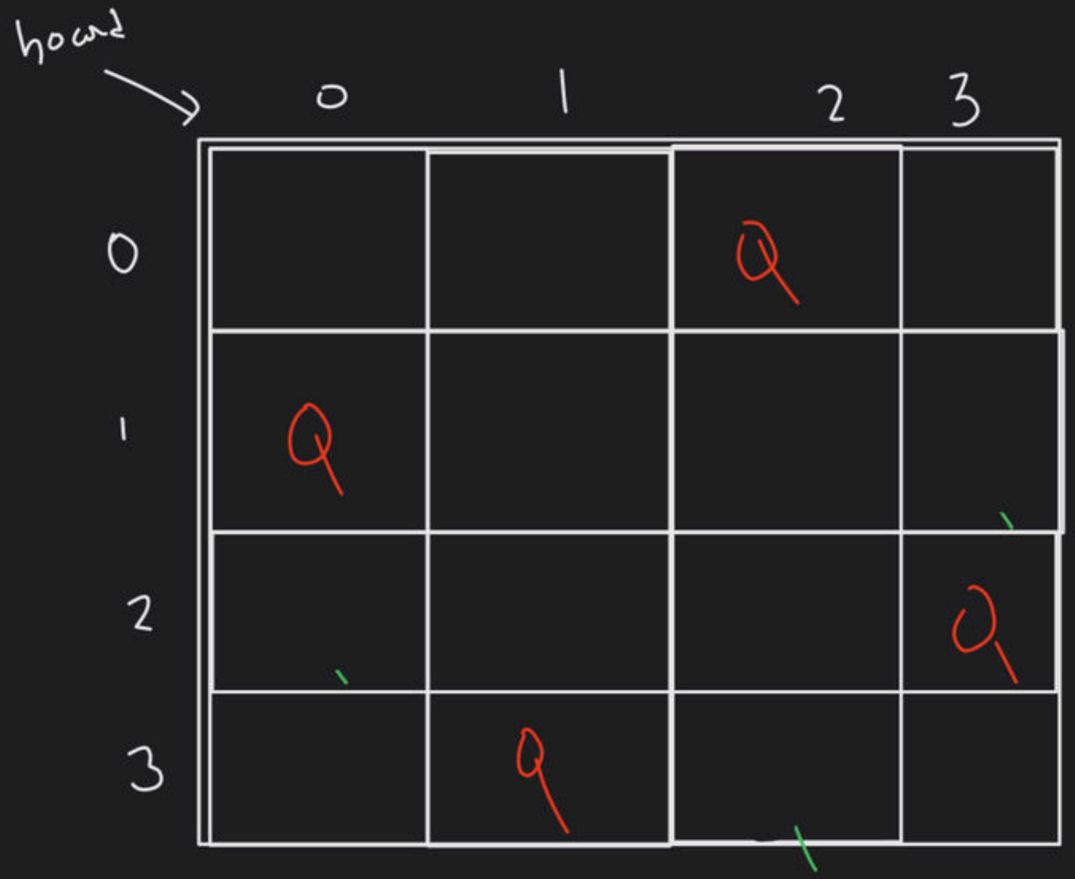
DnC Level-4

Special class

Love Babbar • Mar 24, 2023

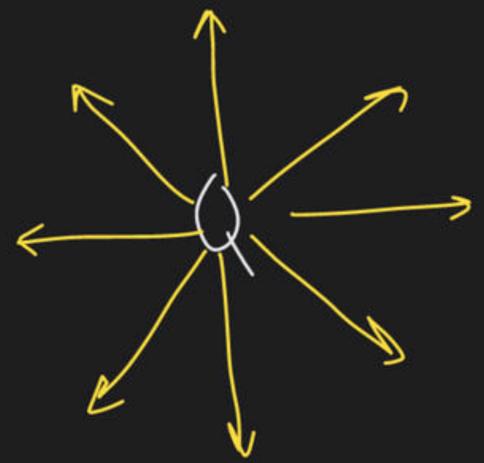
N - Queens

Queen place



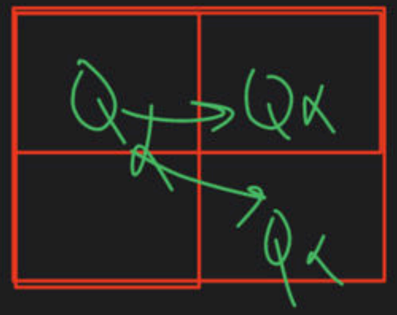
4 Queen

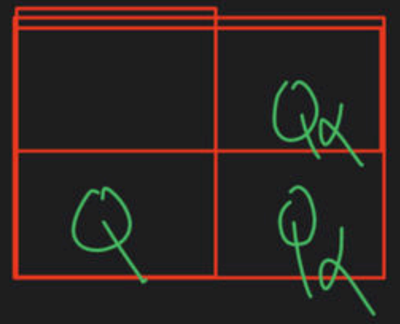
8 type movement



i/p → n → n×n board
→ n queen

N - Queen → such that no Queen can attack the other Queen





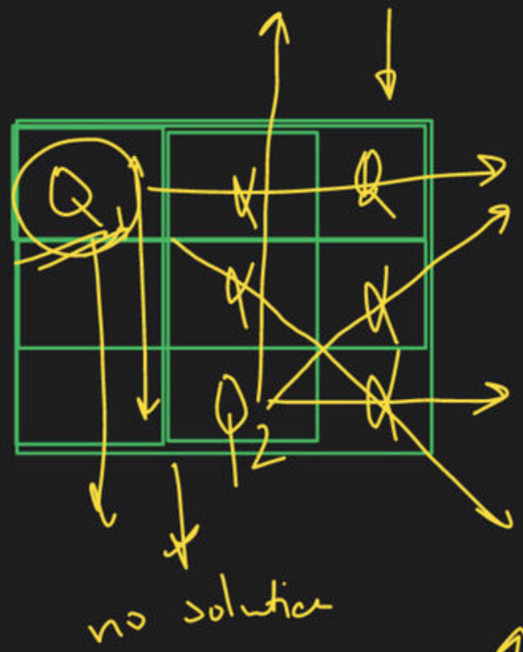
2 < 2 → bound → ✗ no solution

3 tower

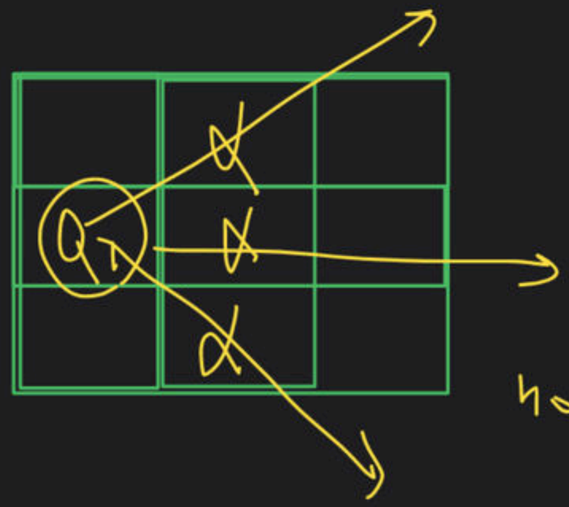
0 0 0

N - Queen

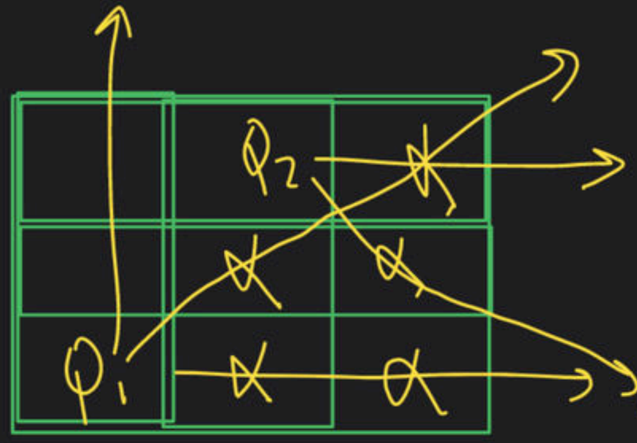
$Q_1, Q_2, Q_3 \dots Q_n$



no solution



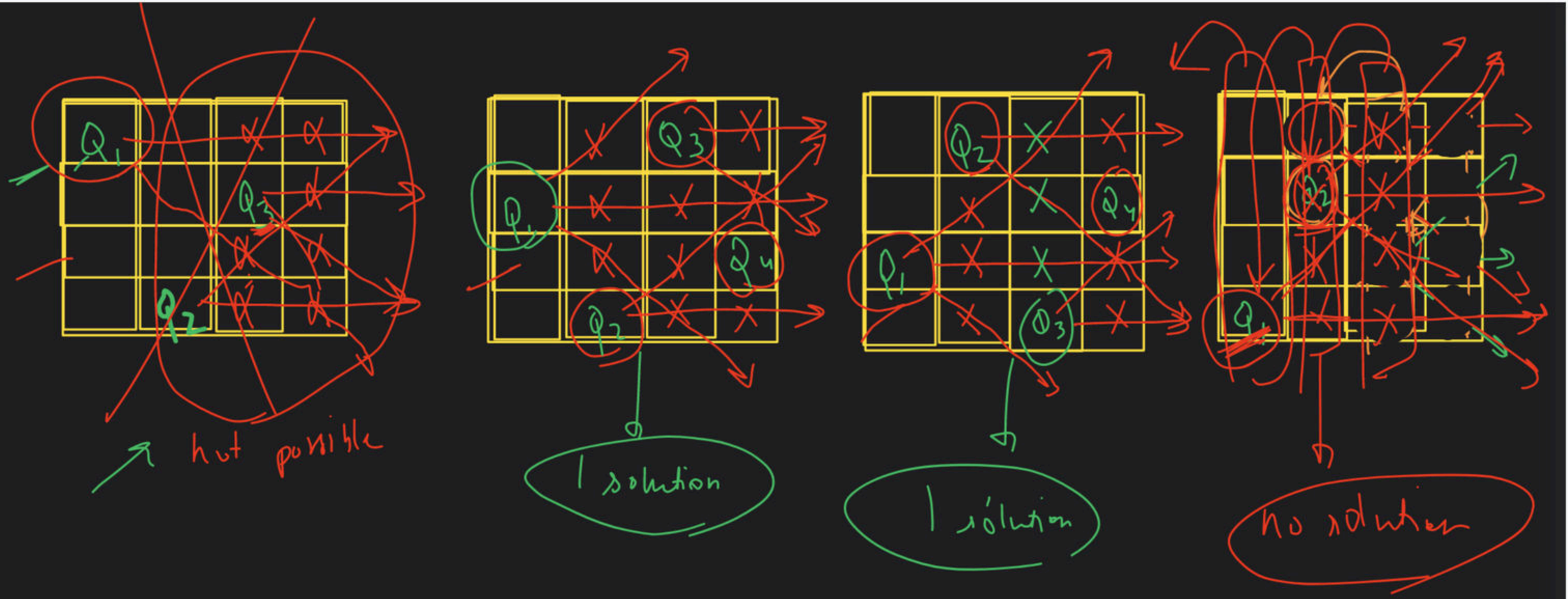
no solution

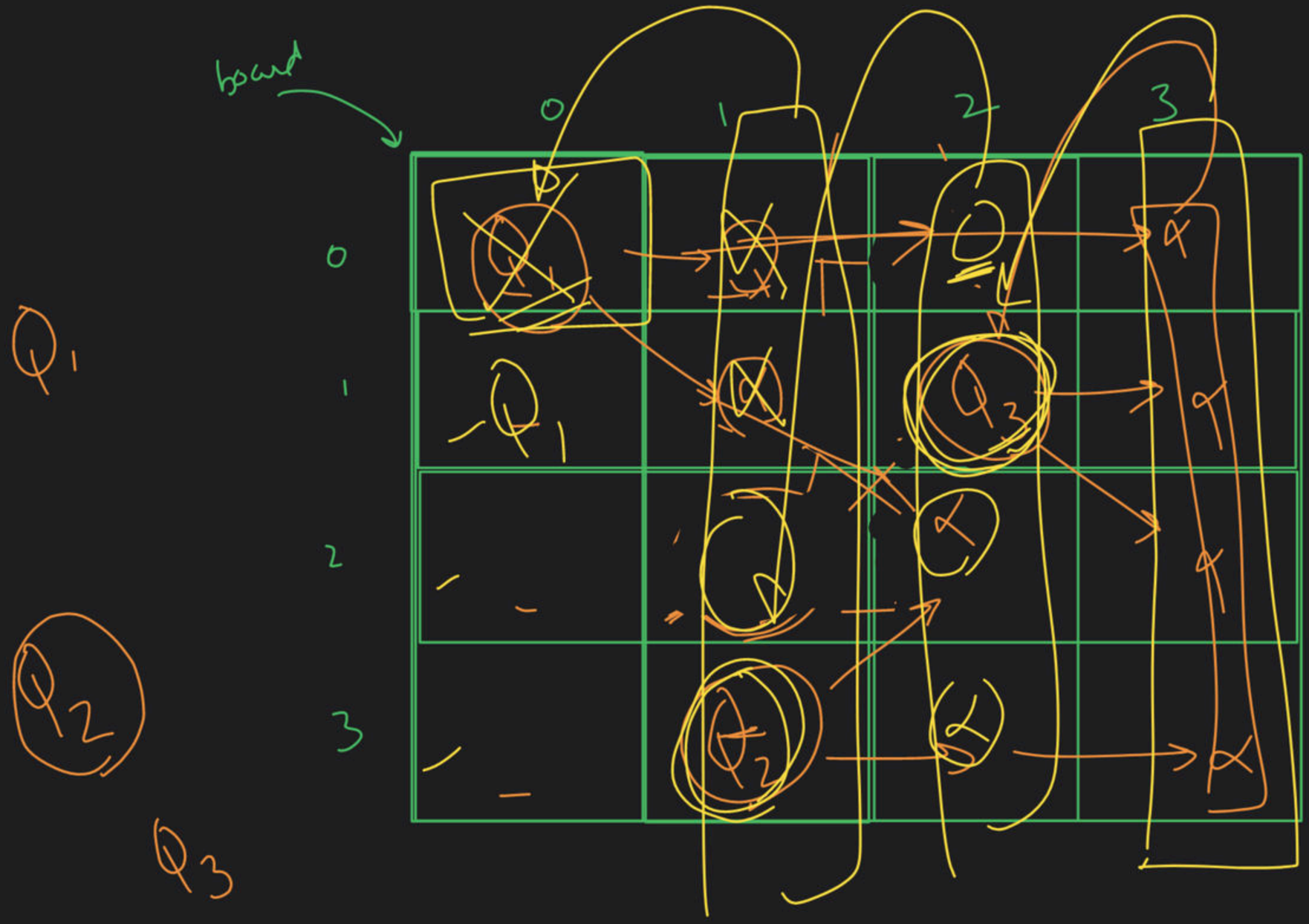


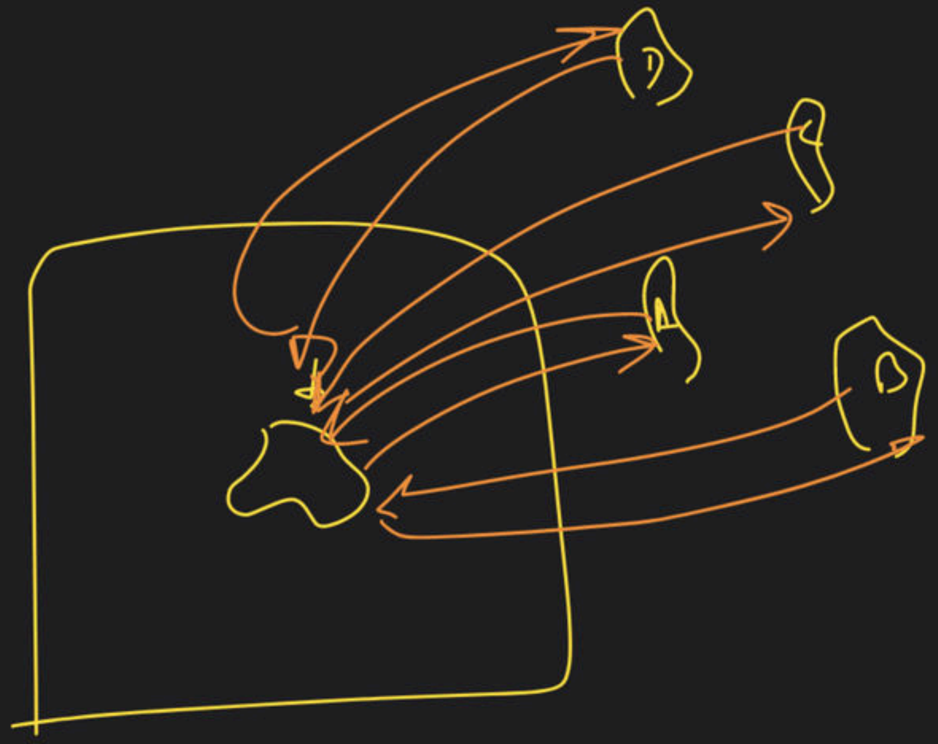
no solution

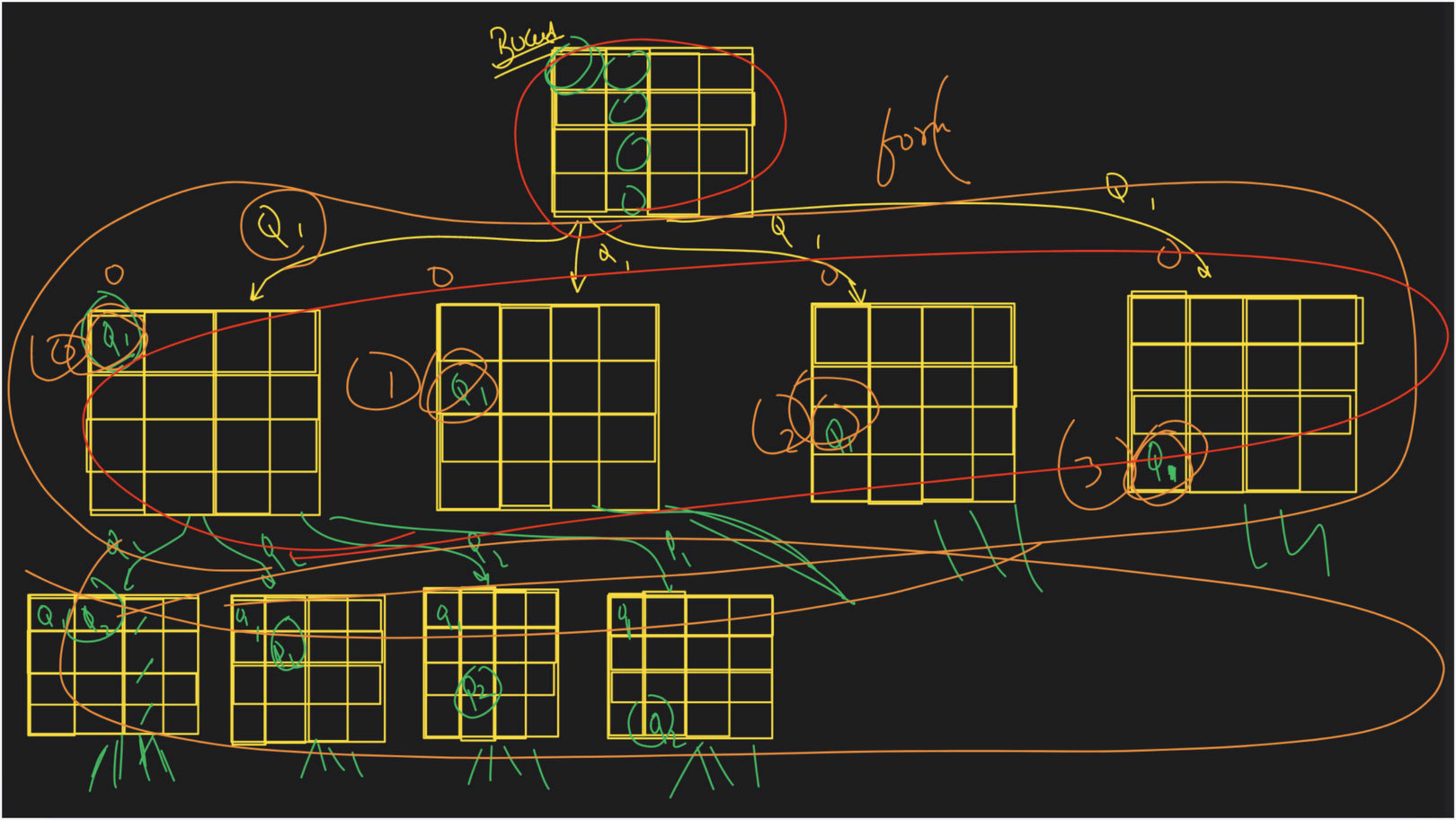
3 x 3

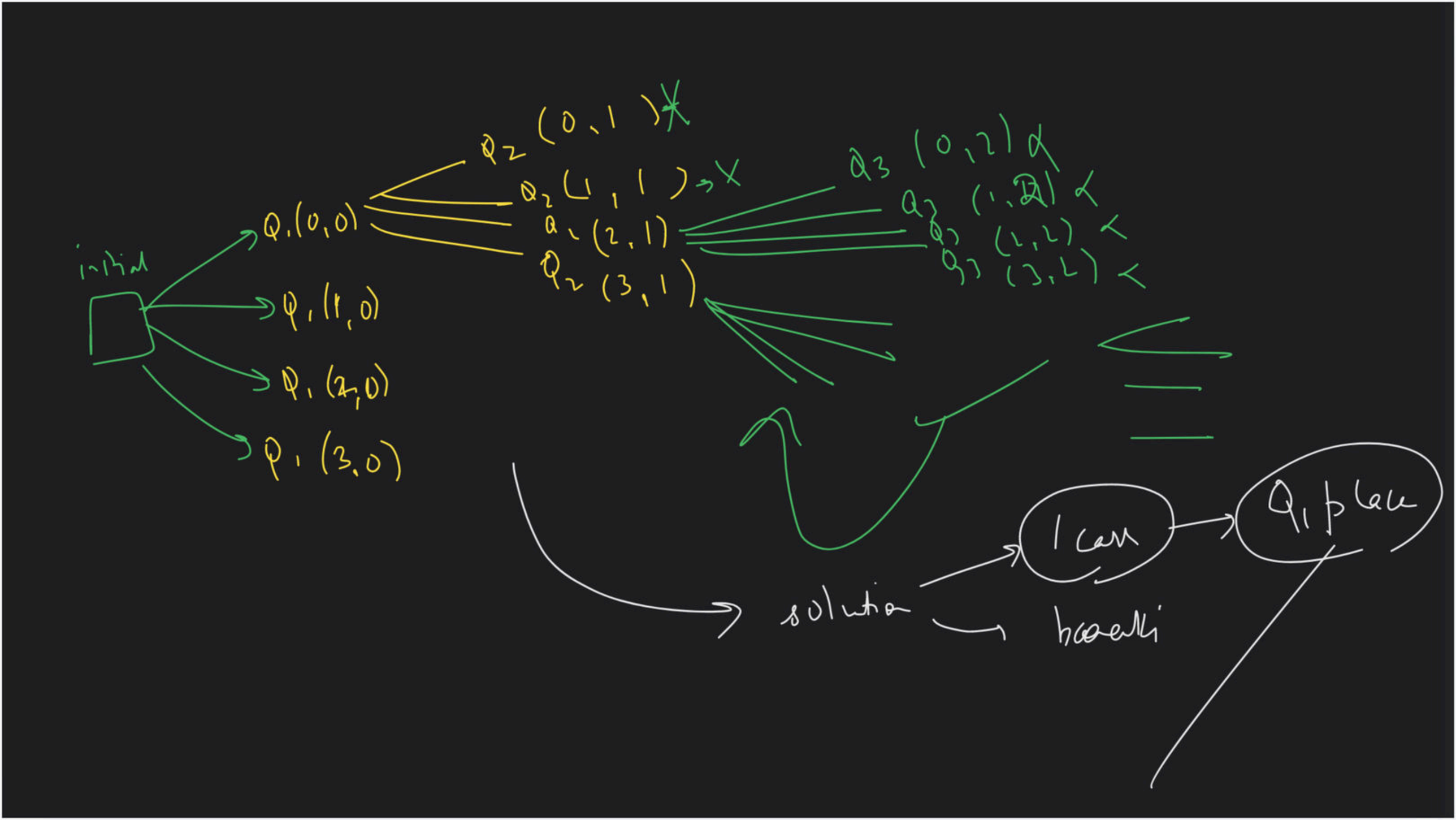
no solution

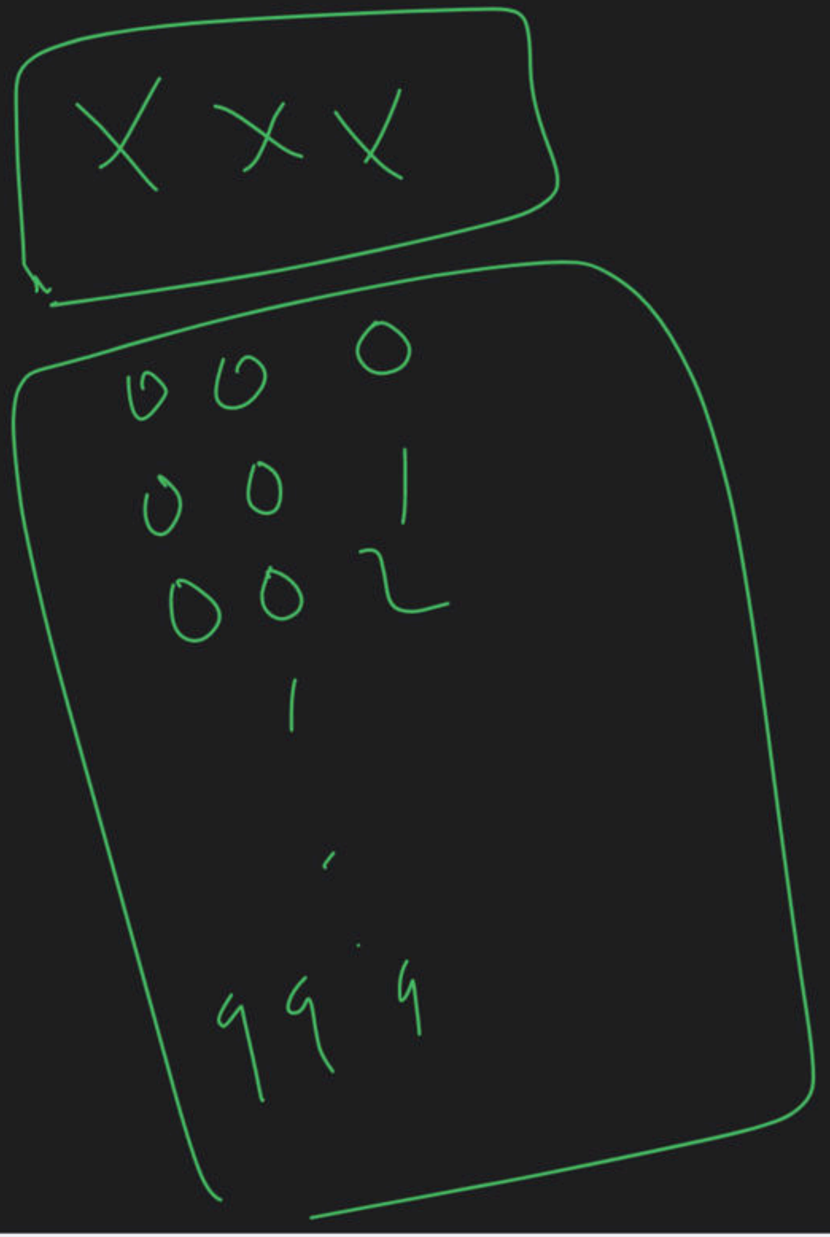

Q1
Q3
Q2
not possible
Q3
Q1
Q4
Q2
1 solution
Q2
Q4
Q1
Q3
1 solution
Q2
Q4
no solution

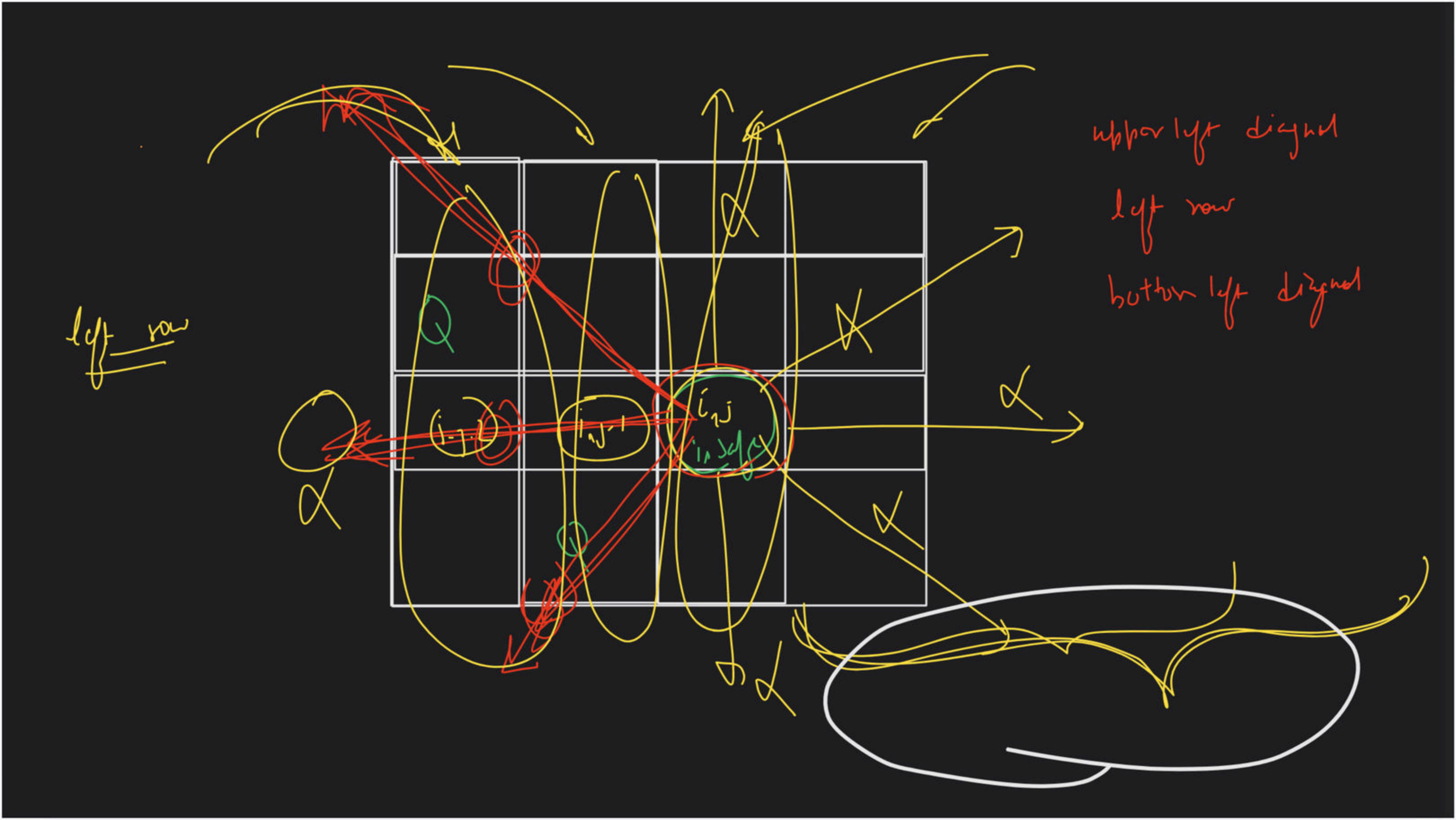

board
0
1
2
3
0
1
2
3
Q1
Q2
Q3

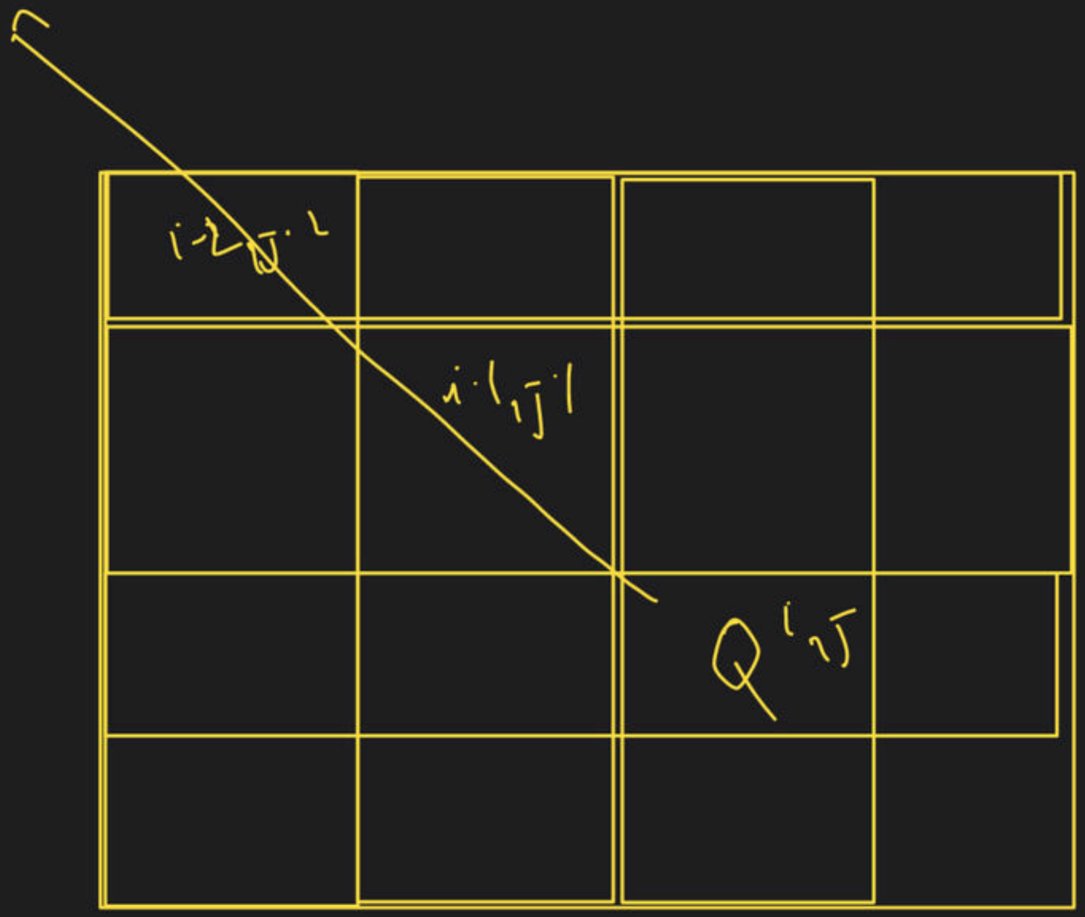


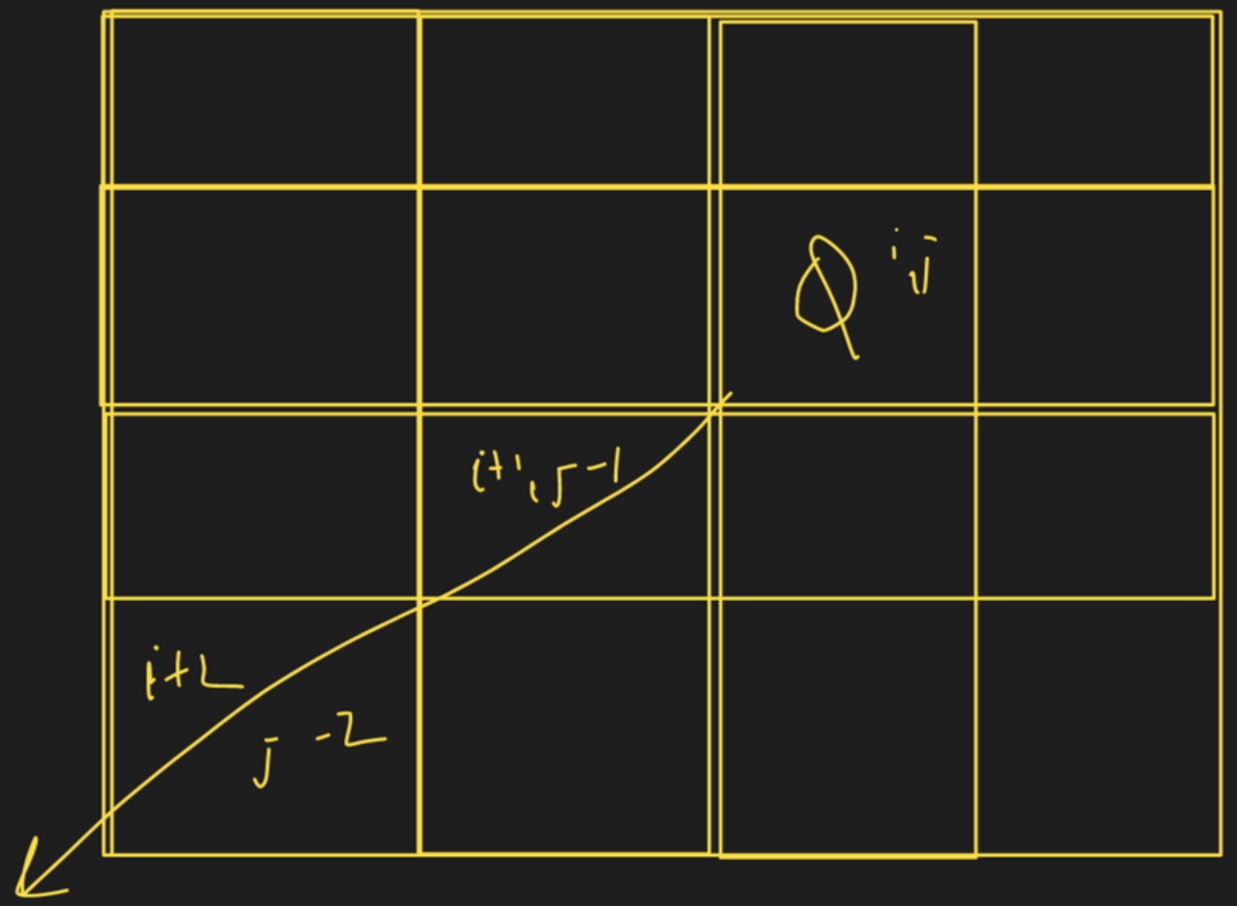

Backtracking
for

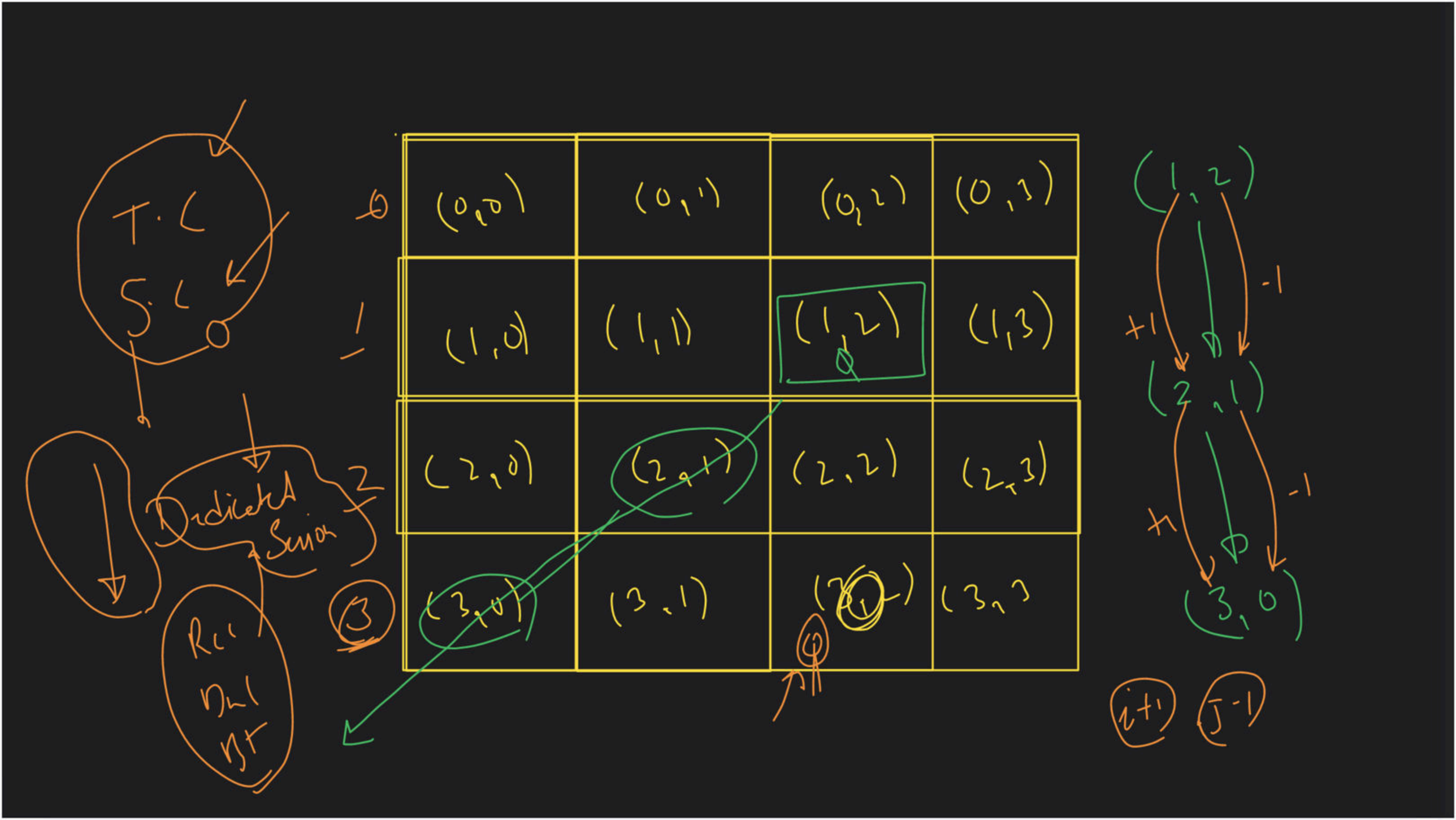

initial
Q1(0,0)
Q1(1,0)
Q1(2,0)
Q1(3,0)
Q2(0,1)
Q2(1,1)
Q2(2,1)
Q2(3,1)
Q3(0,2)
Q3(1,2)
Q3(2,2)
Q3(3,2)
solution
I can
Q1 place

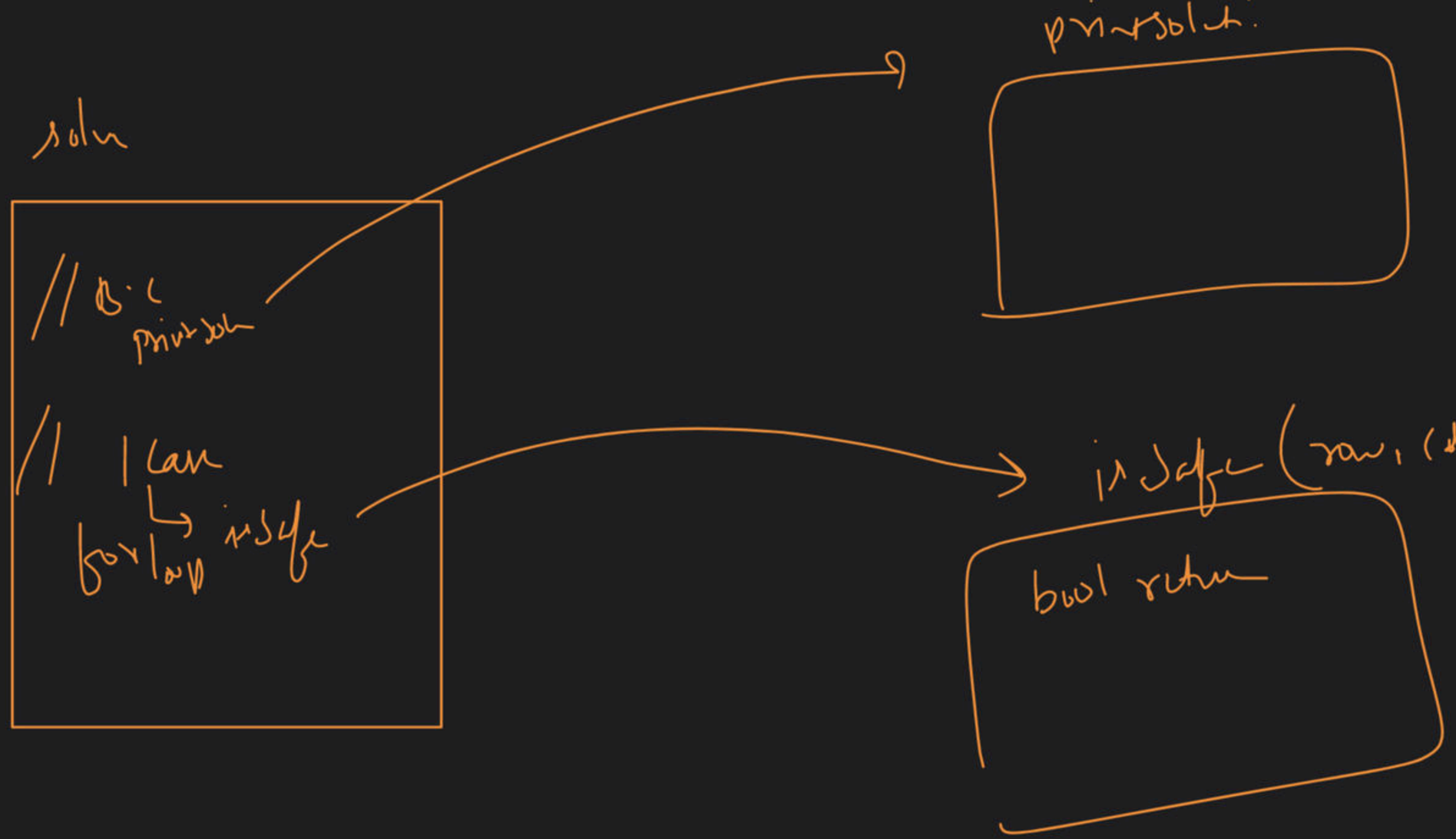

XXX
000
001
002
.
.
.
999

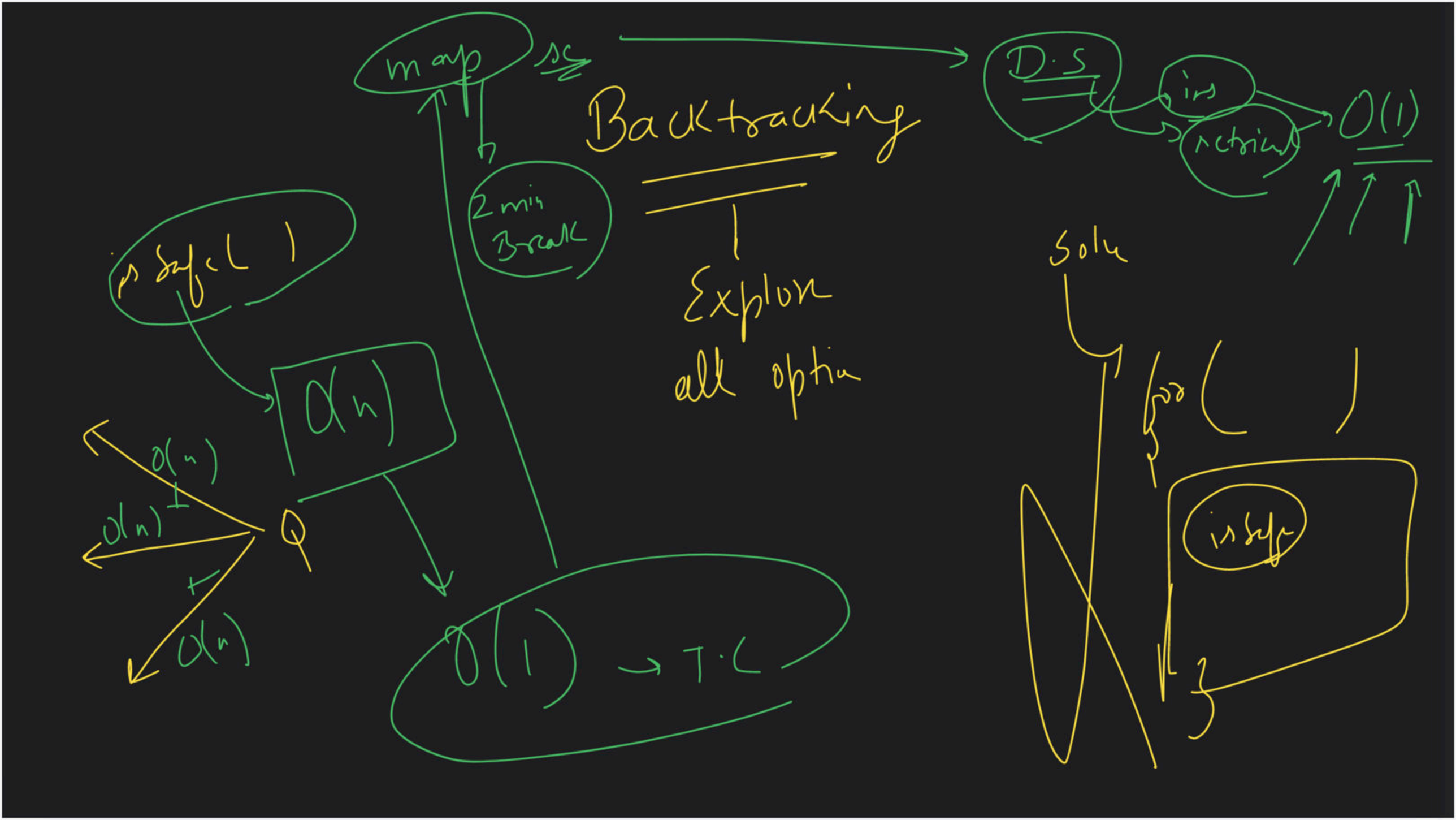

upper left digonal
left row
button left digonal
left row

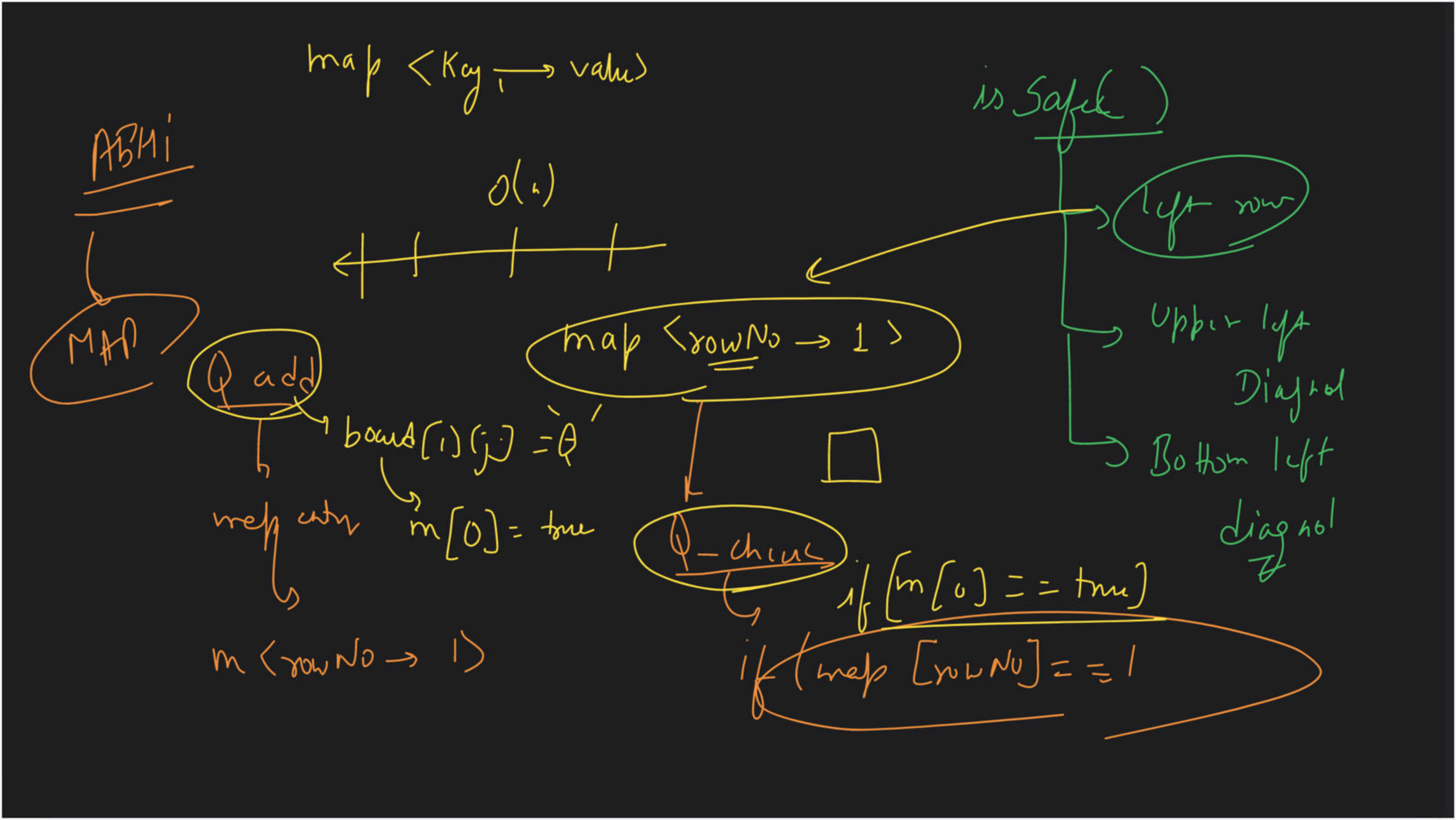


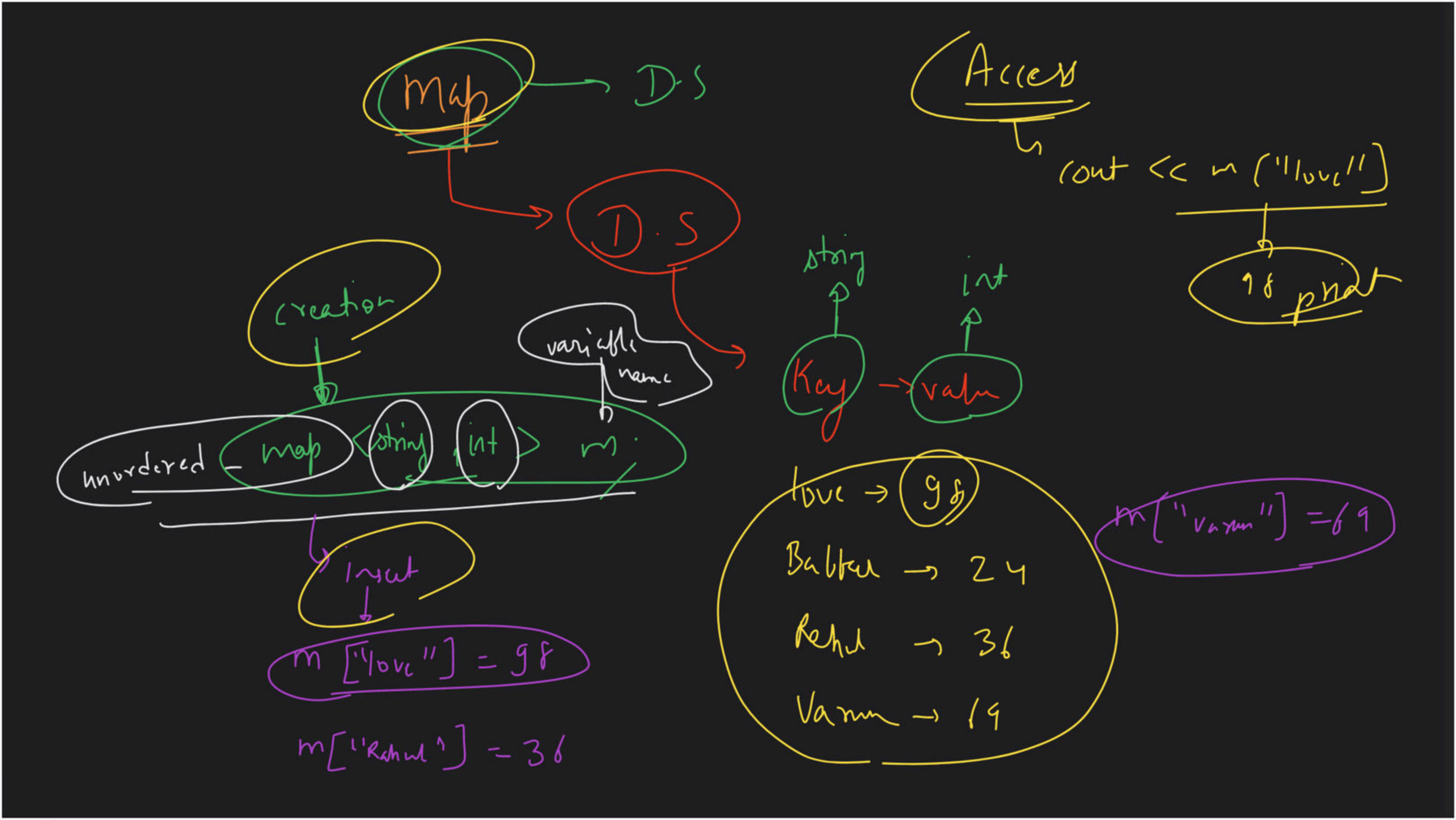

i, j
(i+1, j-1
i+2
j-2


T.C
S.C
Dedicated Server
R.C
Mail
Int
0
1
2
3
(0,0) (0,1) (0,2) (0,3)
(1,0) (1,1) (1,2) (1,3)
(2,0) (2,1) (2,2) (2,3)
(3,0) (3,1) (3,2) (3,3)
(1,2)
+1
-1
(2,1)
+1
-1
(3,0)
i+1
j-1


solve
// 1. Call
printSol
// 2. Call
boolean isSafe
printSol:
isSafe(row, col)
bool return


map
sc
2 min
Backtracking
Explore
all option
D.S
ins
retrieve
O(1)
is safe(
O(n)
O(n)
O(n)
O(n)
Q
O(1) → T.C
Solu
for(
is safe


map <Key → value>
Abhi
O(n)
isSafe()
MAP
add
map <rowNo → 1>
Upper left Diagnol
Bottom left diagnol
boundA[i][j] = 0
m[0] = true
helper call
m <rowNo → 1>
check
if [m[0] == true]
if (map [rowNo] == 1


Map
D.S
D.S
Access
cout << m["love"]
gf print
creation
variable name
unordered map <string, int> m;
string
int
Key
value
insert
m["love"] = gf
m["Rahul"] = 36
love → gf
Babbar → 24
Rahul → 36
Varun → 19
m["Varun"] = 69

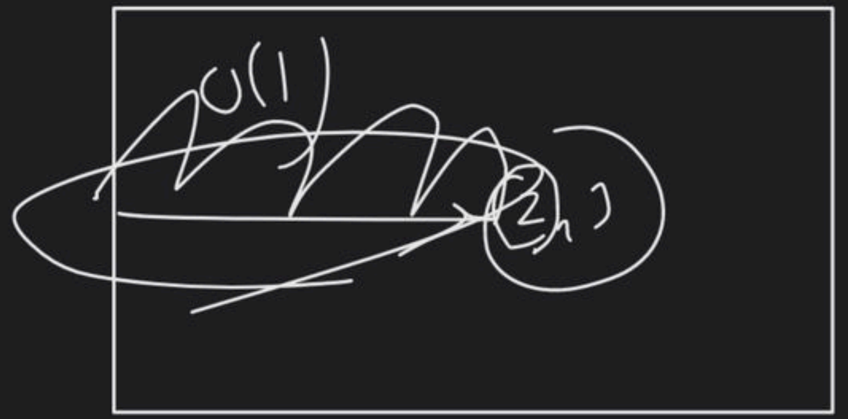
isSafe

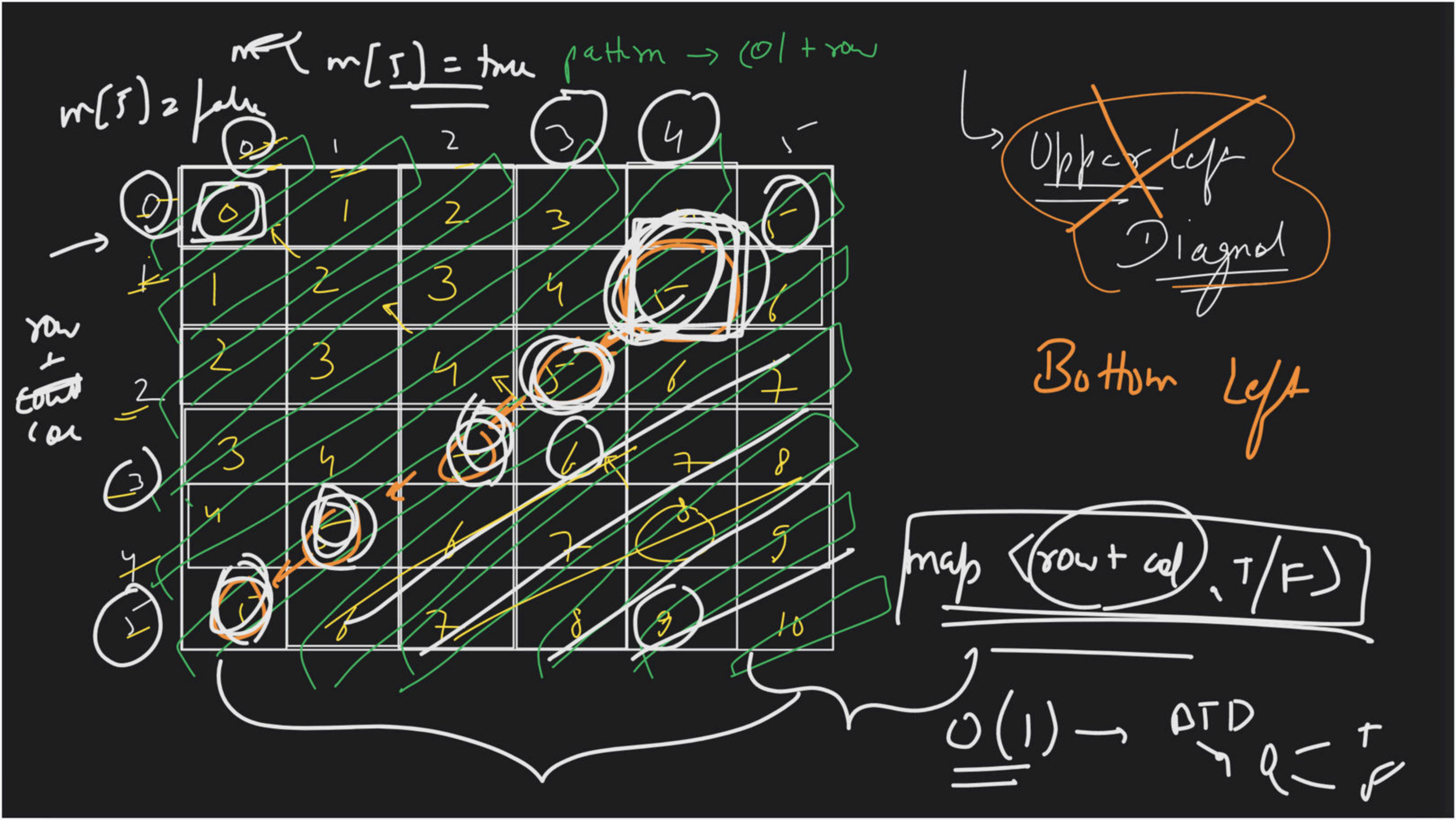
↳ leftRow

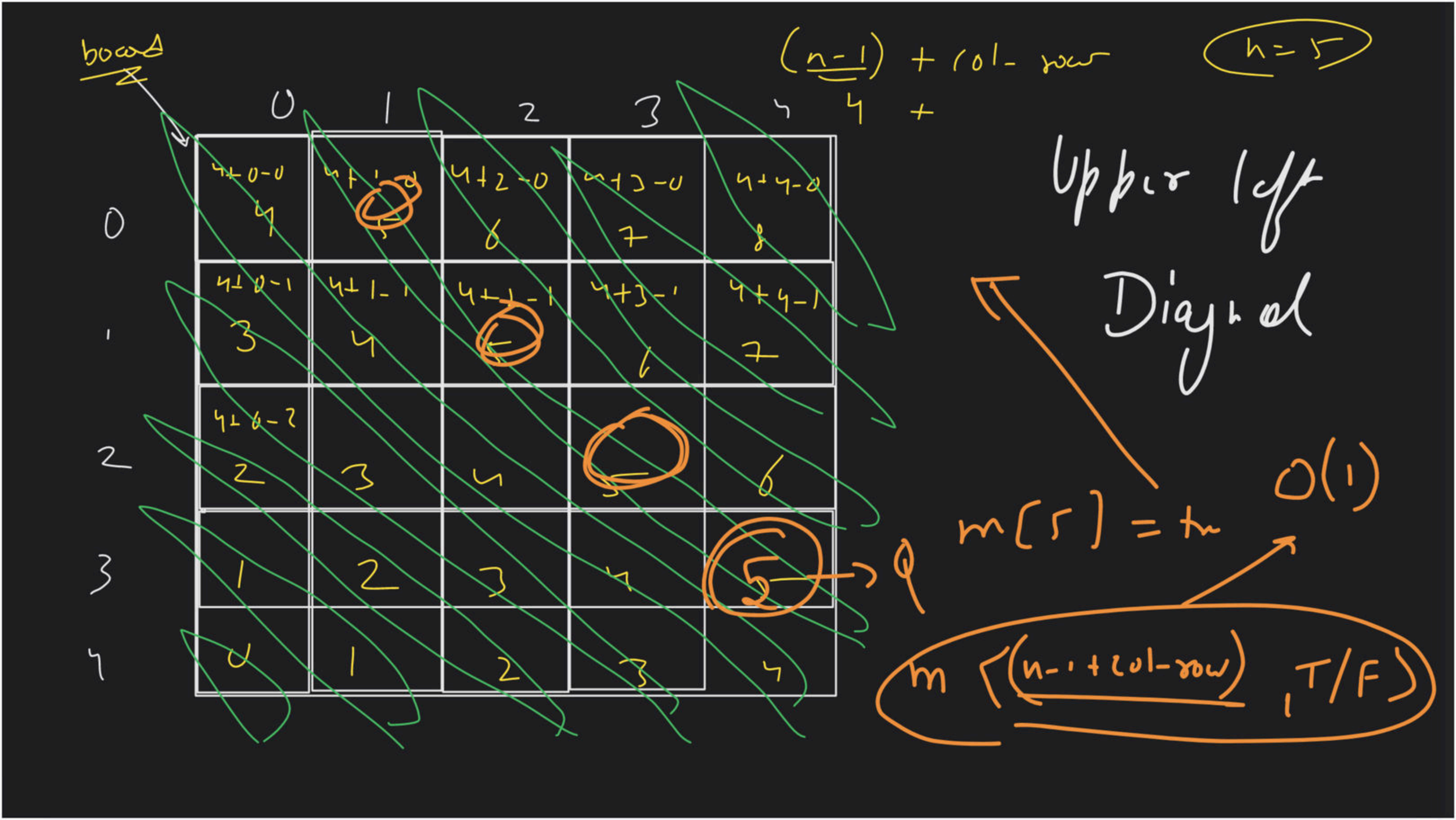
m ((RowNo → True/False)

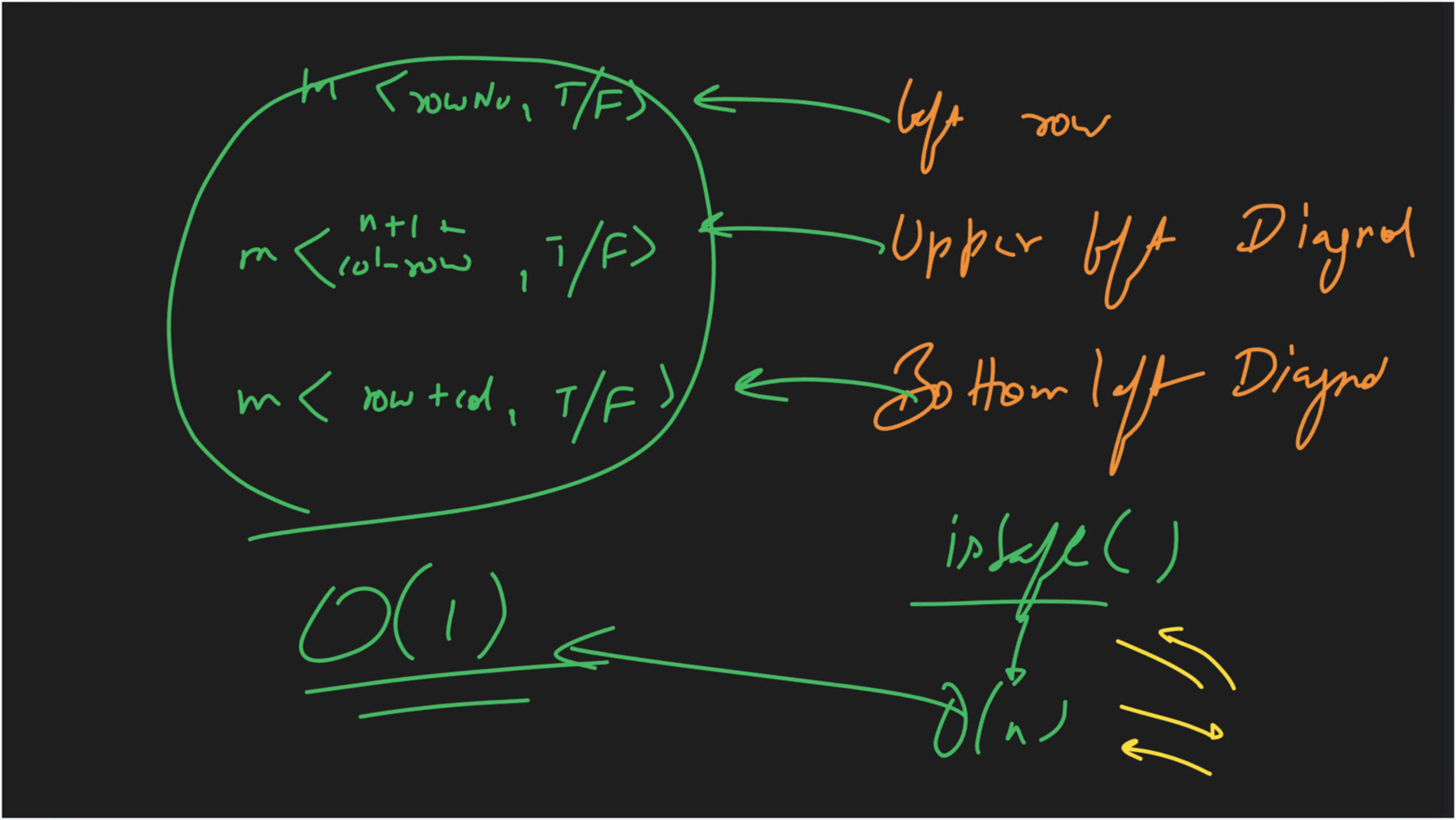
m [int, bool] → O(1)

if (m[2] == true) O(1)




m[i] = true
path → col + row
m[i] = false
Upper left
Diagnol
Bottom Left
row + col
map <row + col, T/F>
O(1) → ATD


board
(n-1) + col - row
n=5
Upper left
Diagonal
m[r] = true
O(1)
m[(n-1+col-row), T/F]


m < row Nu , T/F >
m < n+1 - row , T/F >
col
m < row + col , T/F >
left row
Upper left Diagnol
Bottom left Diagnol
O(1)
isLeaf()
O(n)