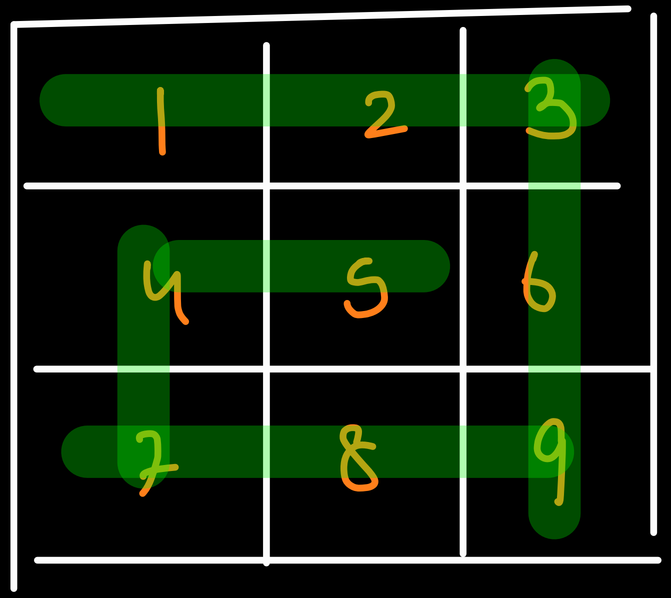
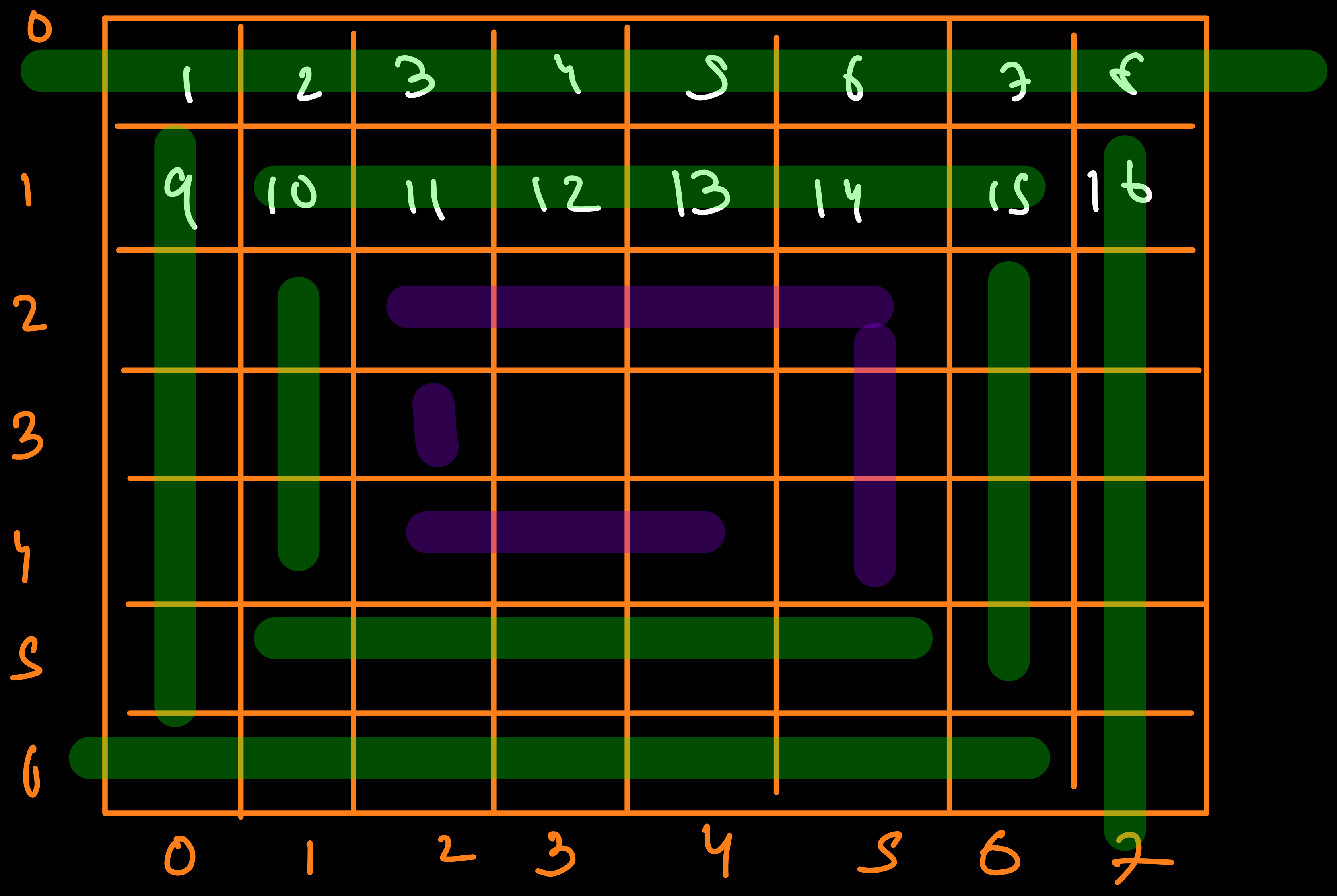


[1, 2, 3, 6, 9, 8, 7, 4, 5]

In one go, we either eliminate a complete row or a complete col



$$\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 1 & 2 & 3 & 4 & 5 \\
1 & 6 & 7 & 8 & 9 & 10 \\
2 & 11 & 12 & 13 & 14 & 15 \\
3 & 16 & 17 & 18 & 19 & 20 \\
4 & 21 & 22 & 23 & 24 & 25
\end{array}$$

$\underline{\underline{5 \times 5}}$

start_row → 0

last_row → 4

start_col → 0

last _ col → 4

→ eliminate start_row

→ eliminate end_col

→ eliminate end_row

→ eliminate start_col

$$\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 1 & 2 & 3 & 4 & 5 \\
1 & 6 & 7 & 8 & 9 & 10 \\
2 & 11 & 12 & 13 & 14 & 15 \\
3 & 16 & 17 & 18 & 19 & 20 \\
4 & 21 & 22 & 23 & 24 & 25
\end{array} \rightarrow \text{grid}$$

5x5

start_row → ~~0~~ 1
last_row → ~~4~~ 3
start_col → ~~0~~ 1
last_col → 3

continue;
break;

→ eliminate start_row (left to right)
→ eliminate end_col
→ eliminate end_row (right to left)
→ eliminate start col (down to up)

we want to repeat the process till the time we don't add all elements of the grid to the result.

result → [1, 2, 3, 4, 5, 10, 15, 20, 25, 24, 23, 22, 21, 16, 11, 6]

$x$ = ~~3~~ ~~2~~ ~~1~~ 0

Total elements in grid → n×m

```
while(count < n×m) {

}
```

① How to eliminate start_row ??

# Requirement ⇒ we need to know start_row, start_col, last_col

code →

denotes current column

```
for(let x = start_col; x <= last_col; x++)
{
    result.push(grid[start_row][x])

}
↳ start_row += 1;
```

$x = \cancel{0} \longrightarrow 4$

$\cancel{1}\ \cancel{2}\ \cancel{3}\ \cancel{4}\ 5$

How to eliminate end-col

# Requirements → last_col, start_row, last_row.

```
for(let x = start_row; x <= last_row; x++)
{
        result.push(grid[x][last_col]
}
last_col -= 1;
```

## How to eliminate last_row

# Requirements → last_row, start_col, last_col

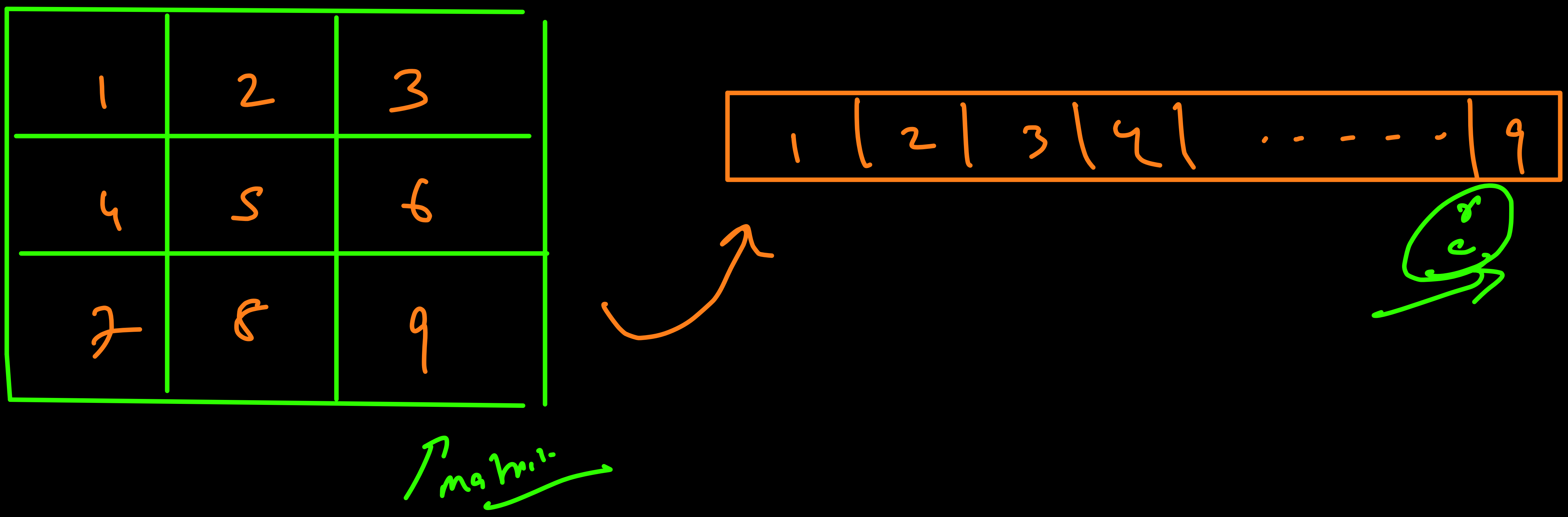
```
for(let x = last_col; x >= start_col; x--)
{
    result.push(grid[last_row][x]);
}
last_row -= 1;
```

$\nearrow 0$

How to eliminate start-col?

# Requirement
↳ start-col, last-row, start-row

```
for (let x = last-row; x >= start-row; x--){
    result.push (grid [x] [start-col]);
}
start-col ++ ;
```



① → create a new matrix of $\underline{\underline{r \times c}}$

row
col

row = ~~0~~ 1
col = 0

$$\begin{array}{c|ccc} & & \downarrow j & \\ \hline i \rightarrow & 1 & 2 & 3 \\ & 4 & 5 & 6 \\ & 7 & 8 & 9 \\ & 10 & 11 & 12 \end{array}$$

$\underline{\underline{4 \times 3}}$

$\rightarrow$ $r = 2$
$\underline{\underline{c = 6}}$

$$\begin{array}{c|cccccc} & \overset{col}{\downarrow} & & & & & \\ \hline & 1 & 2 & 3 & 4 & 5 & 6 \\ row \rightarrow & & & & & & \end{array}$$

$\underline{\underline{2 \times 6}}$

```
                2
for (let i = 0 ; i < m ; i++) {
                 2
  for (let j = 0; j < n ; j++) {
      element = matrix [i] [j];  → current element from original
                                   matrix
      result [row][col] = element;
      col ++;
      if (col == c) {
          row ++
      }
  }
}
```

$i = 1, j = \cancel{0}\,\cancel{1}\,2$

elent $\Rightarrow$ $\cancel{4}\,\cancel{5}\,6$

JOIN THE DARKSIDE

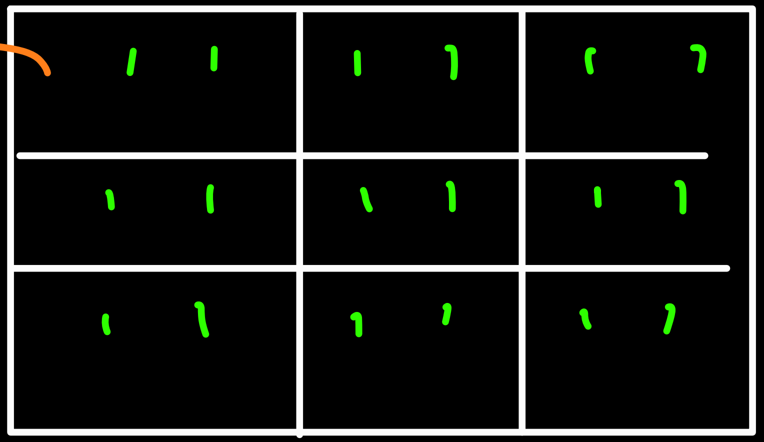
2 players → A → first
B

3x3 grid

player can fill something only on empty sq.

empty square

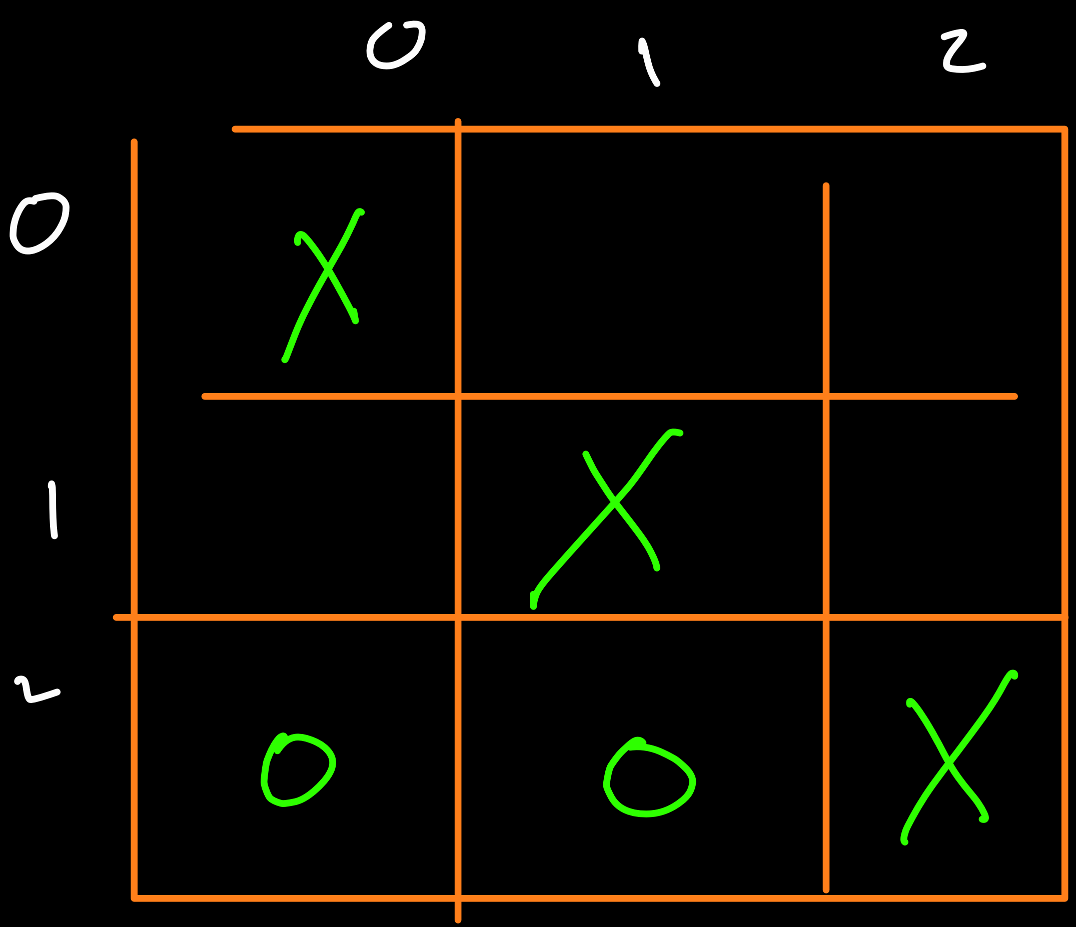
8 pair chan



A → X
B → O

$2^i$

[ [0,0], [2,0], [1,1], [2,1], [2,2] ] → moves



A (X) ← turn

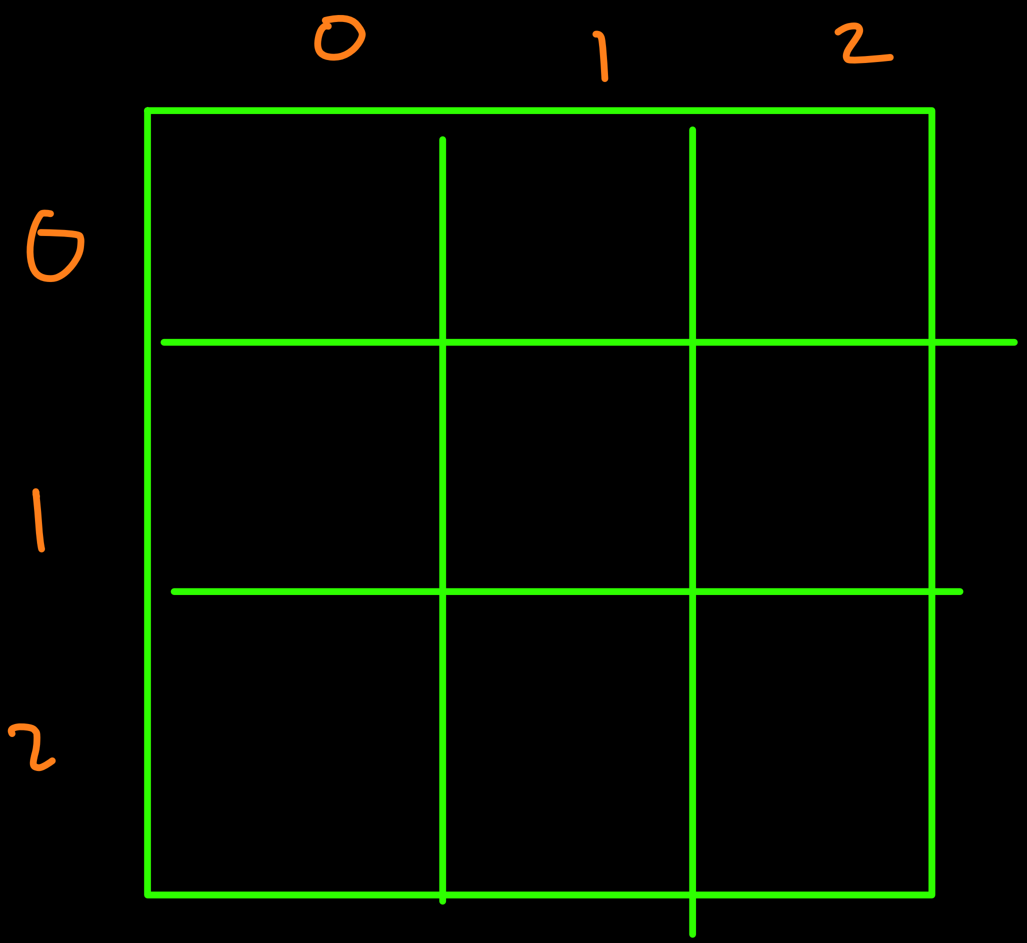
B (O)

State ⇒ "~~Pendy~~"; → A
→ B

turn → turn = (turn+1) % 2

Binary → 0/1

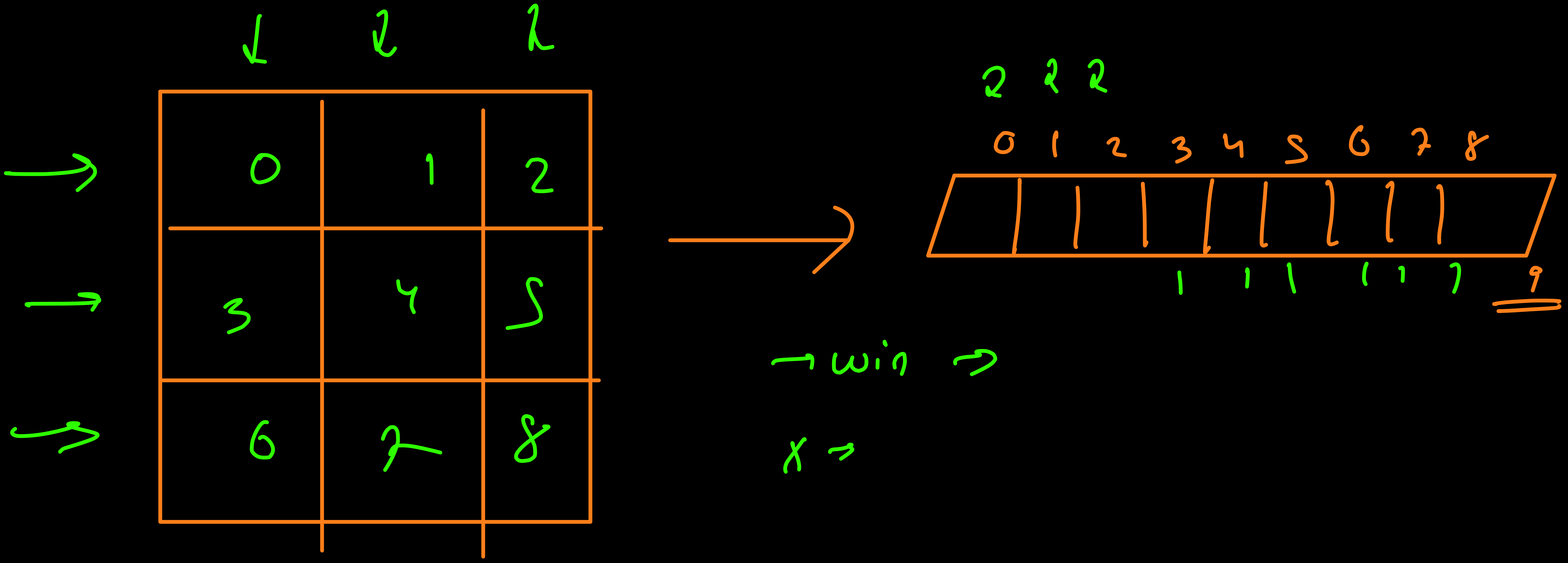
0 → A
1 → B



winning state

1. Any Row
2. Any col
3. Any diag

1. Any Row → [0,0] [0,1] [0,2] ⟷ [1,0] [1,1] [1,2] ⟷ [2,0] [2,1] [2,2]
2. Any Diag → [0,0] [1,1] [2,2] ⟷ [2,0] [1,1] [0,2]
3. Any col →



→ win →

X →