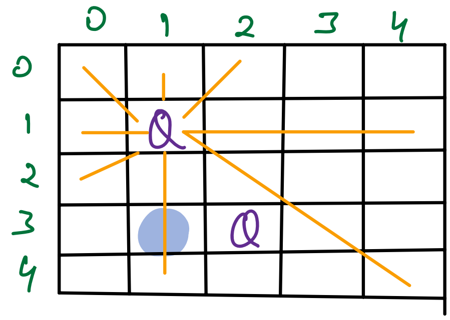
Welcome ☺

Backtracking 2

- N Queens
- Sudoku

Q

Given a N*N chessboard & locaⁿ of 2 queens. Check if they cannot attack each other.



I/P → (1,1) & (3,1) ⇒ false

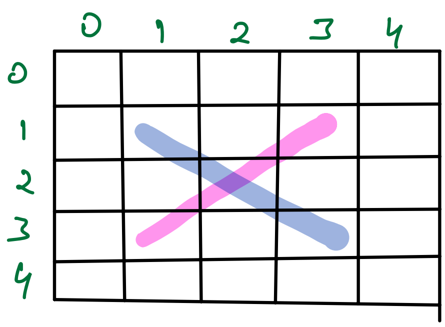
I/P → (1,1) & (3,2) ⇒ True.

Direcⁿ

1) ⟷ same row ($r_1 == r_2$)

2) ↕ same column ($c_1 == c_2$)

3) ↖↘ $r_1 - c_1 == r_2 - c_2$

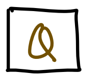


4) ↗↙ $r_1 + c_1 == r_2 + c_2$

Q N Queen's problem

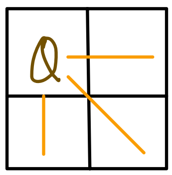
Given a integer N, check if it is possible to place N queens on a N*N chessboard s.t no queens attack each other.

N=1



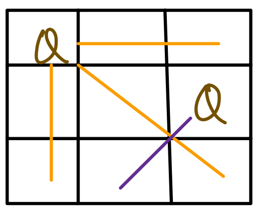
True

N=2

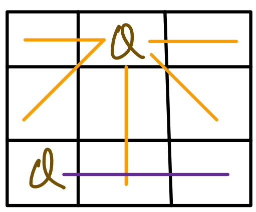


False.

N=3

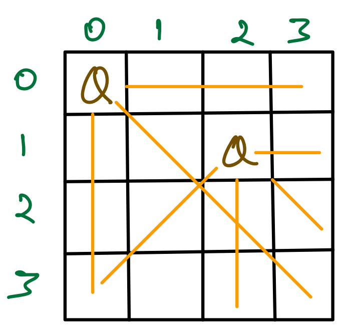


false.

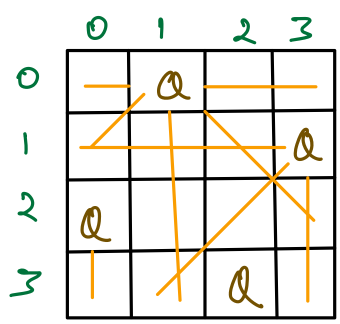


false.

N=4



→ false.



⇒ True

N Queens & N*N chessboard.

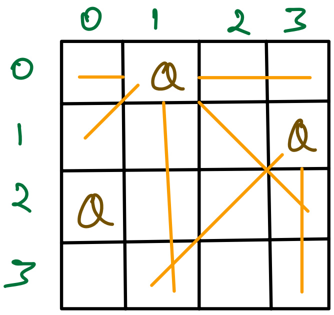
→ Every row should have exactly 1 queen.

→ Every column should have exactly 1 queen

a) Place queen row by row

b) Place queen column by column.

⇒ We don't really need N * N extra space to keep track of placed queens. S.C ⇒ O(N)



⇒ col →

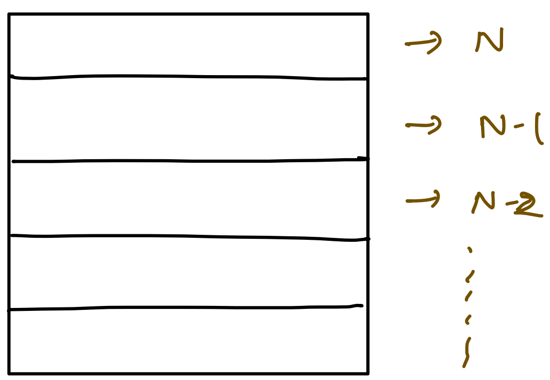
0	1	2	3
1	3	0	

(i, col[i]) → locaⁿ of iᵗʰ queen

Code

```
boolean Nqueens( r 0 , N, col[] )
{
    if( r== N) return true;   // Base case.
    for( c -> 0 to N-1)   // all possibilities.
    {
        if( isValid ( col [], r, c ) // valid possibilities
        {
            col[r] = c      // DO
            if( Nqueens( r+1, N, col[]))  // Recursion
                return True.
            col [r] = -1    // UNDO
        }
    }
    return false.
}
```

```
boolean isValid ( col[] , r, c )        T.C => O(N)
{
    for ( i -> 0 to r-1)
    {
        j = col[i]
        if ( j==c || (i-r)==(j-c) || (i+j)==(r+c))
            return false.
    }
    return True.
}
```



N!

did not consider diagonals. we just considered column.

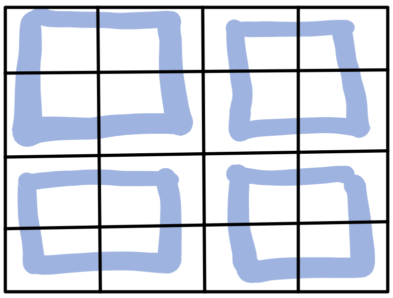
T.C ⇒ O(N! * N) < O(N!) < O(N^N)

S.C ⇒ $O(\underbrace{N}_{\text{1D array}} + \underbrace{N}_{\text{recursion.}})$

Q Solve the given incomplete sudoku

Sudoku → N * N grid where N is a perfect square.
Every row, every column & every block must have unique element.

N = 4



N = 4

1			
	3	2	4
	2		
4		3	2

→

	0	1	2	3
0	2	4	1	3
1	1	3	2	4
2	3	2	4	1
3	4	1	3	2

(1,2) (r,c)

$\left(\frac{1}{2}\right) * 2 = 0$

$\left(\frac{2}{2}\right) * 2 = 2$

$1 - (1 \% 2) = 0$

(3,2)

$3 - 3\%2 = 2$

$2 - 2\%2 = 2$

Code

```
bool sudoku( A[][] , N , r, c )
{
    if( c == N ){
        r++ , c = 0
    }
    if( r == N ){        // Base condⁿ
        return true
    }
    if( A[r][c] > 0 ){   // already filled.
        return sudoku( A , N , r, c+1)
    }

    for ( i → 1 to N ){  // all possibilities
    |
```

valid case & Recursion

```
        A[r][c] = i      // DO

        if ((check(A, N, r, c) && sudoku(A, N, r, c+1))
            return True.

        A[r][c] = -1     // UNDO
    }
    return false.
}

bool check( A[][], N, r, c)        T.C => O(N)
{
    for( i -> 0 to N-1){
        if( i != c && A[r][c] == A[r][i] )
            return false.
        if( i != r && A[r][c] == A[i][c] )
            return false.
    }

    sq = sqrt(N)
    u = r - r%sq
    v = c - c%sq
    for( i -> 0 to (sq-1))
    {
        for( j -> 0 to (sq-1))
        {
            x = u+i
            y = v+j                  excluding current cell.
            if( (x!=r || y!=c) && A[x][y] == A[r][c])
                return false.
        }
    }
    return true
```

}

T.C → N * N ---- N^2 times < $O(N^{N^2})$

S.C → $O(N^2)$

↳ recursive.