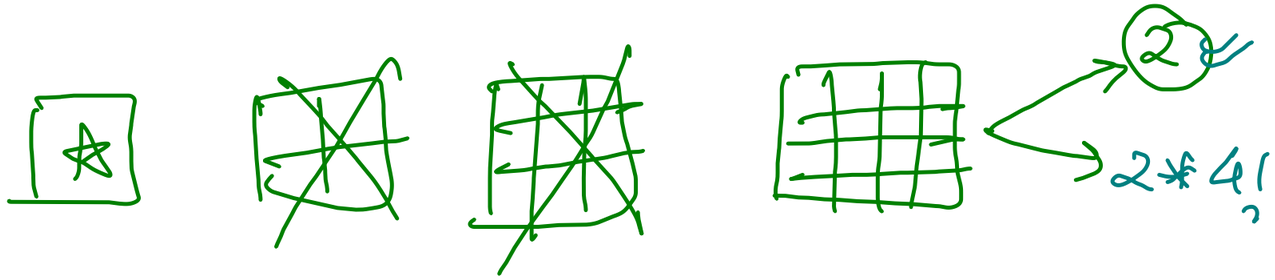
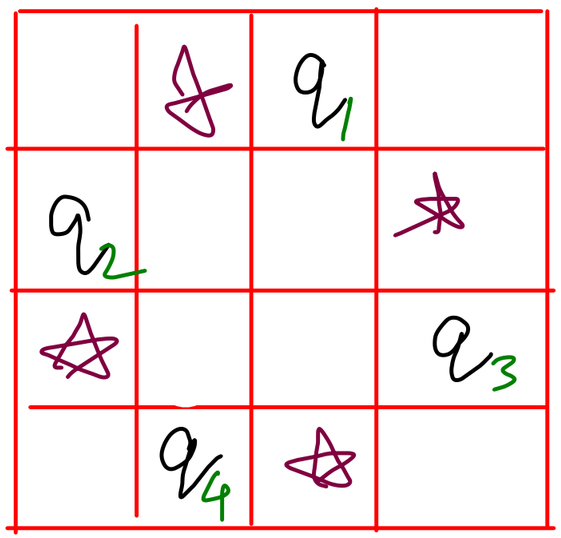
Recursion & Backtracking

Today's Class { lecture - 7 }

N Queen - Combinations	
N Queen - Permutations	
N-Queens - I (IB)	
N-Queens - II	
Using Backtracking	Code
Using Branch & Bound	
Using Bit Manipulation	
N Knight	
Knight's Tour	GfG Article
Knight's Tour	Code





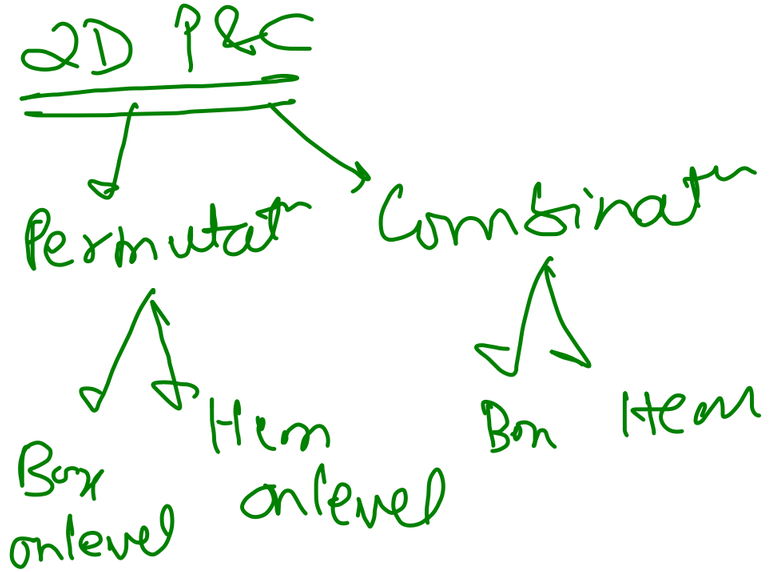
$N^2 = 4 * 4$

N = 4 queens

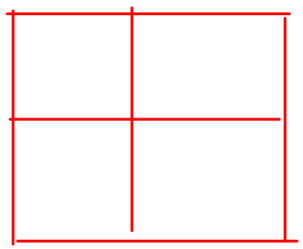
Combination

$\{q_1, q_1, q_1, q_2\}$

constraint → No queen pair should kill each other.



① N Queen - Combination



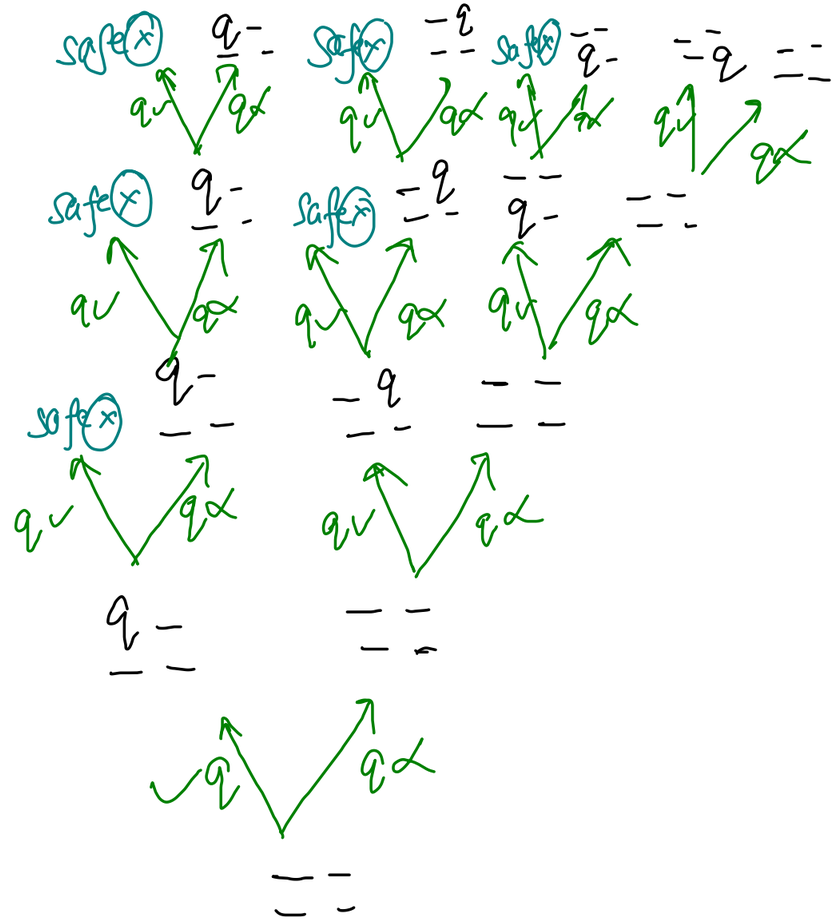
$\{q, \_\}$

2^{N^2}

N = 2 ?

0 valid configurations

(1,1) (1,0) (0,1) (0,0)



N = 9 & 2 + 9

$$(calls)^{height} + (pre + post) * height$$

$$(2^n) + n * (n^2) = O(2^n + n^3)$$

IsQueenSafe

	c0	c1	c2	c3
r0	d_D	d_E	d_g	d_h
r1	d_C	d_D	d_f	d_g
r2	d_B	d_C	d_D	d_f
r3	d_A	d_B	d_C	d_D

(1) Left Diagonal

	c0	c1	c2	c3
r0	d_I	d_{II}	d_{III}	d_{IV}
r1	d_{II}	d_{III}	d_{IV}	d_V
r2	d_{III}	d_{IV}	d_V	d_{VI}
r3	d_{IV}	d_V	d_{VI}	d_{VII}

(2) Right Diagonal

Total left or right diagonals

$= 2 * n - 1$

	c0	c1	c2	c3
r0	r0	r0	r0	r0
r1	r1	r1	r1	r1
r2	r2	r2	r2	r2
r3	r3	r3	r3	r3

(3) Row

	c0	c1	c2	c3
r0	c0	c1	c2	c3
r1	c0	c1	c2	c3
r2	c0	c1	c2	c3
r3	c0	c1	c2	c3

(4) column

Total rows or columns = n

O(N) time, O(1) space

Recursion space: → O(n²)

output: O(n²)

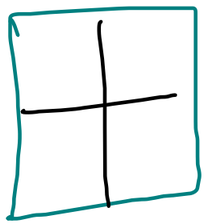
```java
public static boolean isQueenSafe(int r, int c, boolean[][] vis){
    // row (left)
    for(int j=0; j<c; j++){
        if(vis[r][j] == true)
            return false;
    }
    // col (up)
    for(int i=0; i<r; i++){
        if(vis[i][c] == true)
            return false;
    }

    // left diagnol (up)
    int i = r, j = c;
    while(i >= 0 && j >= 0){
        if(vis[i][j] == true){
            return false;
        }
        i--; j--;
    }

    // right diagnol (down)
    i = r; j = c;
    while(j < vis.length && i >= 0){
        if(vis[i][j] == true){
            return false;
        }
        i--; j++;
    }

    return true;
}
```

```java
public static void nqueen(int r, int c, int qpsf, boolean[][] vis) {
    if (qpsf == vis.length) {
        List<String> ans = new ArrayList<>();
        for (int i = 0; i < vis.length; i++) {⟺}
        res.add(ans);
        return;
    }

    if(r == vis.length){
        return;
    }

    if (isQueenSafe(r, c, vis)) {
        vis[r][c] = true;

        if (c == vis.length - 1) {
            nqueen(r + 1, 0, qpsf + 1, vis);
        } else {
            nqueen(r, c + 1, qpsf + 1, vis);
        }

        vis[r][c] = false;
    }

    if (c == vis.length - 1) {
        nqueen(r + 1, 0, qpsf, vis);
    } else {
        nqueen(r, c + 1, qpsf, vis);
    }
}
```

N-Queen Permutation

$\{q_1, q_2\}$ Brute on level

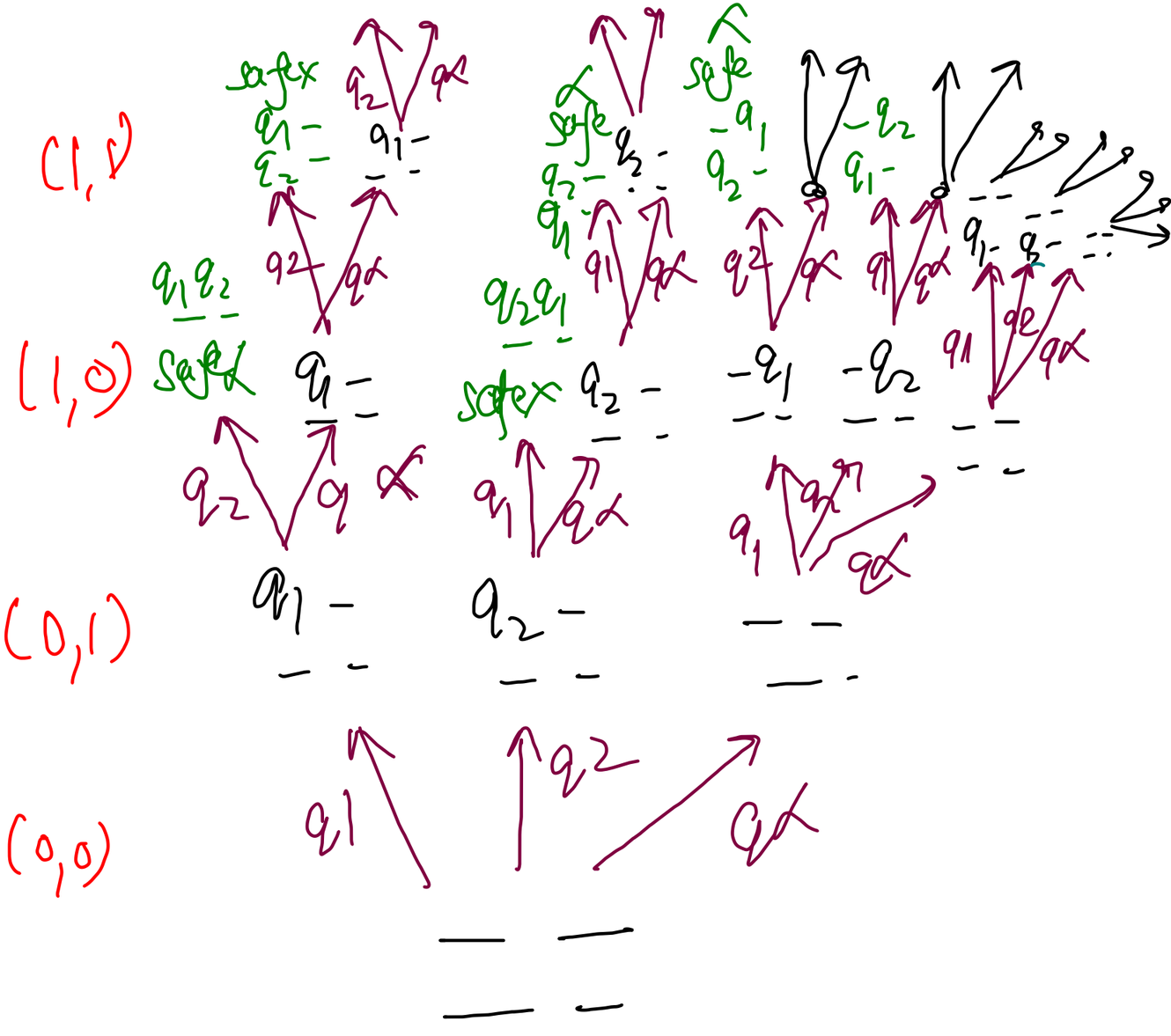
2*2 N = (2)



{0 valid configurations}

N = (21) ⟹ Permutations = 2 * 41 * 0

visited queens ~~array~~ 24 = (48)



Bar on level approach { Order Different }

```java
public static void nqueens(int qpsf, int r, int c, boolean[] queenPlaced, int[][] chess) {
    int n = queenPlaced.length;
    if(qpsf == n){
        for(int i=0; i<n; i++){ }
        System.out.println();
        return;
    }

    if(r == n){ return; }

    if(c == n-1){
        nqueens(qpsf, r+1, 0, queenPlaced, chess);
    } else {
        nqueens(qpsf, r, c+1, queenPlaced, chess);
    }

    for(int i=0; i<n; i++){
        if(queenPlaced[i] == false && isQueenSafe(r, c, chess)){
            queenPlaced[i] = true;
            chess[r][c] = (i + 1);

            if(c == n-1){
                nqueens(qpsf+1, r+1, 0, queenPlaced, chess);
            } else {
                nqueens(qpsf+1, r, c+1, queenPlaced, chess);
            }

            chess[r][c] = 0;
            queenPlaced[i] = false;
        }
    }
}
```

IsQueenSafe

	c0	c1	c2	c3
r0	d_D	d_E	d_g	d_h
r1	d_C	d_D	d_f	d_g
r2	d_B	d_C	d_D	d_f
r3	d_A	d_B	d_C	d_D

(1) Left Diagonal

	c0	c1	c2	c3
r0	d_I	d_{II}	d_{III}	d_{IV}
r1	d_{II}	d_{III}	d_{IV}	d_V
r2	d_{III}	d_{IV}	d_V	d_{VI}
r3	d_{IV}	d_V	d_{VI}	d_{VII}

(2) Right Diagonal

Total left or right diagonals

$= 2 * n - 1$

	c0	c1	c2	c3
r0	r0	r0	r0	r0
r1	r1	r1	r1	r1
r2	r2	r2	r2	r2
r3	r3	r3	r3	r3

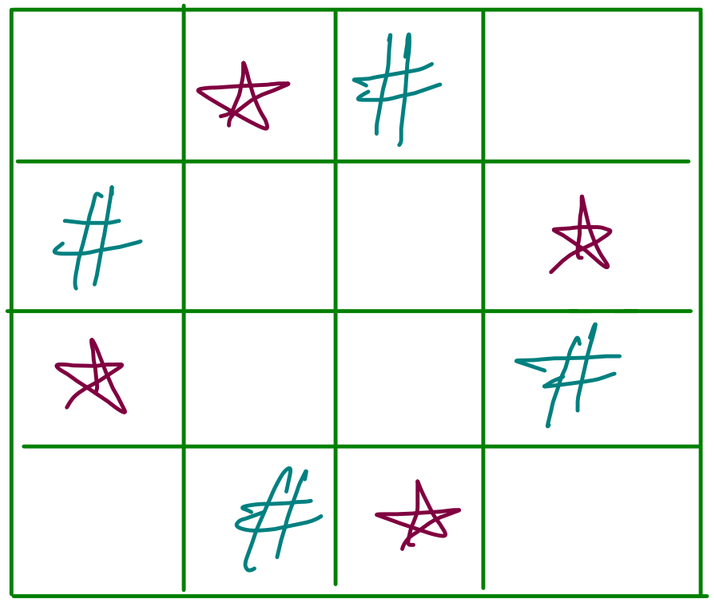
(3) Row

	c0	c1	c2	c3
r0	c0	c1	c2	c3
r1	c0	c1	c2	c3
r2	c0	c1	c2	c3
r3	c0	c1	c2	c3

(4) column

Total rows or columns = n

NQueen {Combinat} {Backtracking}

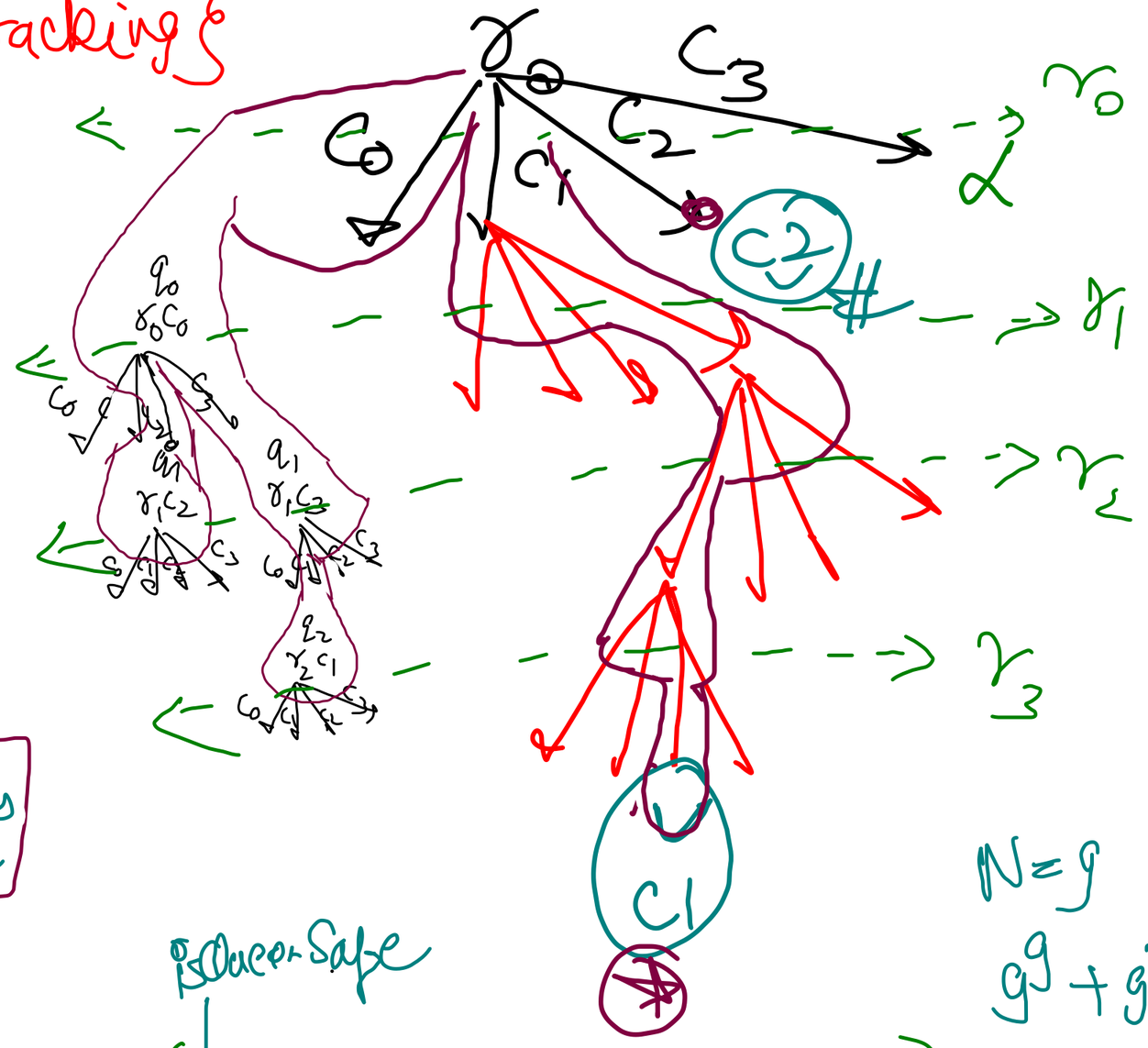


4 rows
4 Columns

4 queens

(rows)
Item on level

row m level
Column as option



$N = 9$

$9^9 + 9^2$

isQueenSafe

$(n)^n + (n) * n$

$\Rightarrow O(n^n + n^2)$

O(N) time, O(1) space

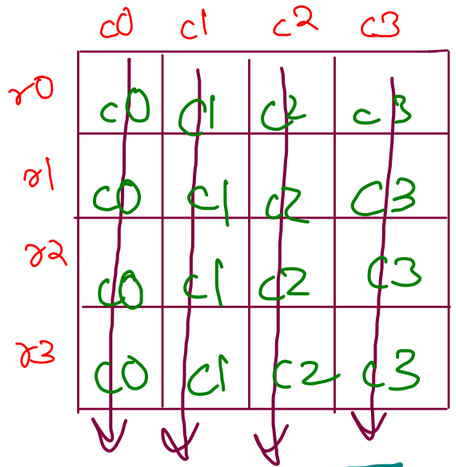
```java
public static boolean isQueenSafe(int r, int c, boolean[][] vis){
    // row (left)
    for(int j=0; j<c; j++){
        if(vis[r][j] == true)
            return false;
    }
    // col (up)
    for(int i=0; i<r; i++){
        if(vis[i][c] == true)
            return false;
    }

    // left diagnol (up)
    int i = r, j = c;
    while(i >= 0 && j >= 0){
        if(vis[i][j] == true){
            return false;
        }
        i--; j--;
    }

    // right diagnol (down)
    i = r; j = c;
    while(j < vis.length && i >= 0){
        if(vis[i][j] == true){
            return false;
        }
        i--; j++;
    }

    return true;
}
```

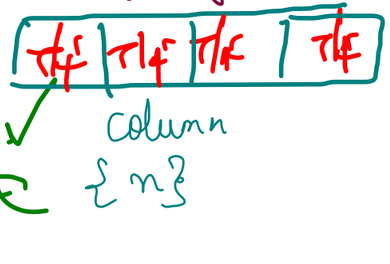
Recursion Call stack ?: → O(n)

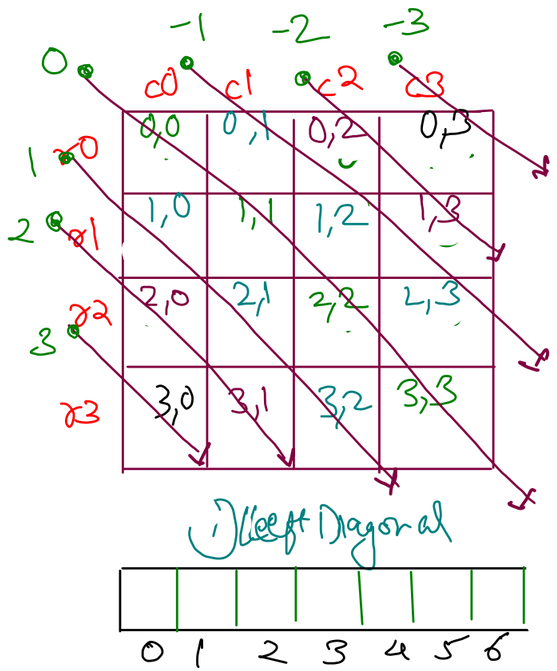
output : O(n)

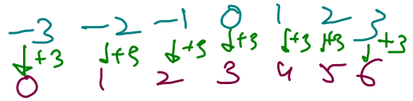
```java
public static void nqueen(int r, boolean[][] vis) {
    if (r == vis.length) {
        List<String> ans = new ArrayList<>();
        for (int i = 0; i < vis.length; i++) {⟺}
        res.add(ans);
        return;
    }


    for(int c=0; c<vis.length; c++){
        if(isQueenSafe(r, c, vis)){
            vis[r][c] = true;
            nqueen(r + 1, vis);
            vis[r][c] = false;
        }
    }
}
```

(3) Branch & Bound {optimize in isQueenSafe} in terms of time complexity



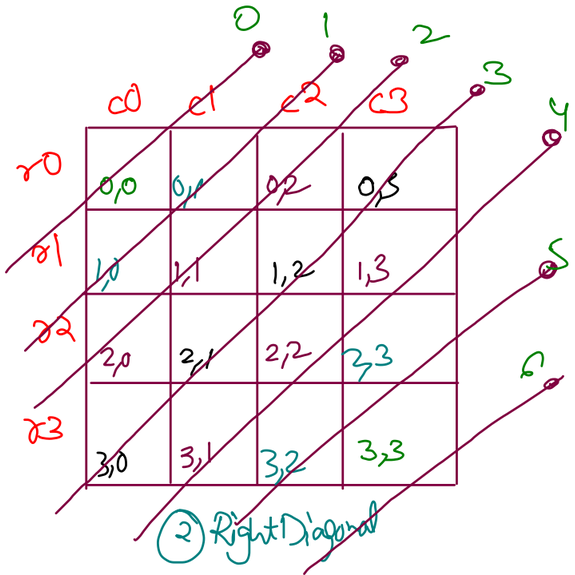






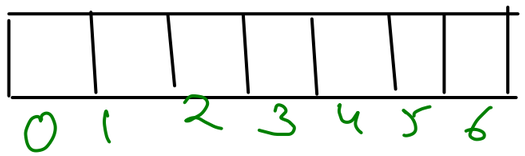
left diagnol for r,c

= ld [r - c + (n-1)]



sum => i+j

rd [r+c] T/F



{2n-1}

```java
public static boolean isQueenSafe(int r, int c, boolean[] ld, boolean[] rd, boolean[] col){
    int n = col.length;
    return ((ld[r - c + n - 1] == true) || (rd[r + c] == true) || (col[c] == true)) ? false : true;
}
```

extra space

$\Rightarrow$ O(1) time
O(n) space

```java
public static void nqueen(int r, boolean[][] chess, boolean[] ld, boolean[] rd, boolean[] col) {
    int n = chess.length;
    if (r == n) {
        List<String> ans = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            String curr = "";
            for (int j = 0; j < n; j++) {
                if (chess[i][j] == true) {
                    curr = curr + "Q";
                } else {
                    curr = curr + ".";
                }
            }
            ans.add(curr);
        }
        res.add(ans);
        return;
    }


    for(int c=0; c<n; c++){
        if(isQueenSafe(r, c, ld, rd, col)){
            chess[r][c] = ld[r - c + n - 1] = rd[r + c] = col[c] = true;
            nqueen(r + 1, chess, ld, rd, col);
            chess[r][c] = ld[r - c + n - 1] = rd[r + c] = col[c] = false;
        }
    }
}
```

O(n²) output

O(n) extra space

Recursion call stack :→ O(n)

Time Comp

$(4)^n + (k)n$

isQueenSafe

$\Rightarrow 4^n + n$

NQueen Combinatn

Time Complexity

Bon on level :→ $2^{n^2} + n^3$: 120 ms

Backtracking :- $4^n + n^2$: 4 ms

Branch & Bound :- $4^n + n$: 3 ms

Space Complexity

Bon on level :→ $O(n^2)$ Recursion call stack

Backtracking :→ $O(n)$ Recursion " "

Branch & Bound :→ $O(N)$ extra space

$O(N)$ Recursion call stack

$O(n^2)$ output (chess)

BT Manipulat :→ $O(1)$ extra space

$O(N)$ Recursion call stack

④ Bit Manipulation

Replacing boolean visited array by bitset

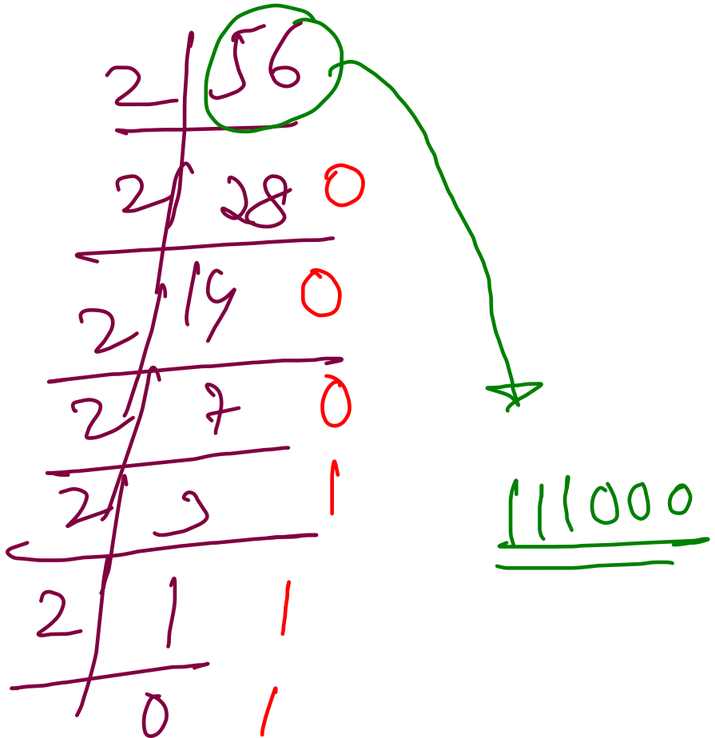
boolean visited array ⇓ O(n) bytes

T/F

bitset ⇓ n bits ⇓ O(1) space

unset / set

{0/1}



integer : 4 bytes : 32 bits

↳ ld

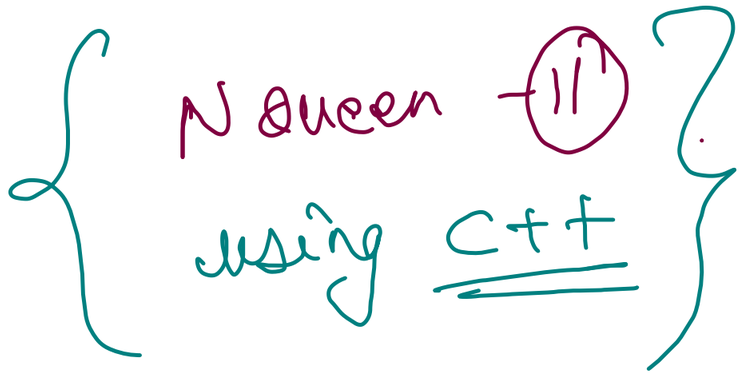
integers ↳ od

integer ↳ col

```java
public static boolean isQueenSafe(int r, int c, int n, BitSet ld, BitSet rd, BitSet col){
    return ((ld.get(r - c + n - 1) == true) || (rd.get(r + c) == true) || (col.get(c) == true)) ? false : true;
}
```

```java
public static void nqueen(int r, boolean[][] chess, BitSet ld, BitSet rd, BitSet col) {
    int n = chess.length;
    if (r == n) {
        List<String> ans = new ArrayList<>();
        for (int i = 0; i < n; i++) {⇔}
        res.add(ans);
        return;
    }


    for(int c=0; c<n; c++){
        if(isQueenSafe(r, c, n, ld, rd, col)){
            chess[r][c] = true;
            ld.set(r - c + n - 1, true);
            rd.set(r + c, true);
            col.set(c, true);

            nqueen(r + 1, chess, ld, rd, col);

            ld.set(r - c + n - 1, false);
            rd.set(r + c, false);
            col.set(c, false);
            chess[r][c] = false;
        }
    }
}
```

```cpp
bitset<1000000> col,d1,d2;
void solve(int i,int n,int &ans)
{
    if(i == n) {ans++; return;}
    for(int j=0;j<n;j++)
    {
        if(!col[j] && !d1[i-j+n-1] && !d2[i+j])
        {
            col[j] = d1[i-j+n-1] = d2[i+j] = 1;
            solve(i+1,n,ans);
            col[j] = d1[i-j+n-1] = d2[i+j] = 0;
        }
    }
}
class Solution {
public:
    int totalNQueens(int n) {
        if(n <= 1) return n;
        int ans = 0;
        solve(0,n,ans);
        return ans;
    }
};
```



Count

R&B - level-2 - lecture 8

Sudoku & Other Puzzles

Puzzles
Valid Sudoku (IB)
Sudoku Solver (IB)
Using Backtracking
Using Bit Manipulation
Crossword Puzzle
Cryptarithmetic Puzzle
Cryptarithmetic Puzzle
Max Score
Max Score

1
2
3
4

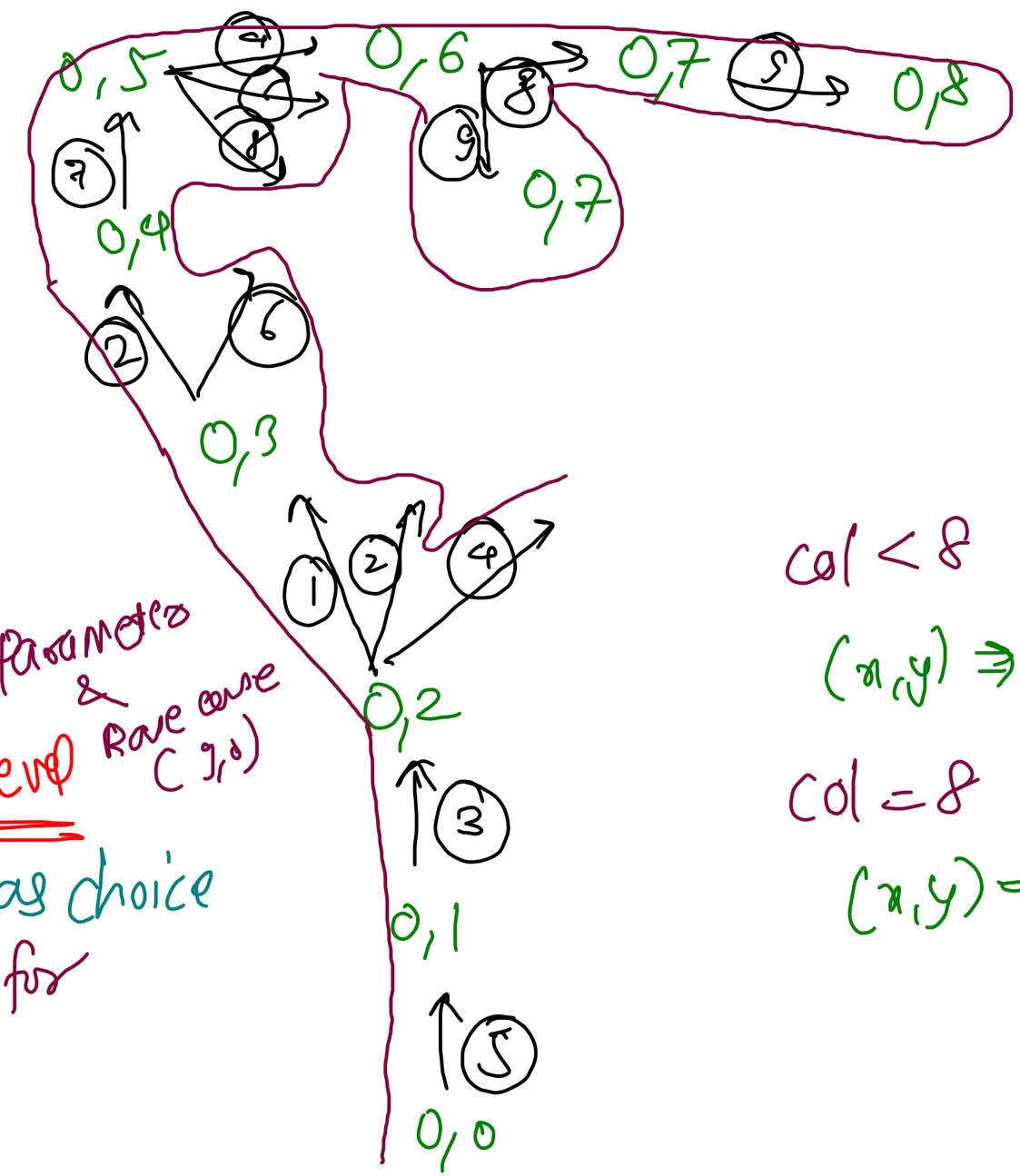
5 { NKnight , Knight's Tour

	0	1	2	3	4	5	6	7	8
0	5	3	4	6	7	3	9	1	2
1	6			1	9	5			
2		9	8					6	
3	8				6				3
4	4			8		3			1
5	7				2				6
6		6					2	8	
7				4	1	9			5
8					8			7	9

(1) In each row, we should have 1-9
↳ 9 columns

(2) In each column, we should have 1-9
↳ 9 rows

(3) In each 3*3 matrix, we should have 1-9



cell → parameters & Row, col (9,0)

Box on level

Items as choice
↳ for

col < 8

(x,y) ⇒ (x, y+1)

col = 8

(x,y) ⇒ (x+1, 0)

	0	1	2	3	4	5	6	7	8
0	5	3			7				
1	6			1	9	5			
2		9	8					6	
3	8				6				3
4	4			8		3			1
5	7				2				6
6		6					2	8	
7				4	1	9			5
8					8			7	9

top-left

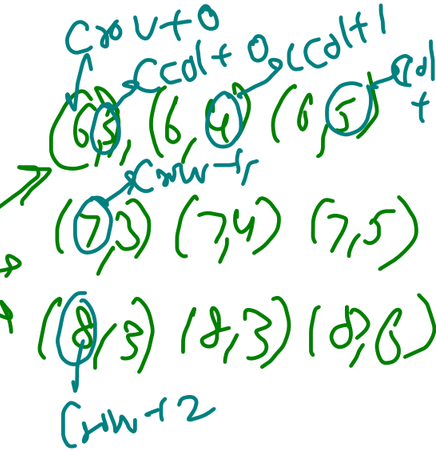
val

0,1,2 $\Rightarrow$ 0 = 0*3 = (val/3)*3

3,4,5 $\Rightarrow$ 3 = 1*3 = (val/3)*3

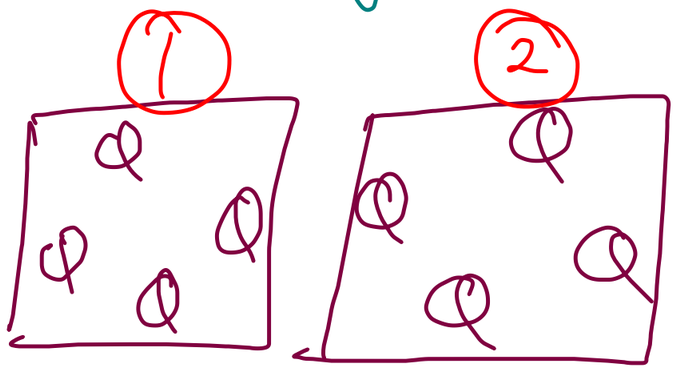
6,7,8 $\Rightarrow$ 6 = 2*3 = (val/3)*3

8,4 $\Rightarrow$ 6,3

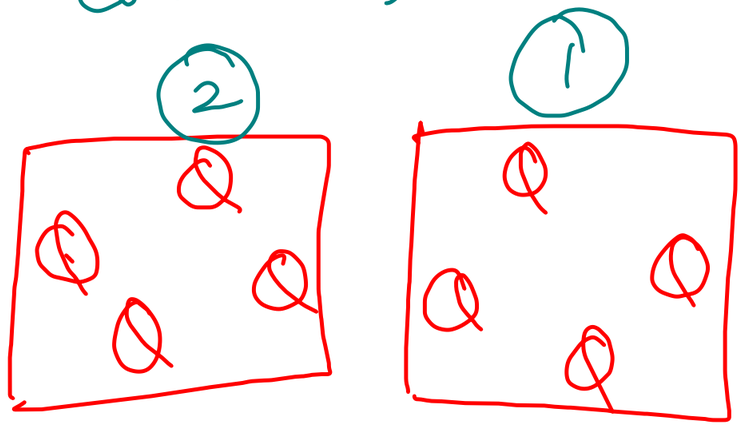


nQueen

row by row

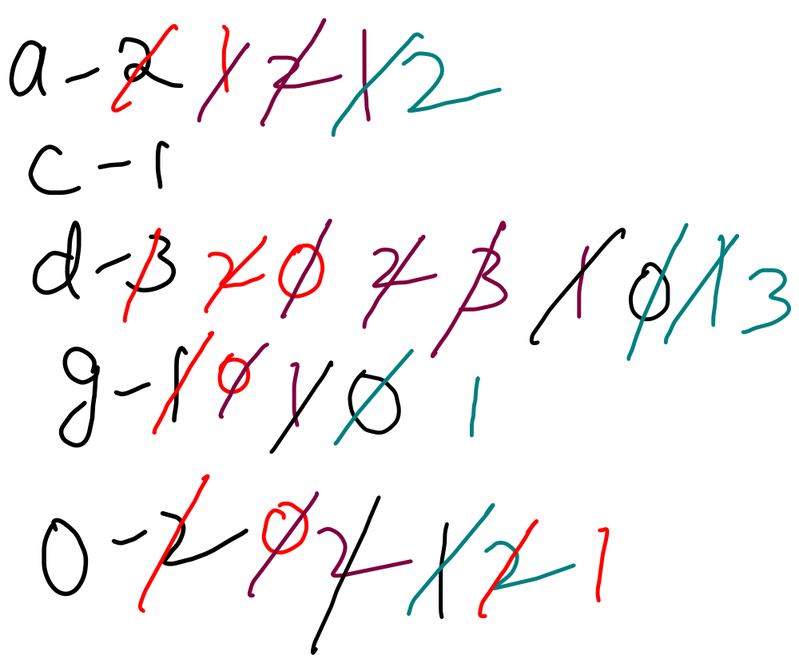


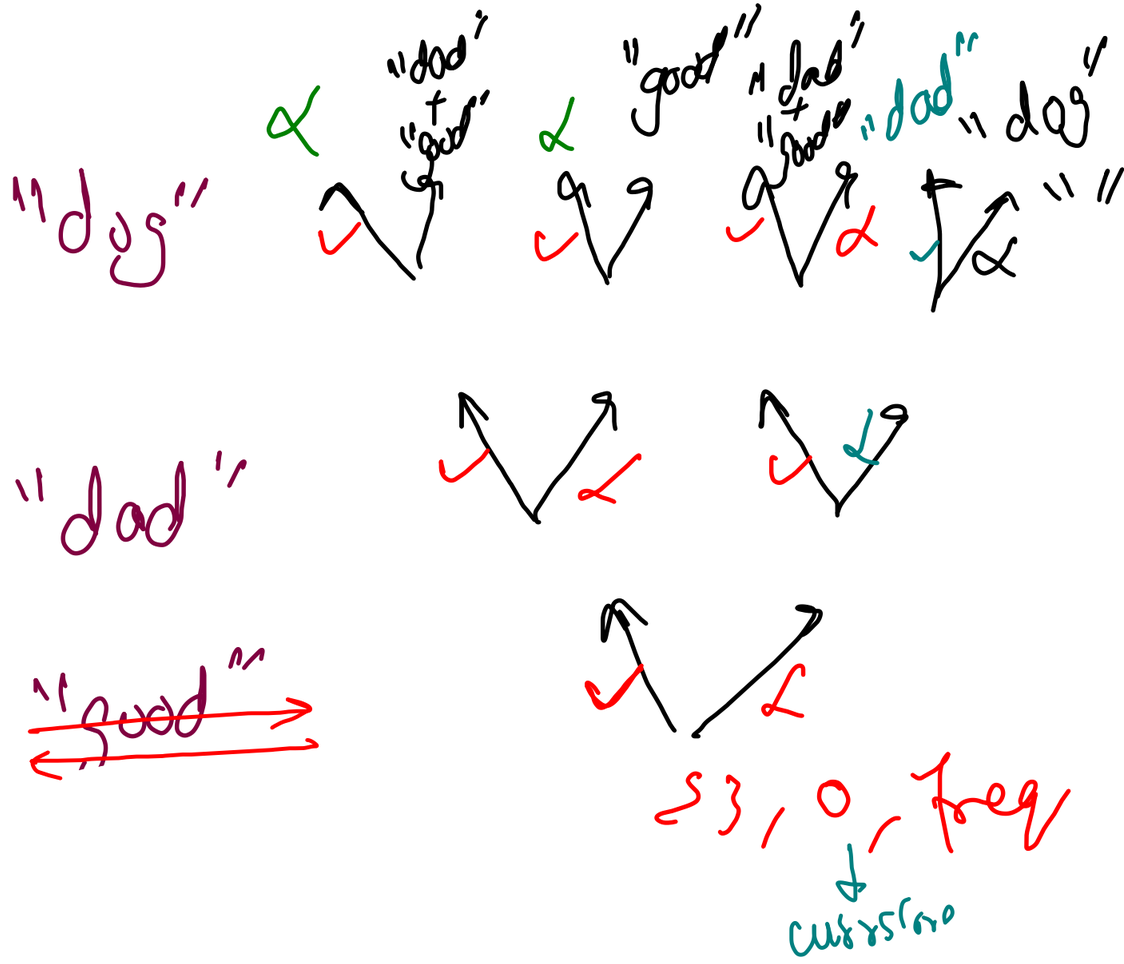
column by column



Max Score

```
Input: words = ["dog","cat","dad","good"], letters =
["a","a","c","d","d","d","g","o","o"], score =
[1,0,9,5,0,0,3,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0]
Output: 23
```





```java
public int maxScore(int idx, String[] words, int[] freq, int[] score){
    if(idx == words.length) return 0;

    // yes
    int currScore = 0;
    boolean flag = true;
    for(char ch: words[idx].toCharArray()){
        freq[ch - 'a']--;
        if(freq[ch - 'a'] < 0){
            flag = false;
        }

        currScore += score[ch - 'a'];
    }

    if(flag == true){
        currScore += maxScore(idx + 1, words, freq, score);
    } else {
        currScore = 0;
    }

    // backtrack
    for(char ch: words[idx].toCharArray()){
        freq[ch - 'a']++;
    }

    // no
    return Math.max(currScore, maxScore(idx + 1, words, freq, score));
}
```

Move Zeroes to End

{ 1, 0, 3, 0, 12, 3, 12 } ⟹ { 1, 3, 12, 0, 0 }

f(0,0) → f(1,0) → f(2,1) → f(3,1) → f(4,2) → f(5,3)

```java
public static void moveZeroes(int idx, int countOfNonZero, int[] nums) {
    if(idx == nums.length){
        int countOfZeros = nums.length - countOfNonZero;
        for(int i=0; i<countOfZeros; i++){
            nums[nums.length - 1 - i] = 0;
        }
        return;
    }

    if(nums[idx] != 0){
        nums[countOfNonZero] = nums[idx];
        moveZeroes(idx + 1, countOfNonZero + 1, nums);
    }
    else {
        moveZeroes(idx + 1, countOfNonZero, nums);
    }
}
```

Today's Questions

1. Crossword
2. Word Search - (I, II, III)
3. Cryptarithmetic
4. Josephus

Cryptarithmetic

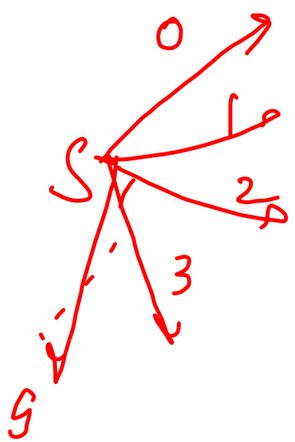
"SEND" + "MORE" = "MONEY"

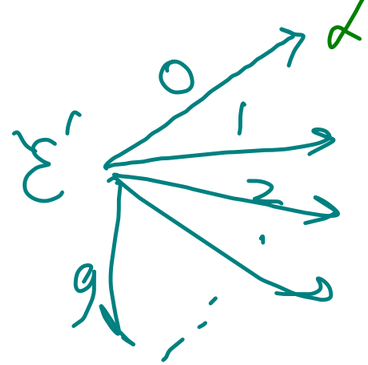
1 : 1 mapping
- char (A-Z)
- digit (0-9)

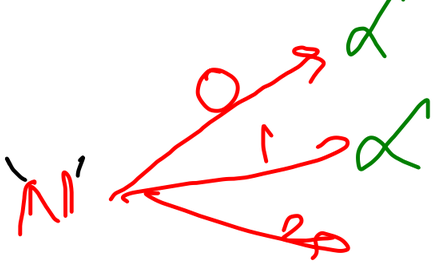
unique char or level

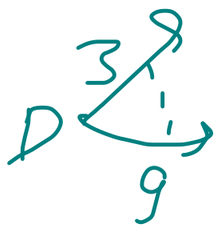
{'S', 'E', 'N', 'D', 'M', 'O', 'R', 'Y'} base case

↳ equivalency check









Josephus

$k = 4$

$n = 8$
0 1 2 3 4 5 6 7 } y
0 1 2 ~~3~~ 4 5 6 7

$n = 7$
0 1 2 3 4 5 6 } x
4 5 6 ~~7~~ 0 1 2

$n = 6$ 0 1 2 ~~4~~ 5 6

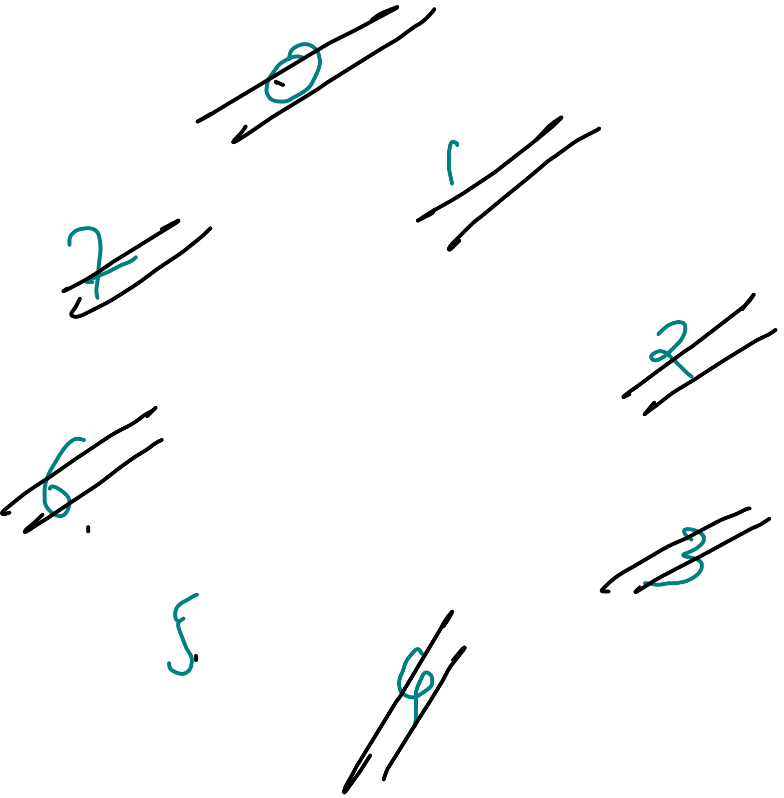
$n = 5$ 5 6 0 ~~1~~ 2

$n = 4$ 2 5 6 ~~0~~

$n = 3$ ~~2~~ 5 6

$n = 2$ 5 ~~6~~

$n = 1$ 5 (0)



{0 1 2 3 4 5 6} x
{4 5 6 7 0 1 2} y

$n = 8$
$k = 4$

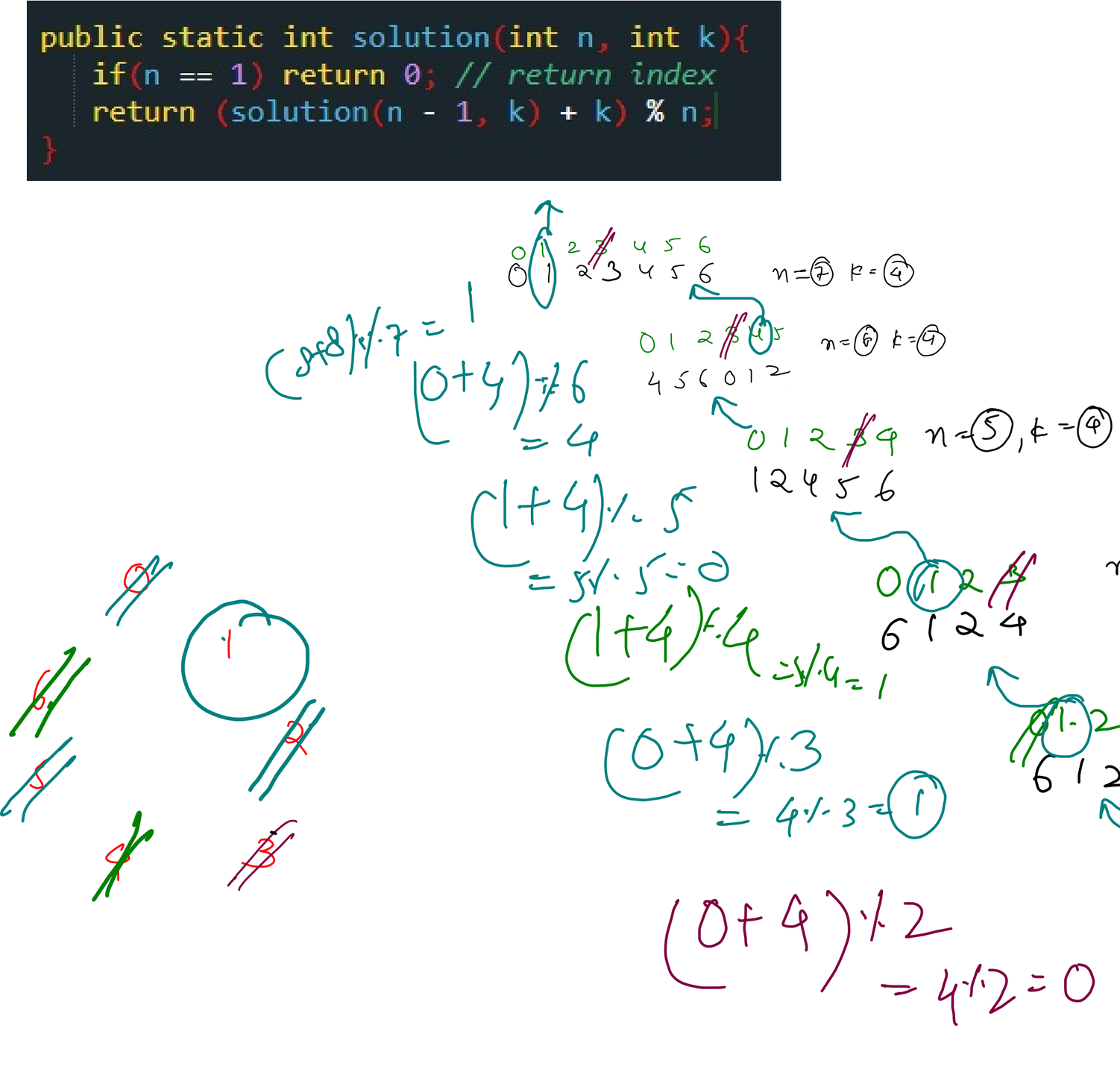
actual value → y = $(x + k) \% n$ ← index

Meeting Exp

$(\text{josephus}(n-1, k) + k) \% n$

Shifting on k
rotation on x

```java
public static int solution(int n, int k){
    if(n == 1) return 0; // return index
    return (solution(n - 1, k) + k) % n;
}
```



Tc ⇒ O(n)

Sc ⇒ O(1) extra space

O(n) Recursion

```java
public static int solution(int n, int k){
    Queue<Integer> q = new ArrayDeque<>();
    for(int i=0; i<n; i++){
        q.add(i);
    }
    int count = 0;
    while(q.size() > 1){
        int val = q.remove();
        count++;

        if(count == k){
            count = 0;
        } else {
            q.add(val);
        }
    }
    return q.peek();
}
```

Constructive algorithm

Queue { brute force

O(n*k) ⇒ Time

O(n) ⇒ Queue space

Lexicographical Nb's

N = 999

1
10 100 101 102 103 104 --- 109
11 110 111 112 113 ---- 119
12 120 121 122 ---- 129
⋮
19 191 192 - - - - 199

2
20 200 201 202 --- 209
21 210 211 ----- 219
22 220 221 --- 229
23 230 231 ---- 239
⋮
29 290 291 --- 299

3
30 → 301 --- 309
31 → 310 --- 319
32 -
⋮
39 → 390 --- 399

4
40, 41, 42, --- 49

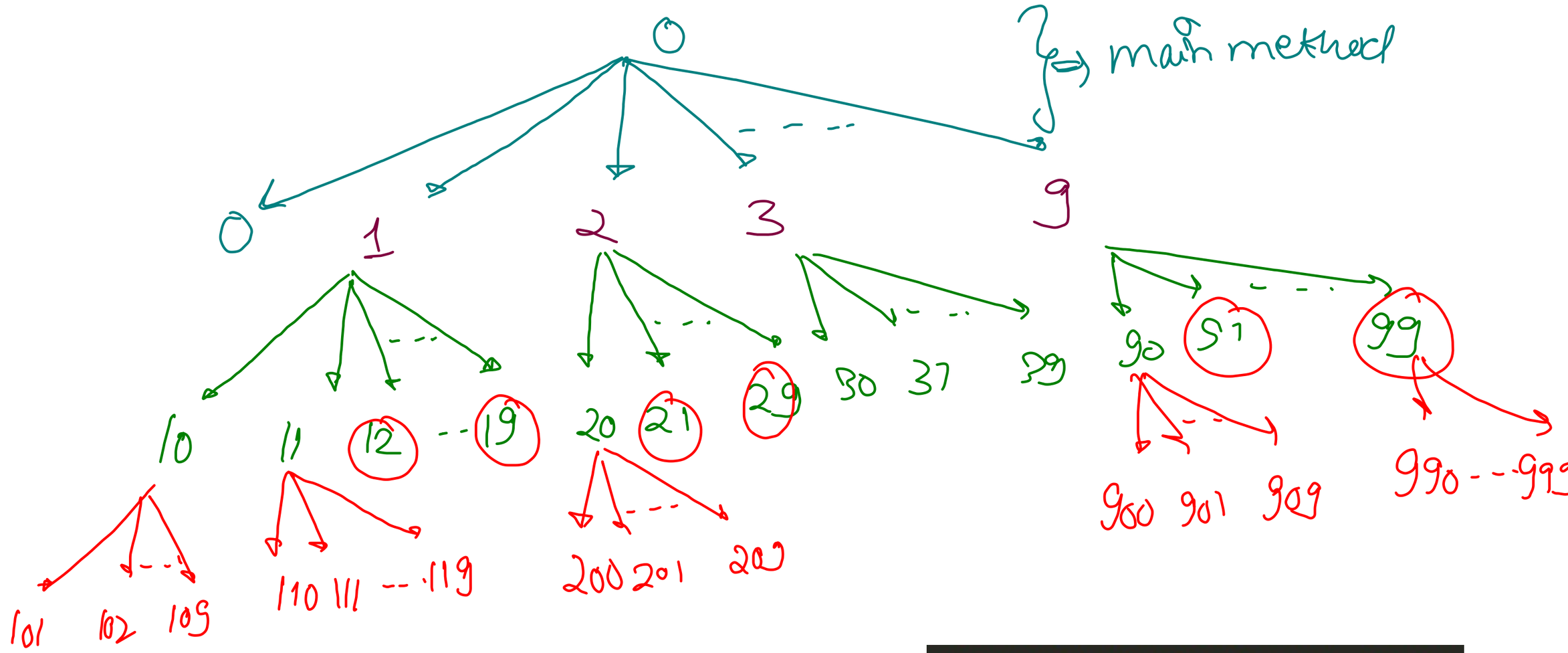
5
50, --- 59

6

7

8

9
90 91 92 --- 99
900 901 --- 909
990-999




```java
public void lexicalOrder(int input, int n){
    if(input > n) return;

    res.add(input);
    for(int i=0; i<10; i++){
        lexicalOrder(input * 10 + i, n);
    }
}
```

R&B - level 2 - lecture 10

1. Crossword
2. Word Search - II

   Word Search - III {Homework}
3. Friends Pairings
4. Largest No in K swaps
5. Restore IP Address

More questions to practice

Paranthesis Addition Ways
Stepping Numbers
Kth Symbol in Grammar
Split into Fibonacci Sequence
Distinct Subsets Permutations

N knights

Crossword

```
++++++++++
+------+++
+++-++++++
+++-++++++
+++-----++
+++-++-+++
++++++-+++
++++++-+++
++++++-+++
++++++++++
```