Backtracking:- (0,0)

	0	1	2	3
0	0	0	9	0
1	0	0	9	0
2	9	9	0	0
3	0	9	0	0

9 → each ex a hurdle

(3,3)

1.

1	1	1	0
0	1	1	0
9	1	1	1
0	0	0	1

2.

1	1	1	0
1	1	1	0
9	1	1	0
0	1	1	1

3.

1	0	1	0
1	1	1	0
9	1	1	0
0	9	0	1

4.

1	0	1	0
1	1	1	0
9	1	1	0
0	1	1	1

```
public static void mazepath(int[][] arr,
                            int sr, int sc, int dr, int dc){
    if (arr[sr][sc] == 9){
        arr[sr][sc] = 1;  display(matrix); return;
    }
    if (sr > dr || sc > dc){
        return;
    }
    if (arr[sr][sc] == 0){
        return;
    }
    arr[sr][sc] = 1;
    mazepath(arr, sr, sc+1, dr, dc);
    mazepath(arr, sr+1, sc, dr, dc);
    arr[sr][sc] = 0;
}
```

1

	0	1	2	3
0	1	0	9	0
1	1	1	9	0
2	9	1	0	0
3	0	9	0	1

Observe
N×N = chess board
N = Queens
↳ How many ways are possible?
N=4
4 Queens on 4×4 chess board.

(0,3)

board →

	0	1	2	3
0				
1				
2				
3				

```
boolean[][] board = new boolean[4][4];
System.out.println(countNQueens(board, 0));
public static int countNQueens(boolean[][] board,
    int row){
    if (row == board.length){
        return 1;
    }
    int count = 0;
    for (int col = 0; col < board.length; col++){
        if (isSafe(board, row, col)){
            board[row][col] = true;
            count = count + countNQueens(board, row+1);
            board[row][col] = false;
        }
    }
    return count;
}
```

	0	1	2	3
0				
1				
2				
3				

```
public boolean isSafe(boolean[][] board, int row
                      int col){
    for (int i = row; i >= 0; i--){
        if (board[i][col]){
            return false;
        }
    }
    for (int i = row, j = col; i >= 0 && j >= 0; i--, j--){
        if (board[i][j]){
            return false;
        }
    }
    for (int i = row, j = col; i >= 0 && j < board.length; i--, j++){
        if (board[i][j]){
            return false;
        }
    }
    return true;
}
```

Q- TOH

n=3

source dest helper

1. Move 1st disc to dest.
2. Move 2nd — — — helper.
3. — 1st — — helper
4. Move 3rd — — dest
5. — 1st — — source
6. — 2nd — — dest
7. — 1st — — dest

7 steps

(3, src, dest, helper)
↳ (2, src, helper, dest).
↳ (1, src, dest, helper);

"src", "dest", "helper"

```
public static void TOH(String src, String dest, String helper){
    if (n == 0){
        return;
    }
    TOH(n-1, src, helper, dest);
    Sys.o(Move 1st disc from src to dest);
    TOH(n-1, helper, dest, src);
}
```

Resume at 8:48 ⇒

Q- Print lexico> end → 1000
curr → 0

lexico - dictionary order.

1
10
100
1000
101
102
103
104
105
106
107
108
109
11
110
111
112
113
114
115
116
117
118
119
12
120
121
122
123

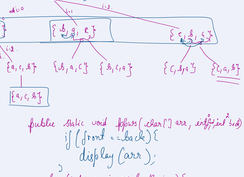
```
public static void lexico(int end, int curr){
    if (curr > end){
        return;
    }
    Sys.o(curr);
    if (curr == 0){
        for (int i = 1; i <= 9; i++){
            lexico(end, curr*10+i);
        }
    } else {
        for (int i = 0; i <= 9; i++){
            lexico(end, curr*10+i);
        }
    }
}
```

if (curr == 0)

(1×10+0) => 10
(10×10+0) => 100

end, curr
(1000, 0)

Q> print permutations.
char[]arr = {'a', 'b', 'c'};




```
public static void permute(char[] arr, int front, int back){
    if (front == back){
        display(arr);
    }
    for (int i = 0; i < arr.length; i++){
        swap(arr, front, i);
        permute(arr, front+1, back);
        swap(arr, front, i);
    }
}
```

(difficult)

→ DS (easy)

Recursion-3
↳ Recursion → 4

Friday ↳ 7:00pm