BT ∝

Rec.

DP ∝

Introduction to Backtracking Technique

Special class

Vishal Pachauri • Oct 1, 2022

→ What is BackTracking?

20-25

26

30+k

4-5

4.5 m-x

web dev

Course

→ 4-10

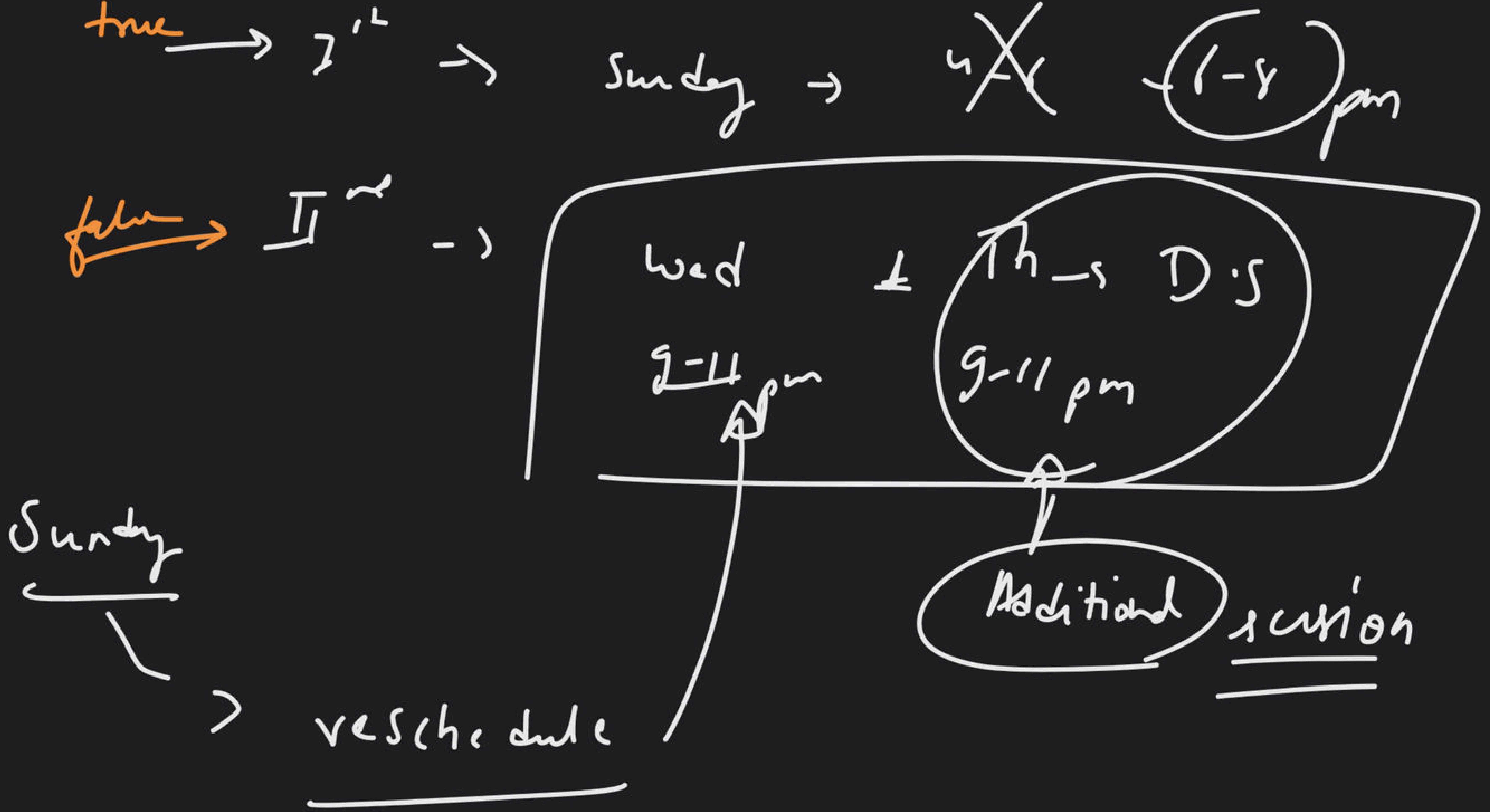
→ 8-9 pm

or

→ Wed + Thu

8.11

0.5 9-11



true → Ist → Sunday → ~~4-6~~ 1-4 pm

false → IInd → Wed & Thus D.S

9-11 pm

9-11 pm

Additional session

Sunday → reschedule

→ what is BT ?

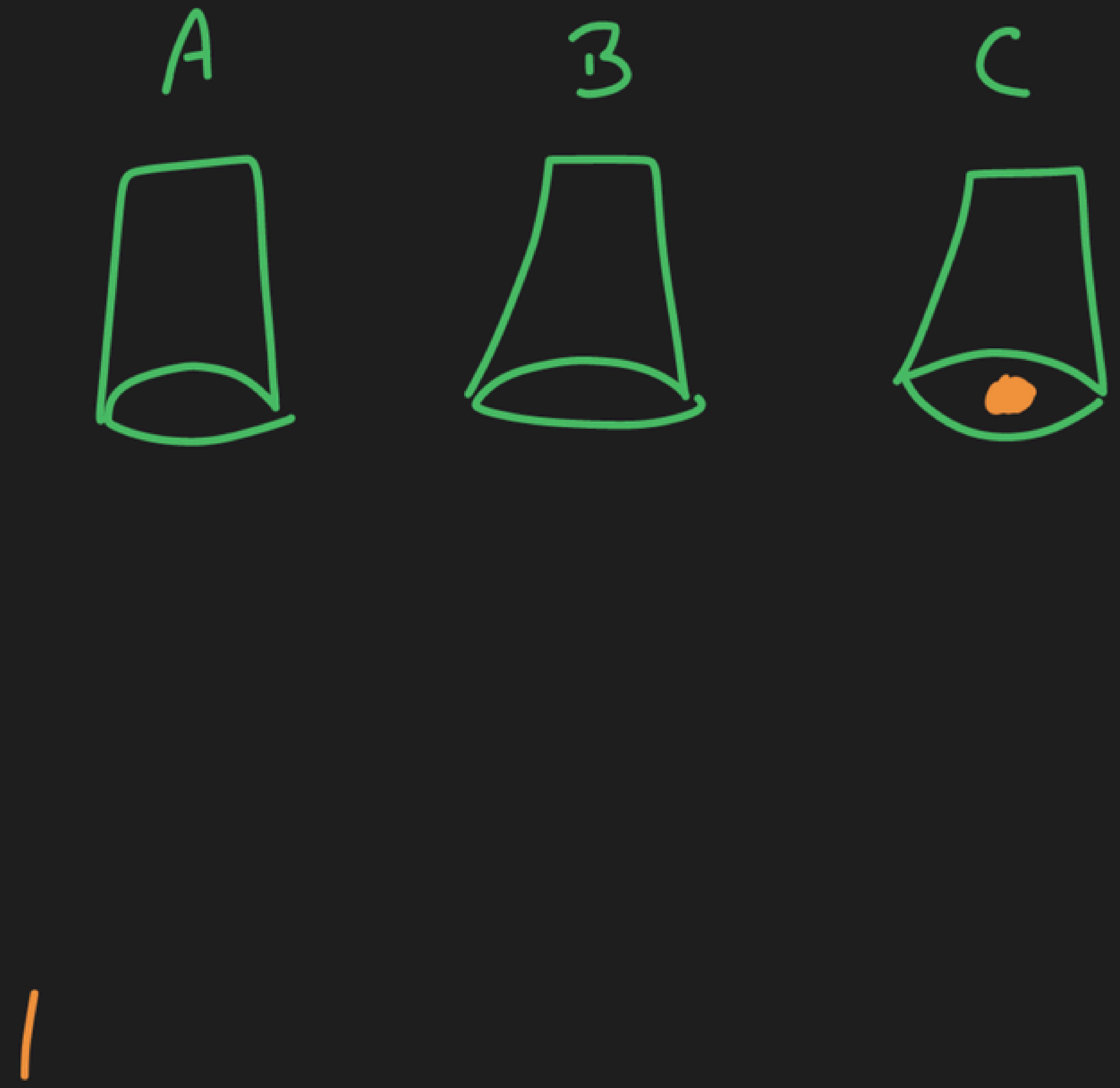
find sone ka sikka?

✓ A → X

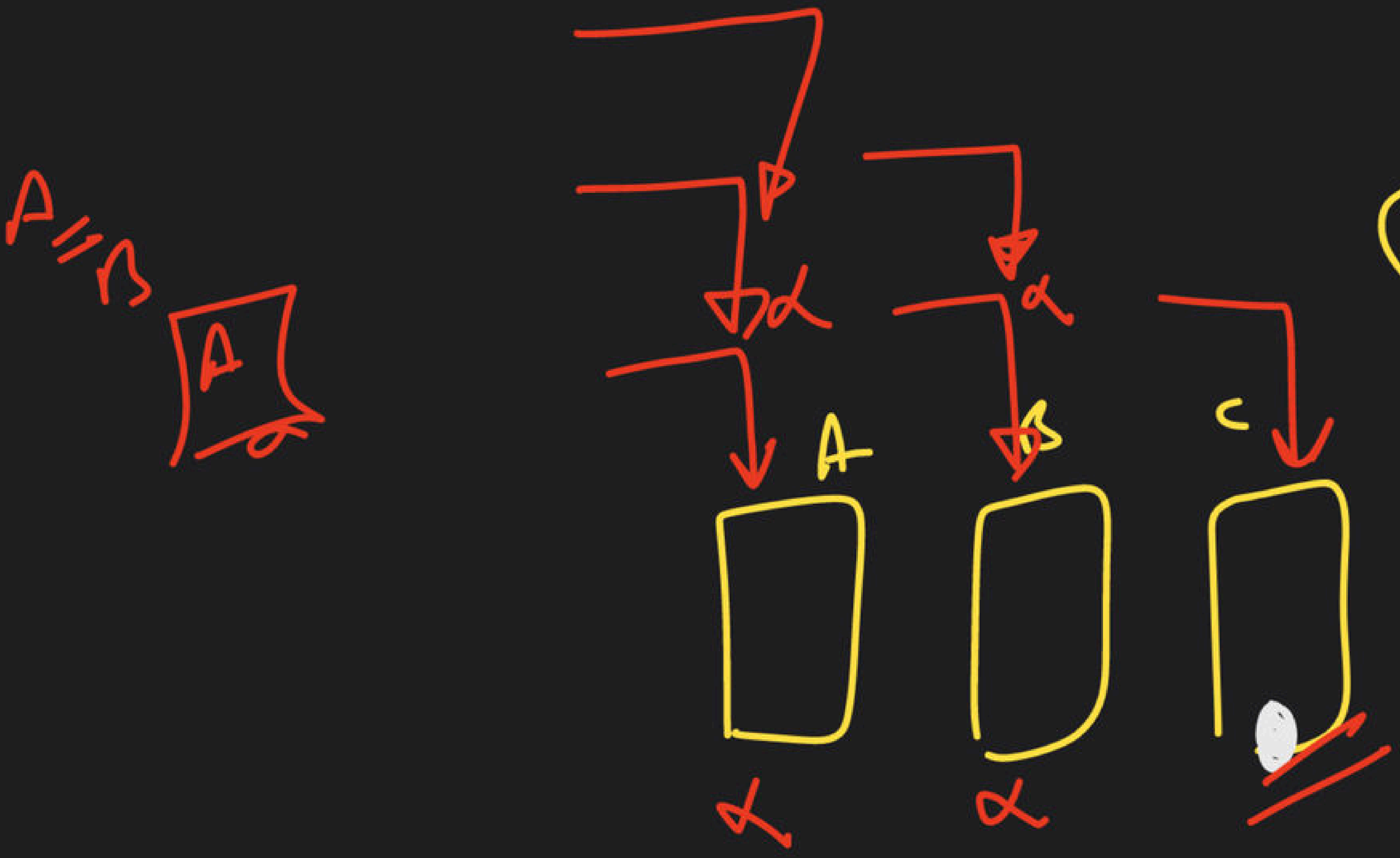
✓ B → x

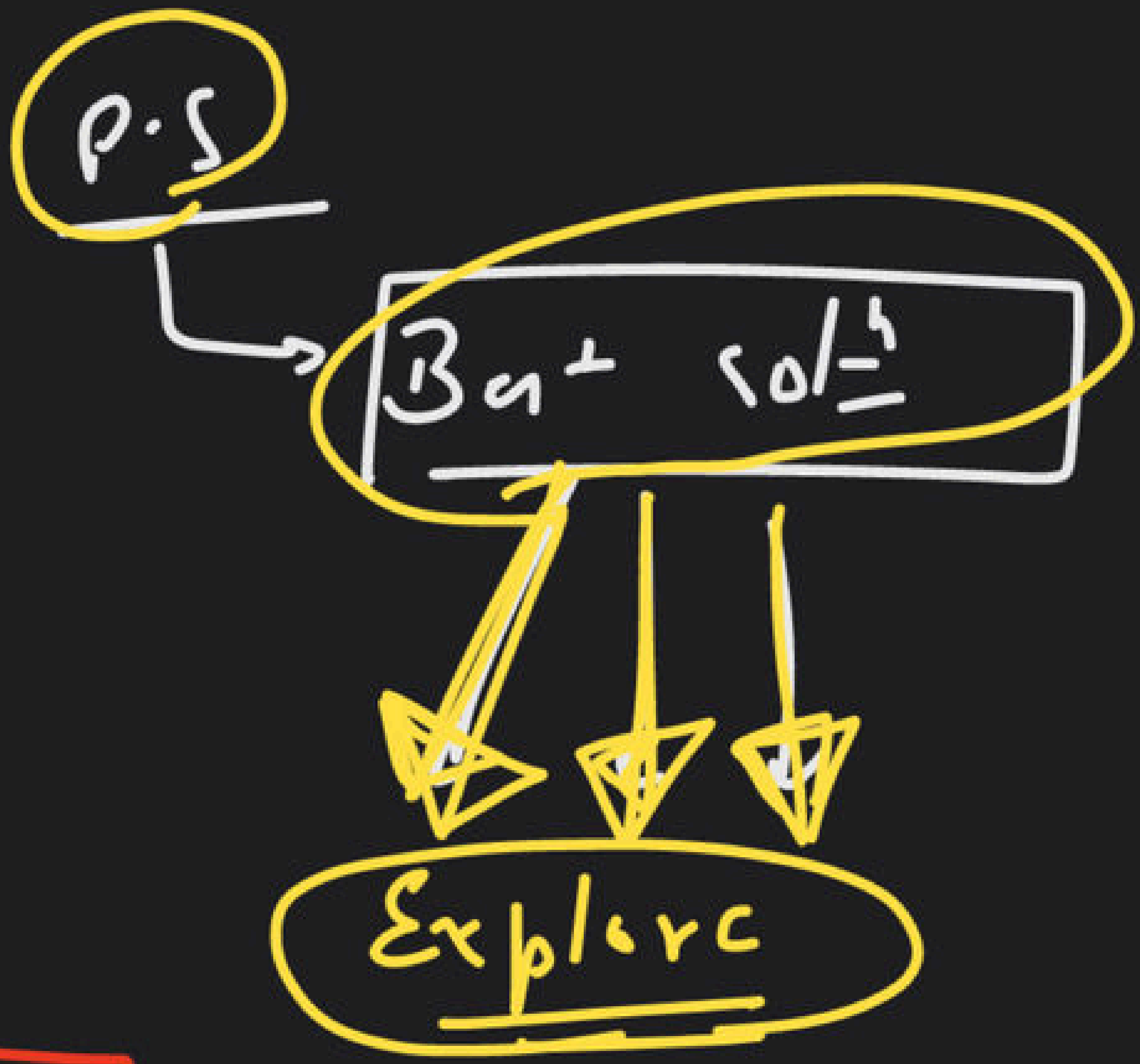
✓ C → ✓

2 min → join again guys


A
B
C

→ Back·Tracking





Permutations of String

Ques-1

i/p → "abc"

0 1 2

$3! = \boxed{6}$

o/p →

a c b

a b c

b a c

b c a

c a b

c b a

6 p

no of permutatio $= \boxed{n!}$

i/p -> "ab"

2!

a b

b a

2 p~

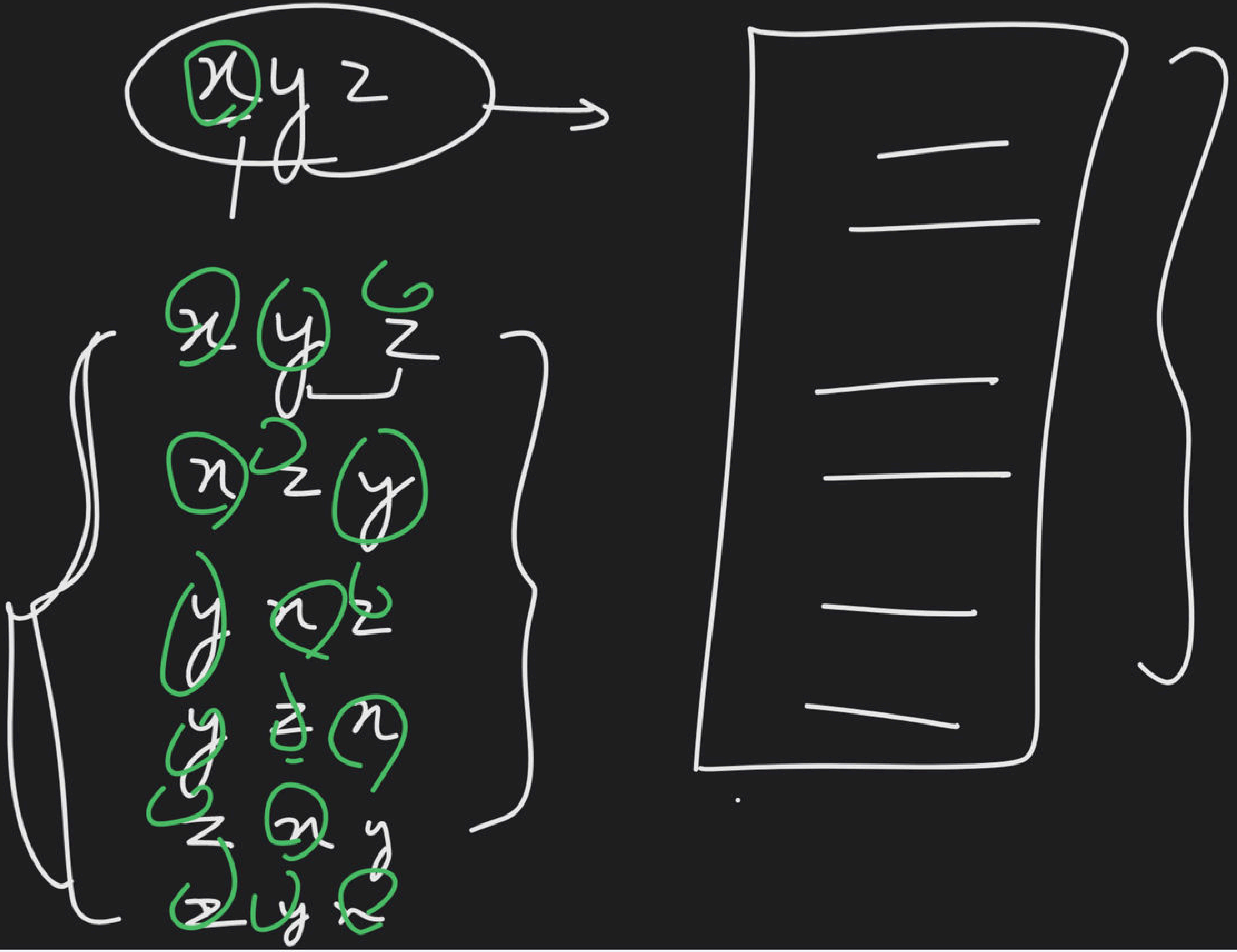
i/p -> abcd

$4! = \boxed{24}$

i/p -> abcd

abcd	bacd	c —	d —
abdc	badc		
acbd	bcad	—	—
acdb	bcdc	—	
adbc	bdac		— —
adcb	bdca		—

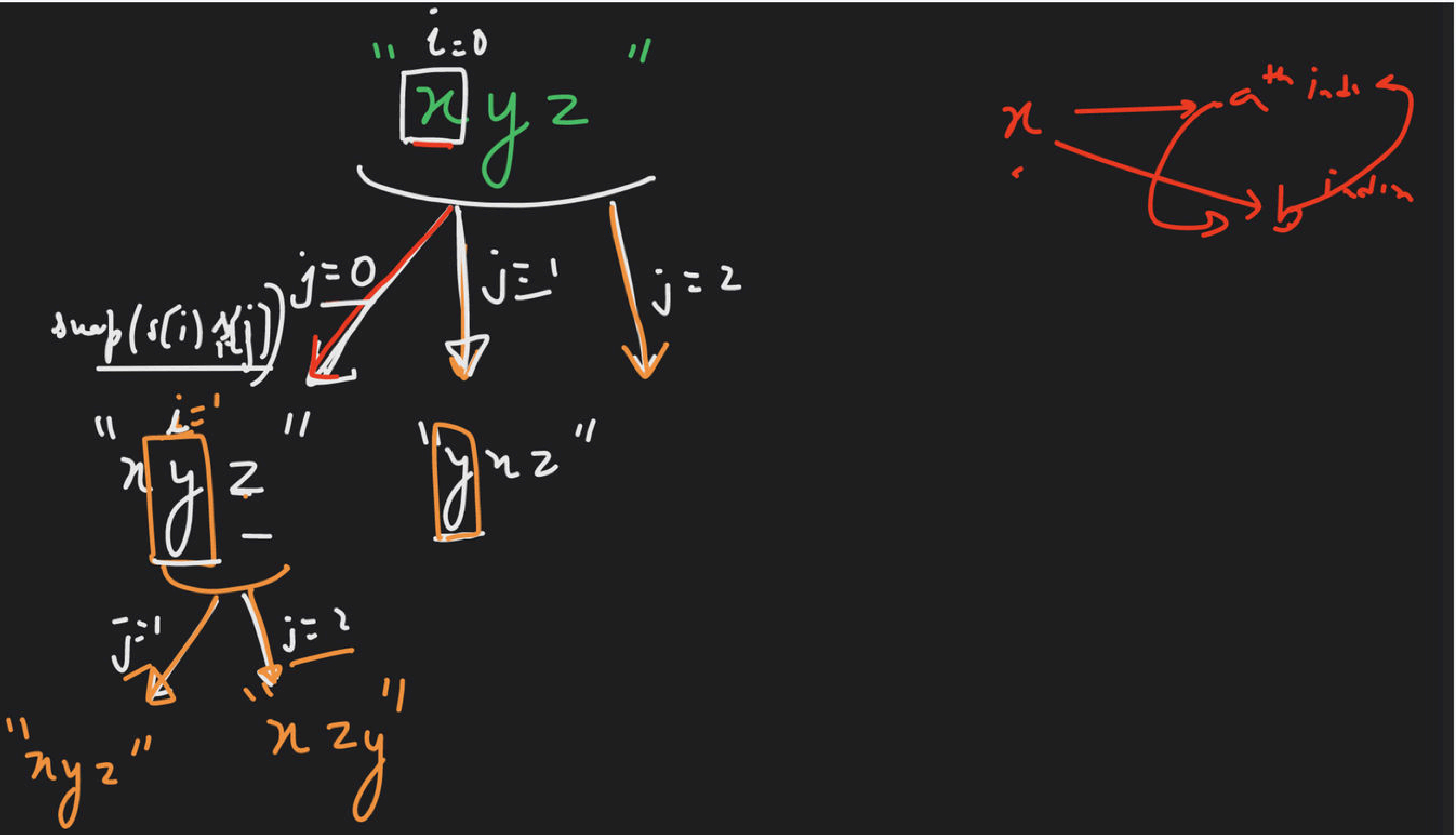
24 p —



2 min

swap

3! = 6 pcm

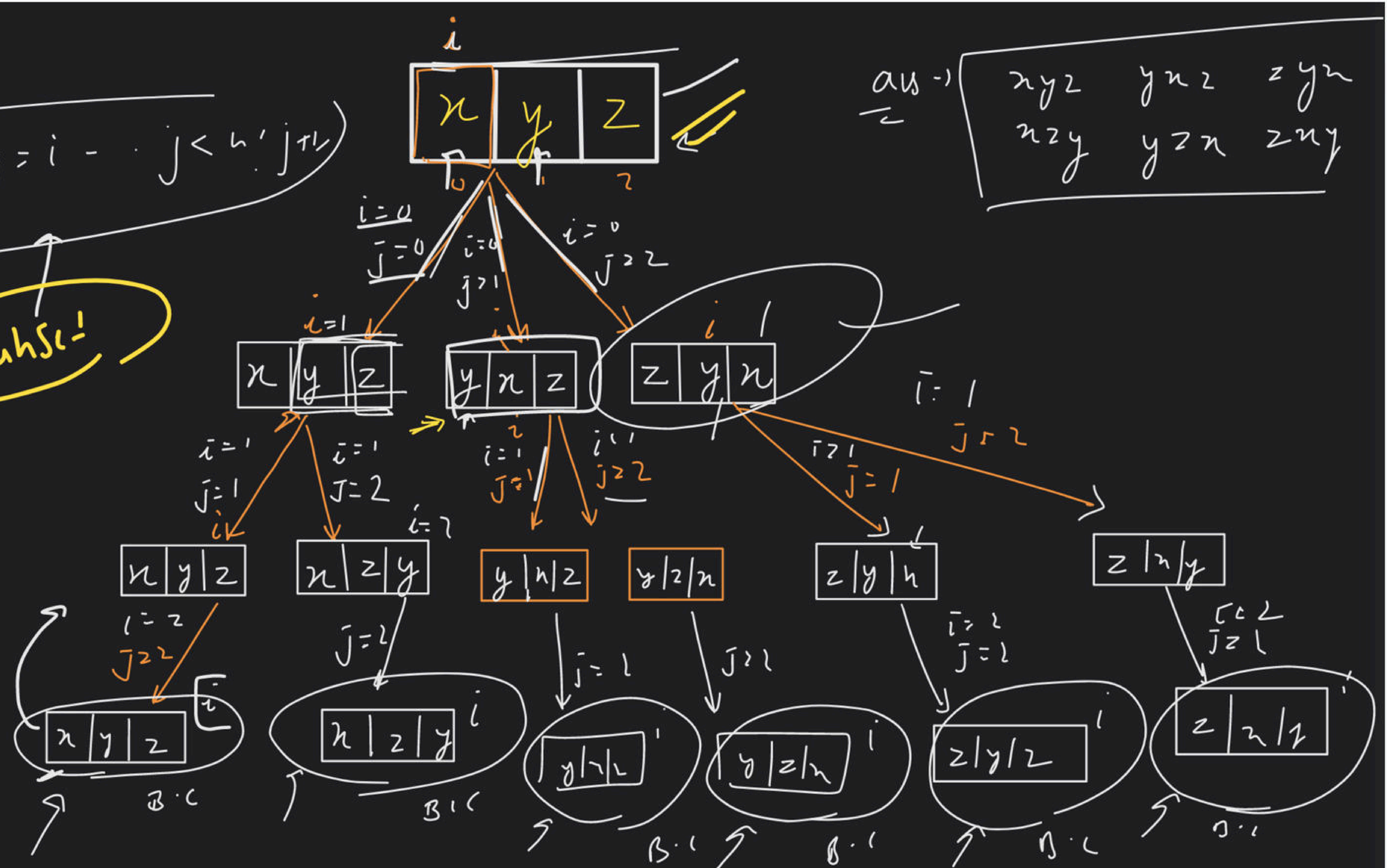

i=0
"xyz"
j=0
j=1
j=2
swap(s[i], s[j])
i=1
"xyz"
"yxz"
j=1
j=2
"xyz"
"xzy"
x
ath indx
bth indx

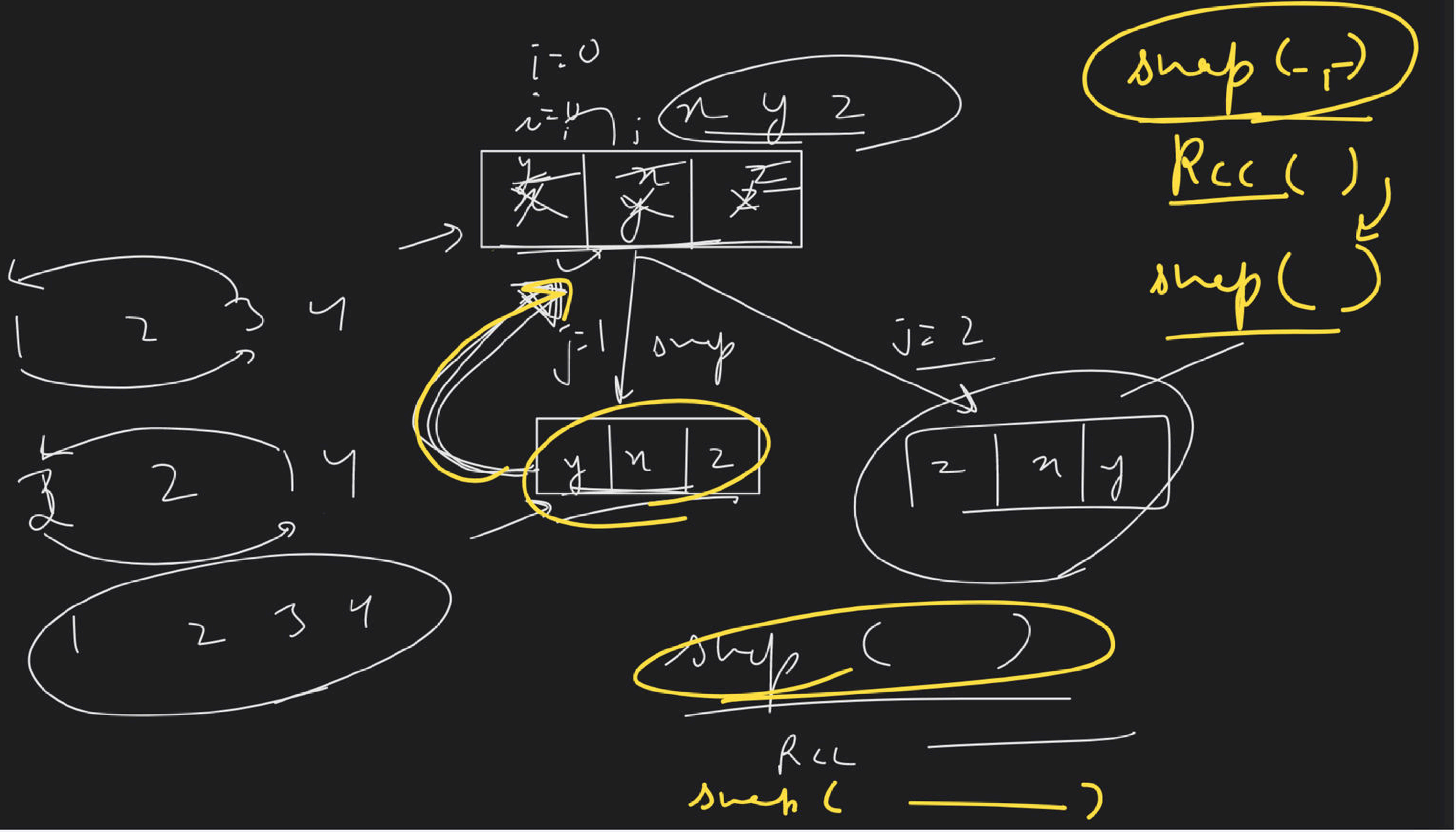
```
for ( int j = i - ; j < n ; j++
```

Hashset!

ans ->

xyz yxz zyx
xzy yzx zxy



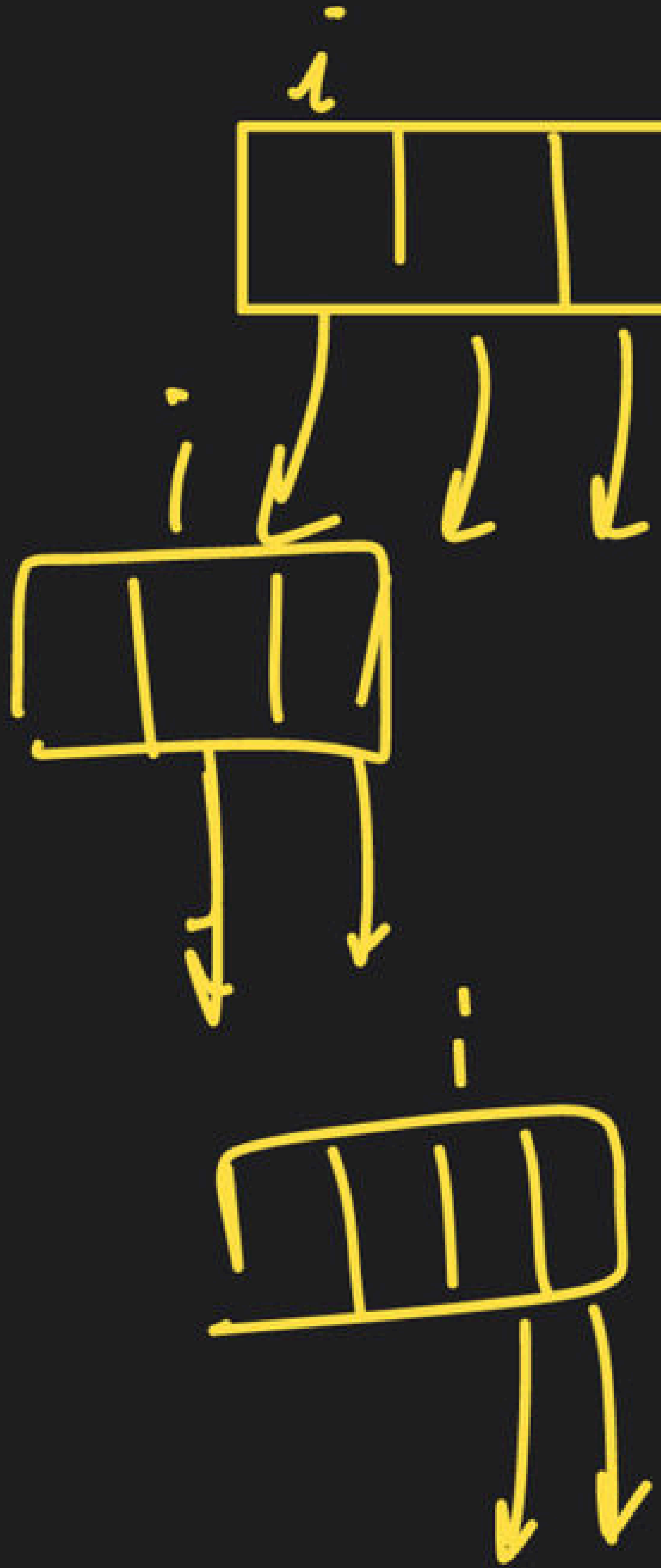

i=0
i=1
j
x y z
y x z
x y z
j=1
swap
y x z
j=2
z x y
1 2 3 4
3 2 1 4
1 2 3 4
swap (_,_)
Rec ( )
swap ( )
swap ( )
Rec
swap ( )

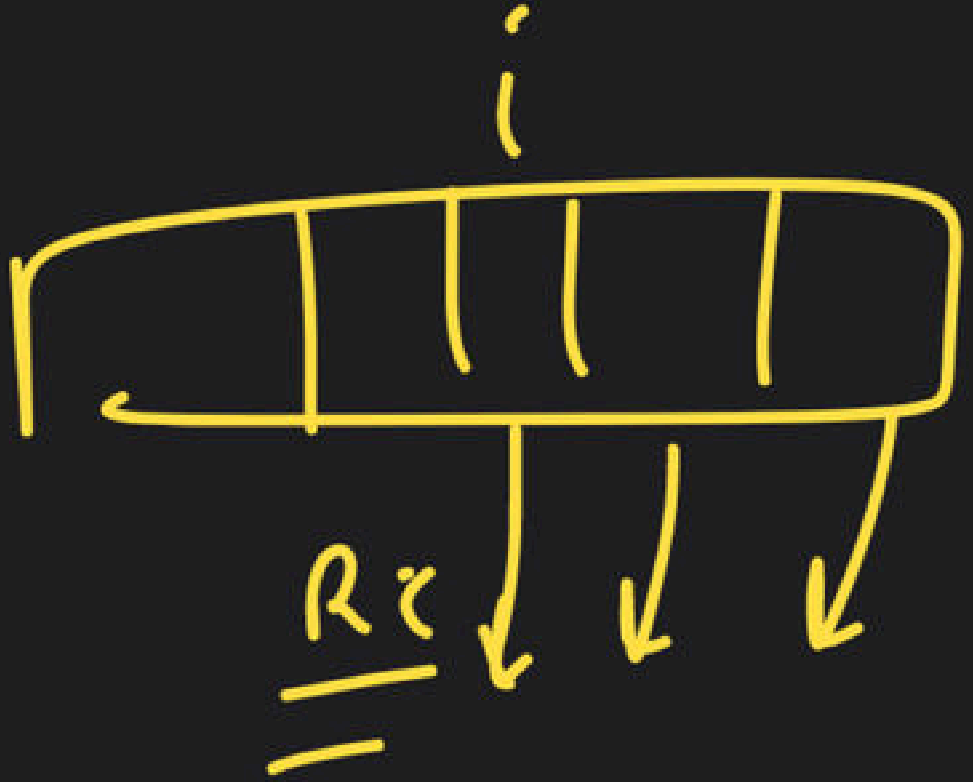
```
void solve( i , ____ , ____ )
{
        // Base Case

        if(i>= str.size())
        {      //ans store
               return;
        }
        for(int j=i ; j<n; j++)
        {      swap(str[i], str[j])
               solve (i+1 , _ , _ )
               swap(str[i] , str[j]);        BT
                                             B.T
        }
}
```

Tmin

↓

Java?





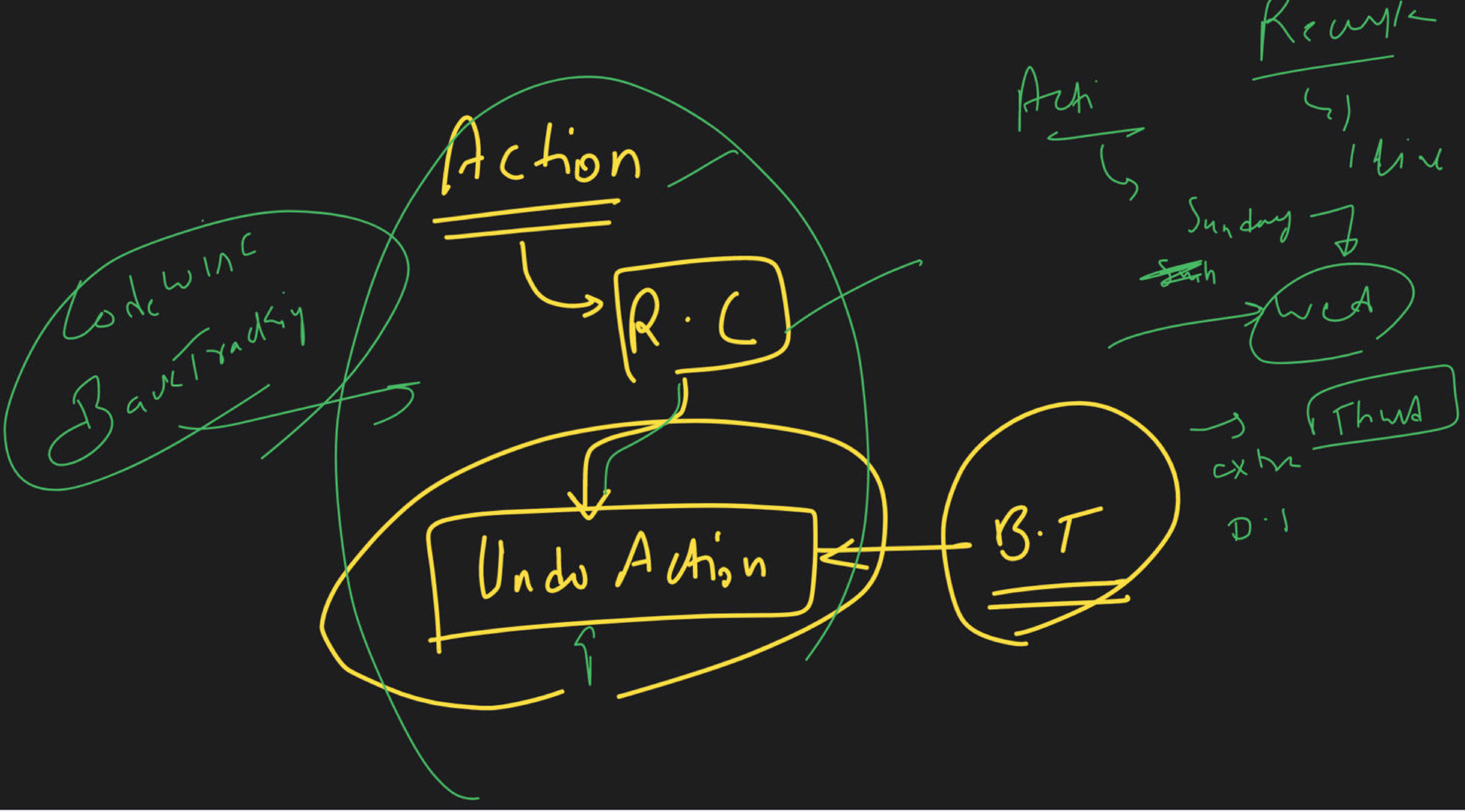
for(j=i; — <n)

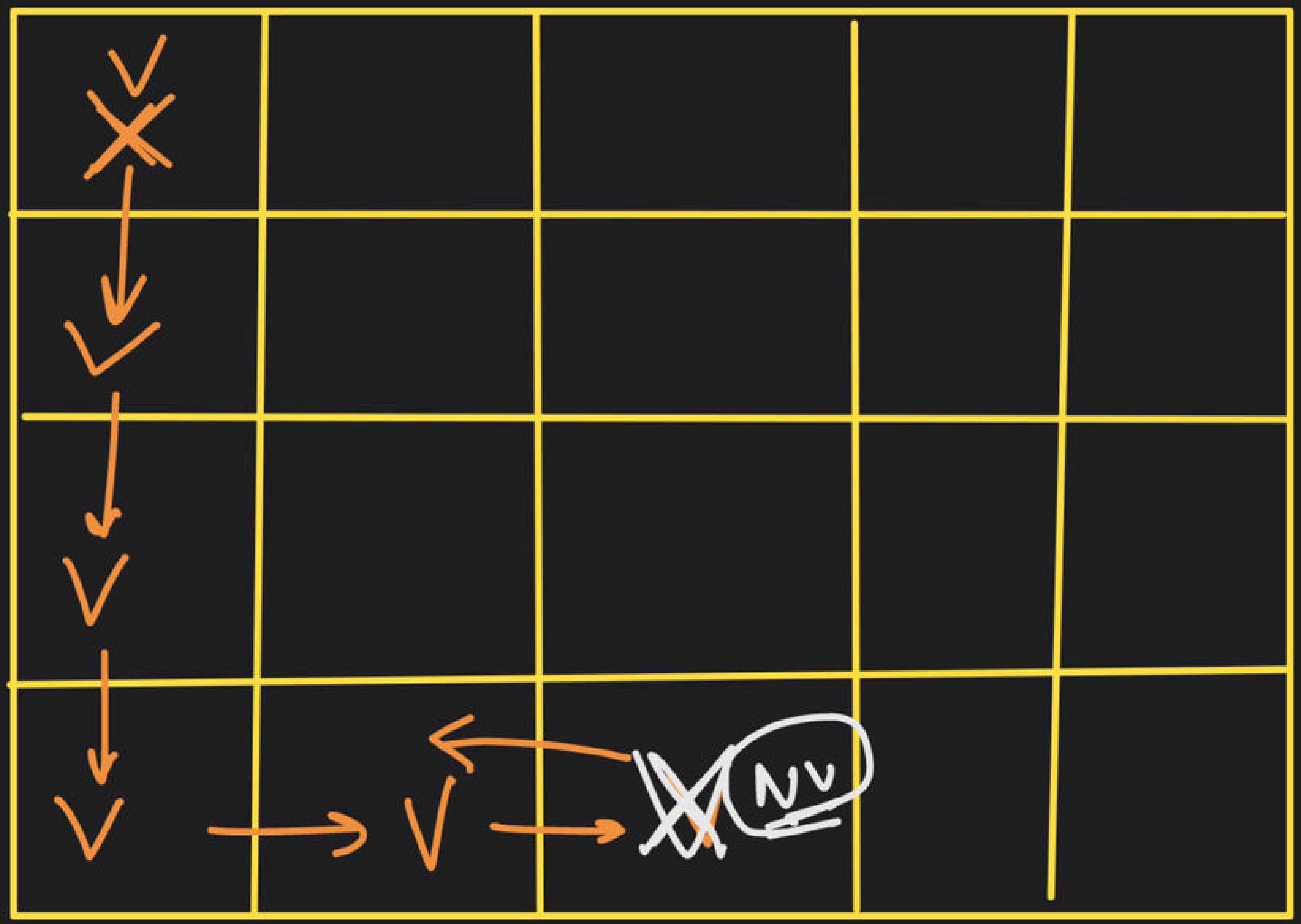
H/W

$\boxed{xzz} \longrightarrow$ permutation - ?

abcb →

abbb →


Codewinc
BackTrackiy
Action
R · C
Undo Action
B · T
Recursi
1 line
Acti
Sunday
~~Sun~~h
WCA
Third
extra
D · 1



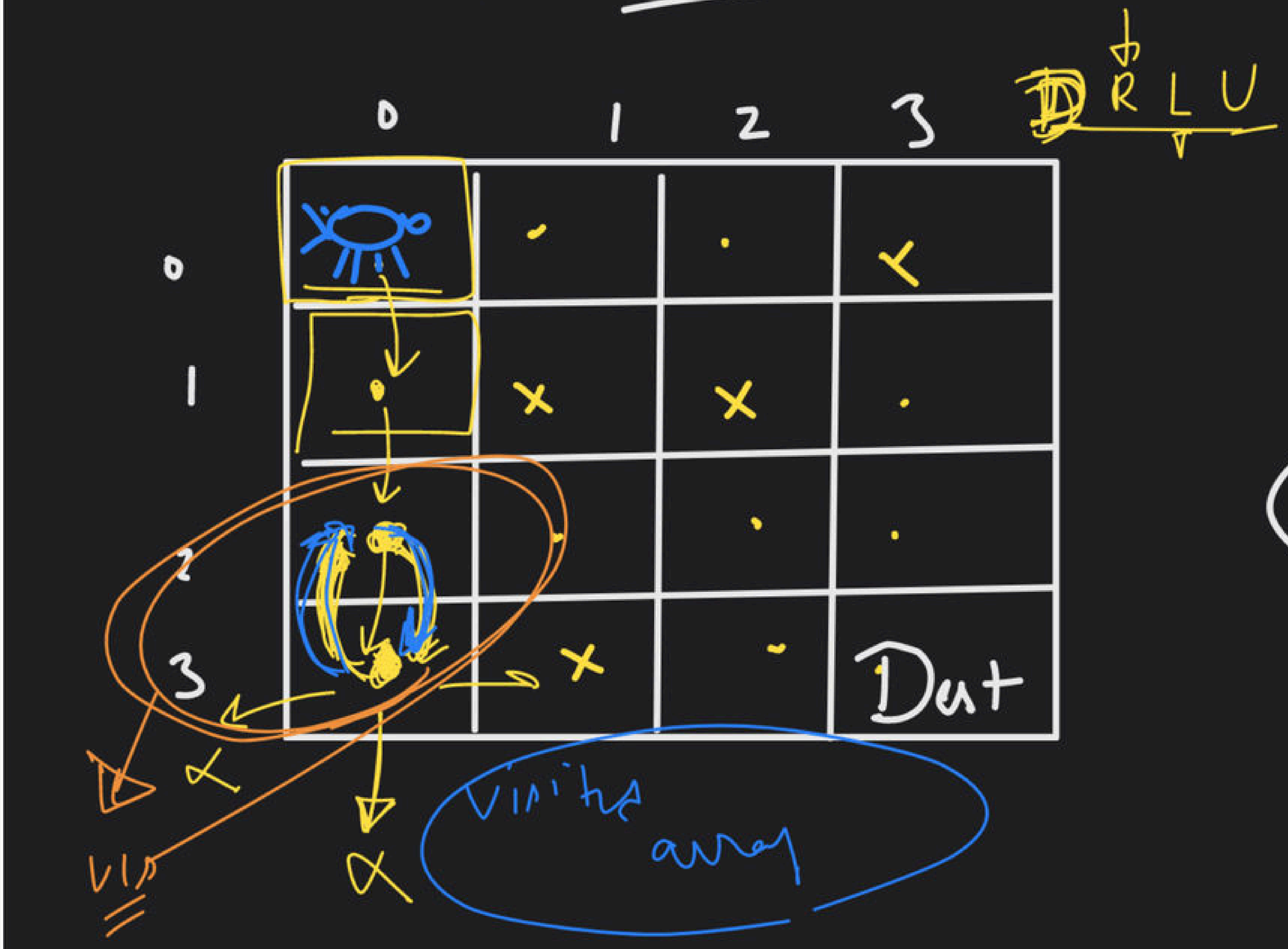
→ mark Vis

→ Rec

→ mark NV

B.T

→ Rat in a maze



	0	1	2	3
0		·	·	X
1	·	X	X	·
2			·	·
3		X	·	Dest

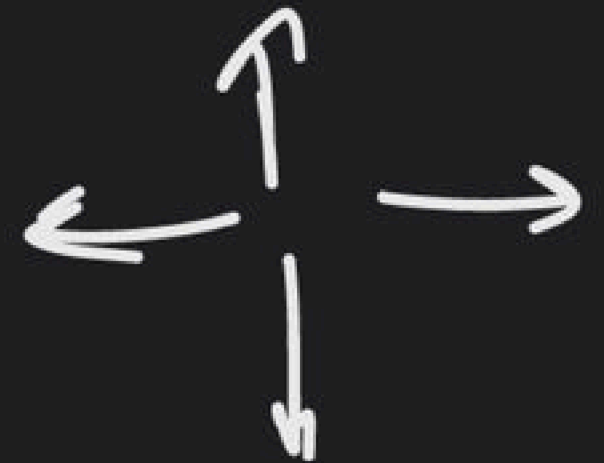
□ → open → 0 → ·

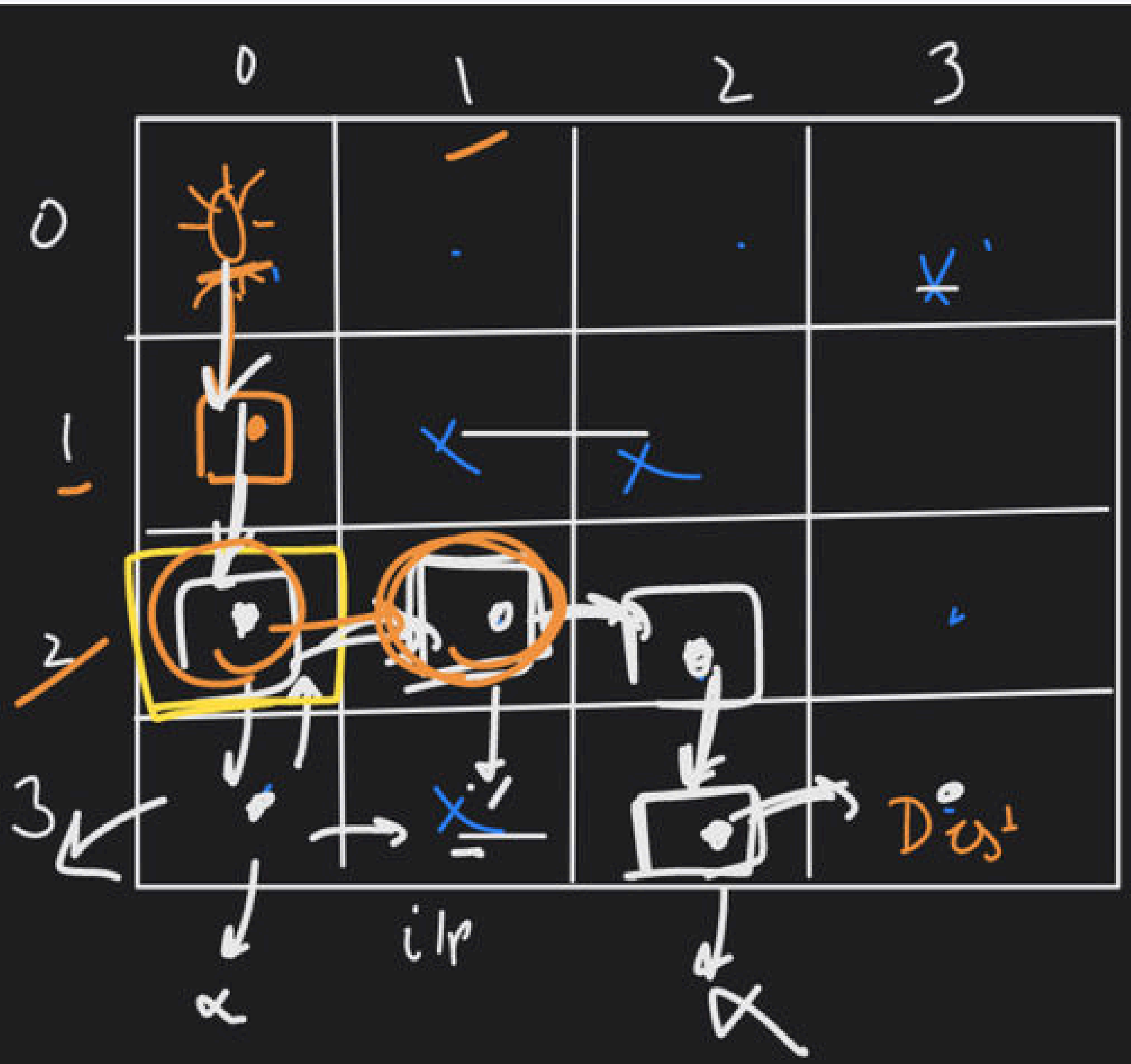
□ → clos'd → 1 → X

src -> (0, 0)

dest → (3, 3)

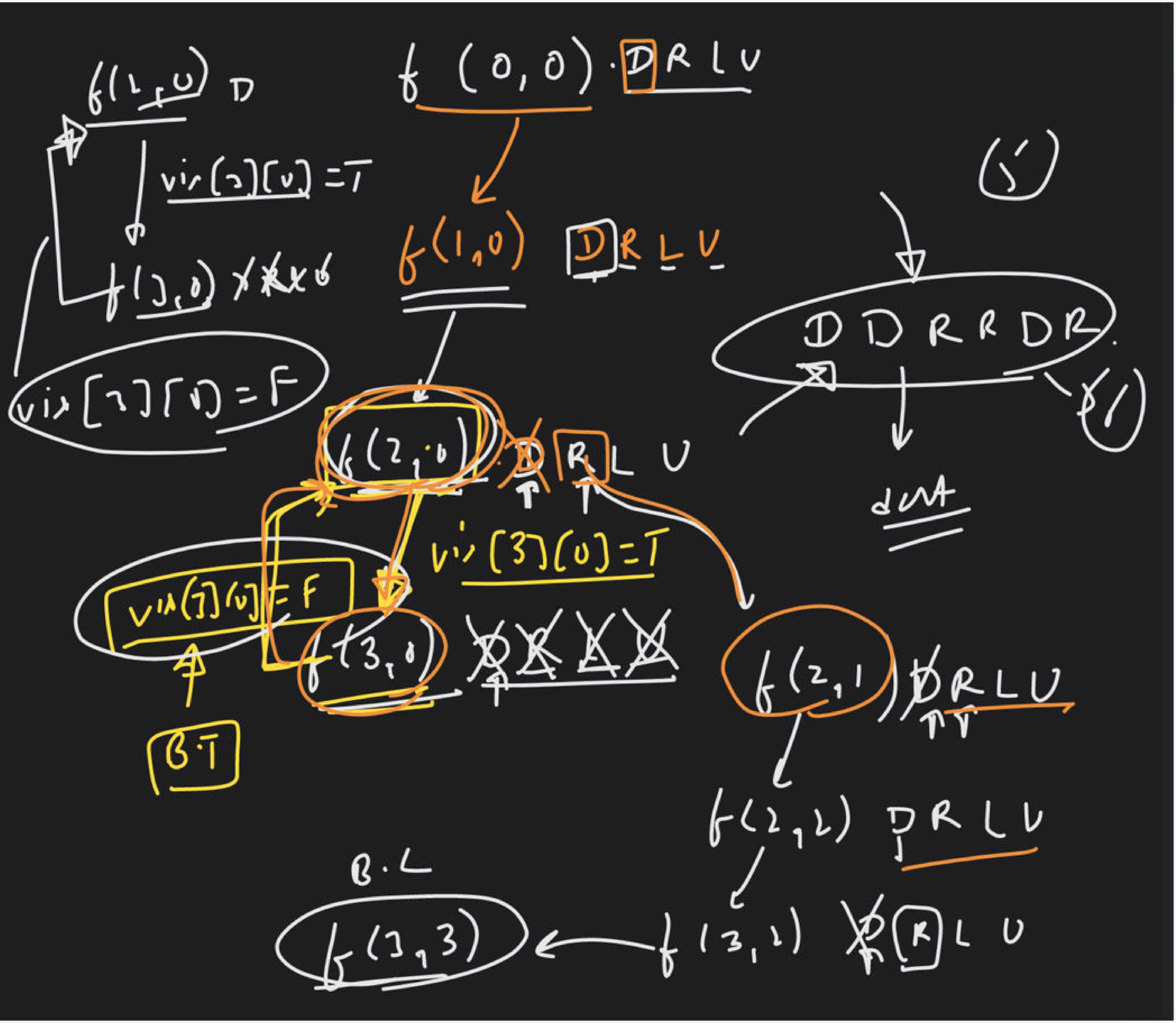
4 muvemat →





	0	1	2	3
0	T ~~F~~	F	F	F
1	T ~~F~~	F	F	F
2	T ~~F~~	T ~~F~~	T ~~F~~	F
3	F ~~F~~	F	T ~~F~~	F

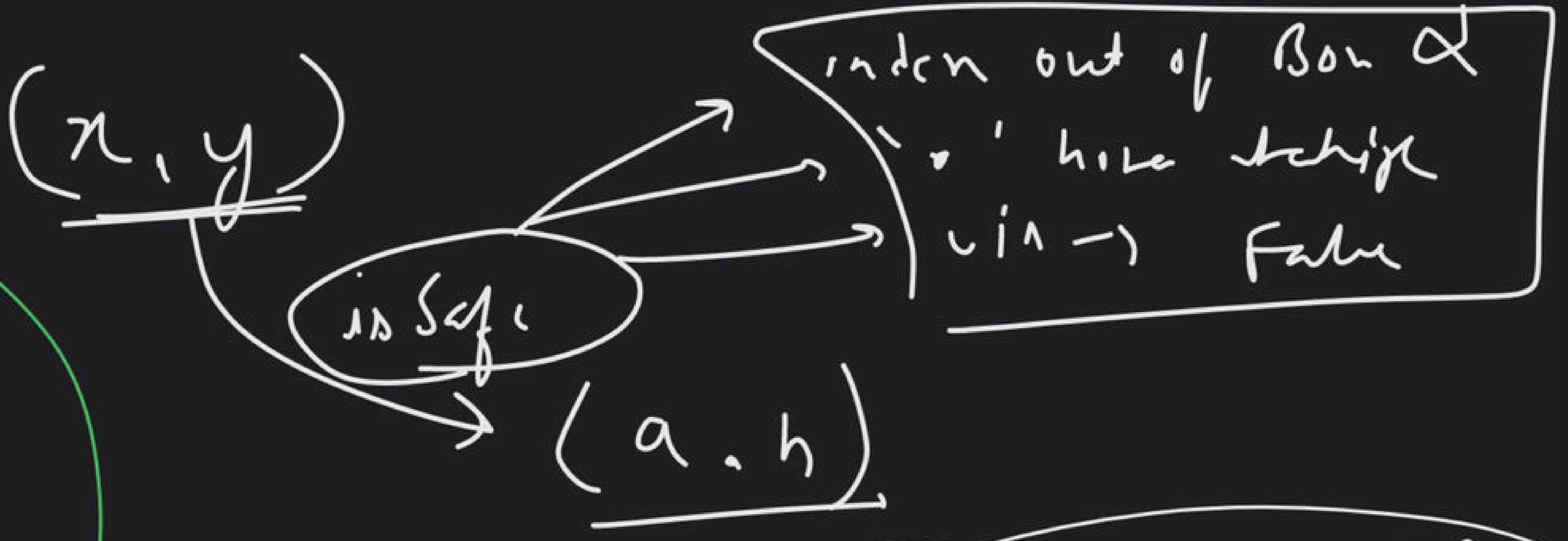
visited



H/W

Knights tour

Bishop GFG pr



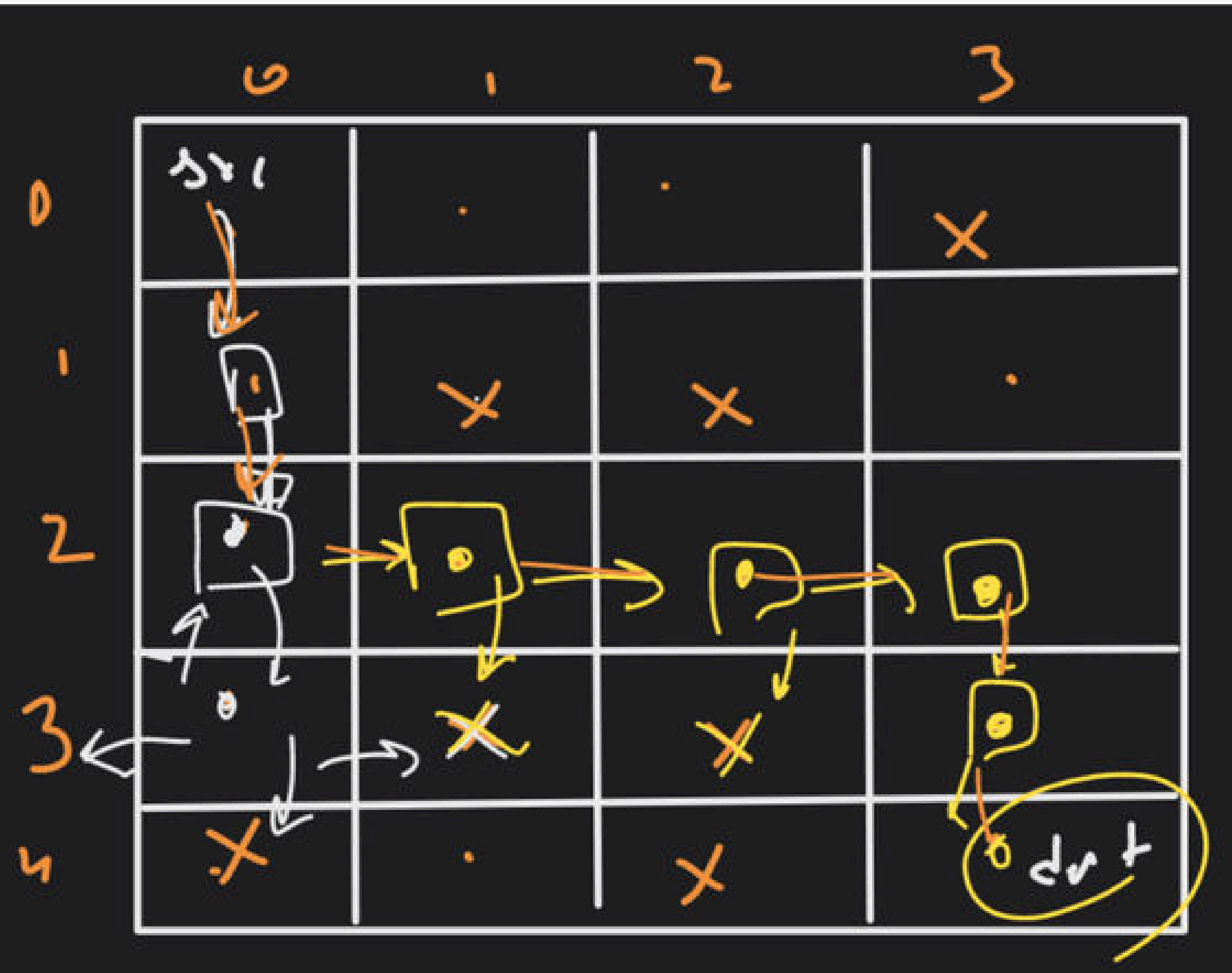
index out of Box &
'o' have already
visit → False

$n \times m$

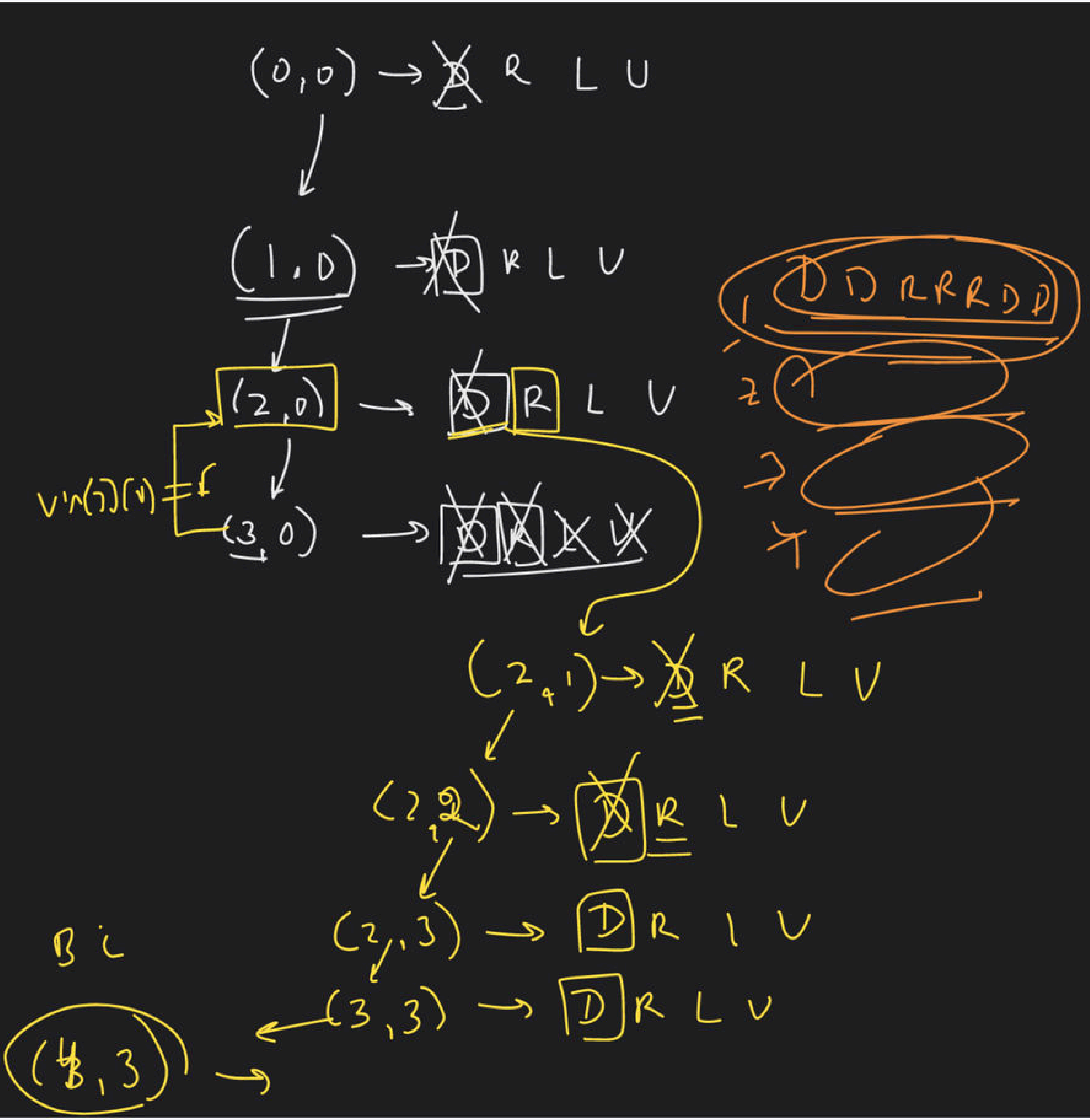
T.C → □ n^2 — □

S.C → n^2

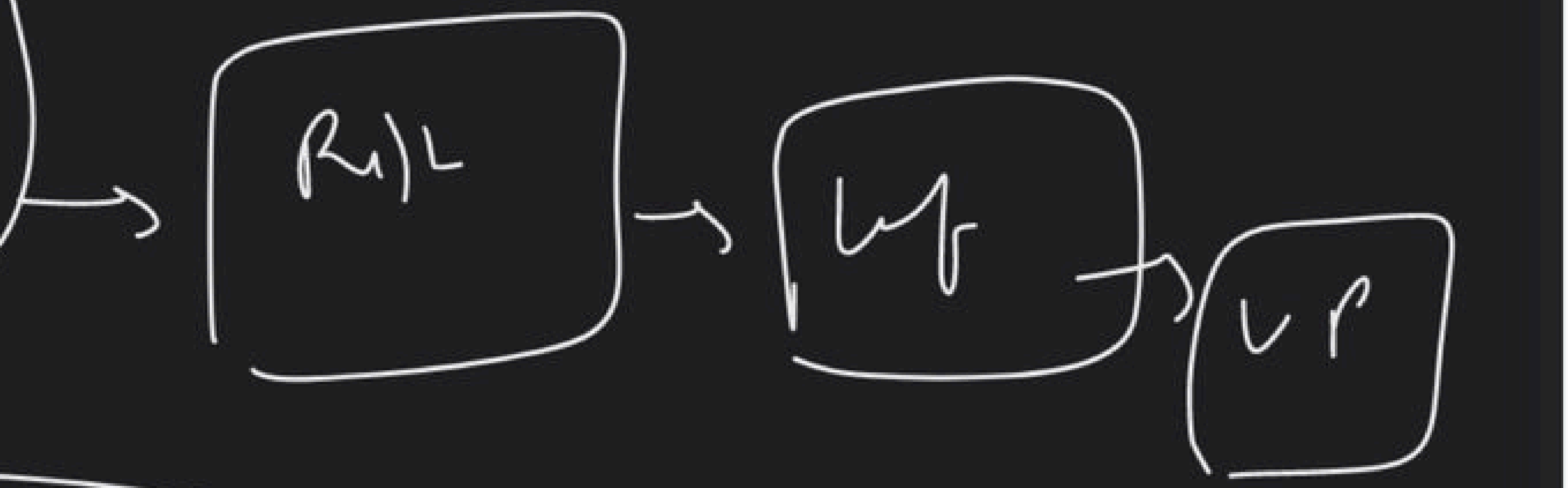
Google



	0	1	2	3
0	T ~~F~~	F	F	F
1	T ~~F~~	F	F	F
2	T ~~F~~	~~F~~ T	~~F~~ T	~~F~~ T
3	F	F	F	~~F~~ T
4	F	F	F	~~F~~



```
bool solve (srcx , srcy , dstx , dsty , board{} , vis)
{
        // B.Case
        if ( srcx == dstx && srcy == dsty )
            return true;

        if ( isSafe (down) )
        {      vis[ ][ ] = T
               Rec Call
               vis[ ][ ] = F
        }
```



Ques
↳ True or False

DRLU

```
if (D or R or L or U)
    return T
return false;
}
```

```
main()
{
    bool ans = solve( 0,0 , 4,4 , board{} , vis)
}
```

```
isSafe( newx , newy , board , vis)
{
if ( newx>=0 && newx < rowSize
     newy>=0 && newy < colSize
     vis[newx][newy] = F
     board[newx][newy] = '.' )
     return true
  else
       return false;
}
```

$$\begin{tikzcd}
 & & (-1, 0) & & \\
 & & (i-1, j) & & \\
(0, -1) & (i, j-1) \arrow[l] & i, j \arrow[l] \arrow[r] \arrow[u] \arrow[d] & (i, j+1) \arrow[r] & \overset{+}{(0, 1)} \\
 & & (i+1, j) \arrow[d] & & \\
 & & (+1, 0) & &
\end{tikzcd}$$