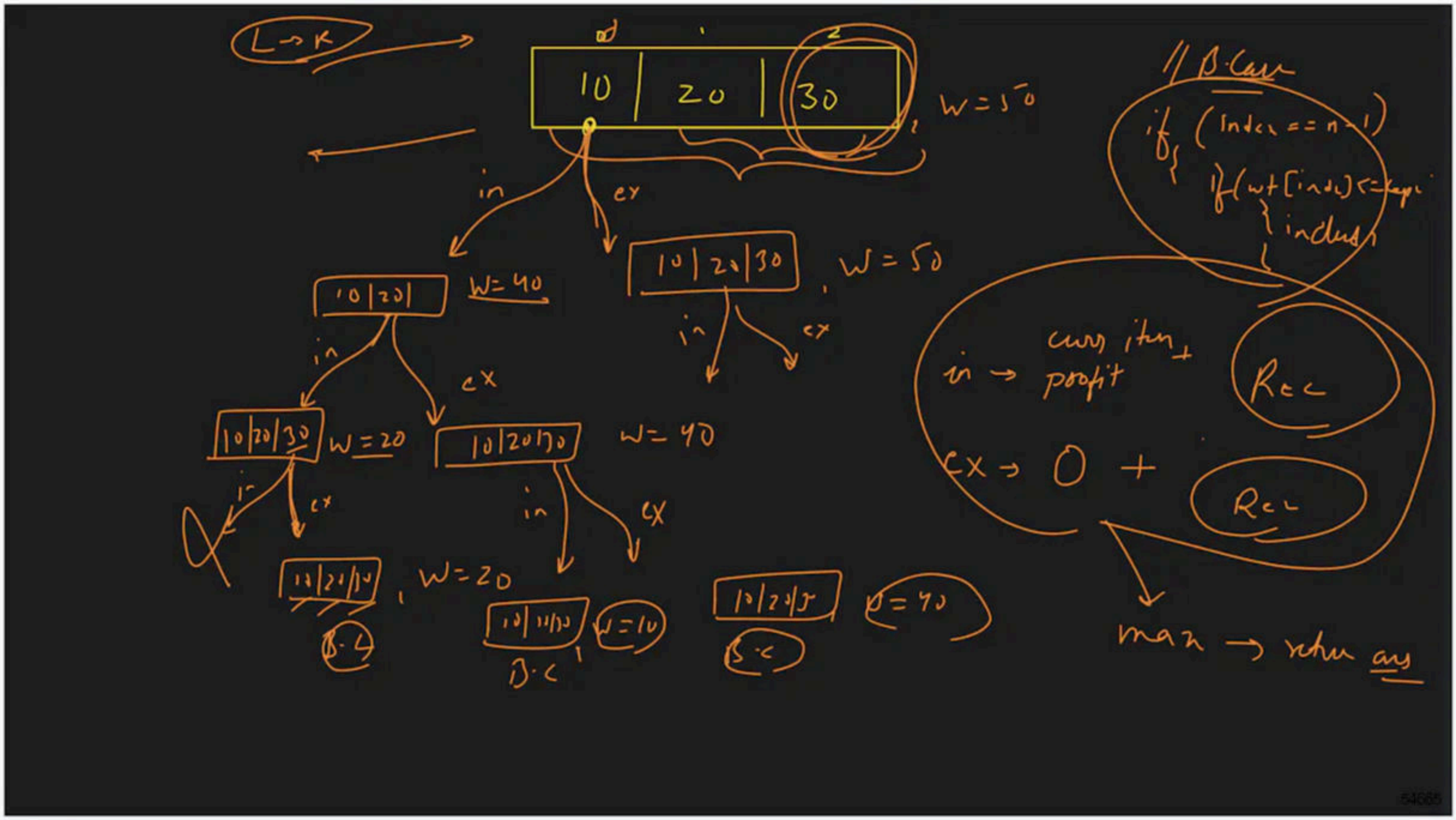
weight

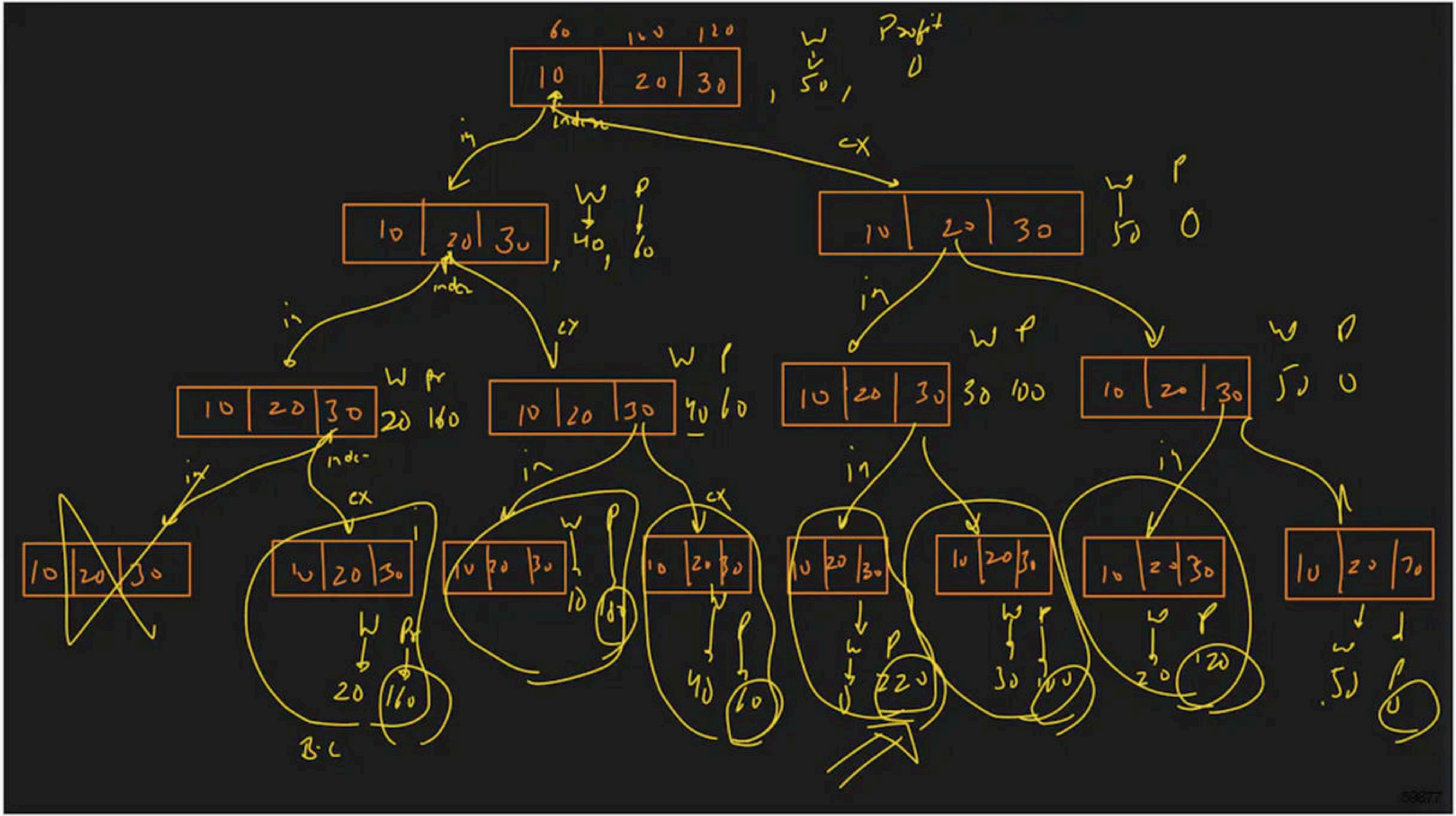
10	20	30
0	1	2

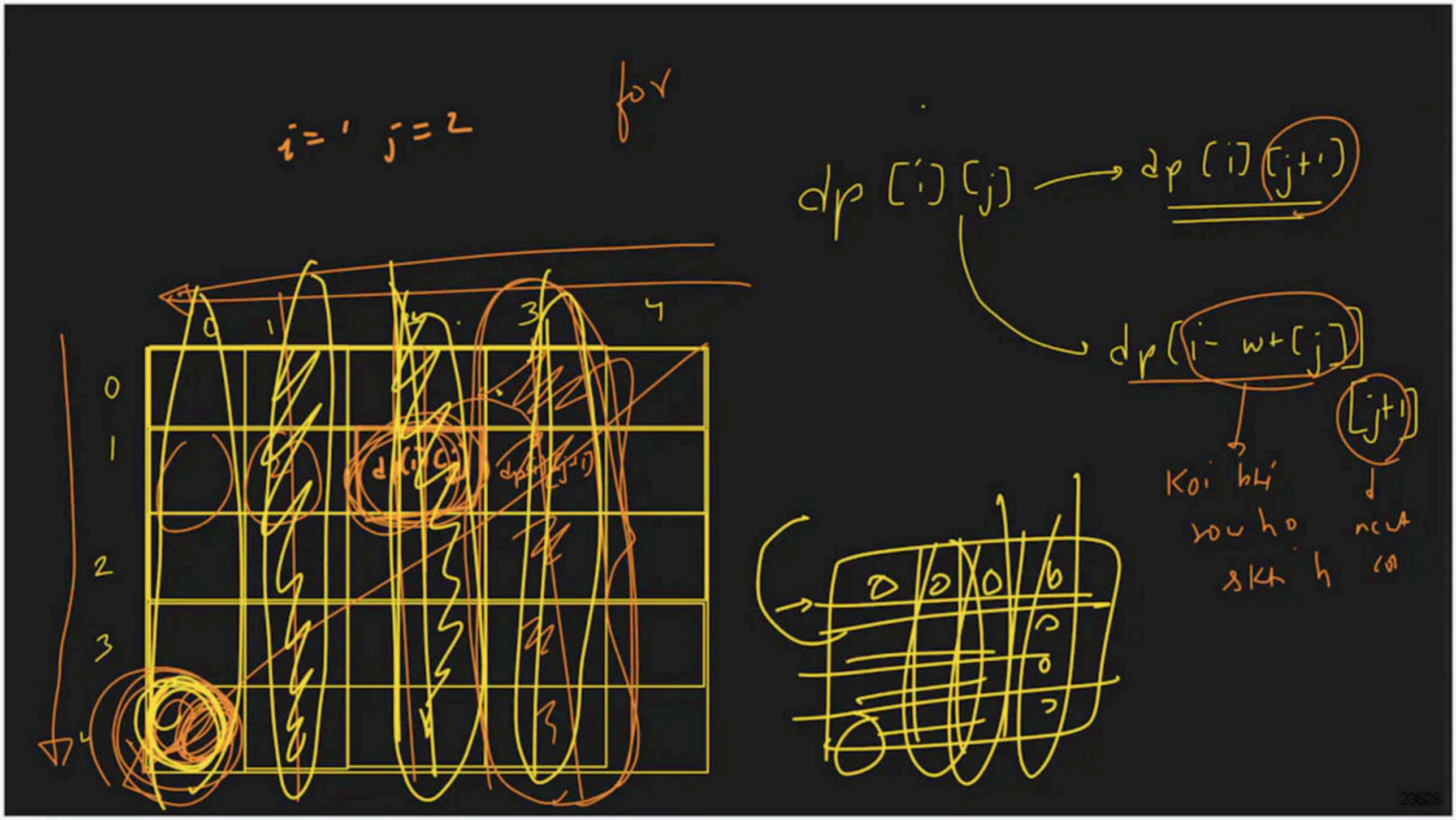
Value / profit

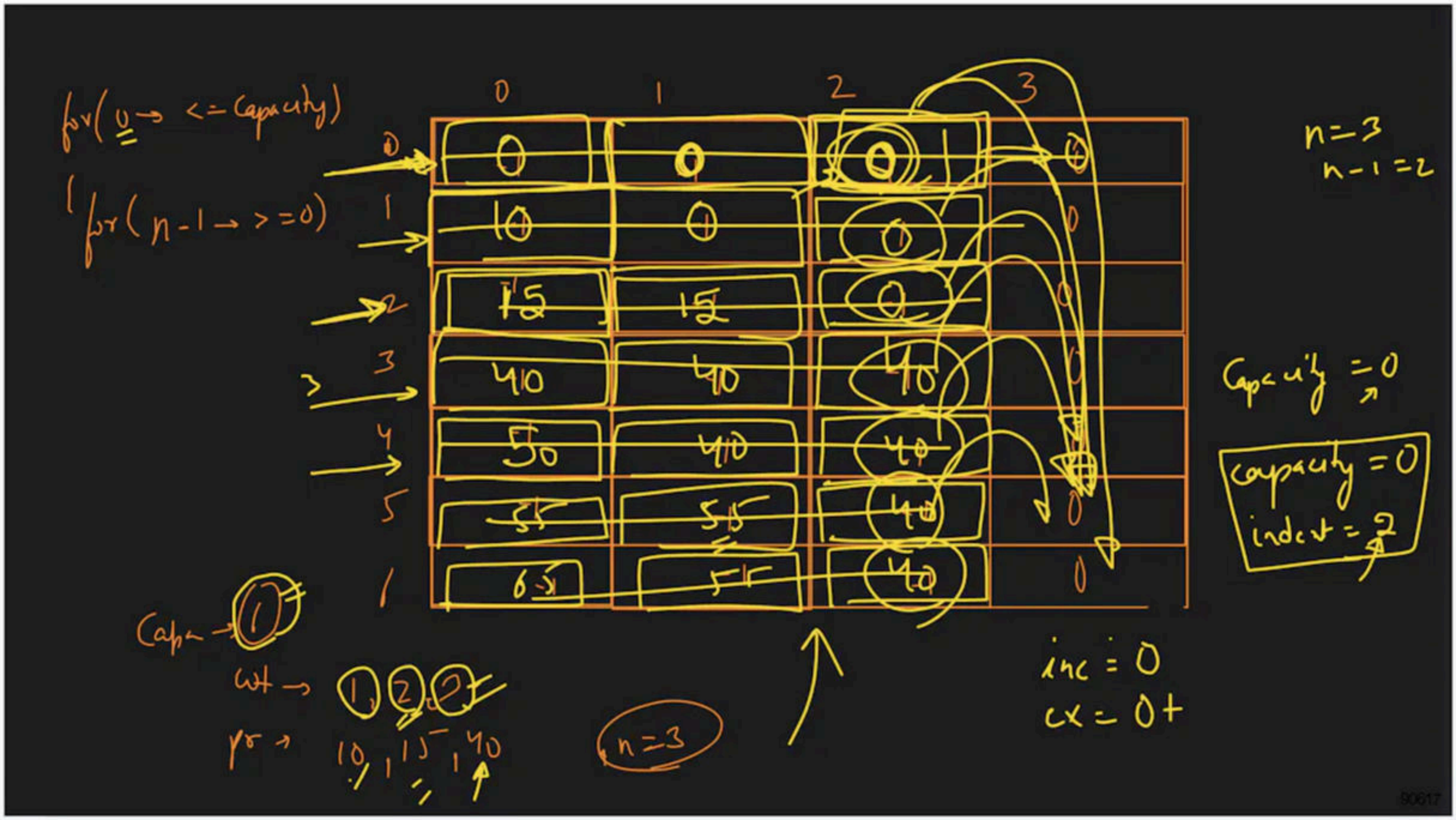
60	100	120
0	1	2




L->K
0
1
2
10
20
30
W=50
in
ex
10|20
W=40
10|20|30
W=50
10|20|30
W=20
10|20|30
W=40
W=20
W=10
W=40
// B-Case
if (index == n-1)
if(wt[index]<=kapa
includ
curr item +
in → profit
Rec
ex → 0 +
Rec
max → retun ans


60 100 120
10 | 20 | 30
W 50
Profit 0
index
in
ex
W P
40 60
50 0
W Pr
20 160
30 100
B.C


i=1 j=2
for
dp[i][j] → dp[i][j+1]
dp[i-wt[j]][j+1]
Koi bhi
soucho
skta h
ncha
0 1 2 3 4
0 1 2 3 4


for(i → <= capacity)
for(n-1 → >=0)
0
1
2
3
n=3
n-1=2
Capacity = 0
capacity = 0
index = 2
inc = 0
cx = 0+
n=3
wt →
pr →
10
15
40
50
60

wt = 2
capa = 0
→ i = 0
cc → 0

dp[i][j] ⟶ dp[i][j+1]

dp[i - wt[j]][j+1]

curr

0
-1
-1
-1
-1
-1
-1

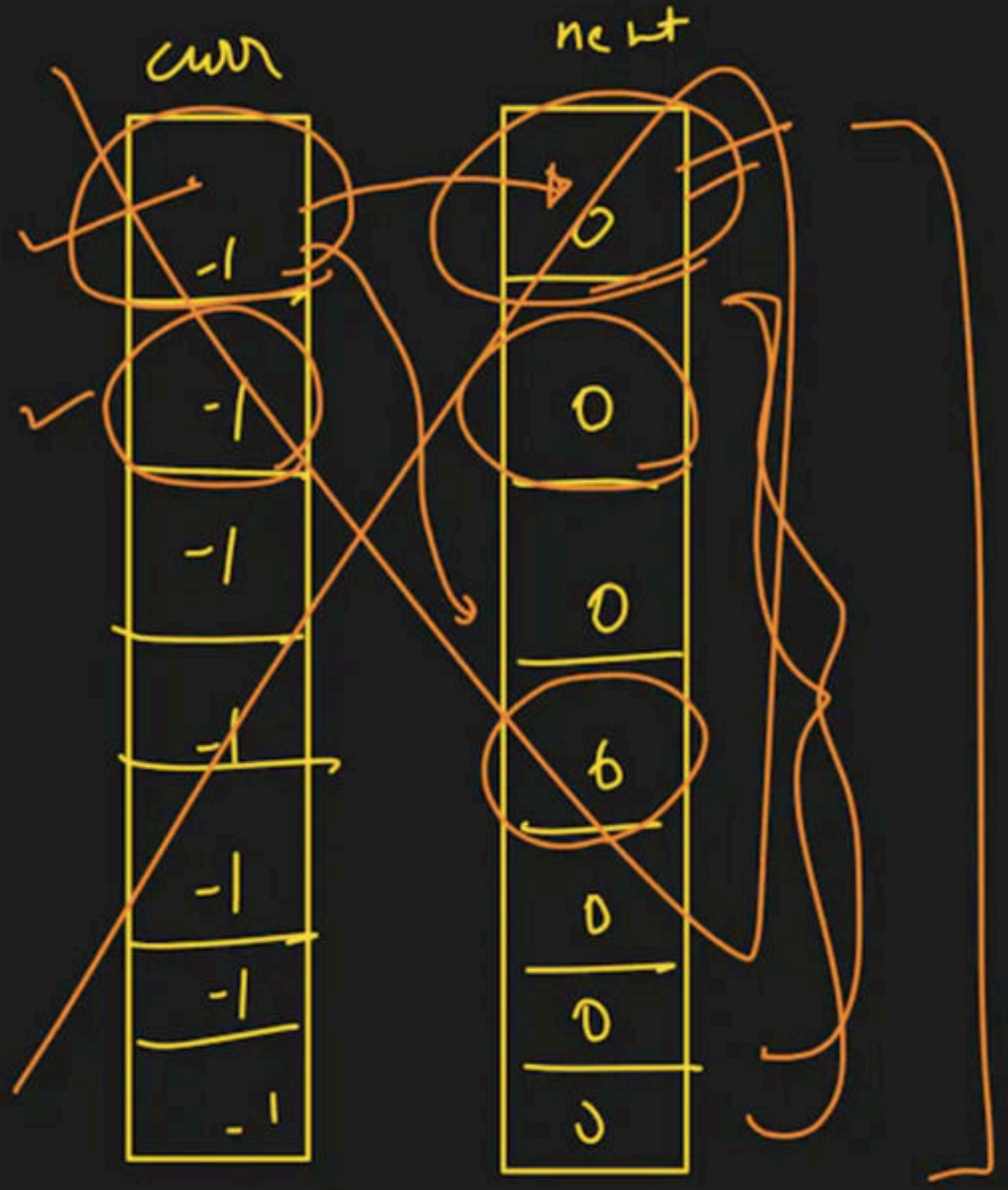
next

0
0
0
0
0
0
0

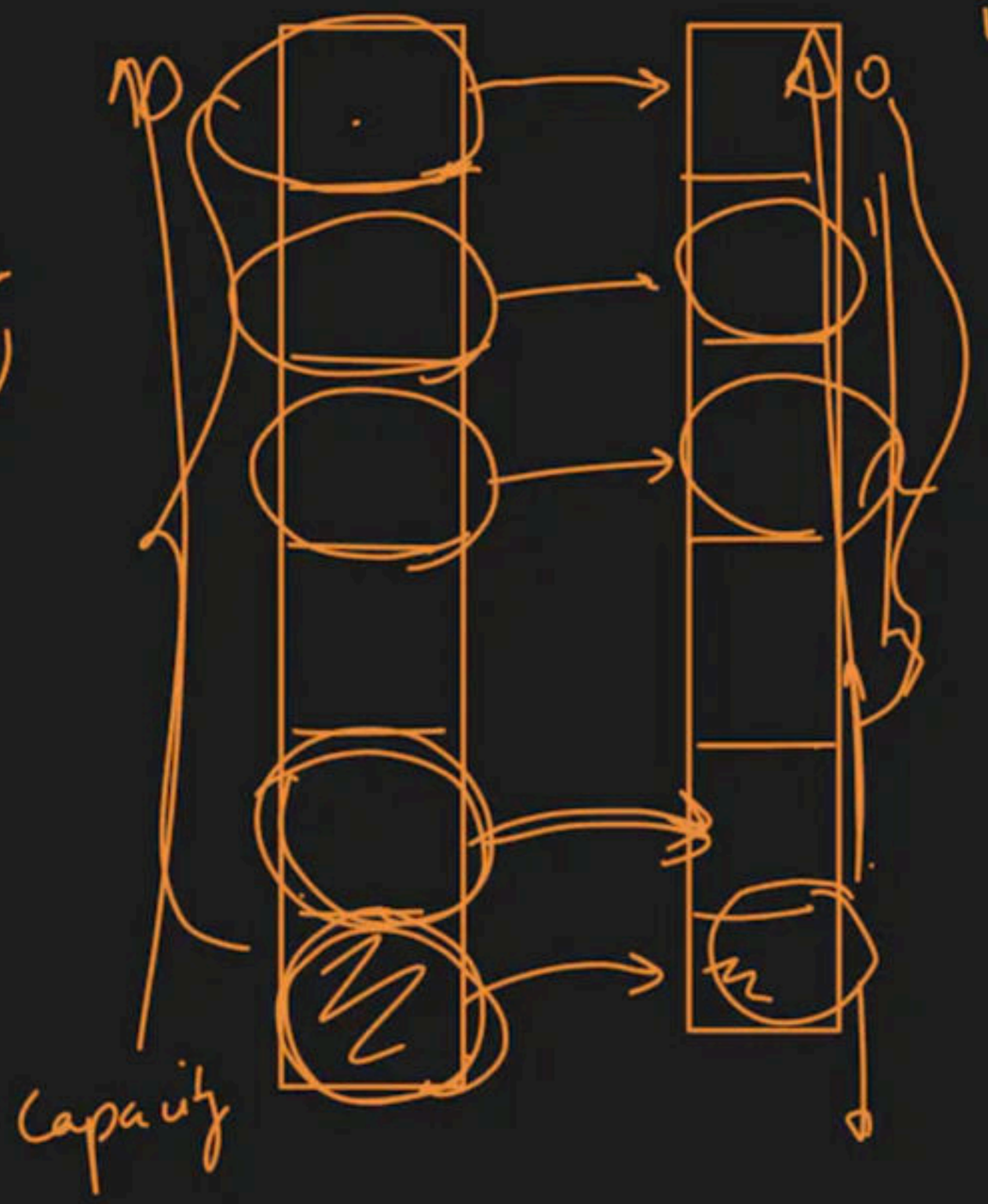
capacity → 0

j → (n-1) → 0

0	2	3


curr
next
-1
-1
-1
-1
-1
-1
-1
0
0
0
0
0
0
0

```
for (col
    n-1 ---> 0) ---> j
{
  for (row
       0 ---------> capacity) ---> i
  {
  }
}
```



Dynamic Programming Class-4

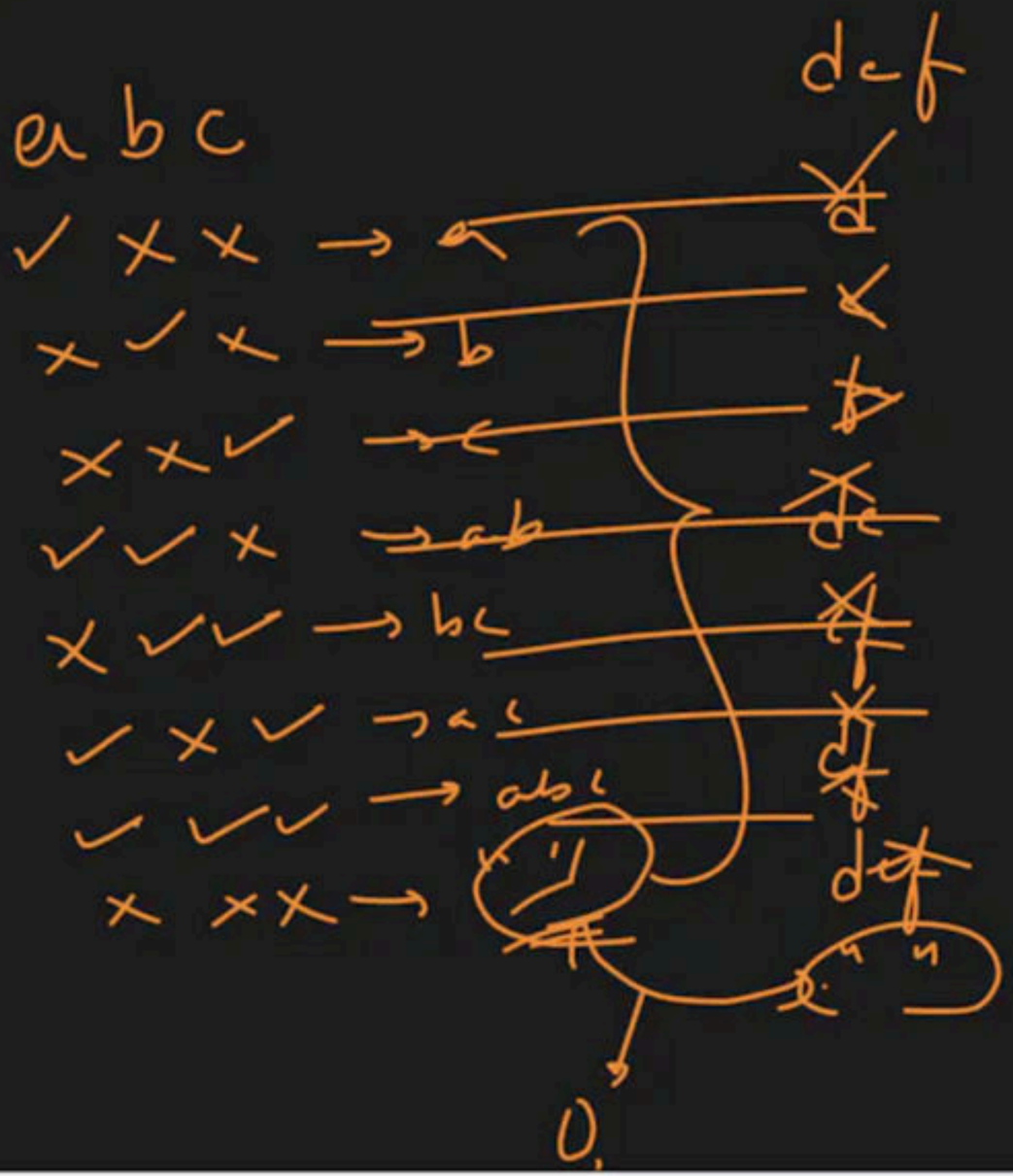
Special class

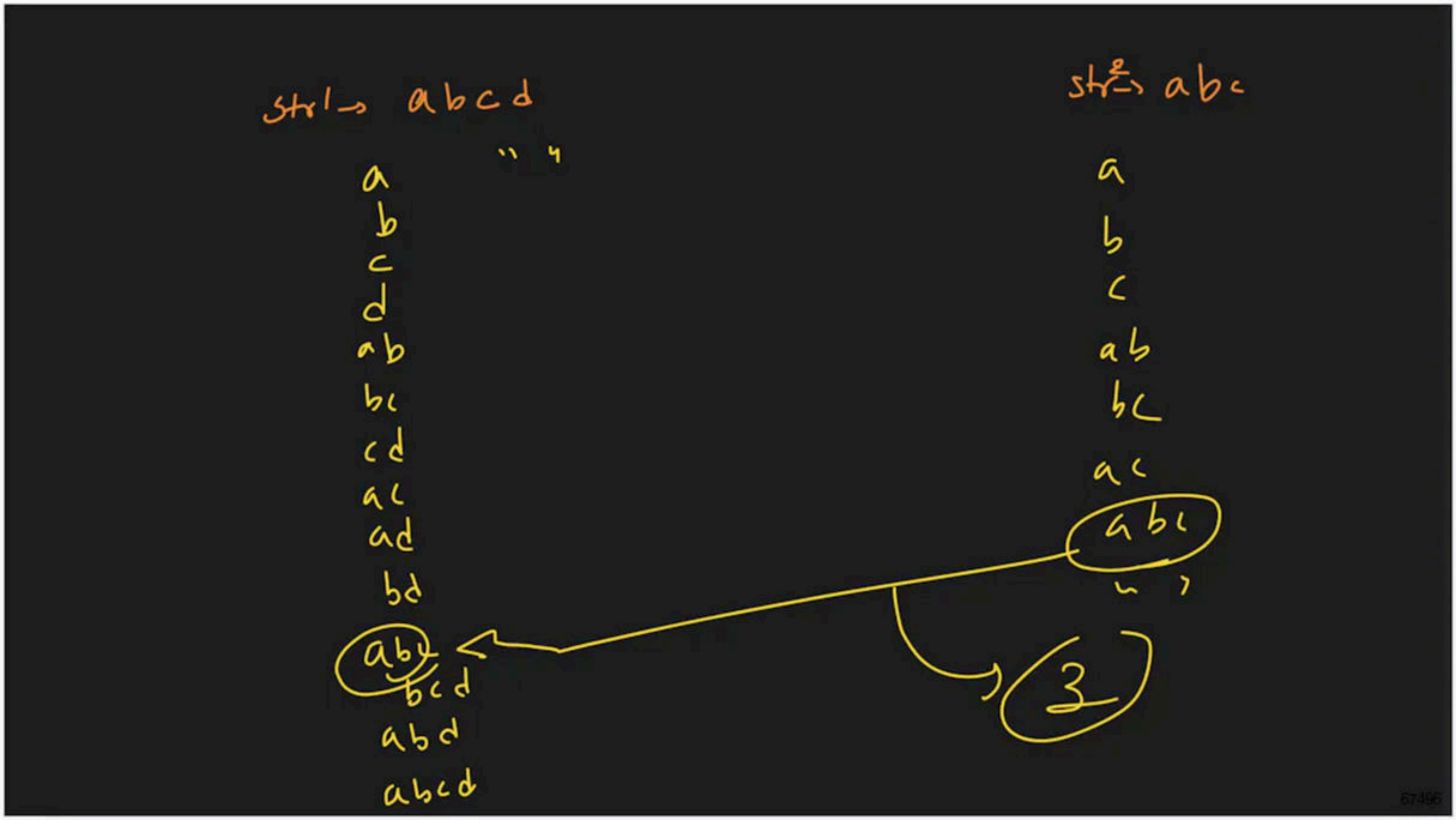
Love Babbar • Jan 11, 2024

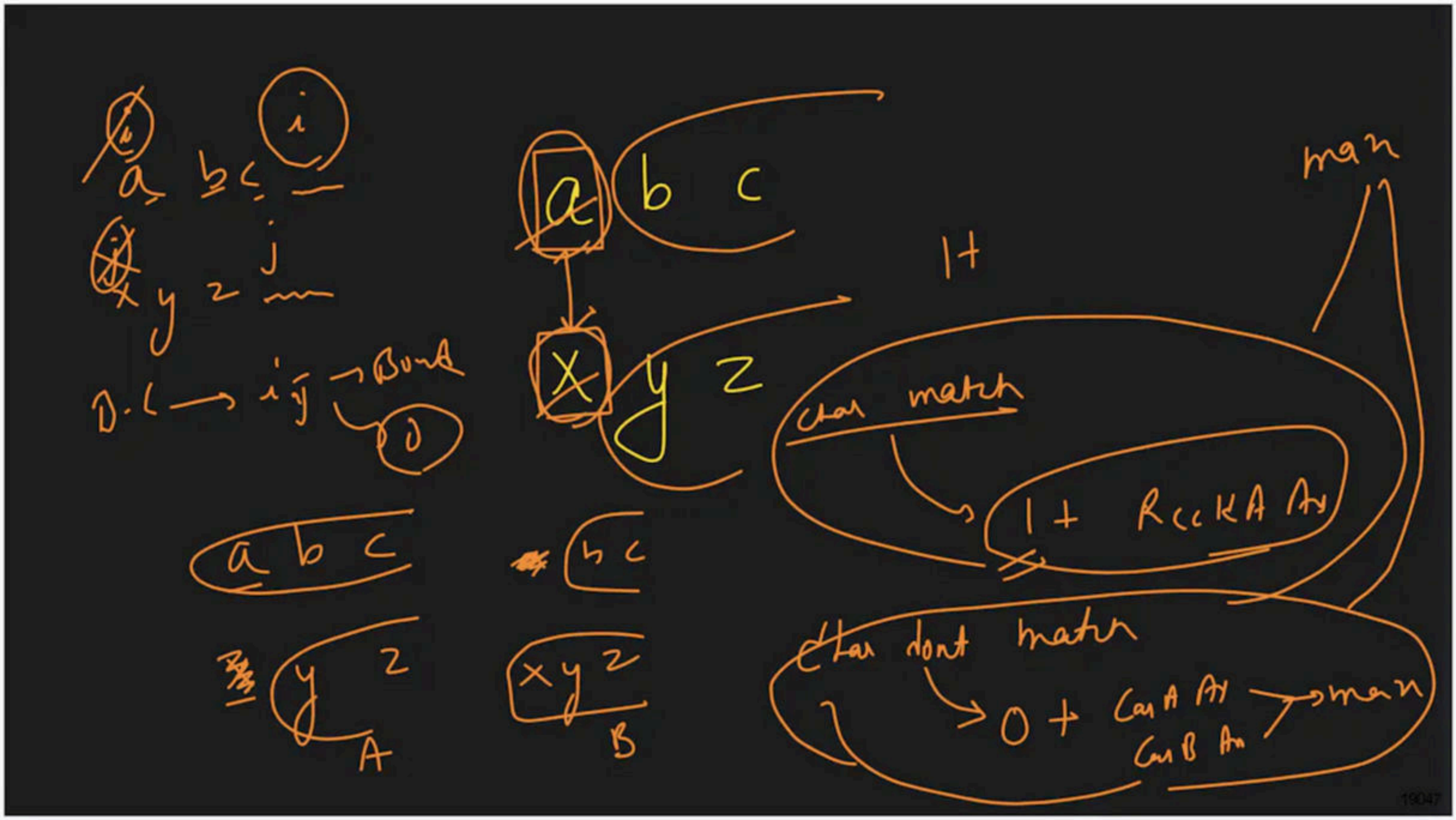
→ Longest Common Subsequence :-

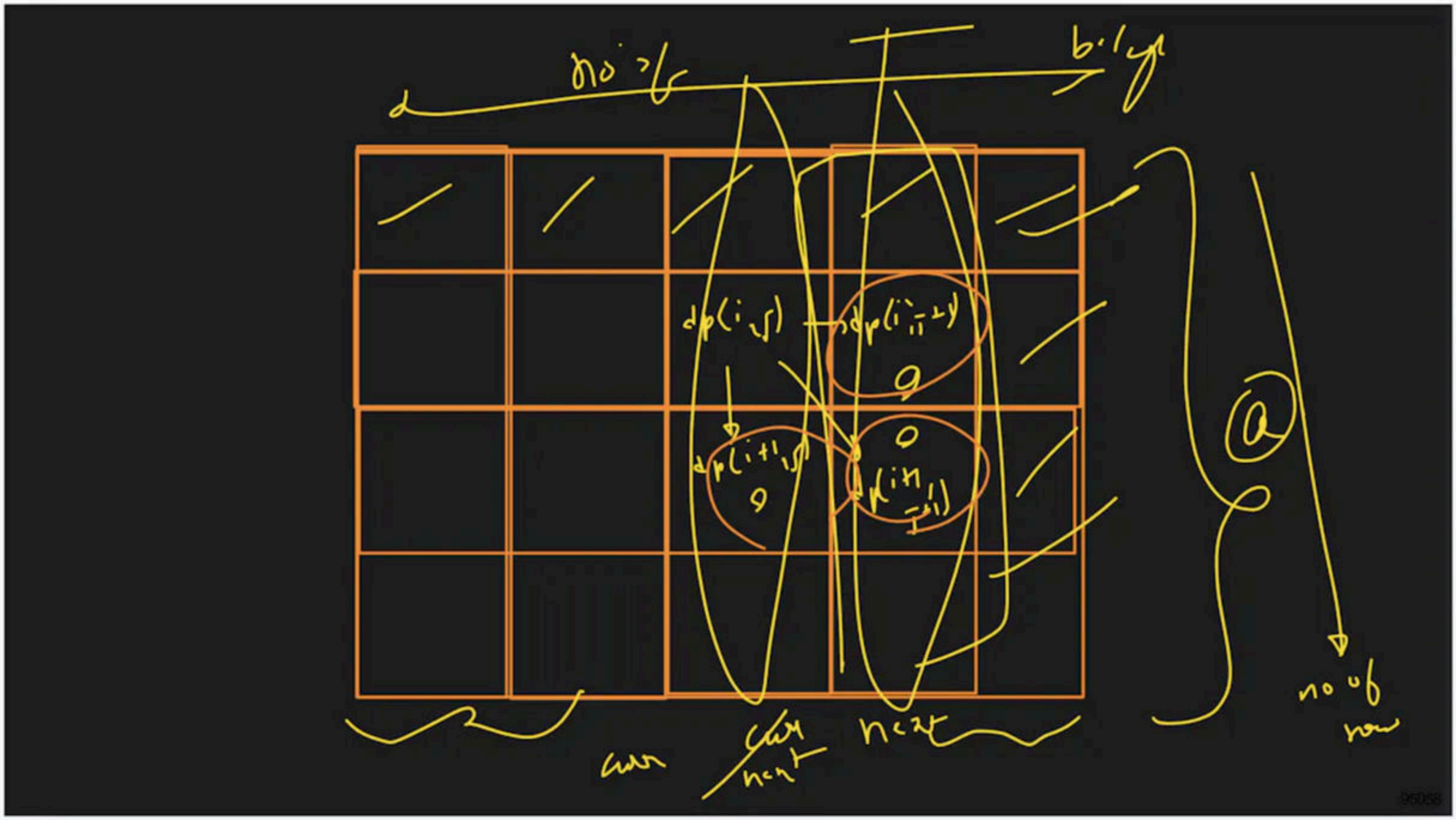
str1 ⟹ abc

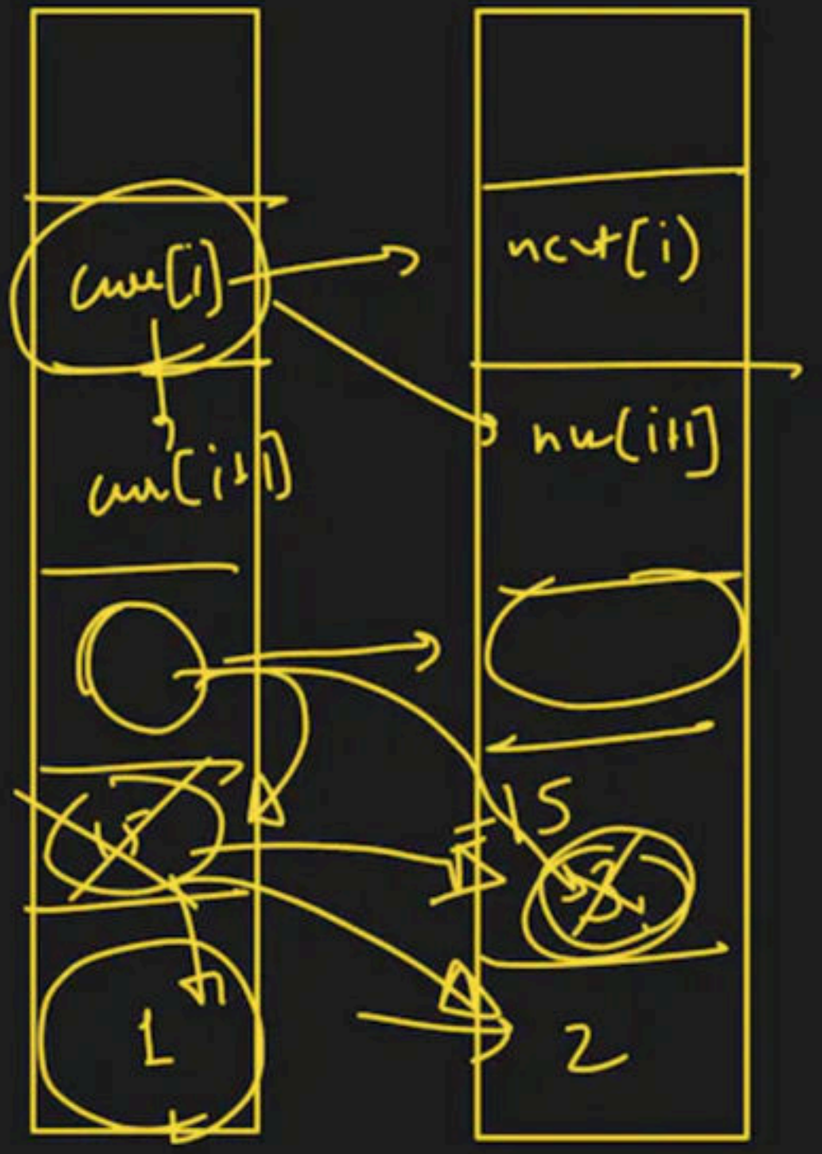
str2 ⟹ def

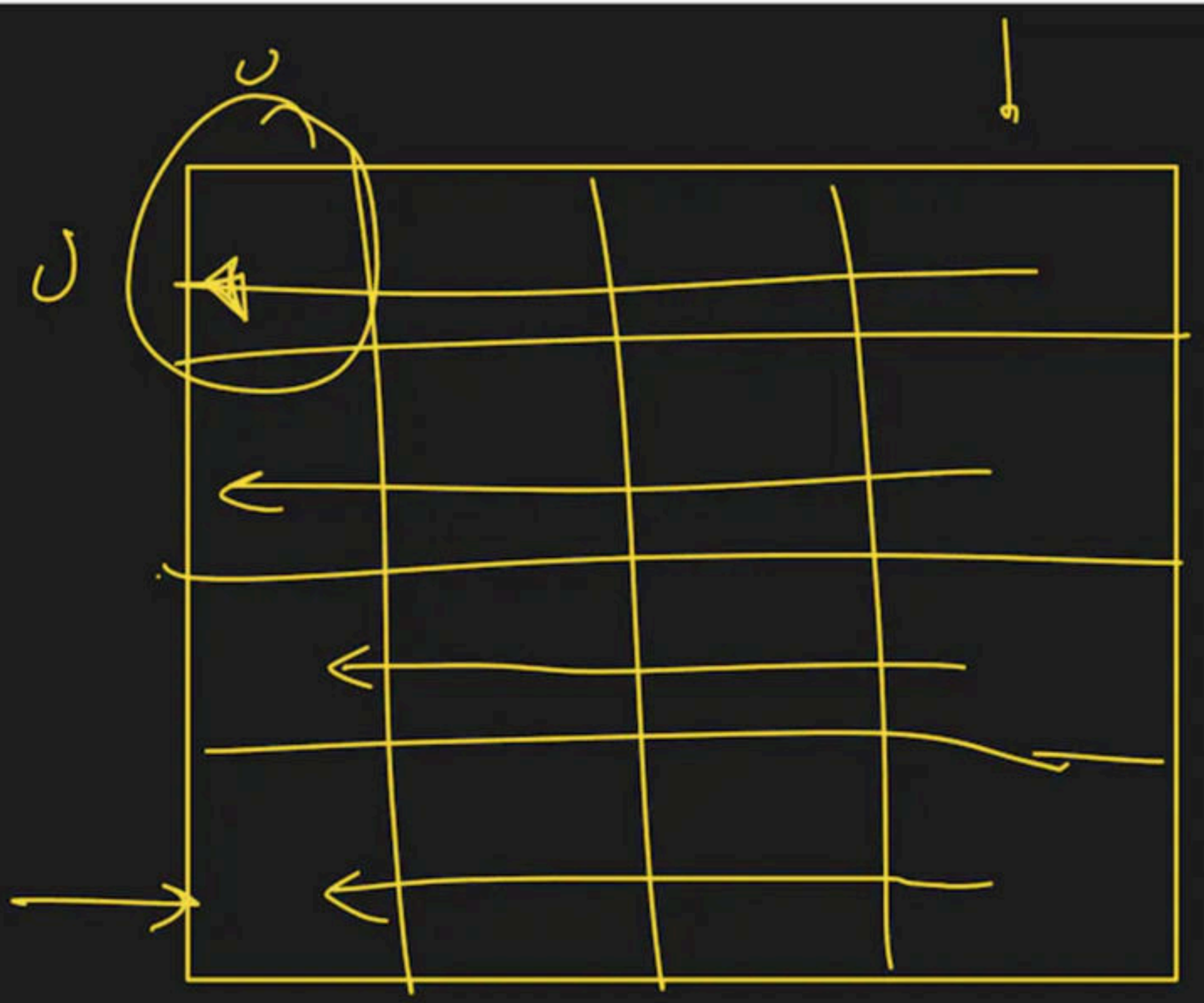


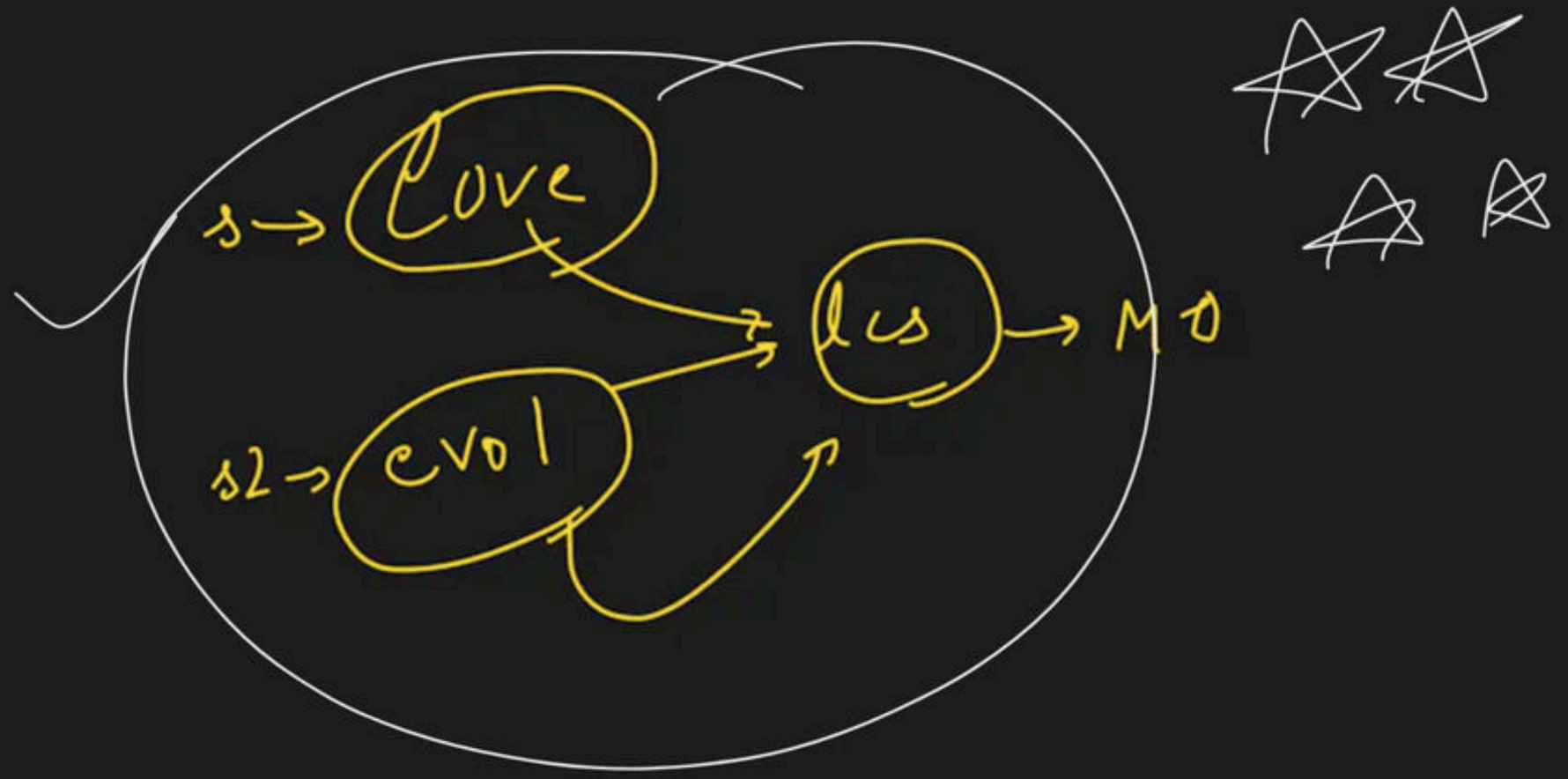

str1 -> abcd
a
b
c
d
ab
bc
cd
ac
ad
bd
abc
bcd
abd
abcd
str2 -> abc
a
b
c
ab
bc
ac
abc
3

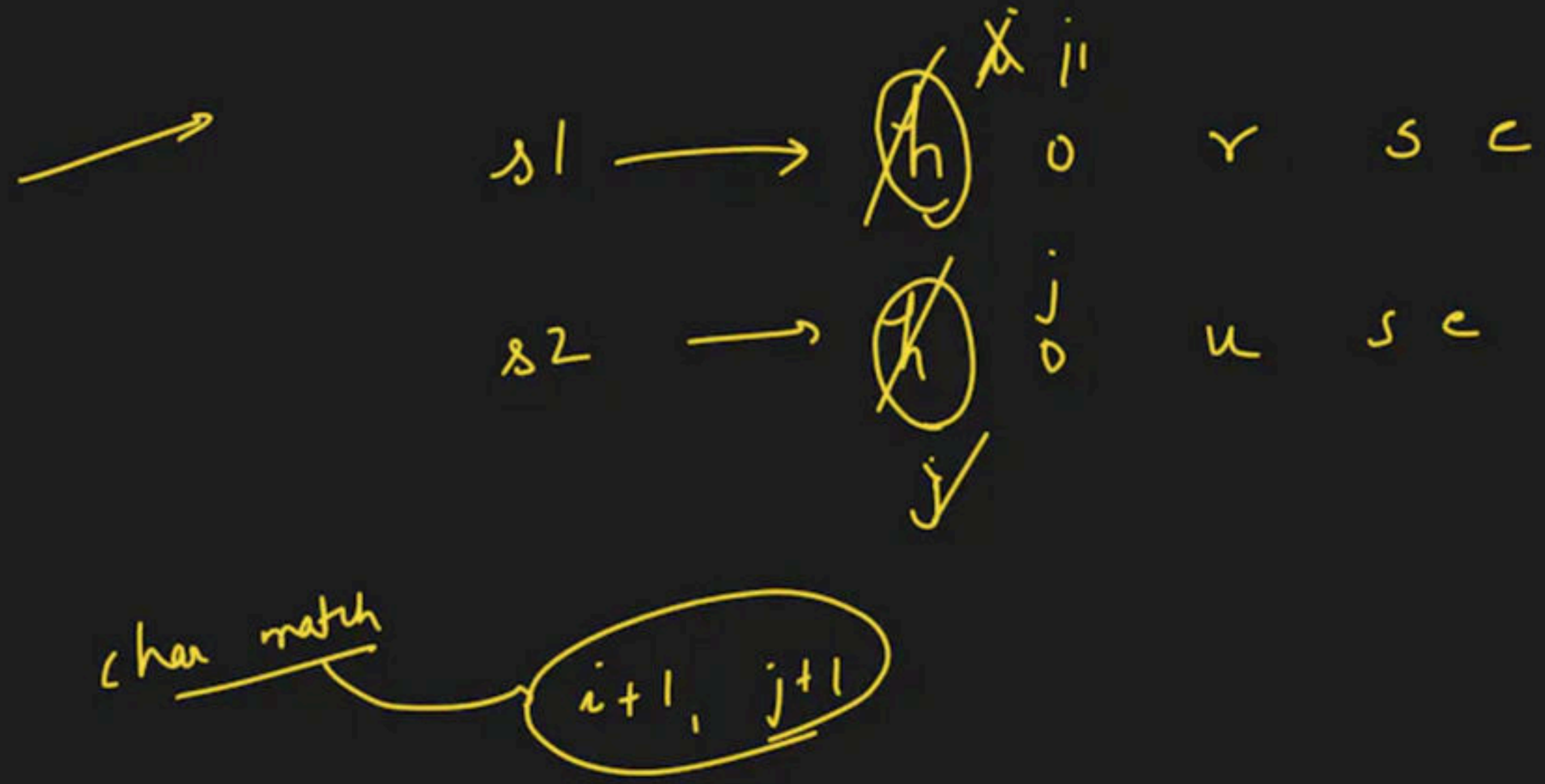

a b c
i
x y z
j
D.C → i j → Bond
0
a
b c
x
y z
main
It
char match
It RecKA An
char dont match
0 + CanA An
CanB An
main
abc
bc
y z
A
x y z
B

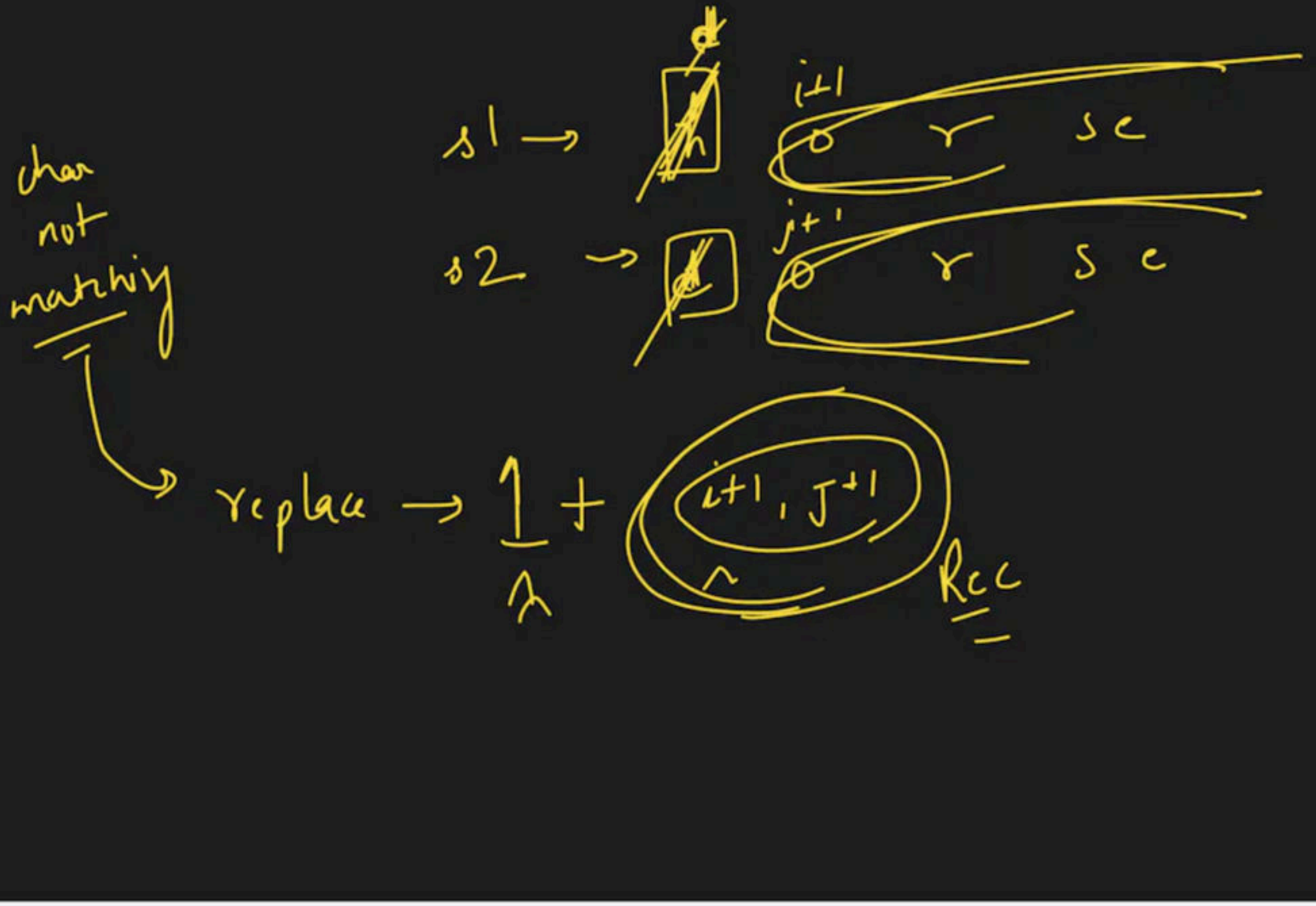


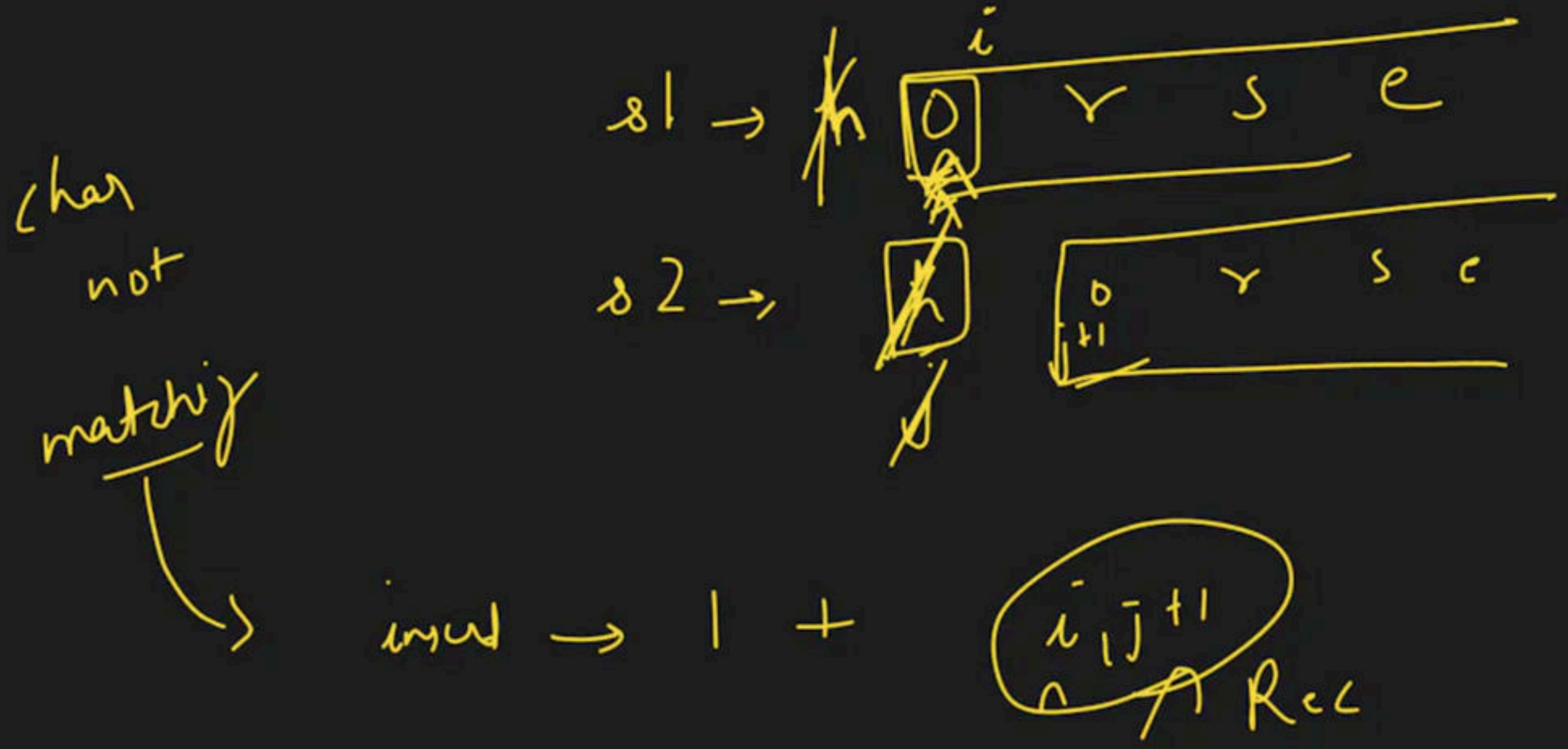

next[i]
num[i]
num[i+1]
num[i+1]
5
1
2


C
C


s → Love
s2 → evol
ls
→ MO


s1
h
i
0
r
s
e
s2
k
j
0
u
s
e
j
char match
i+1, j+1


char not matching
s1 →
i+1
s2 →
j+1
replace → 1 +
i+1, j+1
Rec


i
s1 →
0
s
e
s2 →
0
s
c
+1
char
not
matching
→ ans → 1 + i, j+1
Rec

char not matching

$s \rightarrow$ ~~X~~ O S C (i+1)

$s2 \rightarrow$ O S C (j)

delete $\rightarrow 1 + (i+1, j)$ Rec

Char match

→ Rec → (i+1, j+1)

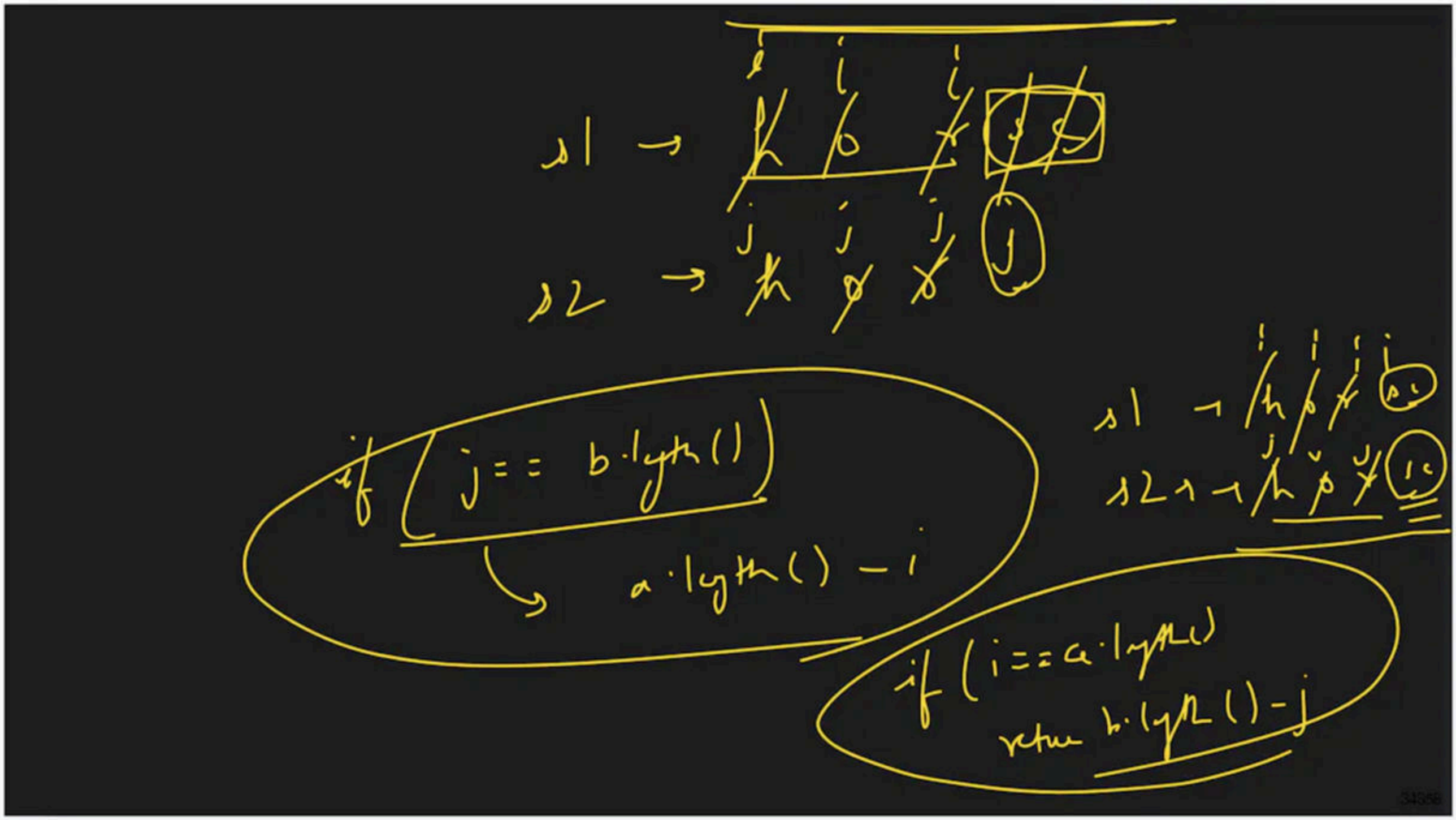
Char not match

→ replace → 1 + (i+1, j+1) Rec

insert → 1 + (i, j+1) Rec

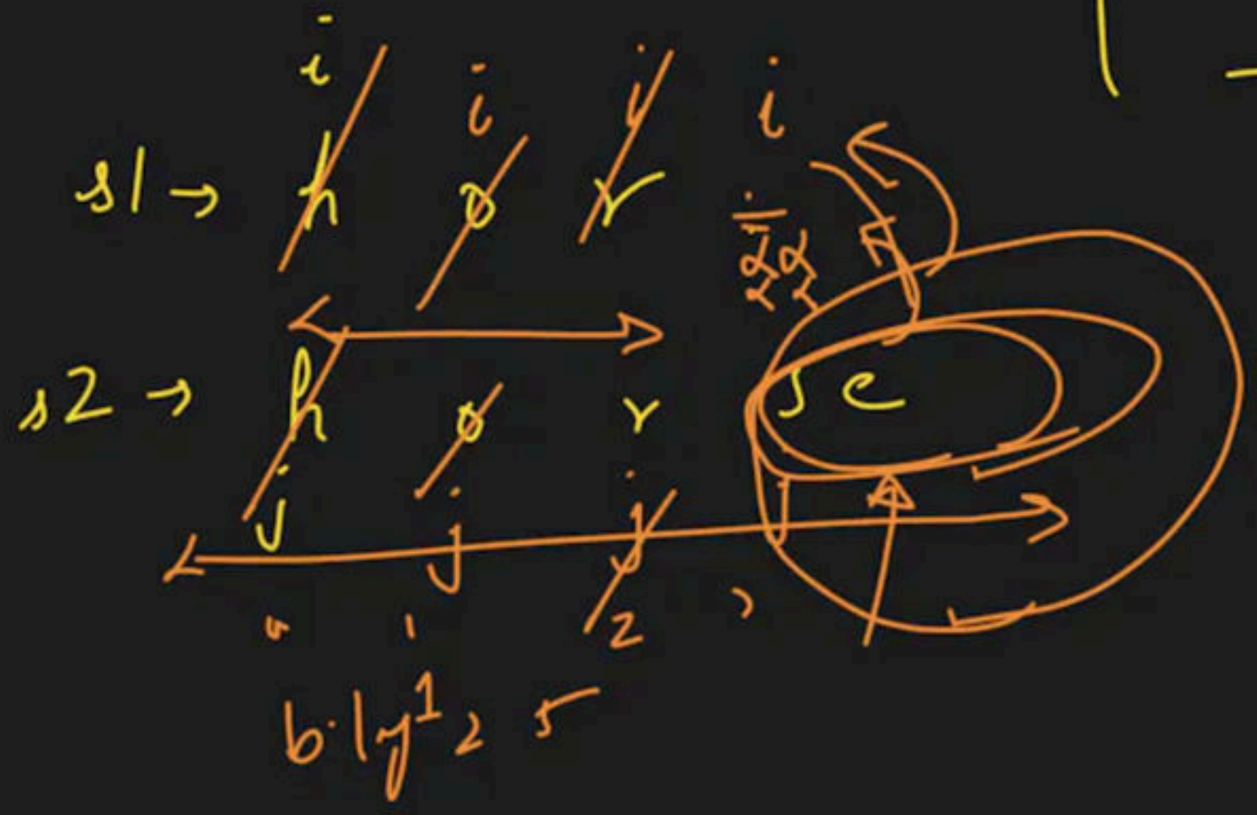
deletion → 1 + (i+1, j) Rc

min


s1 →
s2 →
if (j == b.length())
a.length() - i
if (i == a.length()
return b.length() - j

Rec → B.C

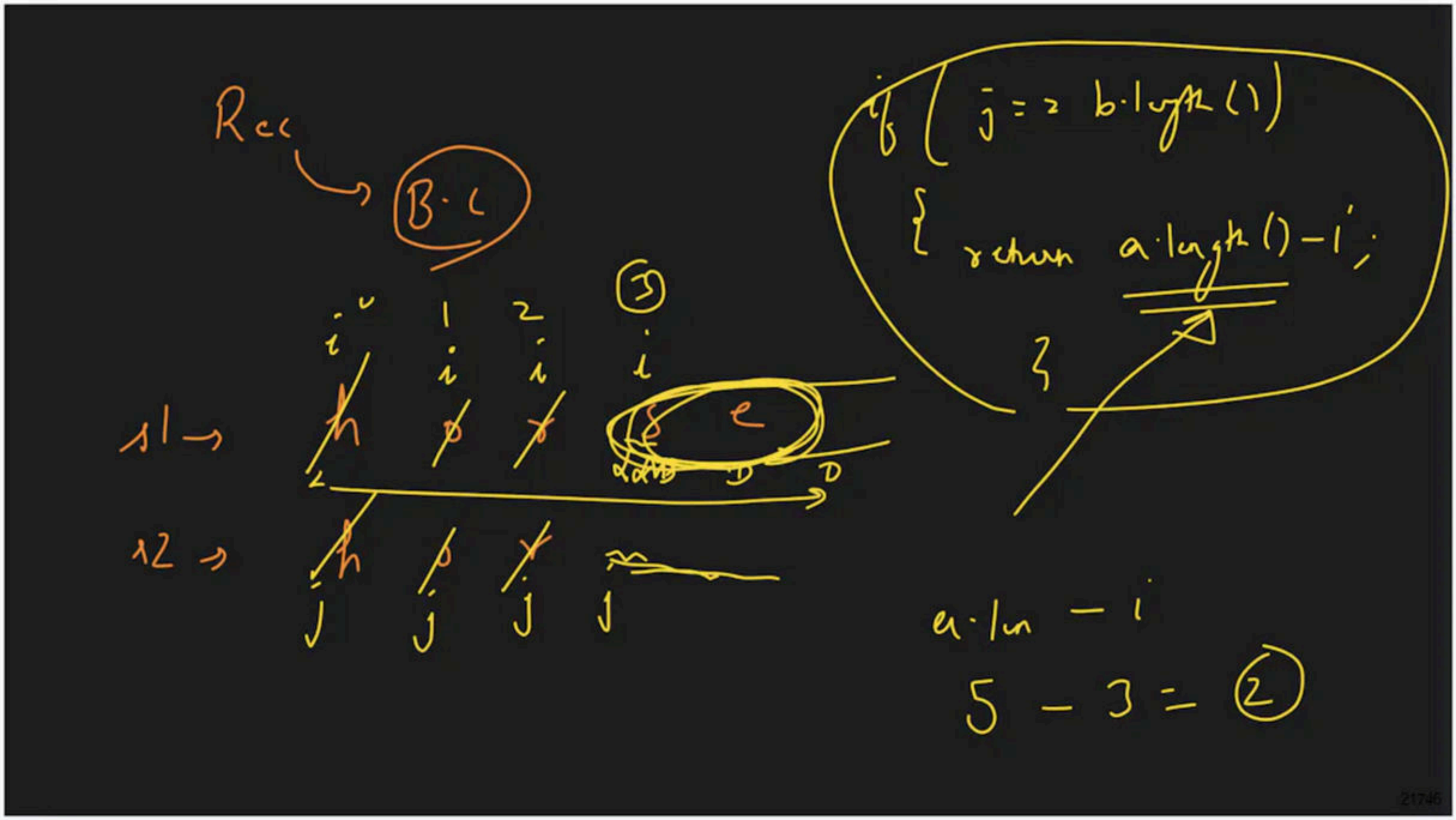
```
if (i == a.length())
{
    return b.length() - j
}
```

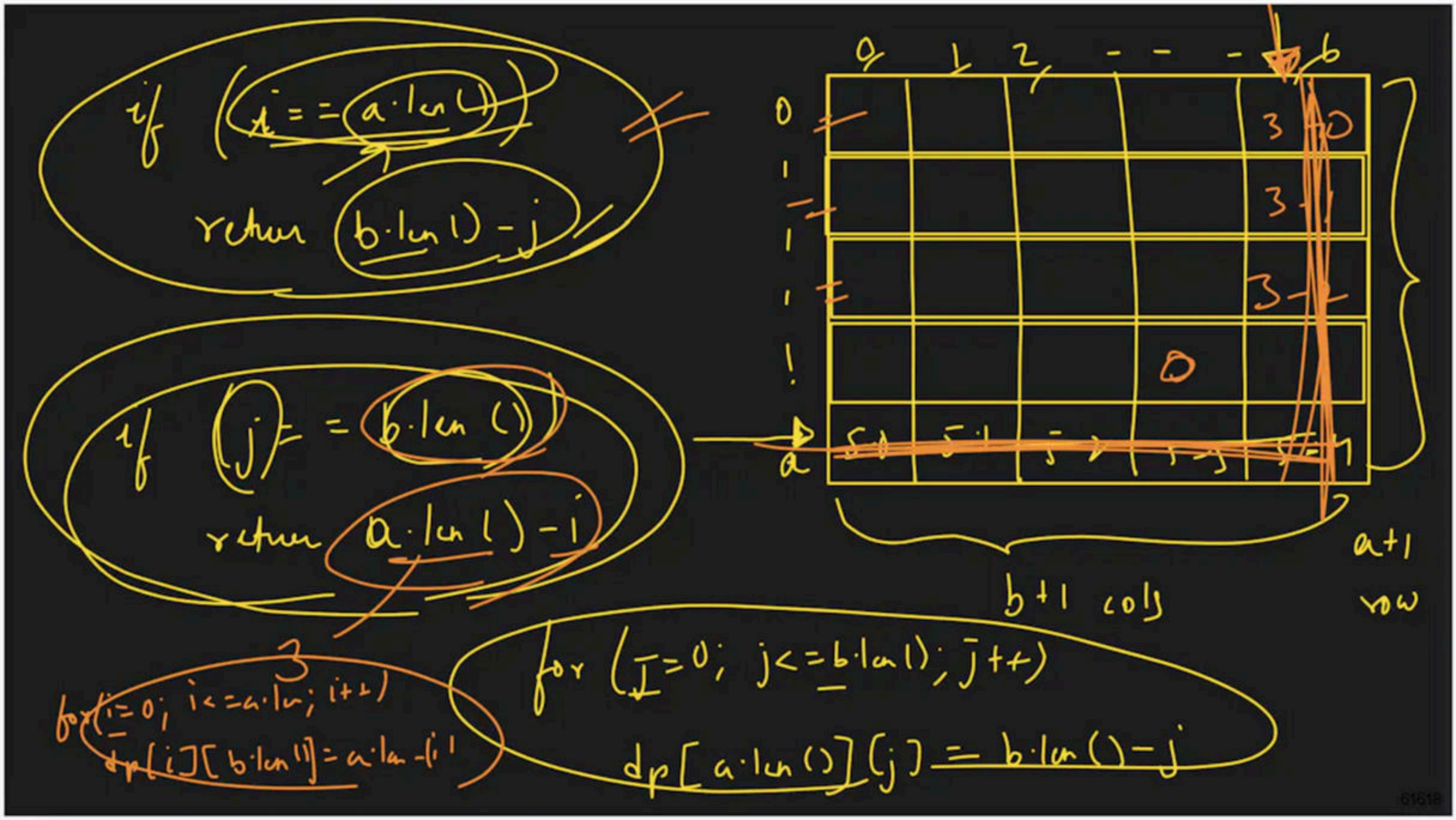


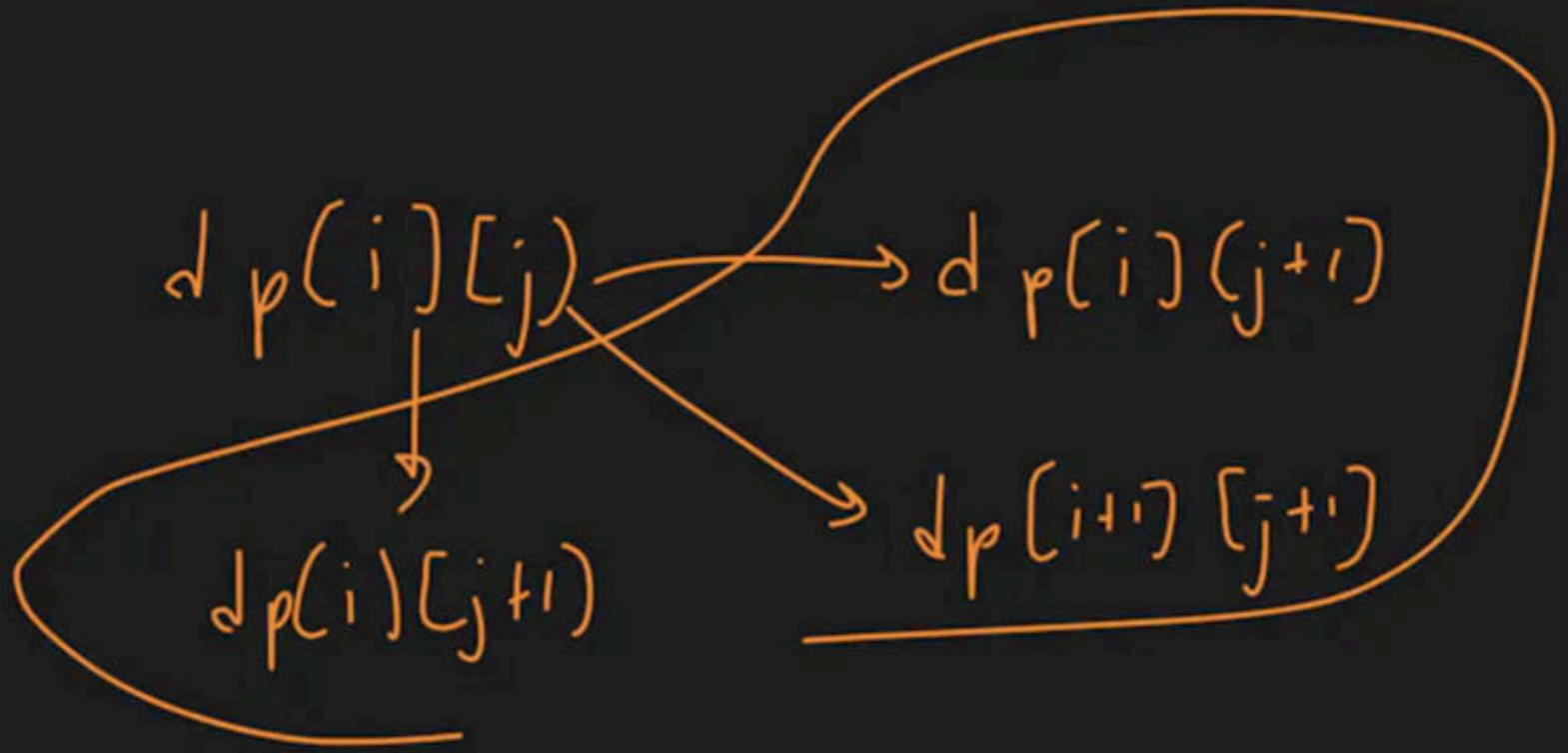
J = 3

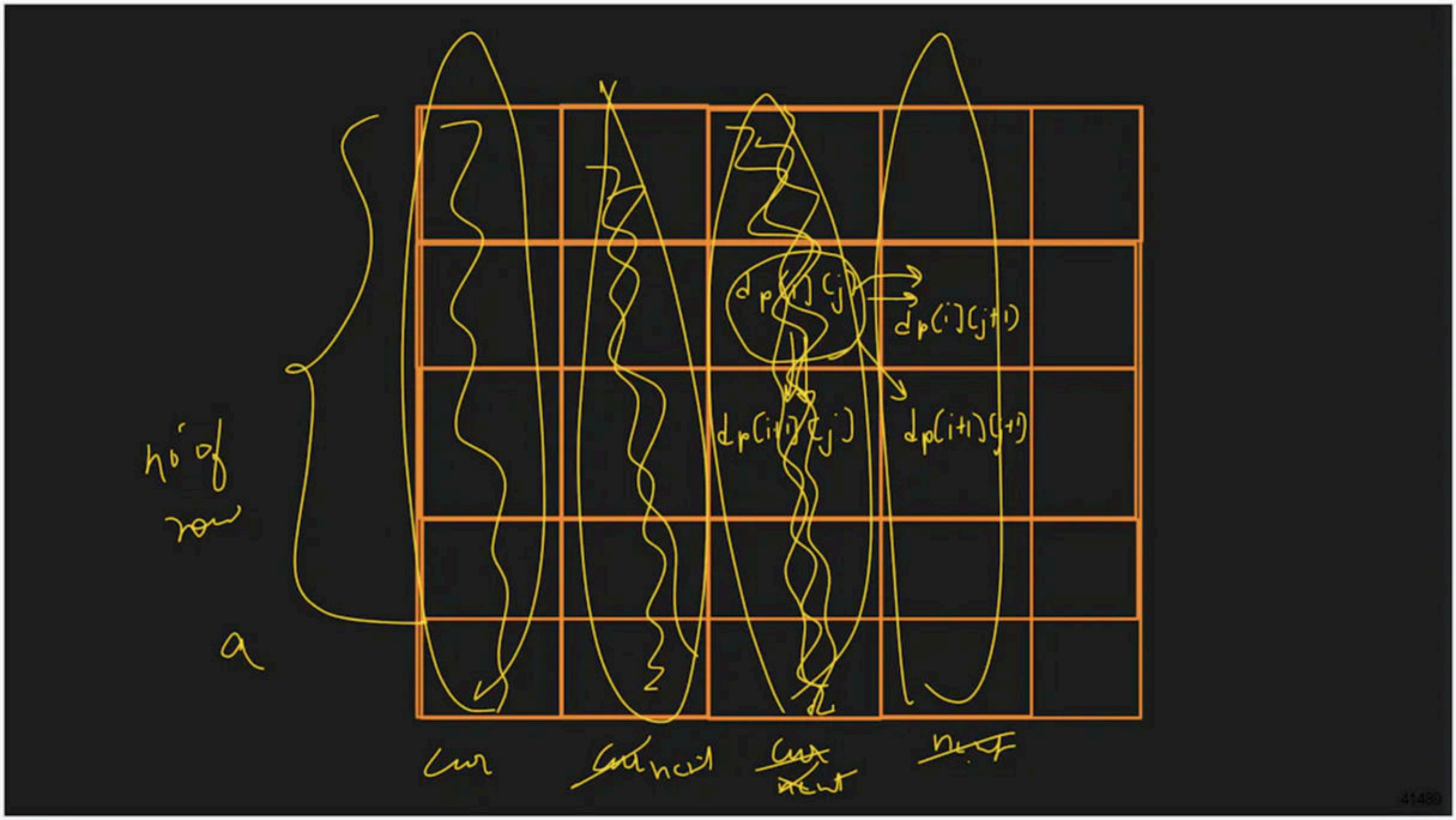
b.length − j

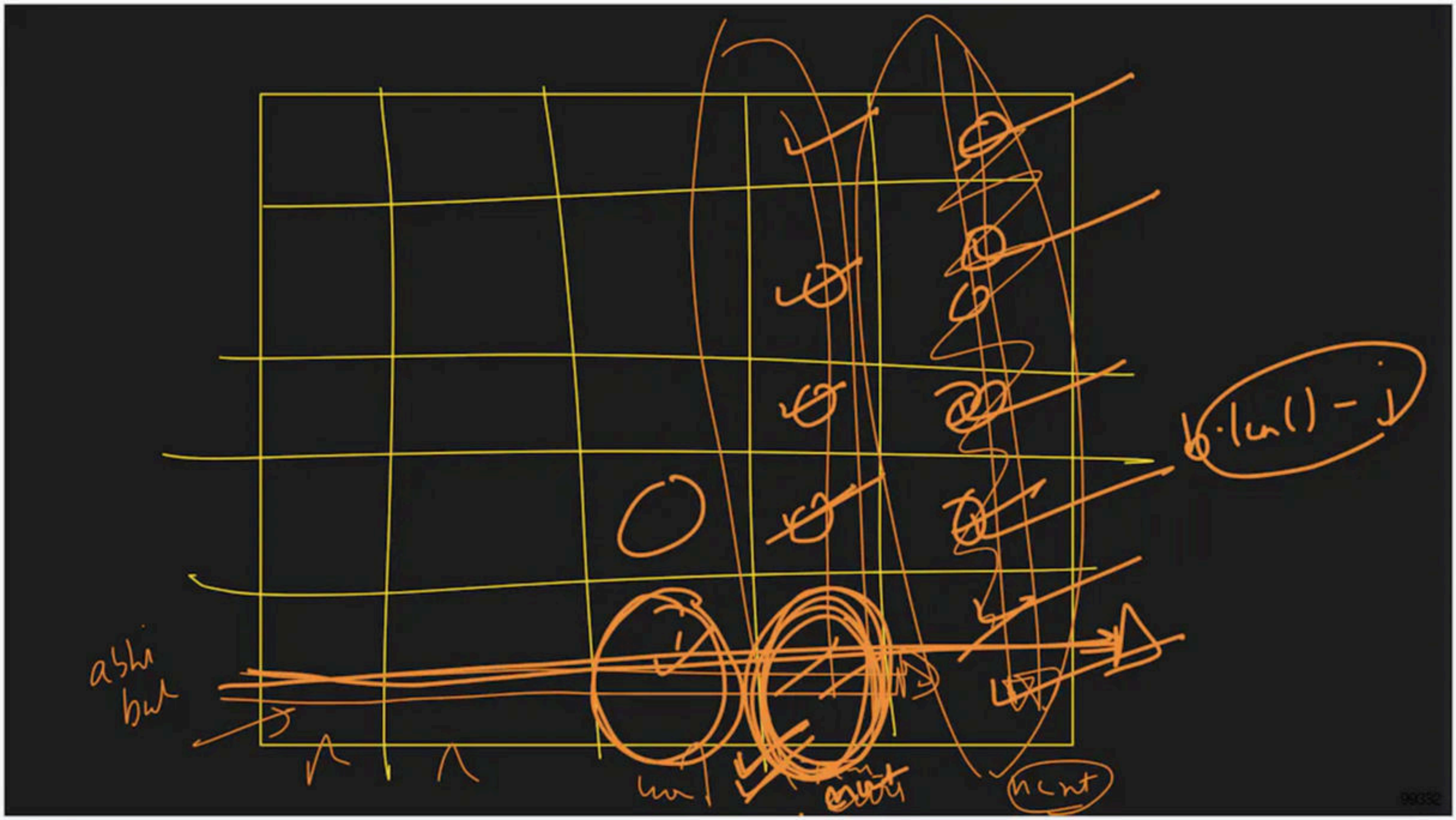
5 − 3 = 2

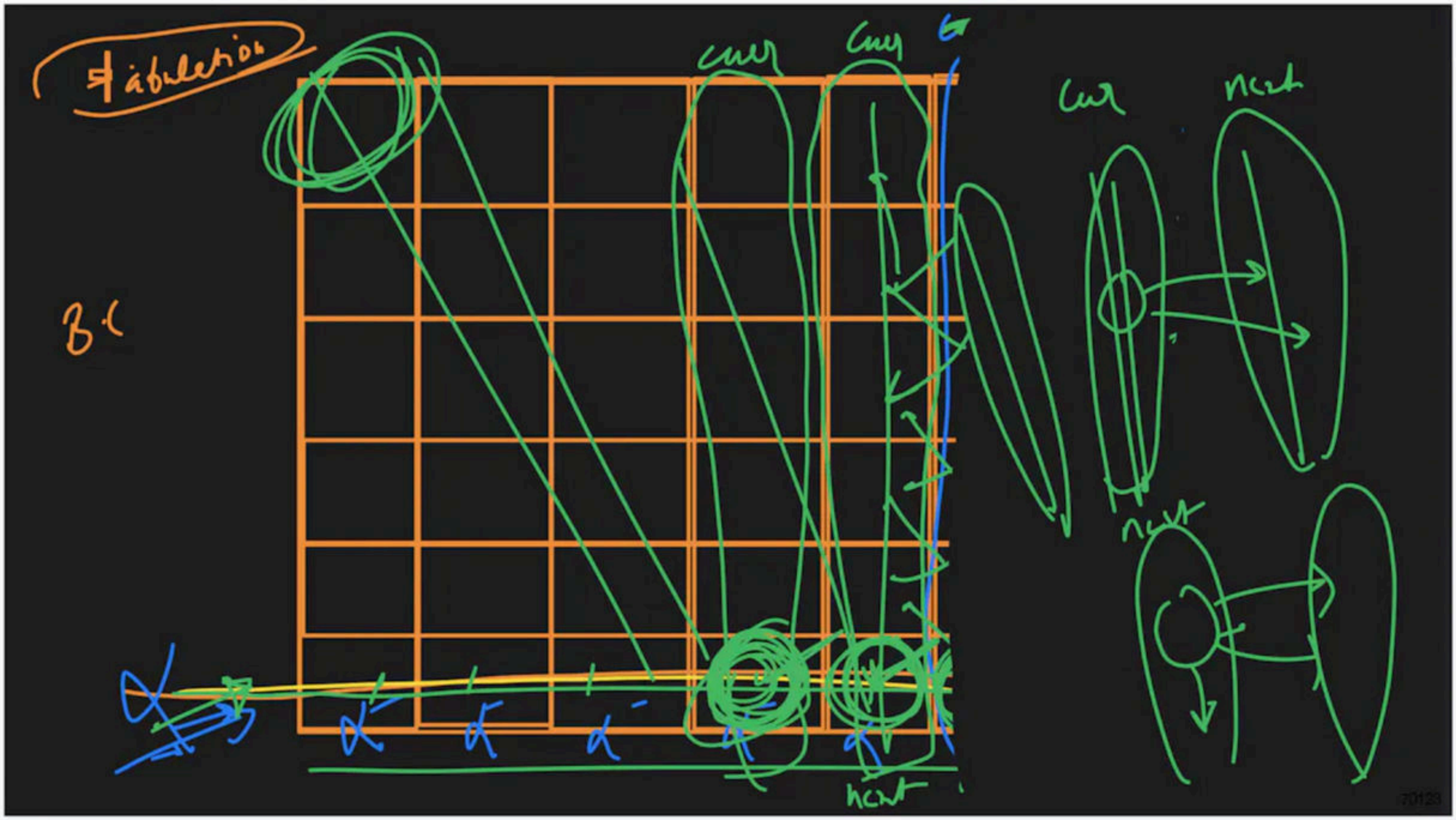

Rec
B.C
if (j == b.length())
{ return a.length() - i ;
}
a.len - i
5 - 3 = 2

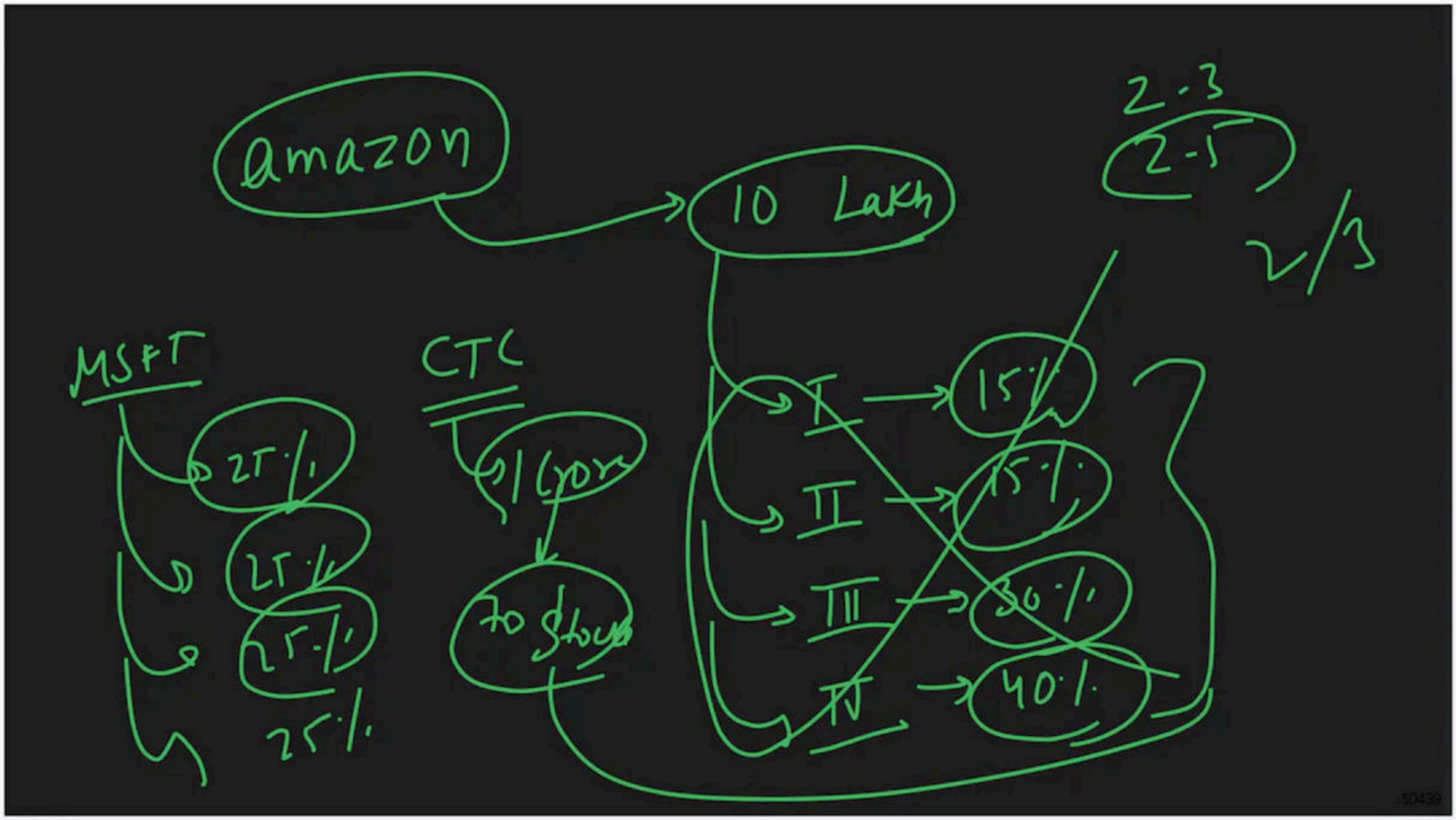

if (i == a.length())
return b.length() - j
if (j == b.length())
return a.length() - i
for (i=0; i<=a.length; i++)
dp[i][b.length()] = a.length - i
for (j=0; j<=b.length(); j++)
dp[a.length()][j] = b.length() - j
0 1 2 - - - b
0
a
3
3
3
0
b+1 col
a+1 row

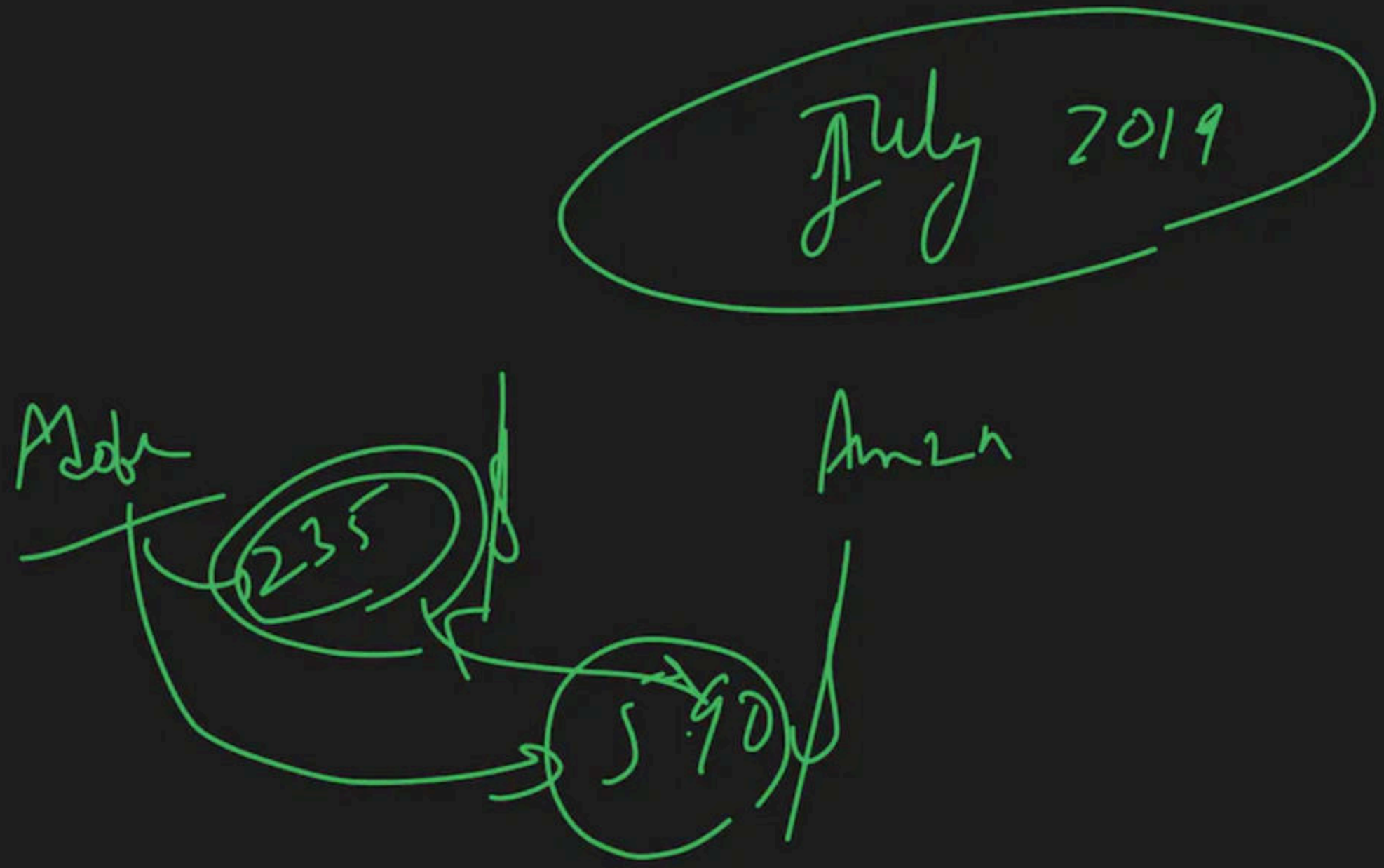

dp[i][j]
dp[i][j+1]
dp[i+1][j]
dp[i+1][j+1]

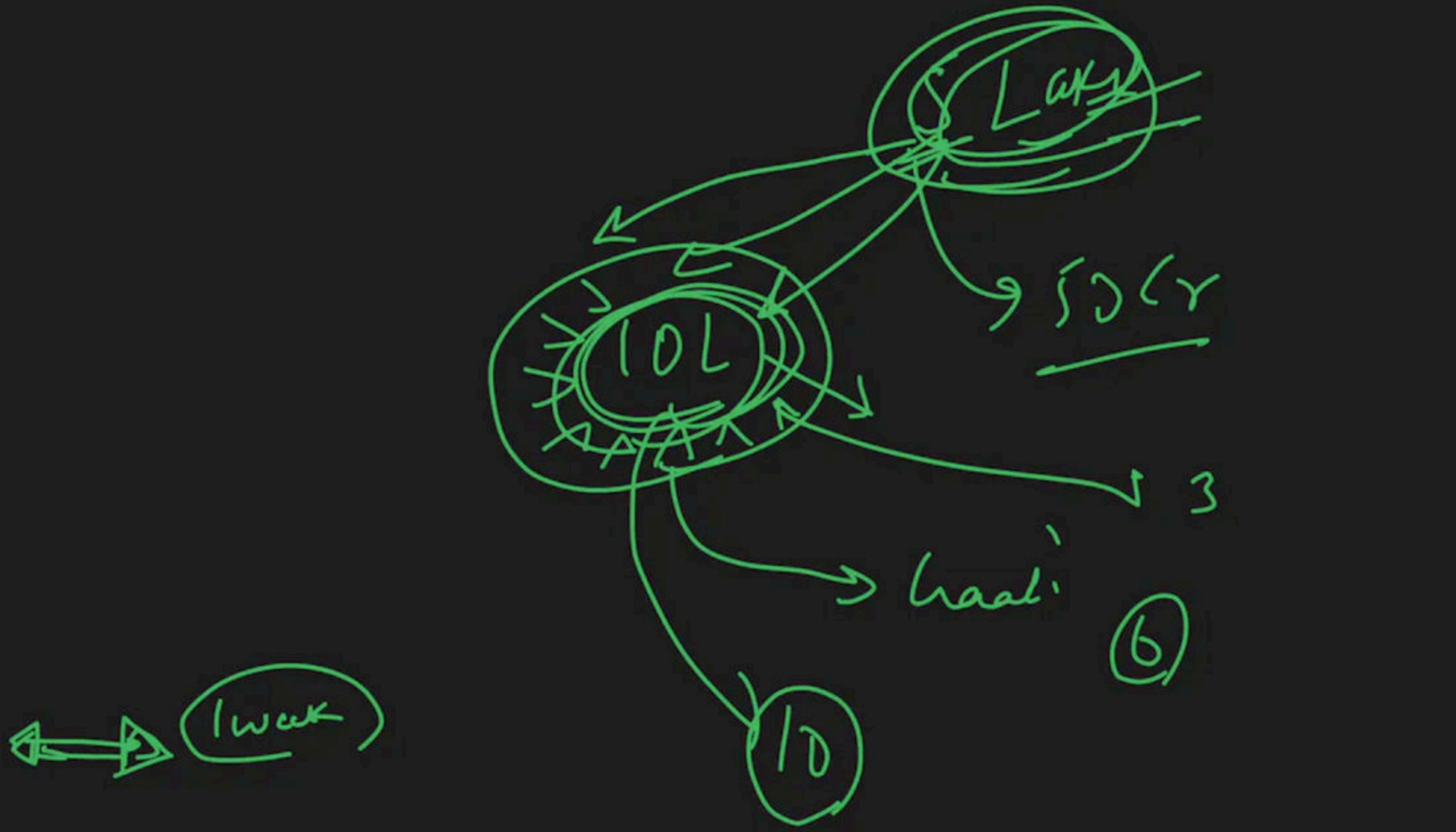

dp[i][j]
dp[i][j+1]
dp[i+1][j]
dp[i+1][j+1]
no of row
a

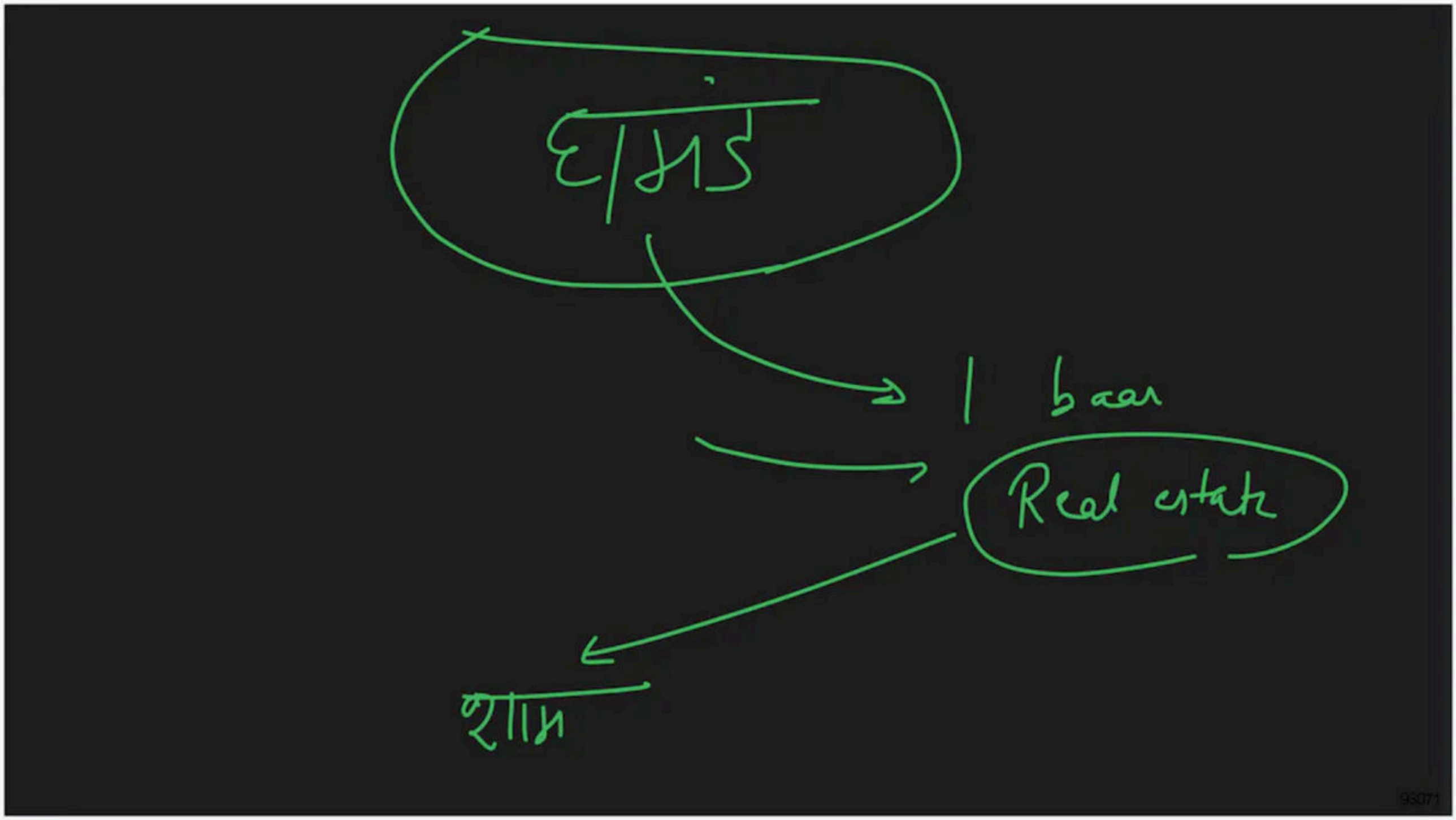

ashi
bul
next


Tabulation
curr
curr
curr
next
next
next


Amazon
10 Lakh
2·3
2·5
2/3
MSFT
CTC
I
II
III
IV
15%
15%
30%
40%
25%
1 Crore
70 Shows


July 2019
Mobile
235
Amazon
590


Lak
IOL
3
6
10


Real estate

1. Knapsack => unbounded
2. B.S wale
3. LAP =>
4. LIS =>
5. Club