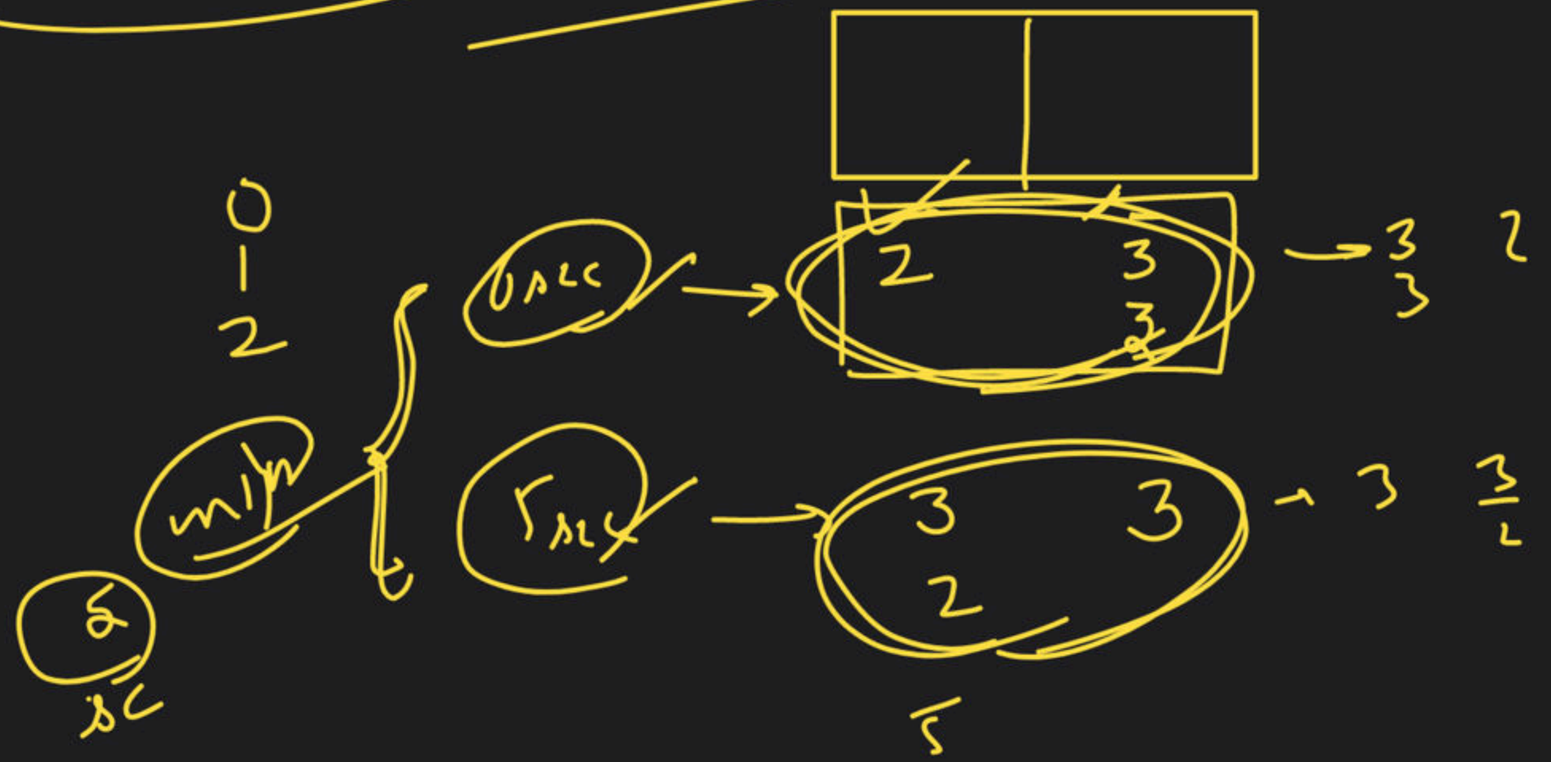
Solving Hard Problems using Backtracking - Level 2

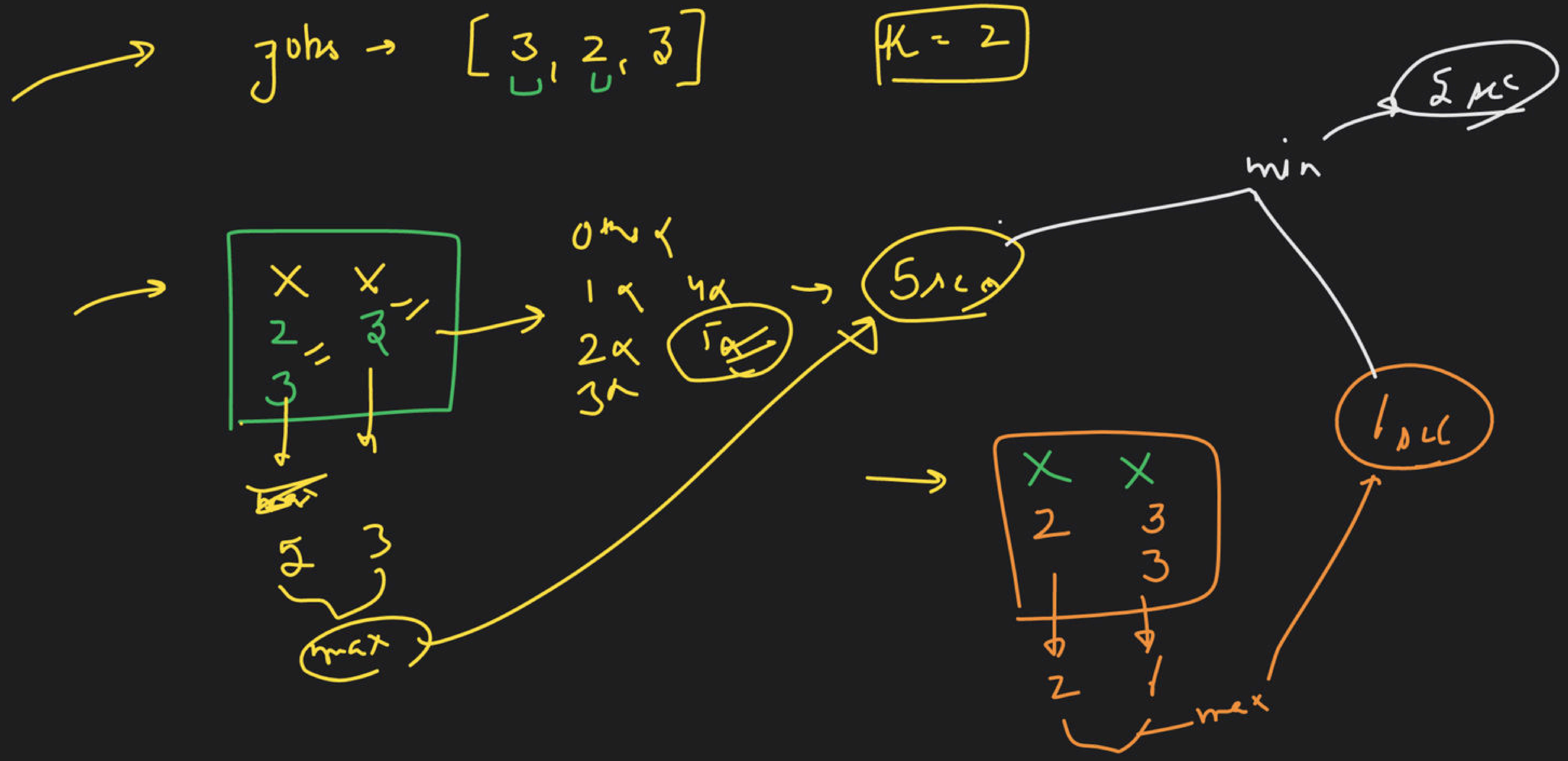
Special class

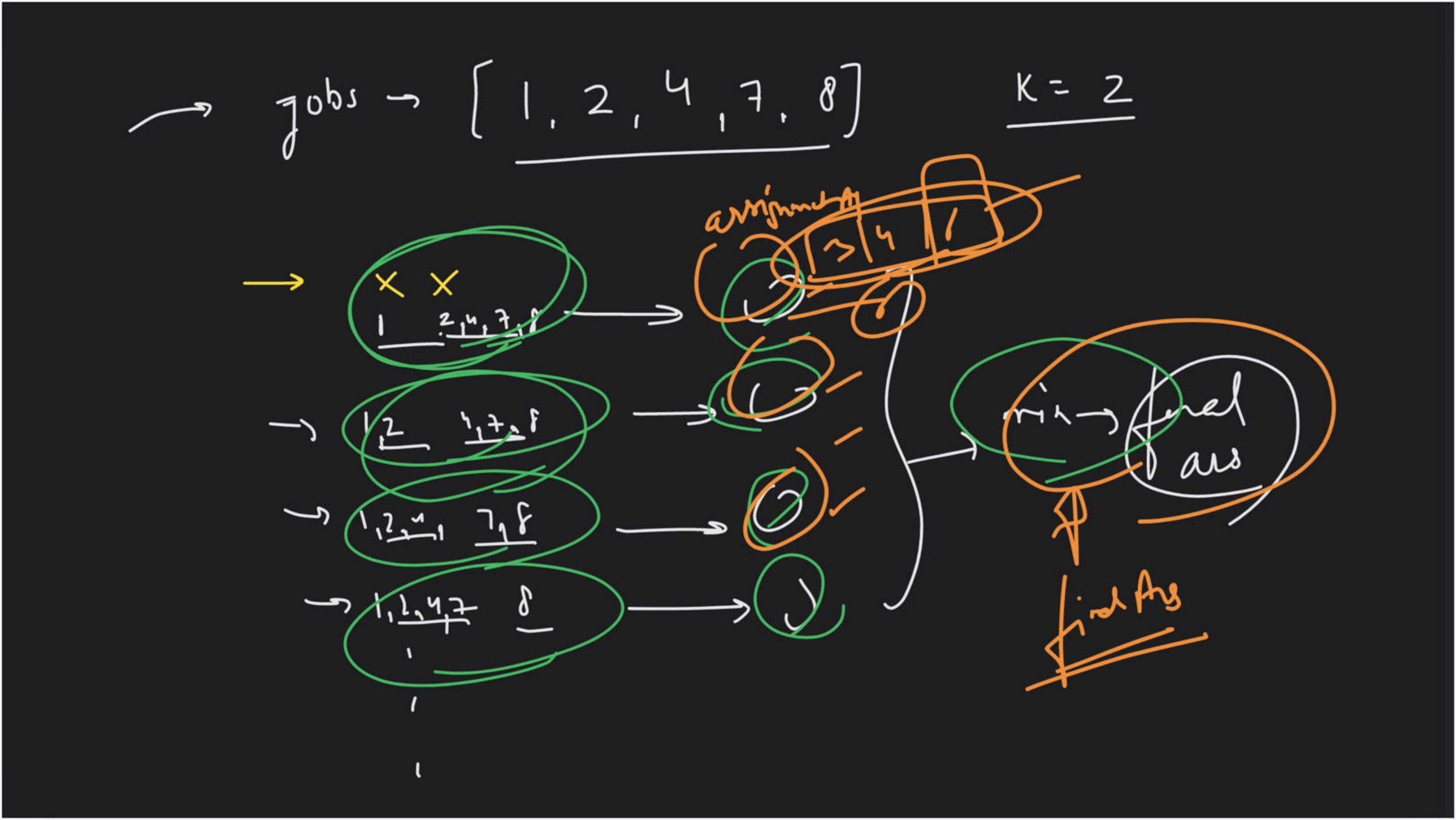
Love Babbar • Oct 15, 2022

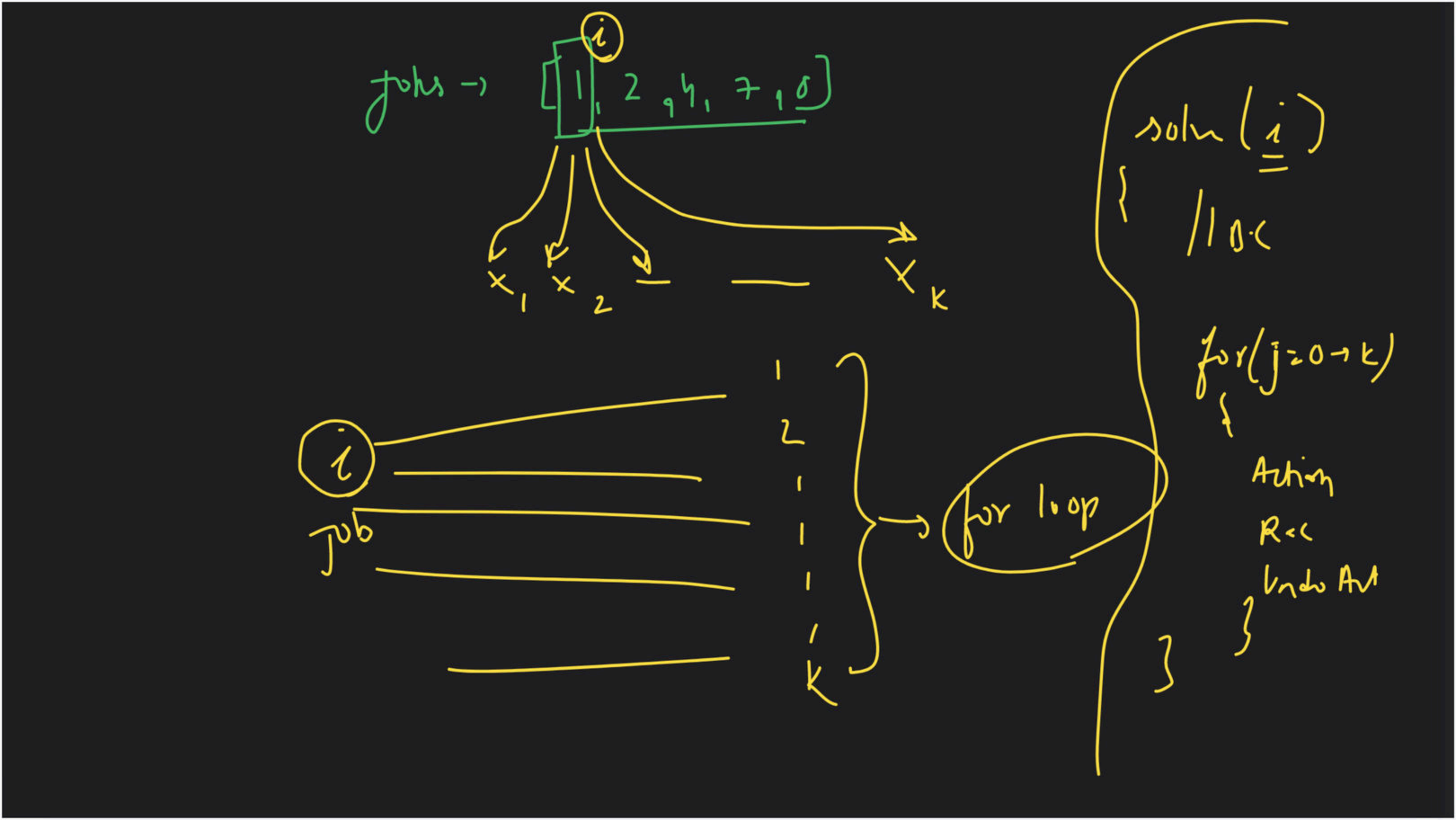
① find minimum time to finish all jobs.

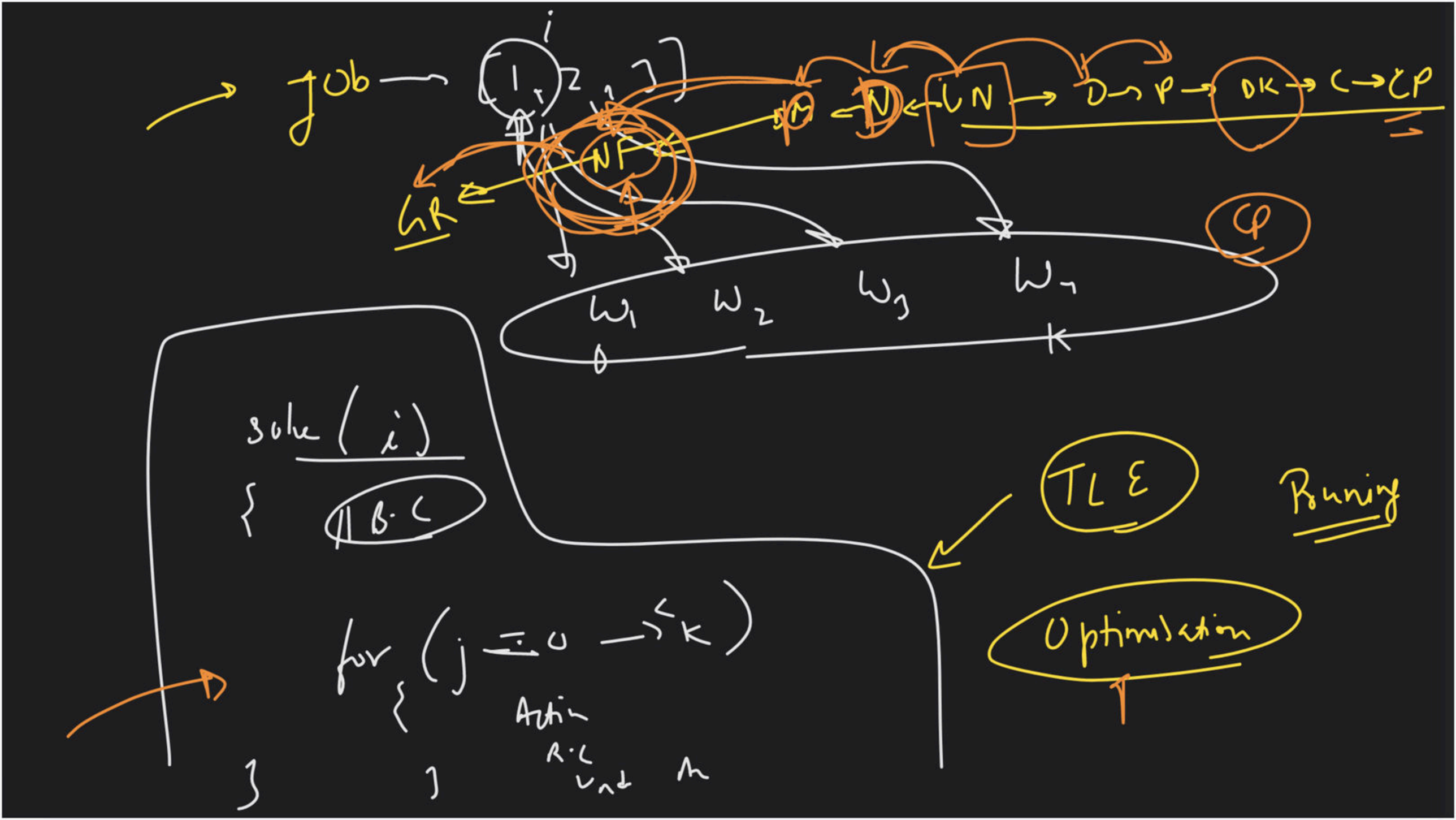
Jobs → [3, 2, 3] k = 2

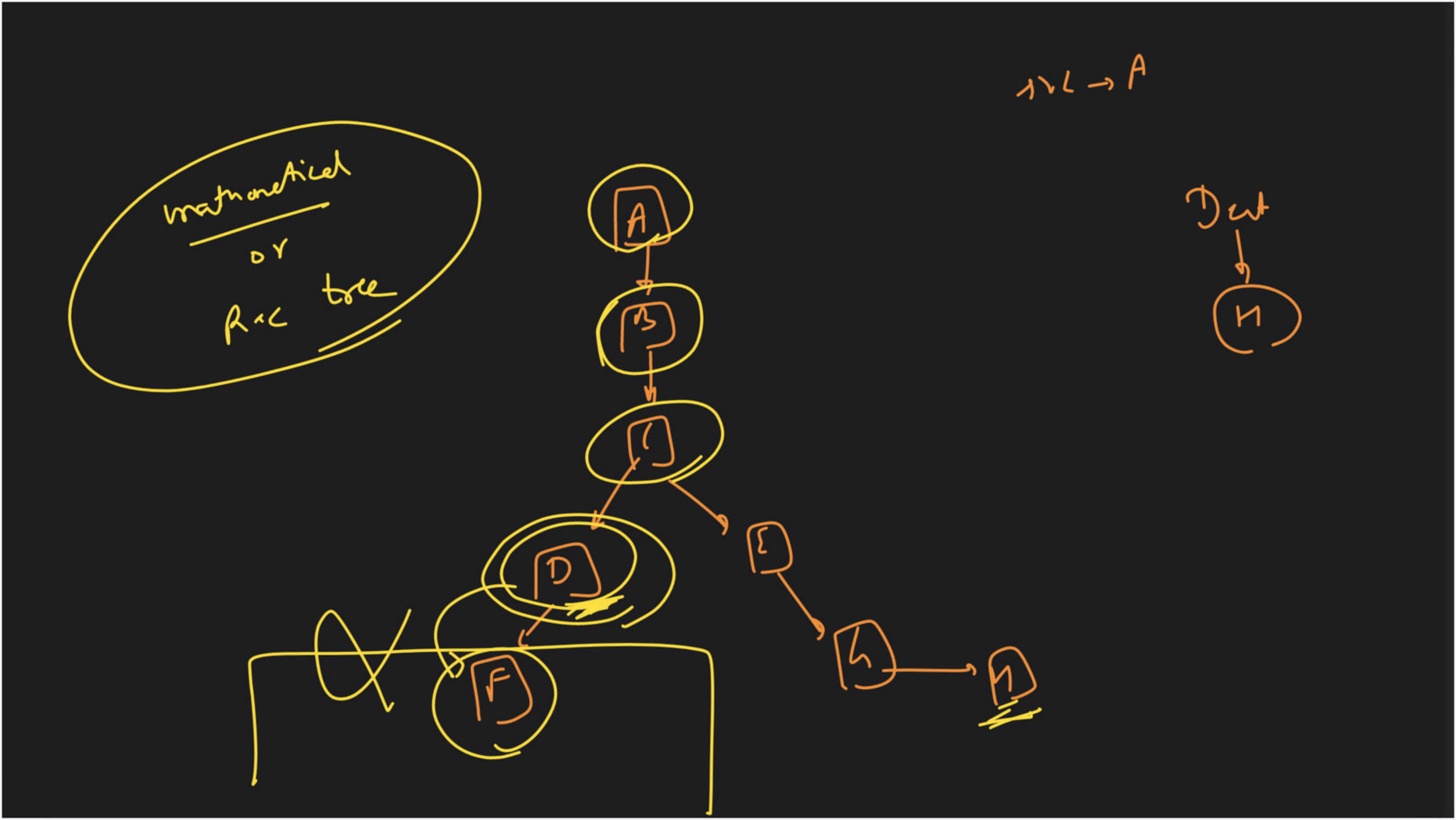



jobs → [3, 2, 3]
K = 2
5 sec
min
1 sec
max
max

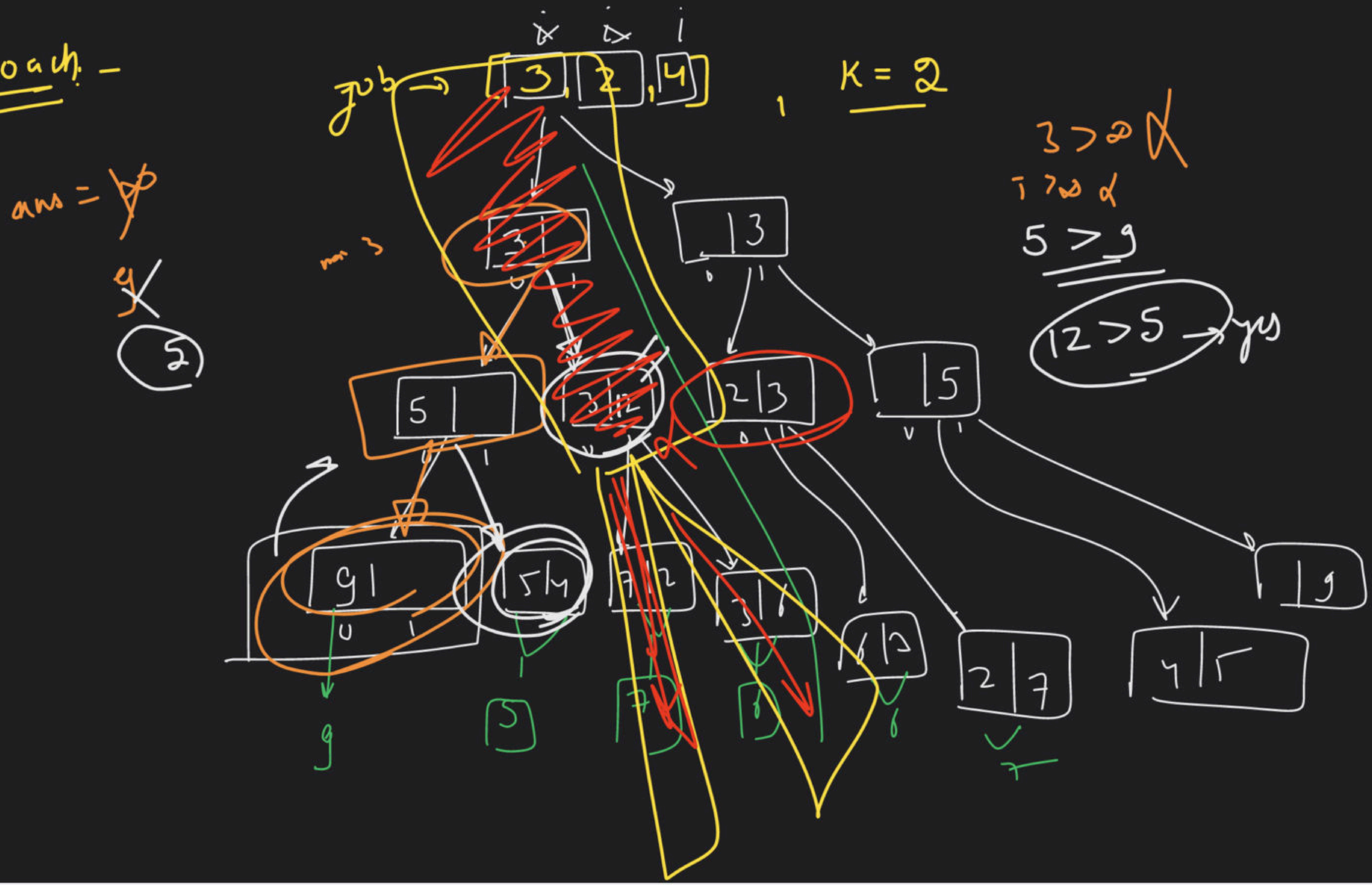

jobs → [1, 2, 4, 7, 8]
K = 2
assignment
3
4
1
min → final ans
final Ans

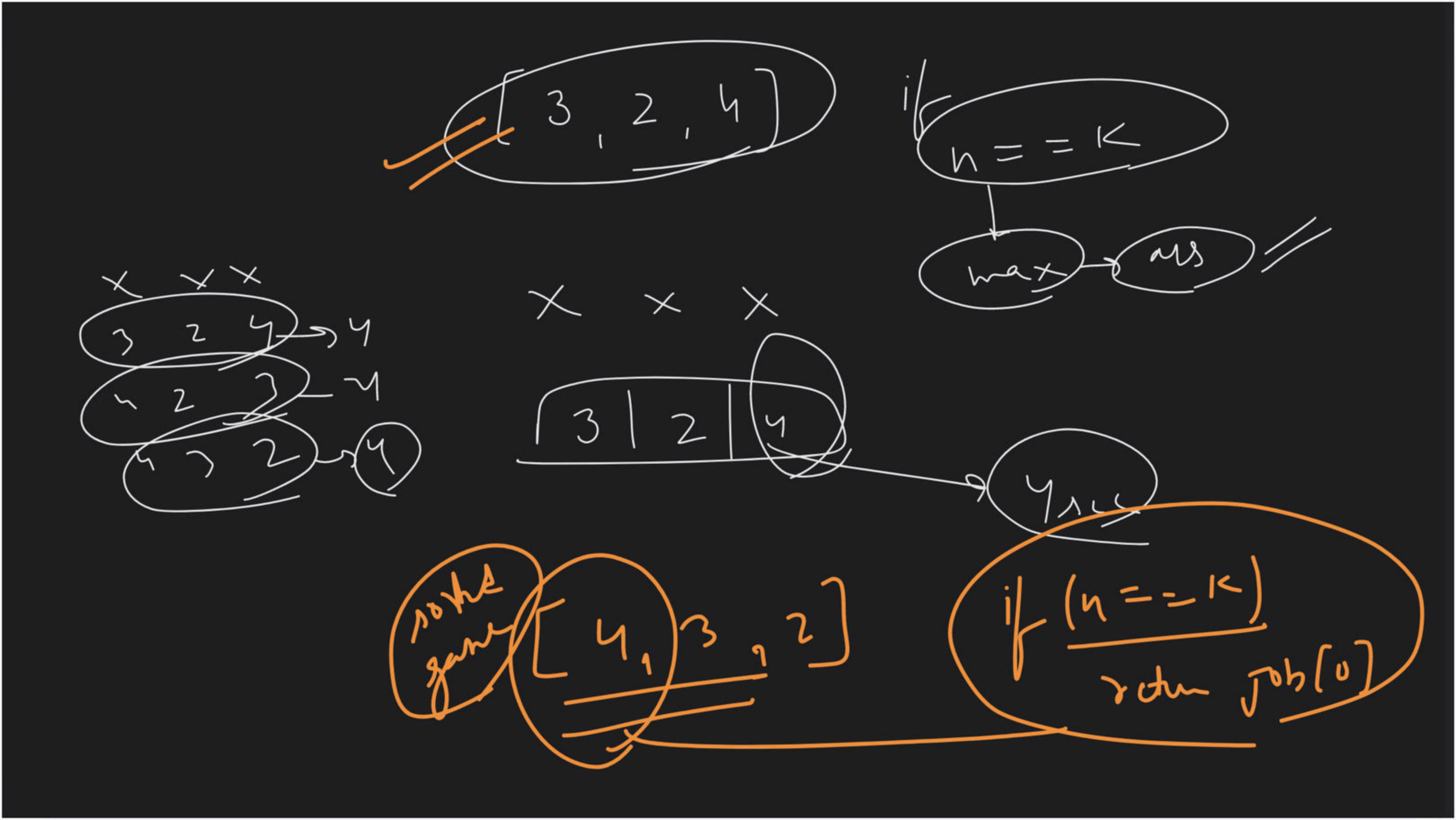

jobs →
1 2 4 7 10
i
x1
x2
xk
i
Job
1
2
k
for loop
soln (i)
{
// O.C
for(j=0→k)
{
Action
Rec
Undo Act
}
}

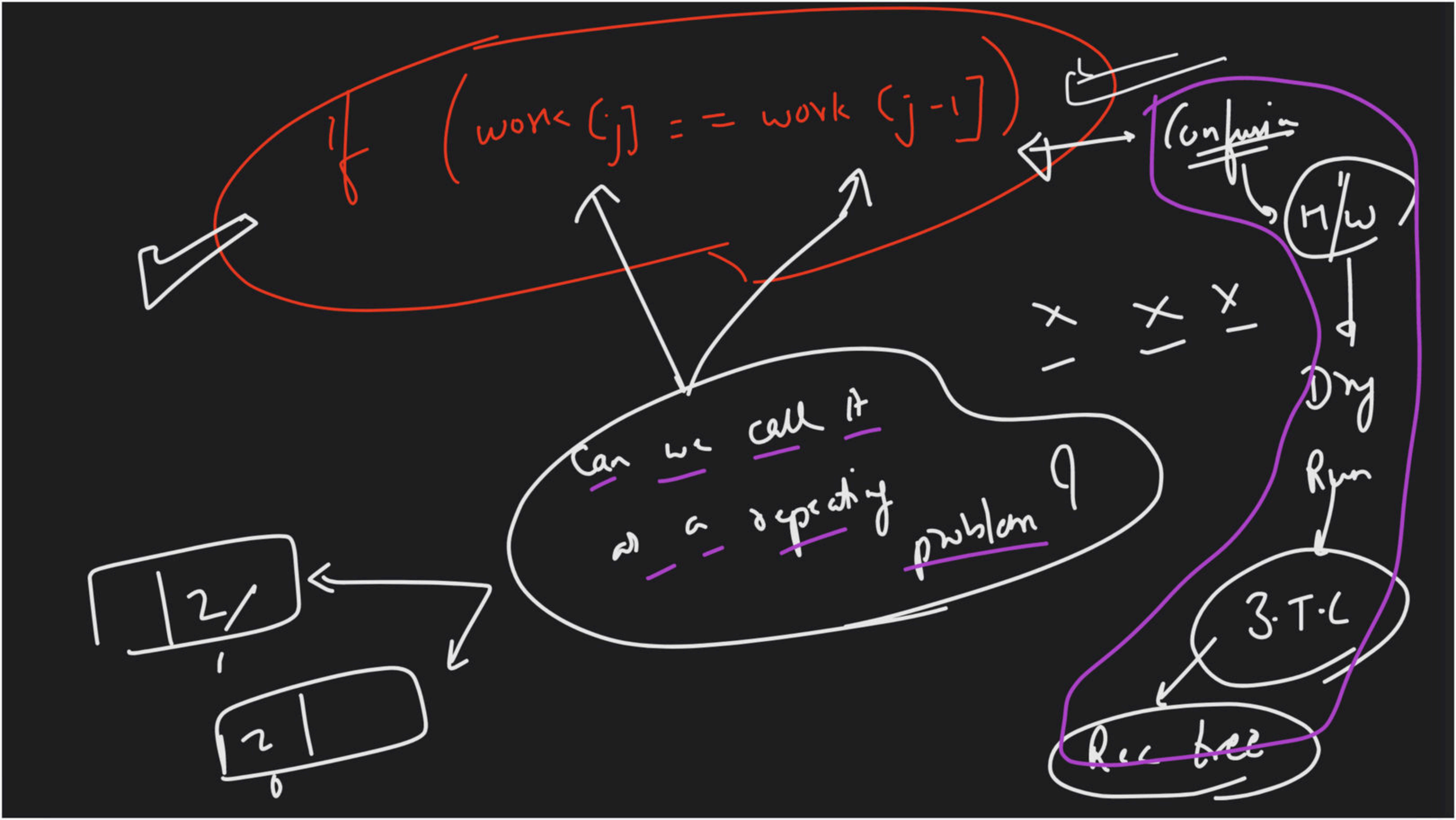

Job
NF
WR
W1
W2
W3
Wn
Solve (i)
{
B.C
for (j = 0 → k)
{
Action
R.C
Val
}
}
TLE
Running
Optimisation


Mathematical
OR
R-C tree
A
B
C
D
E
F
G
H
Dent
H

Approach -



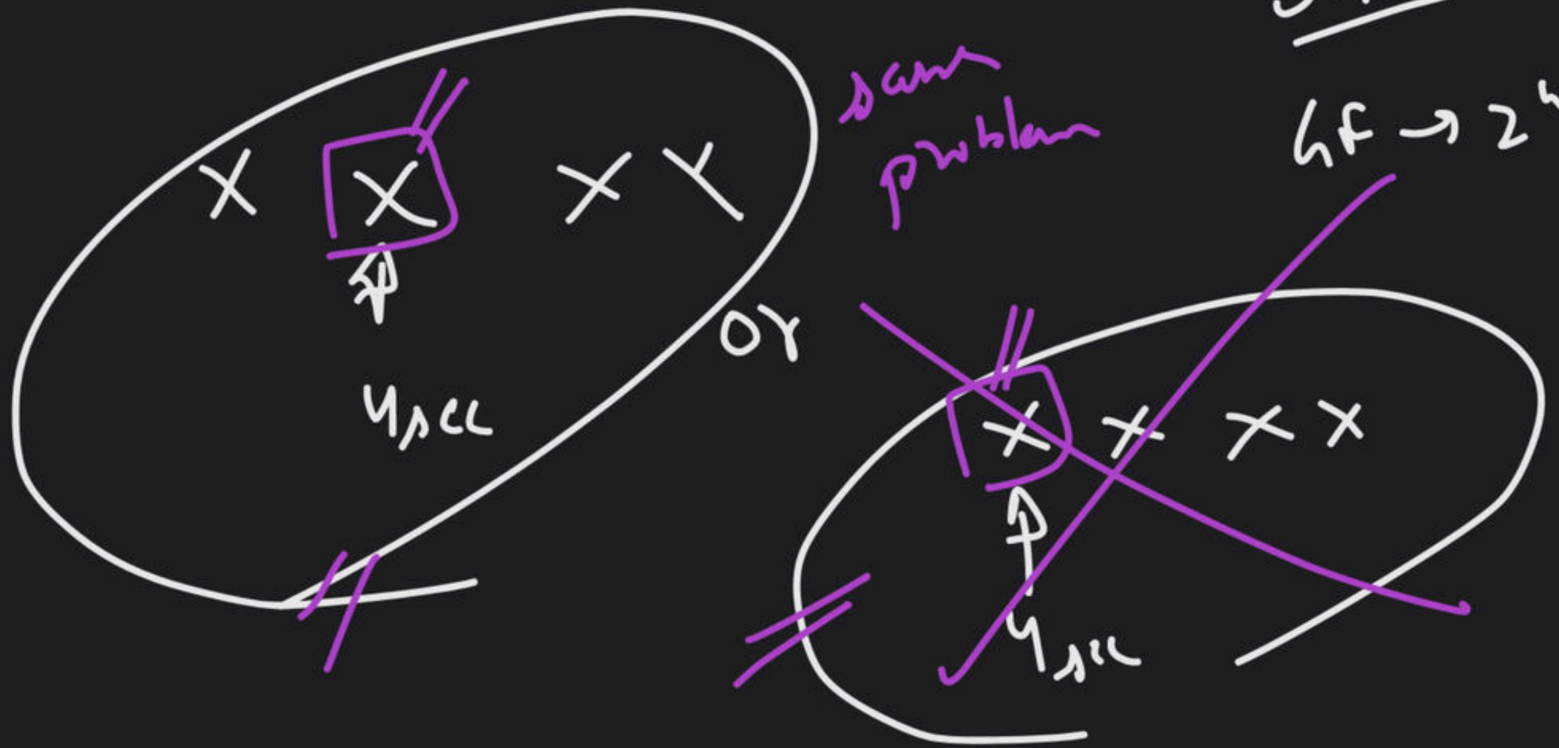

3, 2, 4
if
n == k
max
ans
3 2 4
4 2 3
1 3 2
4
4
4
3 | 2 | 4
4,1,1
solve
game
[4, 3, 2]
if (n == k)
return job[0]


if (work[j] == work[j-1])
Can we call it
as a deadlock
problem ?
2
2
Confusion
H/W
Dry
Run
3.T.C
Rec bee

2×2 = 4

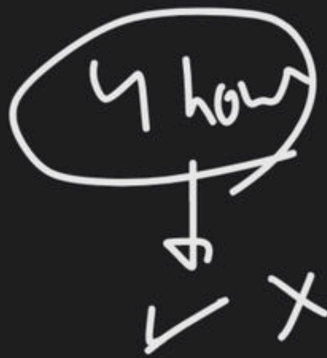
2×2α

A B C D



Brick

4f → 2^4d



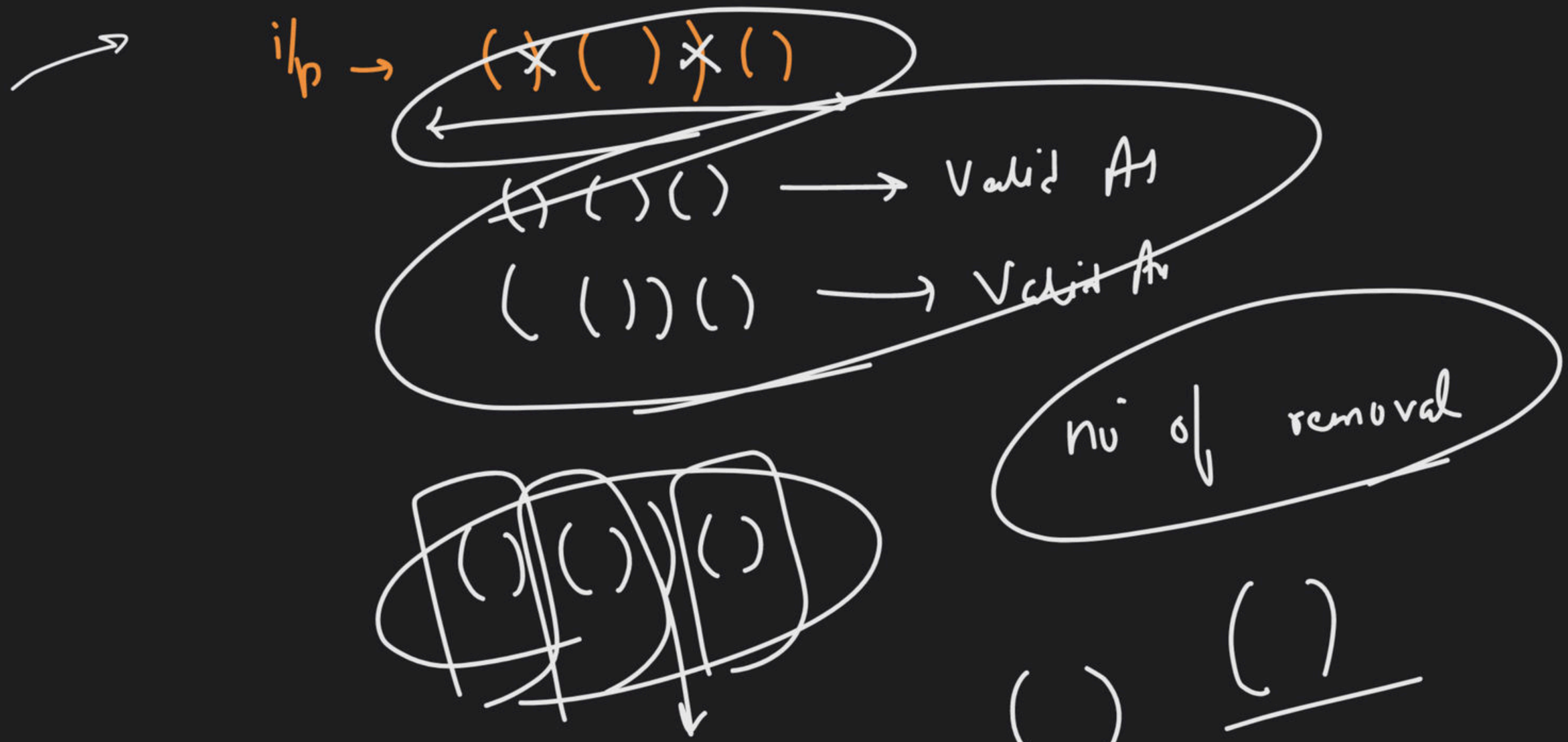
Remove Invalid parenthesis → HARD

2 min

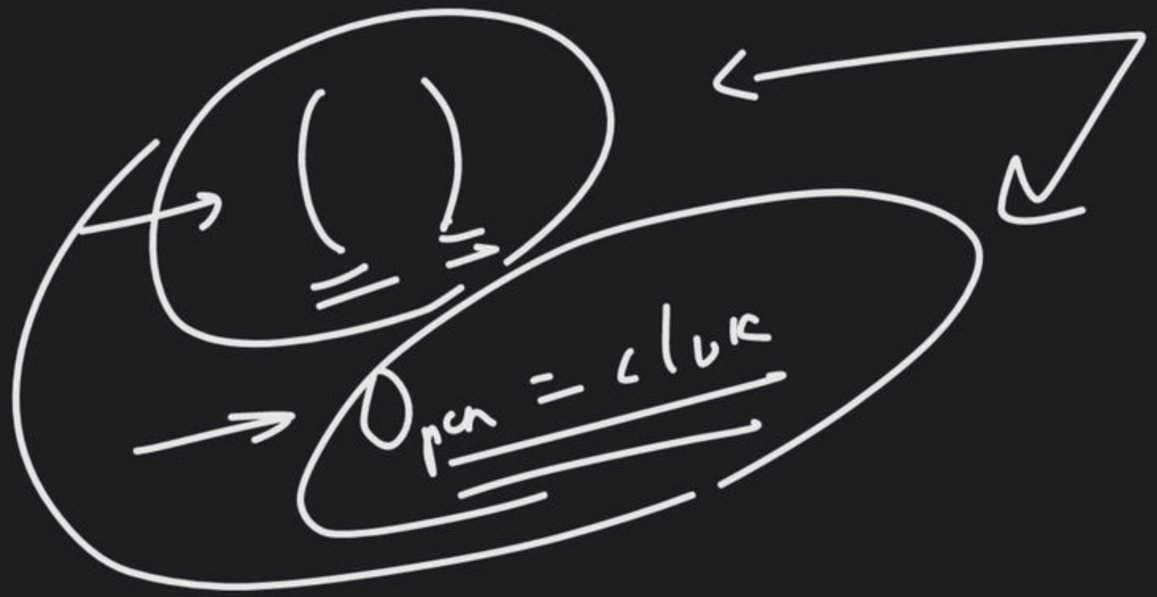
Basics

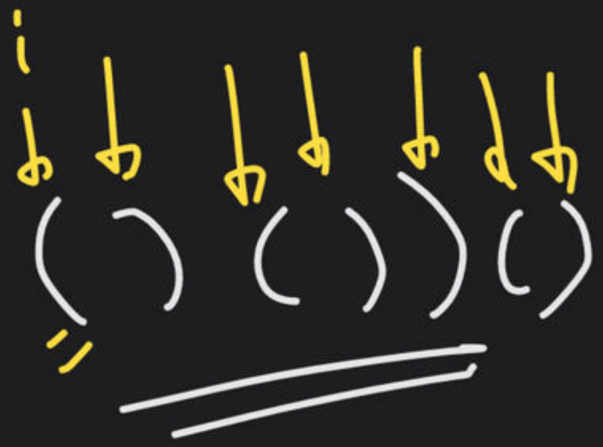
- Valid Parenthesis → Stack → Easy
  - ↳ Leetcode 20
- Make min of steps to make string valid → Medium
- Min Addition to make parenthesis Valid → Medium

H/W


i/p →
( ) ( ) ( )
( ) ( ) ( ) → Valid A1
(( )) ( ) → Valid A2
no of removal
( )
( )

Valid parenthesis





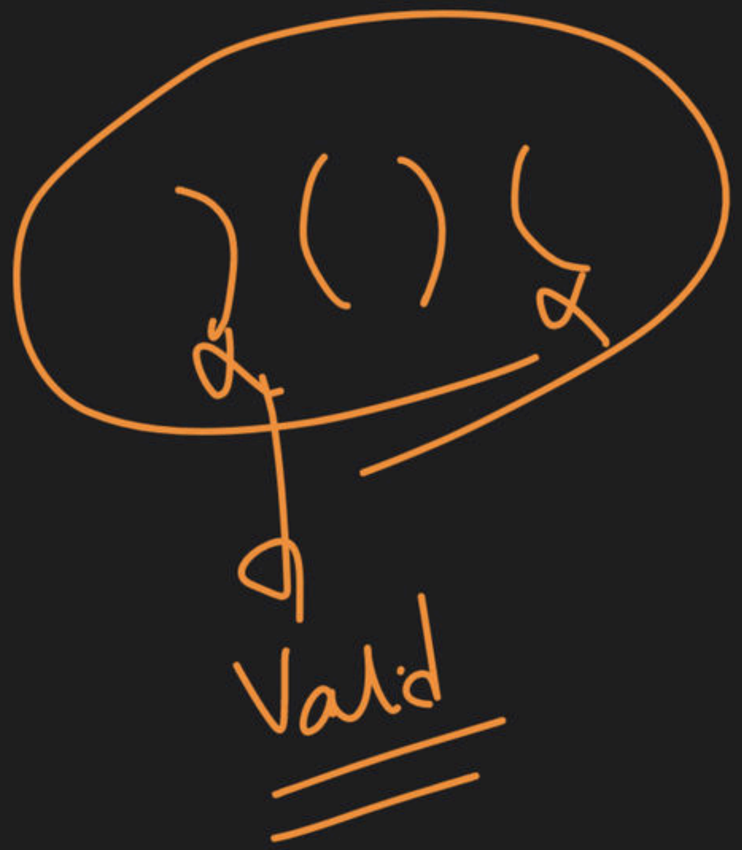
stack

no of removal → `(` → st - push

`)` → st.top → `(` → st.pop

NO

st.push(`)`)



st.size()

↳ no of removal

map<string, int>

(( ))) ( )

z → len

no of removed ↳ 1

→ ( ))) ( ) α

→ ( ))) ( ) α

(( )) ( )

→ (( )) ( )

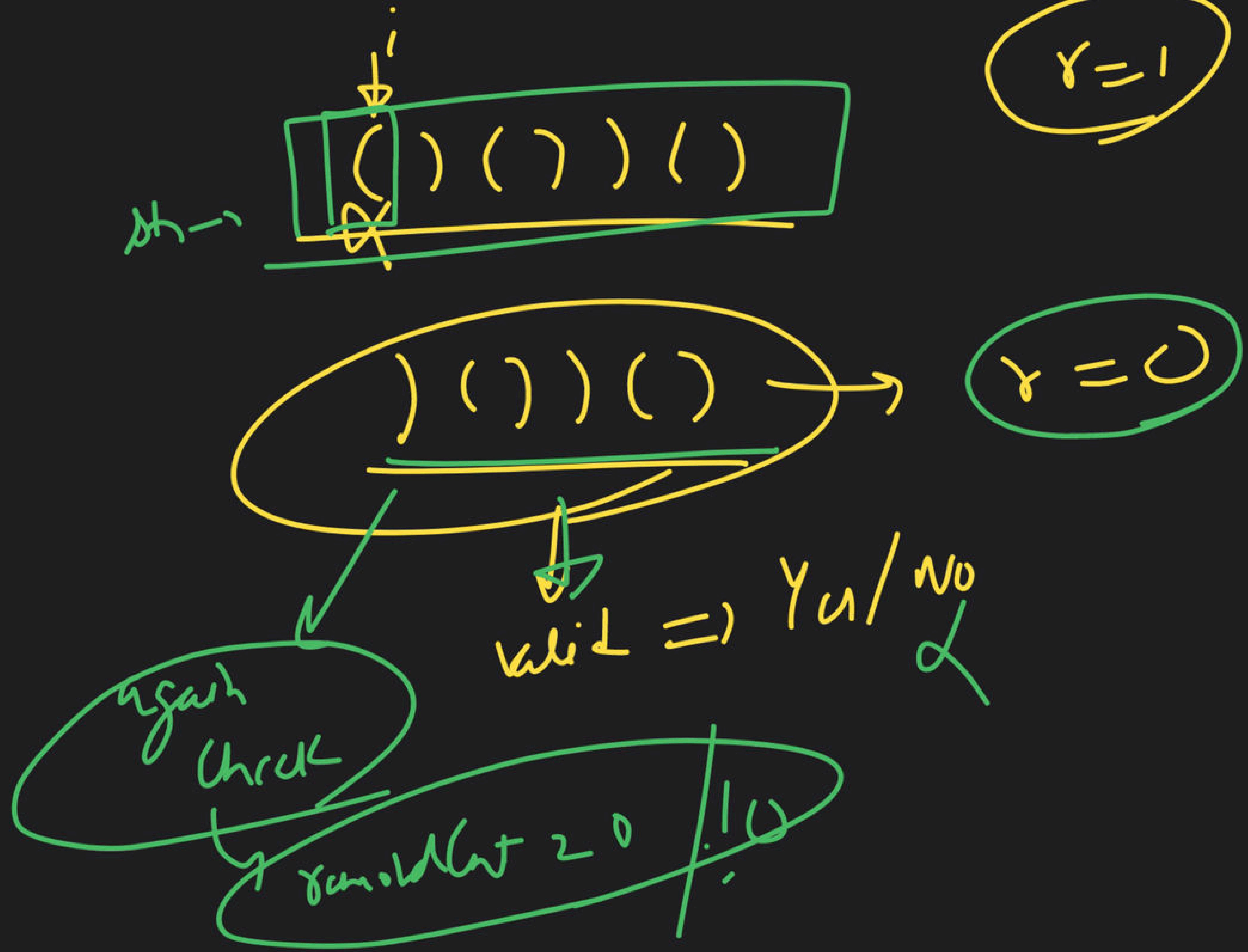
→ (( )) ( )

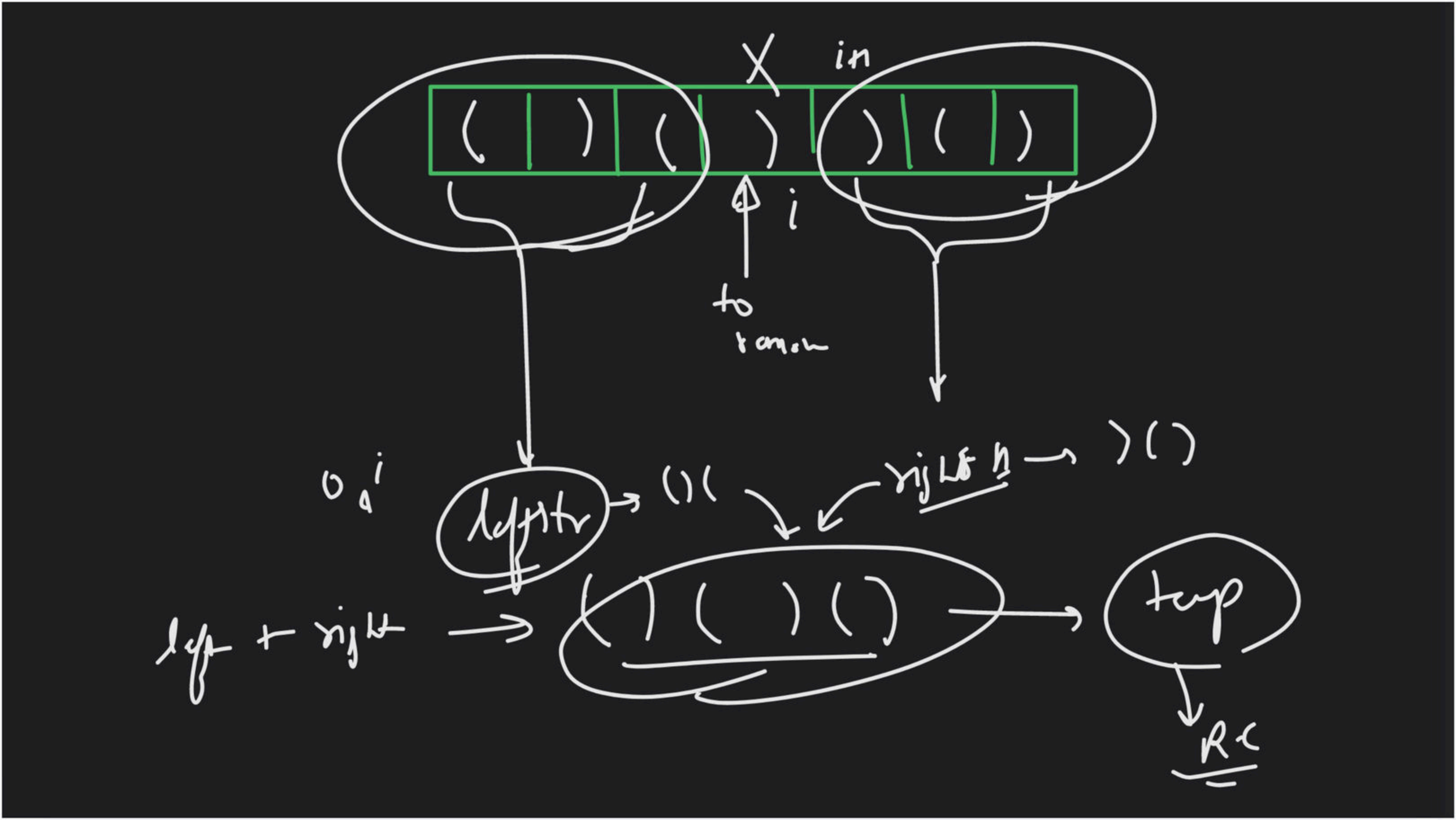
1

(( ))) α

(( ))))

2


i
str
( ) ( )) ( )
r = 1
) ( )) ( )
r = 0
valid => Yes / No
again check
removedCnt 2 0


X
in
(
)
(
)
)
(
)
i
to
token
0
i
left str
)(
right str
)()
left + right
temp
Rc

Valid Arithmetic Puzzle

Leetcode