Today's Agenda :-

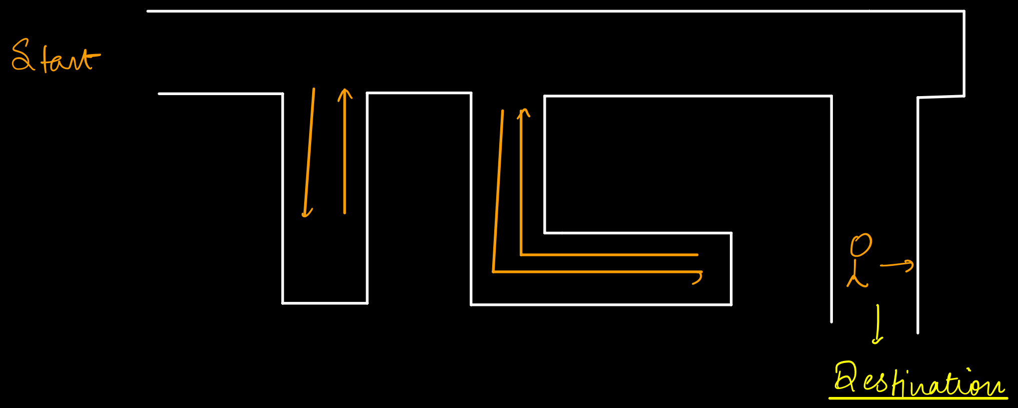
1) Intro to Backtracking

2) Print all N digit nos using 1 & 2.

3) N- Queen

4) Subset Sum

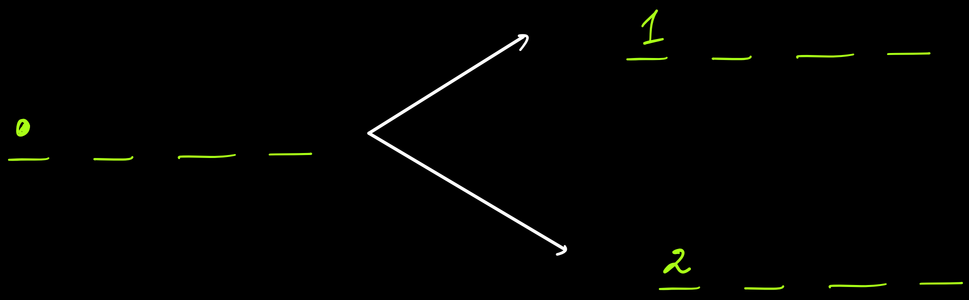
Backtracking :- Generate all possibilities & find the answer.



Q1) Print all N digit numbers that can be formed by 1 or 2 or both in increasing order.

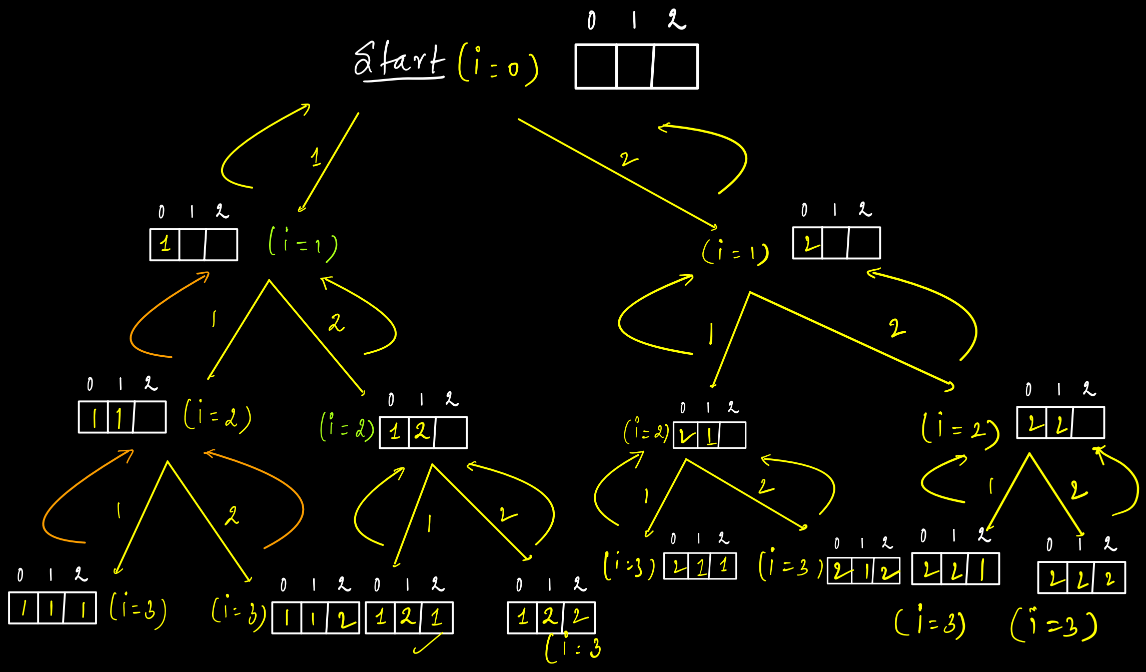
N = 1	1 2
N = 2	11 12 21 22
N = 3	111 112 121 122 211 212 221 222

N = 4



N = 3

$\underline{0}$ $\underline{1}$ $\underline{2}$ i = 3 → print the no'



O/P ✓

1 1 1 ✓

1 1 2 ✓

1 2 1 ✓

1 2 2 ✓

2 1 1 ✓

2 1 2 ✓

2 2 1 ✓

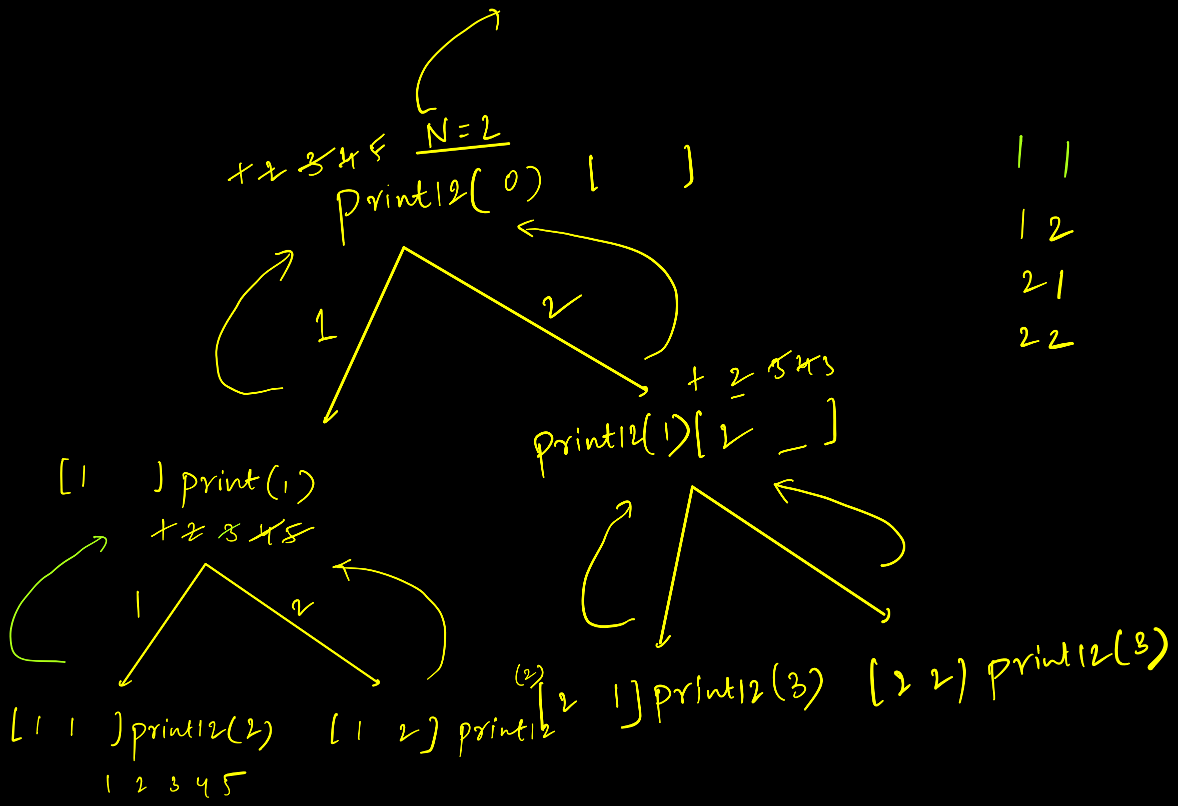
2 2 2 ✓

```
main() {
    int A[N]
    print12(N,0,A)
}
```

Assumption :- print all N digit nos containing 1 or 2.

```
void print12(int N, int i, int A[]) {

    if(i = = N) { print(A); return; }
    A[i] = 1
    print12(N, i+1, A) ;
    A[i] = 2
    print12(N, i+1, A);

}
```

```
void print12(int N, int i, int A[]) {
    if(i == N){ print(A); return; }
    A[i] = 1
    print12(N, i+1, A);
    A[i] = 2
    print12(N, i+1, A);
}
```



1 1

1 2

2 1

2 2

$\underline{2} * \underline{2} * \underline{2} * \underline{2} * \underline{2} * \quad \underline{2} = 2^N$ subsets

{1, 2} ←

[ ]

$\left\{ \begin{matrix} \{1\} & \{1,2\} \\ \{2\} & \{ \} \end{matrix} \right\}$ $\underline{2^2}$

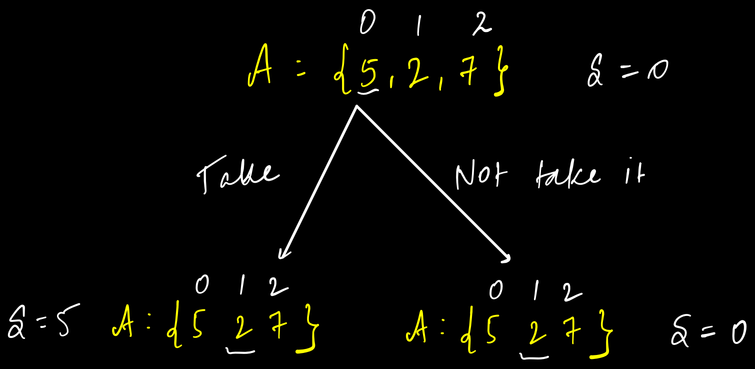
Q2) Given N array elements count No. of Subsets with sum = K ?

A = { 10, 2, 7, 6, 1, 5 }   K = 8 ⟶ 3

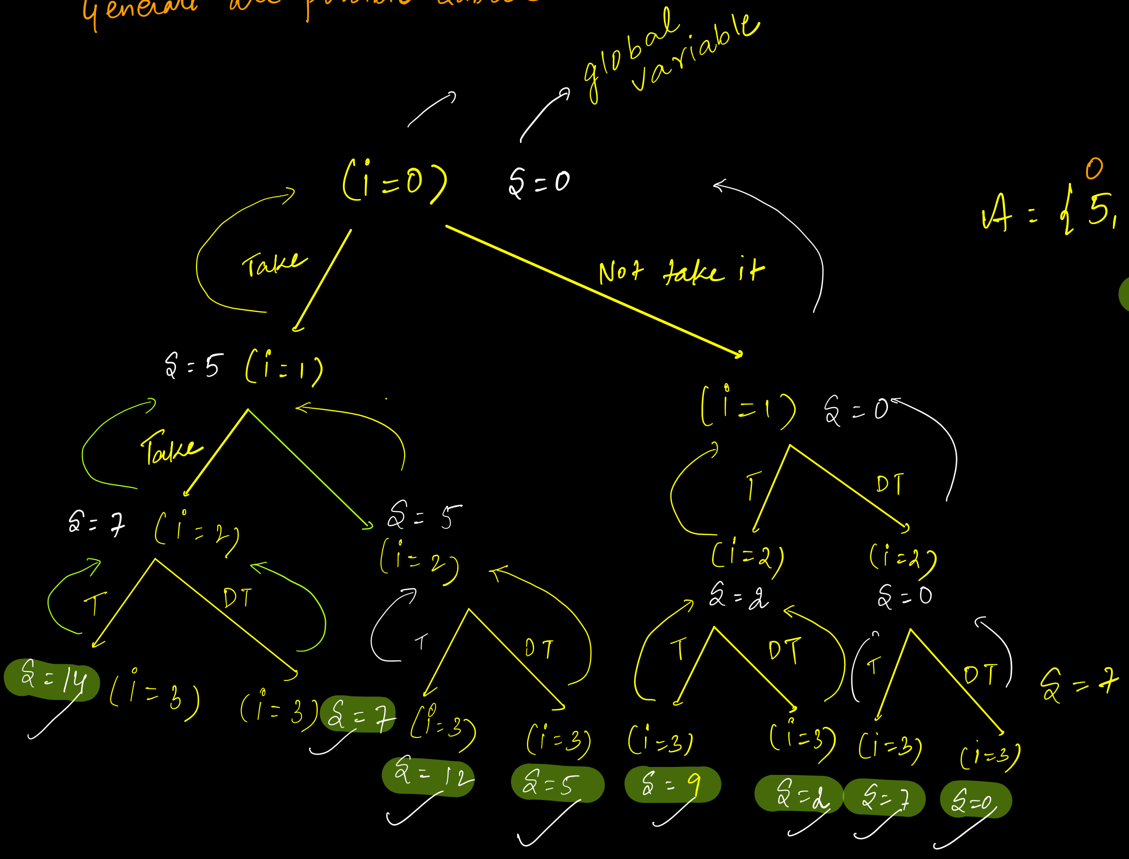
{2, 6}, {7, 1}, {2, 1, 5}

A = {5, 2, 7}   K = 7 ⟶ 2

{7}, {5, 2}



Generate all possible Subsets



$A = \{\overset{0}{5}, \overset{1}{2}, \overset{2}{7}\}$

{5}

{2} ✓

{7} ✓

{5, 2} ✓

{5, 7} ✓

{2, 7} ✓

{5, 2, 7} ✓

{ } ✓

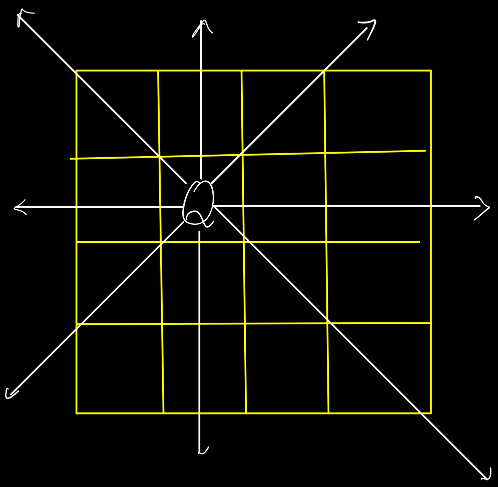
$c = \cancel{0} + 2$

```
if(i == N){
|   if(S == K) c++
}   return
```

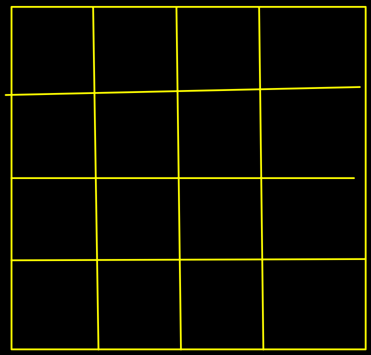
```
Cnt = 0

void subsetSum ( int A[ ], int i, int sum, int k) {
        if (i == len(A)) {
            if(sum == k) { cnt ++ }
            return ;
        }

        sum = sum + A[i];

        subsetSum( A, i+1, sum, k);

        sum = sum - A[i];

        subsetSum( A, i+1, sum, k) ;
}


main () {

        subsetSum( A, 0, 0, k) ✓

        return cnt
}
```

TODO:
Please dry run |
draw out

3Q) Given a N X N matrix, print all valid configurations of placing N queens so that none of queeers attack each other.

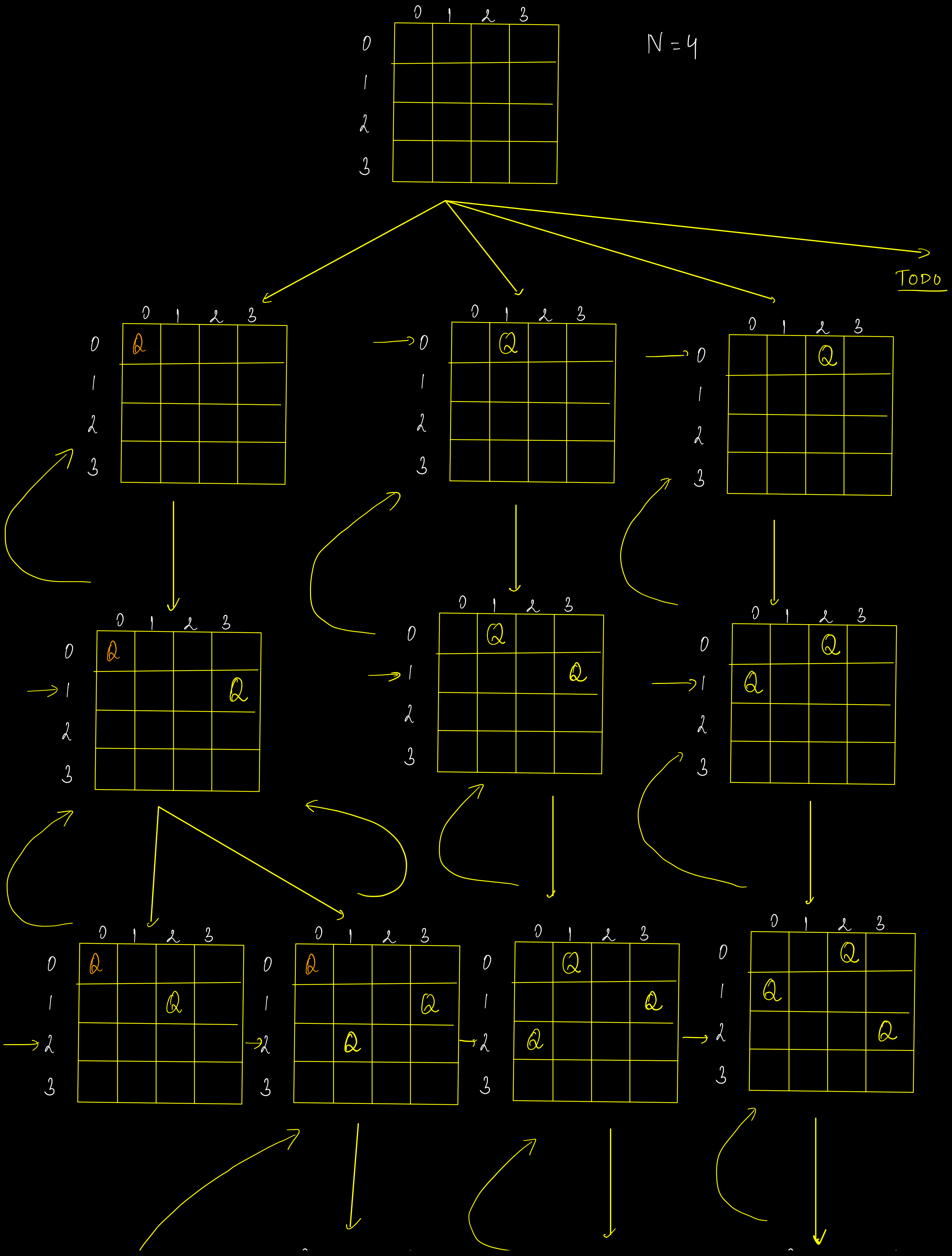


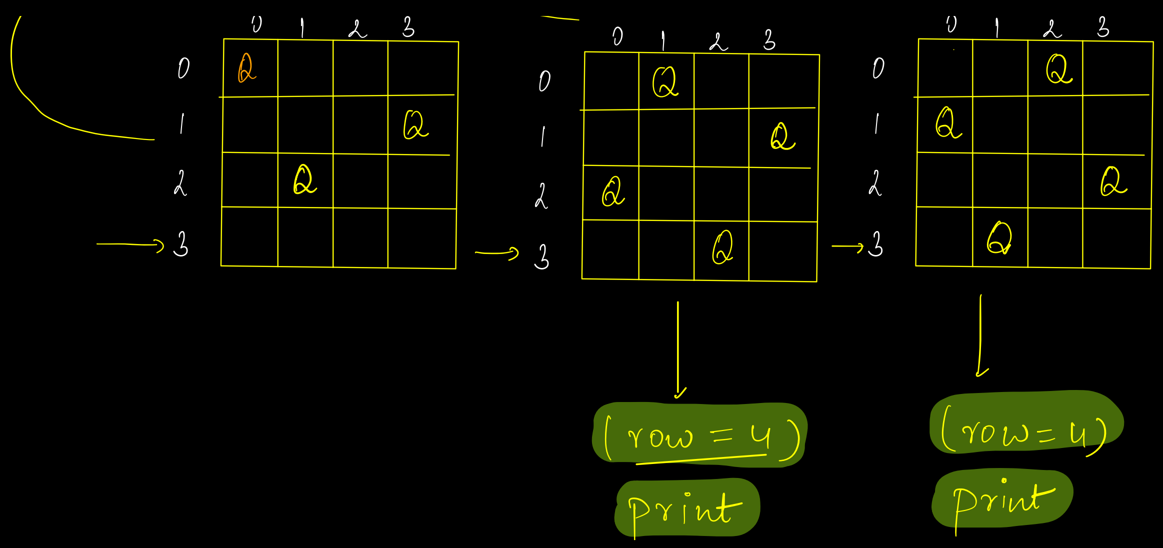
N = 4



4 Queens

1) Each queens in a different row.

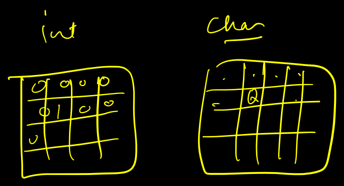

N = 4
TODO



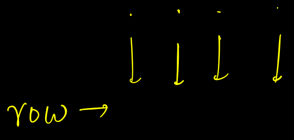
1) Are we filling the queens row by row ?

2) If you are at a row, are you trying every single column ?

3) Before placing a queen , are your ensuring safety ?



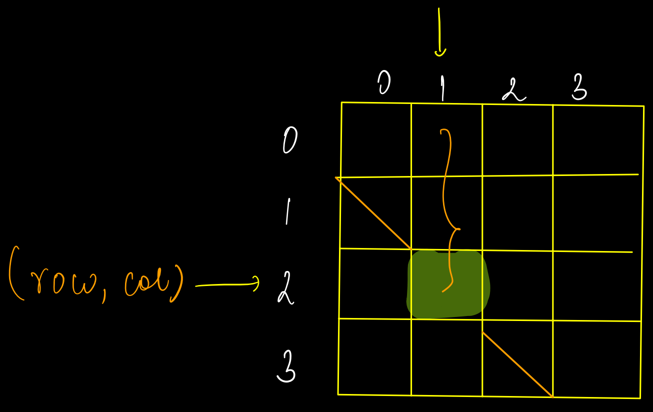
```
void NQueen ( int N, int row, int mat[ ][ ]) {
    if(row == N) {
        print(mat);
        return ;
    }
    for(int col = 0; col < N; col ++ ){
        //try placing a queen at (row, col)
        if (checkSafety (mat, row, col)){
            mat[row][col] = 1
            NQueen( N, row +1, mat);
            mat[row][col] = 0
        }
    }
}
```



checkSafety:
- → check for a queen in col
- → diagonal check

$T(N) = N * T(N-1) + 1$

```
bool checkSafety (mat, row, col) {
        // col  } ret F
        // diag }
        ret T;
}
```



```
→ for(int i=0; i<N; i++){
      if(mat[i][col]==1){
          return false
      }
  }
```

Diagonal check

Try on your own

TC & SC

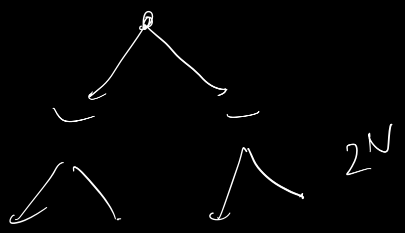
Please try to write

✓ Recurrence Relation

✓ Recursive Tree method

1   2

Very similar to fibonacci



2 things

① Please try to — /

② Recap Imp - / will be uploaded by today

③ //