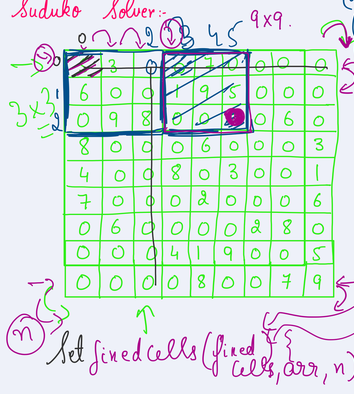
Lecture 13

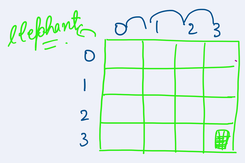
Friday, 1 May 2020 6:43 PM

Recursion :- 4.

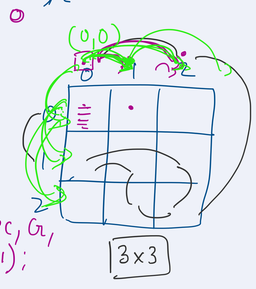
Suduko Solver :-

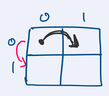


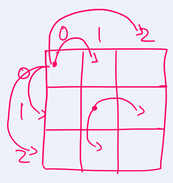
```
public static void main(String[] args){
    int[][] arr = {{ }};
    int n = 9;
    boolean [][] fixedCells = new boolean[n][n];
    setFixedCells(arr, fixedCells, n);
    boolean ans = SudukoSolver(arr, fixedCells, 0, 0, 9);
    if(ans){
        display(arr);
    } else{
    }
}

setFixedCells(fixedCells, arr, n){
    for (int i=0; i<n; i++){
        for (int j=0; j<n; j++){
            if(arr[i][j] != 0){
                fixedCell[i][j] = true;
            }
        }
    }
}

SudukoSolver(arr, fixedCells, int row, int col, n);
    if(row == n){
        return true;
    }
    if(col == n){
        return Su_( , row+1, 0, n);
    }
    if(fixedCells[row][col]){
        return SudukoSolver(arr, fixedCells, row, col+1, n);
    }
    for (int num=1; num<=9; num++){
        boolean canPlace = canPlace(board, row, col, n, num);
        if(canPlace){
            arr[row][col] = num;
            boolean Solverest = SudukoSolver(arr, fixedCells, row, col+1, n);
            if(Solverest){
                return true;
            } else{
                arr[row][col] = 0;
            }
        }
    }
    return false;

canPlace(board, row, col, n, num){
    for (int i=0; i<board.length; i++){
        if(board[i][col] == num){
            return false;
        }
        if(board[row][i] == num){
            return false;
        }
    }
    int sqrtN = Math.Sqrt(n);
    int startrow = (row/sqrtN) * sqrtN
    int startCol = (col/sqrtN) * sqrtN;
    for (int i = startrow; i < startrow + sqrtN; i++){
        for (int j = startCol; j < startCol + sqrtN; j++){
            if(b[i][j] == num){
}
```



```
public static void main(String[] args){
    Sysо(elephantways(2, 2, 0, 0);
}

public static int elephantways(int er, int ec, int cr, int cc){
    if(er == cr && ec == cc){
        return 1;
    }
    if(cr > er || cc > ec){
        return 0;
    }
    int Count = 0;
    for (int i = cc; i < ec; i++){
        Count = Count + elep_ways(er, ec, cr, i+1);
    }
    for (int i = cr; i < er; i++){
        Count = Count + elep_ways(er, ec, i+1, cc);
    }
    return Count;
}
```

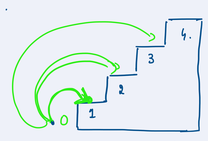






```
for (int i = cr; i < er; i++){
    Count = Count + ways(er, ec, cr, i+1);
}

for (int i = cr; i < er; i++){
    Count = Count + ways(er, ec, i+1, cc);
}
```

Q:- n-stairs.



total ways by which you can reach n?

n = 4.

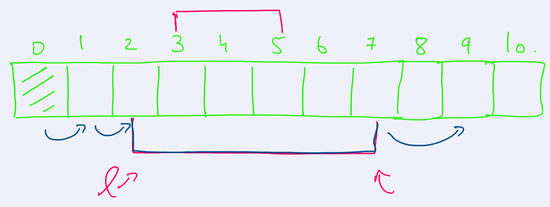
1, 2, 3, 4 → 1
1, 3, 4 → 2
1, 4 → 3
1, 2, 4 → 4

3, 4.
1,

```
public static int nstairs(int n){
    if(n == 0){
        return 1;
    }
    if(n < 0){
        return 0;
    }
    ans = nstairs(n-1) + nstairs(n-2)
        + nstairs(n-3);
}
```

Resume at 8:50 pm.

Q:-

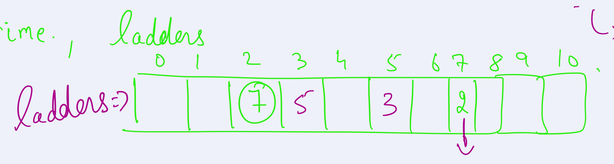


0, 1, 2, 7, END.

0, 1, 3, 5, 8, END.

prime.
↳ make ladders
↳ dp

prime, ladders



• Recursion & Backtracking
↳ Binary Search
↳ MergeSort
↳ QuickSort
↳ Tomorrow

Tomorrow.