PW SKILLS

Dynamic Programming

Checklist

1. Memoization & Tabulation
2. 1D DP Problems
3. 2D DP Grid Problems
4. Knapsack & Unbounded Knapsack Problems
5. DP on Strings
6. MCM problems and DP on Trees

What is Dynamic Programming?

PW SKILLS

misleading name

Advanced / Optimised Recursion

Prerequisites:

1) Recursion → Basics, Understanding, multiple calls
2) Arrays, 2D arrays → Basics
3) Hashmaps & Trees → Basics

Ques:

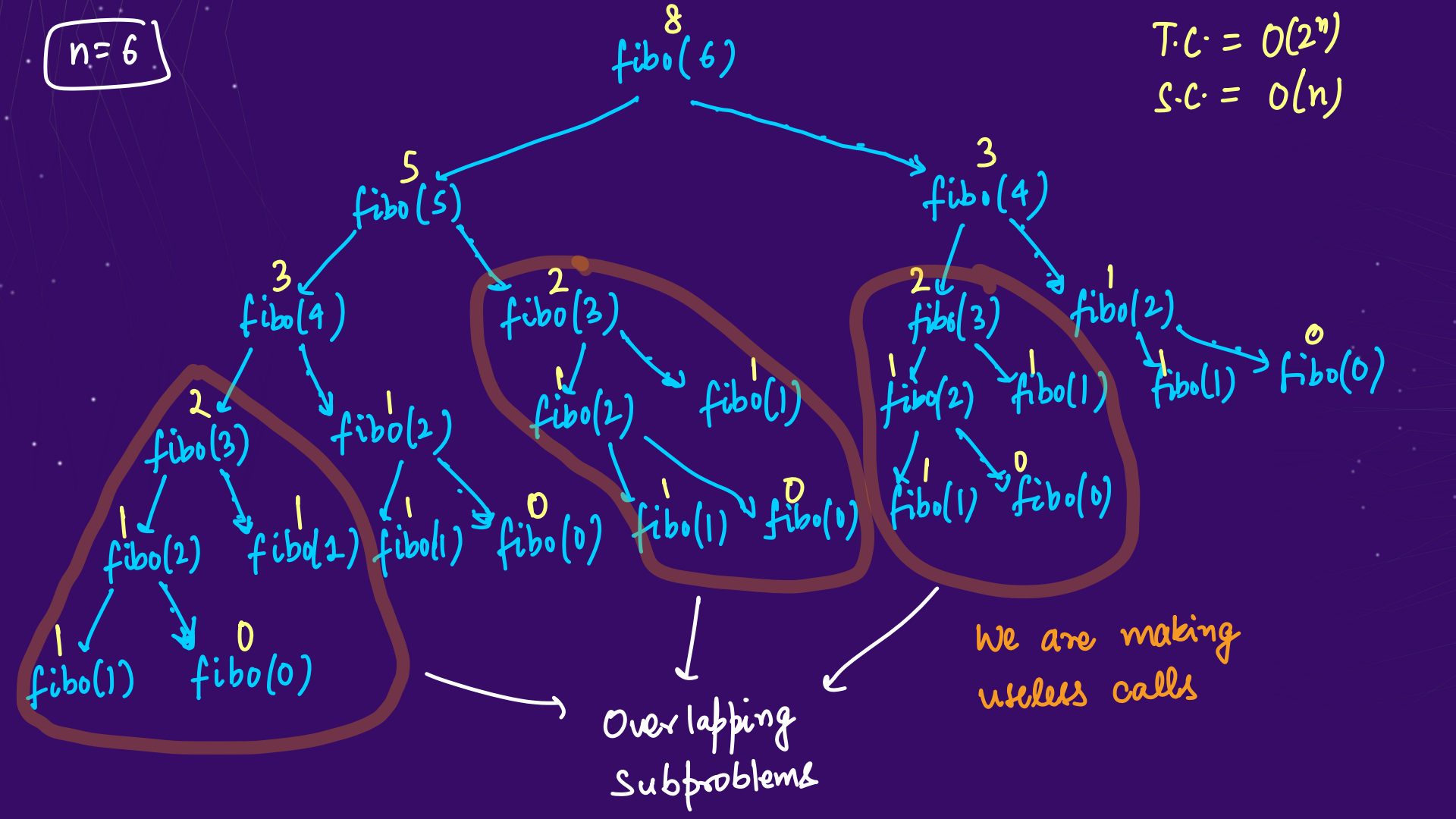
PW SKILLS

Q : Fibonacci Number (Recursion)

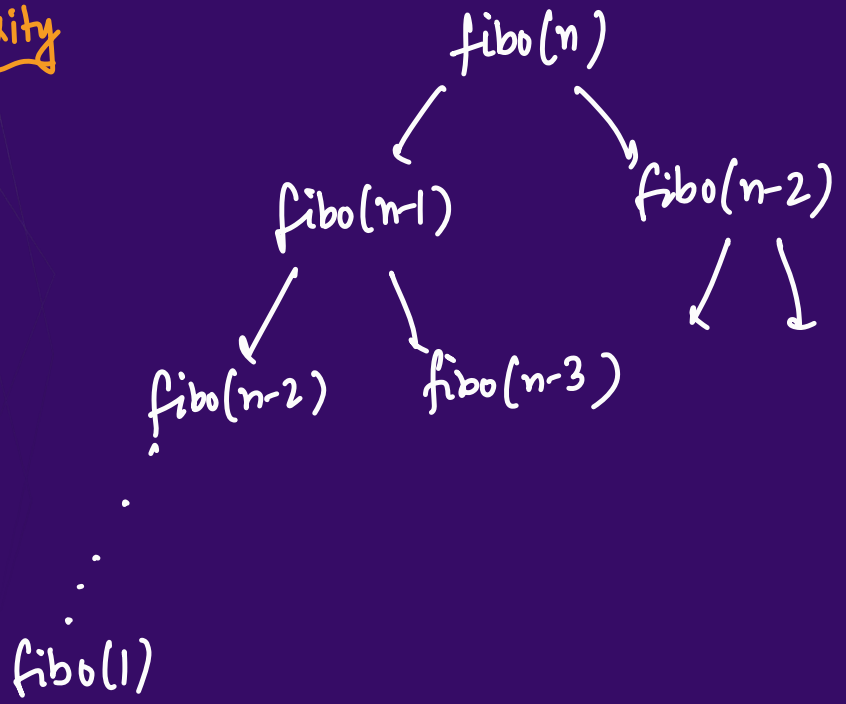
	0	1	1	2	3	5	8	13	21	34	55	89 . . .
n =	0	1	2	3	4	5	6	7	8	9	10	11

$$fibo(n) = fibo(n-1) + fibo(n-2)$$

[Leetcode 509]


n = 6
T.C. = O(2^n)
S.C. = O(n)
8
fibo(6)
5
fibo(5)
3
fibo(4)
2
fibo(3)
1
fibo(2)
0
fibo(0)
1
fibo(1)
2
fibo(3)
1
fibo(2)
1
fibo(1)
0
fibo(0)
1
fibo(1)
0
fibo(0)
3
fibo(4)
2
fibo(3)
1
fibo(2)
1
fibo(1)
0
fibo(0)
1
fibo(1)
1
fibo(2)
1
fibo(1)
0
fibo(0)
Overlapping Subproblems
We are making useless calls

Time Complexity Discussion



For Calculating T.C. count the total no. of calls

$1 + 2 + 4 + 8 \ldots\ldots 2^{n-1}$

$\simeq 2^n$

It is a balanced binary tree with 'n' levels

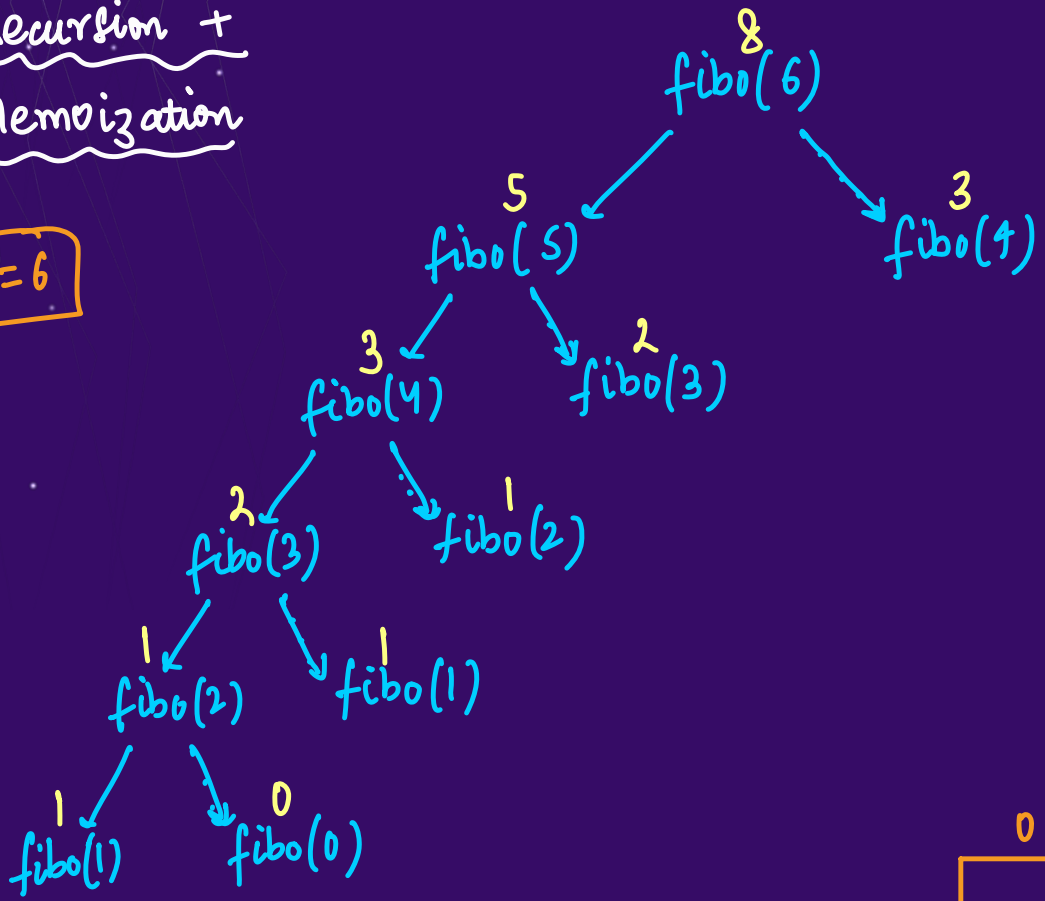
So, no. of nodes / calls are approximately 2^n

$O(\log n) > O(n) > O(n \log n) > O(n^2) > O(n^2 \log n) > O(n^3) >> O(2^n)$

very slow

Recursion + Memoization

n = 6

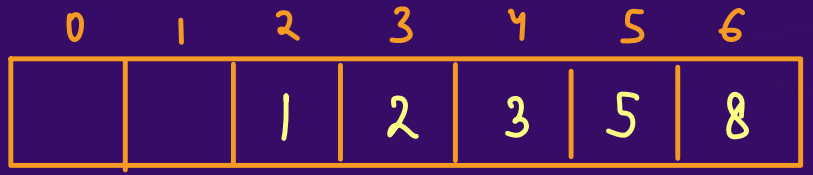


Time Complexity :

Total Calls → 2n - 1

T.C. = O(n)

S.C. = O(n)



Ques:

Q : Fibonacci Number (Recursion + Memoization)

```
public int fibo(int n, int[] dp) {
    if(n<=1) return n;
    if(dp[n]!=0) return dp[n];
    return dp[n] = fibo(n-1,dp) + fibo(n-2,dp);
}
public int fib(int n) {
    int[] dp = new int[n+1]; // index from 0 to n
    return fibo(n,dp);
}
```

[Leetcode 509]

Recursion + Memoization = Top-Down DP = Recursive DP

Ques:

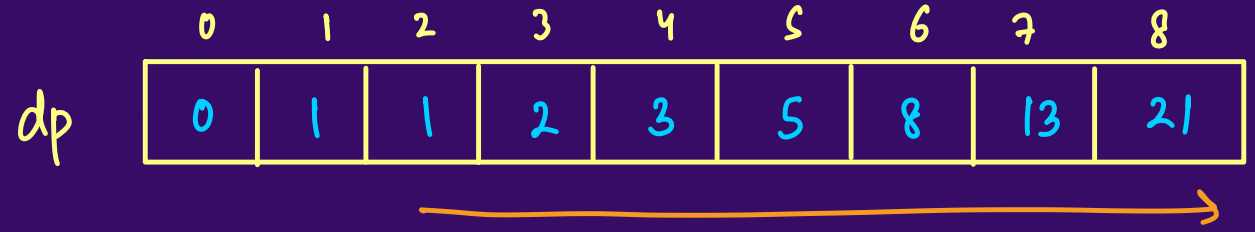
Bottom-Up DP = Iterative DP

PW SKILLS

Q : Fibonacci Number (Tabulation)

T.C. = O(n)
S.C. = O(n)

n = 8



$$dp[i] = dp[i-1] + dp[i-2]$$

DP → Using previous results to compute new result

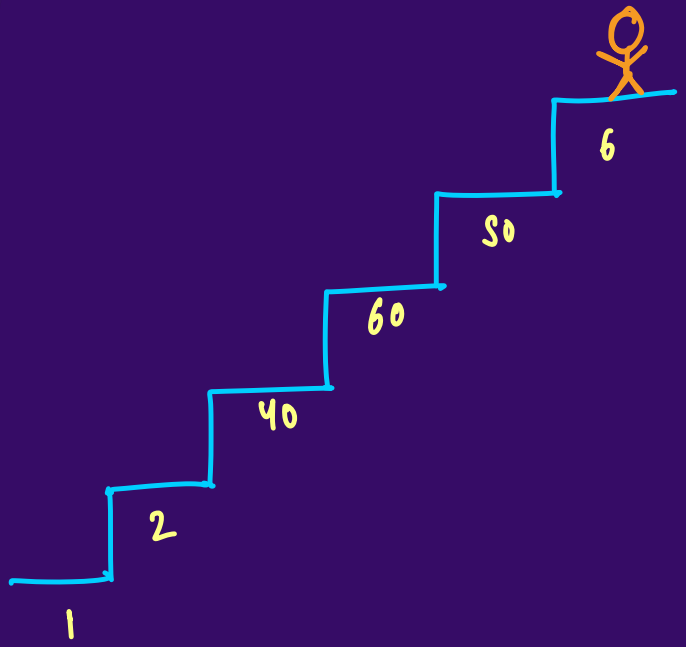
[Leetcode 509]

Ques: $minCost(i) = arr[i] + min(minCost(i-1), minCost(i-2))$

PW SKILLS

Q : Min Cost Climbing Stairs

	0	1	2	3	4	5
cost = {	1,	2,	3,	4,	50,	6 }



greedy (cost) = 6 + 50 + 40 + 1

= 97 ✗

minCost = 60 + 2

= 62 ✓

[Leetcode 746]

Ques:

Q : Min Cost Climbing Stairs

Recursion → TLE

```
public int minCost(int[] cost, int idx) {
    if(idx==0 || idx==1) return cost[idx];
    return cost[idx] + Math.min(minCost(cost,idx-1),minCost(cost,idx-2));
}
public int minCostClimbingStairs(int[] cost) {
    int n = cost.length;
    return Math.min(minCost(cost,n-1),minCost(cost,n-2));
}
```

T.C. = $O(2^n)$

A.S. = $O(n)$ → Recursive Stack Space

[Leetcode 746]

Ques:

Q : Min Cost Climbing Stairs

Memoization

```
public int minCost(int[] cost, int idx, int[] dp) {
    if(idx==0 || idx==1) return cost[idx];
    if(dp[idx]!=-1) return dp[idx];
    return dp[idx] = cost[idx] + Math.min(minCost(cost,idx-1,dp),minCost(cost,idx-2,dp));
}
public int minCostClimbingStairs(int[] cost) {
    int n = cost.length;
    // n is going from n-1 to 0
    int[] dp = new int[n];
    Arrays.fill(dp,-1);
    return Math.min(minCost(cost,n-1,dp),minCost(cost,n-2,dp));
}
```

T.C. = O(n)

A.S. = O(n)

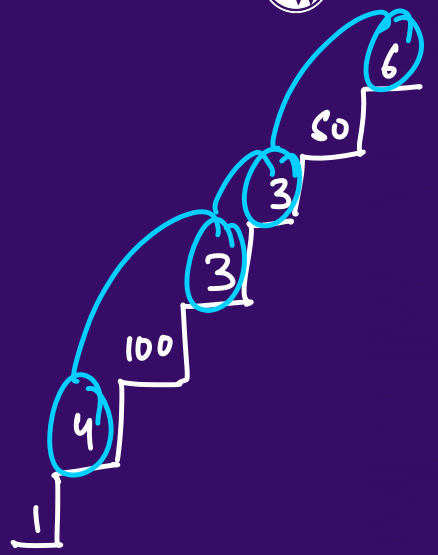
[Leetcode 746]

Ques:

Q : Min Cost Climbing Stairs (Tabulation)

$cost = \{ 1, 4, 100, 3, 3, 50, 6 \}$

$dp = \{ 1, 4, 101, 7, 10, 57, 16 \}$



$$dp[i] = cost[i] + \min(dp[i-1], dp[i-2])$$

[Leetcode 746]

Ques:

Q : Min Cost Climbing Stairs (Tabulation)

```java
public int minCostClimbingStairs(int[] cost) {
    int n = cost.length;
    int[] dp = new int[n];
    dp[0] = cost[0]; dp[1] = cost[1];
    for(int i=2;i<n;i++){
        dp[i] = cost[i] + Math.min(dp[i-2],dp[i-1]);
    }
    return Math.min(dp[n-2],dp[n-1]);
}
```

T.C. = O(n)

A.S. = O(n)

[Leetcode 746]

Homework:

PW SKILLS

Q : Nth Tribonacci Number

1) Recursion

2) Recursion + Memoization

3) Tabulation

[Leetcode 1137]

Ques:

Q : House Robber → Max^m amount that can be stolen

arr = 5 15 6 20 22 1

take & skip algorithm

Greedy Approaches: 5 + 6 + 22 = 33 ✗

15 + 20 + 1 = 36

Max Amount : 15 + 22 = 37 ✓

[Leetcode 198]

Ques:

Q : House Robber (Recursion + Memoization)

```java
public int amount(int[] nums, int i, int[] dp) {
    if(i>=nums.length) return 0;
    if(dp[i]!=-1) return dp[i];
    int take = nums[i] + amount(nums,i+2,dp);
    int skip = amount(nums,i+1,dp);
    return dp[i] = Math.max(take,skip);
}
public int rob(int[] nums) {
    // 'i' varies from 0 to n-1
    // dp[i] will store the value of amount(i)
    int[] dp = new int[nums.length]; Arrays.fill(dp,-1);
    return amount(nums,0,dp);
}
```

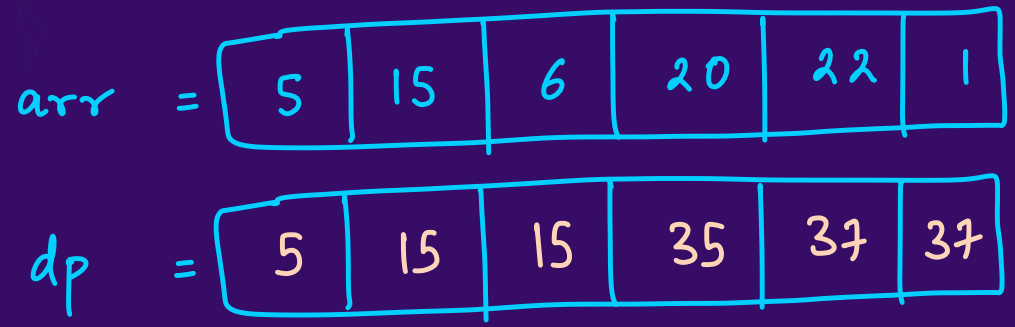
[Leetcode 198]

Ques:

Q : House Robber (Tabulation)

$dp[0] = arr[0]$

$dp[1] = \max(arr[0], arr[1])$



$$dp[i] = \max(arr[i] + dp[i-2],\ dp[i-1])$$

take skip

dp[i] stores the max^m money that the robber can rob from 0th to ith house

[Leetcode 198]

Ques:

Q : House Robber (Tabulation) (Bottom-Up DP)

```java
public int rob(int[] arr) {
    int n = arr.length;
    int[] dp = new int[n];
    dp[0] = arr[0];
    if(n>1) dp[1] = Math.max(arr[0],arr[1]);
    for(int i=2;i<n;i++){
        dp[i] = Math.max(arr[i]+dp[i-2], dp[i-1]);
    }
    return dp[n-1];
}
```

[Leetcode 198]

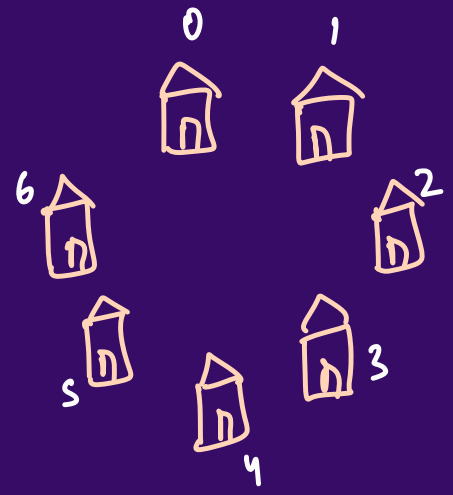
Homework:

PW SKILLS

Q : House Robber II

1

Street is Circular



[Leetcode 213]

Ques:

PW SKILLS

Q : Friends Pairing Problem

You are given a number 'n'. Denoting that there are 'n' friends Find out the no. of ways in which a person can either pair up or stay single.

Ex : a , b , c, d

⇒ ab , cd ⇒ ac , bd ⇒ ad , bc

⇒ ab, c, d ⇒ ac, b, d ⇒ ad, b, c

⇒ cd, a, b ⇒ bd, a, c ⇒ bc, a,d

⇒ a, b, c, d

Ques:

recursive formula

Q : Friends Pairing Problem

n = 3

a, b, c

a, b, c

ab, c

ac, b

bc, a

n = 4

a, b, c, d

a (b, c, d)
- a, b, c, d
- a, bc, d
- a, bd, c
- a, cd, b

ab (cd)
- ab, cd
- ab, c, d

ac (db)
- ac, bd
- ac, b, d

ad (bc)
- ad, bc
- ad, b, c

$$pair(n) = pair(n-1) + (n-1)^4 \, pair(n-2)$$

Ques:

Q : Friends Pairing Problem

generate all permutations

ab ≠ ba

$$pairing(n) = n * [pairing(n-1) + (n-1) * pairing(n-2)]$$

single

pair

nahi hoga

$$pair(4) = 4 * [pair(3) + 3 * pair(2)]$$

$$= 4[4+6] = 4[10] = 40$$ wrong

Ques:

Q : Friends Pairing Problem (Recursion + Memoization)

```java
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int[] dp = new int[n+1];
    Arrays.fill(dp,-1);
    System.out.println(pair(n,dp));
}

private static int pair(int n, int[] dp) {
    if(n<=2) return n;
    if(dp[n]!=-1) return dp[n];
    return dp[n] = pair(n-1,dp) + (n-1)*pair(n-2,dp);
}
```

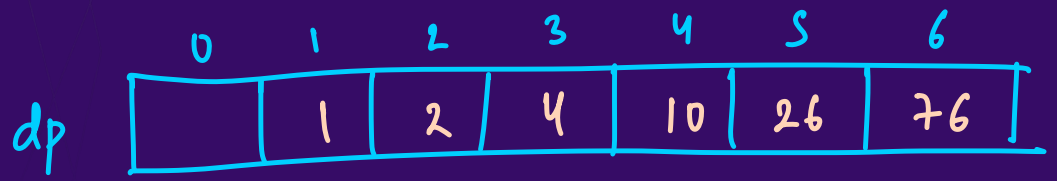
T.C. = O(n)

S.C. = O(n)

Ques:

Q : Friends Pairing Problem (Tabulation)

n = 6




```
private static int friend(int n) {
    int[] dp = new int[n+1];
    dp[1] = 1;
    if(n>1) dp[2] = 2;
    for(int i=3;i<=n;i++){
        dp[i] = dp[i-1] + (i-1)*dp[i-2];
    }
    return dp[n];
}
```

$pair(n) = pair(n-1) + (n-1)^* pair(n-2)$

$dp[i] = dp[i-1] + (i-1)^* dp[i-2]$

T.C. = O(n)

A.S. = O(n)

Homework:

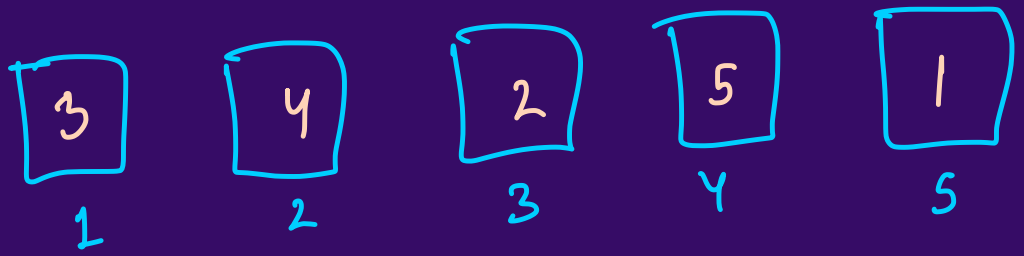
$$n!\left(\frac{1}{n!}-\frac{1}{(n-1)!}+\frac{1}{(n-2)!}\cdots\right)$$

Q : Count Derangements

$n = 5$

there are 'n' postcards & 'n' houses

The postman delivers postcards to every house such that each person does not get his/her postcard → No. of ways



Homework:

Q : Count Derangements

$\frac{\alpha}{1}$

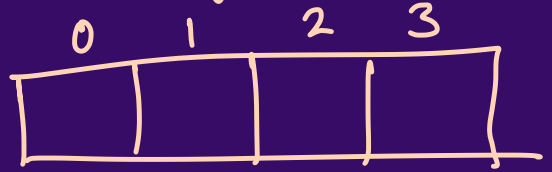
$n=1 \rightarrow 0$

$\frac{3}{1}$ $\frac{1}{2}$ $\frac{2}{3}$

$\frac{2}{1}$ $\frac{1}{2}$ $n=2 \rightarrow 1$

$\frac{2}{1}$ $\frac{3}{2}$ $\frac{1}{3}$

Hint → array



$n=4$

$n=3 \rightarrow 2$

Homework:

PW SKILLS

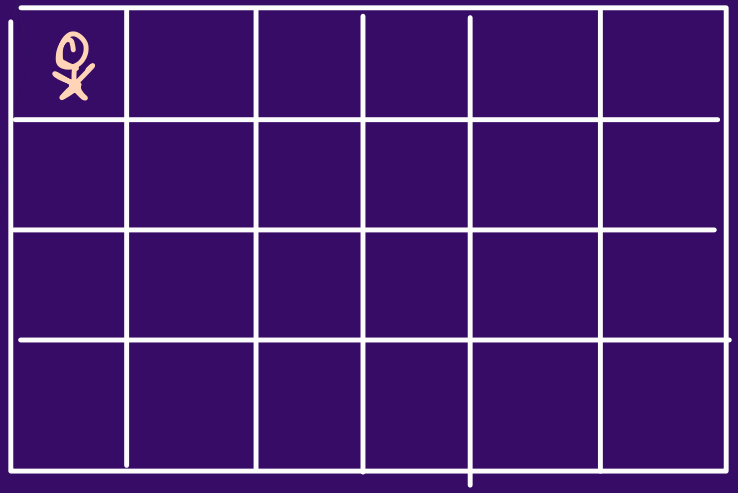
Q: Climbing Stairs

- Recursion
- Recursion + Memoization
- Tabulation

[leetcode - 70]

Ques:

Q : Unique Paths → m & n where there is a m*n grid



4×6

the person has to reach (m-1, n-1) from (0, 0) only using right & down at a time.

$$ways(m,n) = ways(m-1,n) + ways(m,n-1)$$

down right

[Leetcode 62]

Ques:

Q : Unique Paths

```java
public int uniquePaths(int m, int n) {
    // row : 0 to m-1 | col : 0 to n-1
    int[][] dp = new int[m][n];
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            dp[i][j] = -1;
        }
    }
    return paths(0,0,m,n,dp);
}

private int paths(int row, int col, Integer m, Integer n, int[][] dp) {
    if(row>=m || col>=n) return 0;
    if(row==m-1 && col==n-1) return 1;
    if(dp[row][col]!=-1) return dp[row][col];
    int rightWays = paths(row,col+1,m,n,dp);
    int downWays = paths(row+1,col,m,n,dp);
    return dp[row][col] = rightWays + downWays;
}
```

$T.C. = O(m*n)$

$A.S. = O(m*n)$

[Leetcode 62]

Ques:

Q : Unique Paths (Tabulation)

Right or Down

dp

	1	1	1	1	1
1	2	3	4	5	6
1	3	6	10	15	21
1	4	10	20	35	56

4X6

$$dp[i][j] = dp[i][j-1] + dp[i-1][j]$$

[Leetcode 62]

Ques:

Q : Unique Paths (Tabulation)

```
public int uniquePaths(int m, int n) {
    int[][] dp = new int[m][n];
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            if(i==0 || j==0) dp[i][j] = 1;
            else dp[i][j] = dp[i][j-1] + dp[i-1][j];
        }
    }
    return dp[m-1][n-1];
}
```

T.C. = O(m*n)

S.C. = O(m*n)

[Leetcode 62]

Ques:

Prefix Sum

SKILLS

Q : Minimum Path Sum (Tabulation)

2	3	9	4
8	4	1	2
2	1	3	3

arr

2	5	14	18
10	9	10	12
12	10	13	15

dp

$$dp[i][j] = arr[i][j] + \min(dp[i-1][j], dp[i][j-1])$$

[Leetcode 64]

Homework:

PW SKILLS

Q : Minimum Path Sum (Recursion + Memoization)

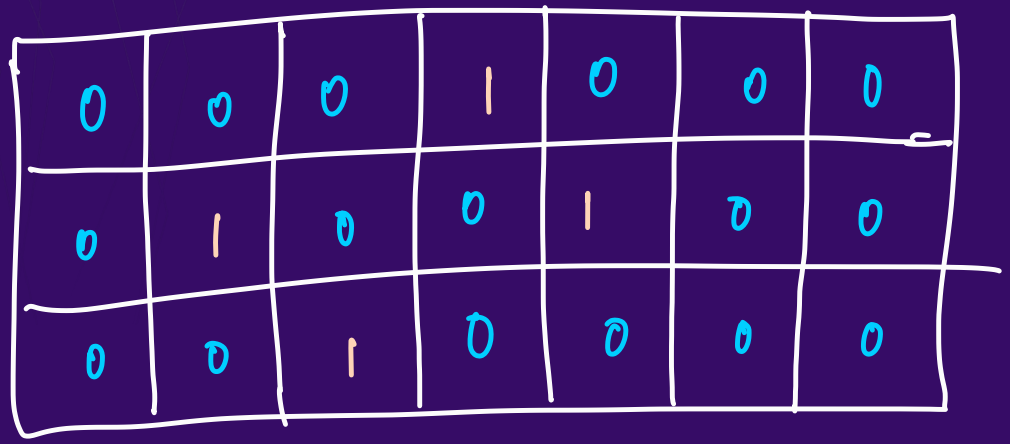
Top-Down DP

[Leetcode 64]

Homework:

PW SKILLS

Q: Unique Paths II



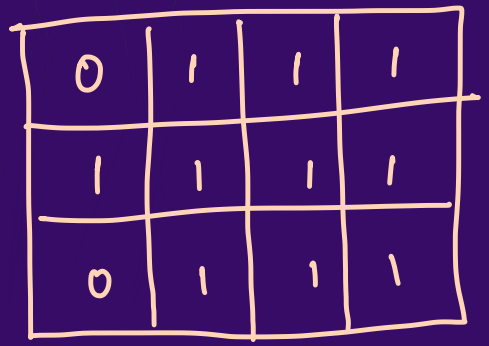
Hint ⇒ Ps → -1

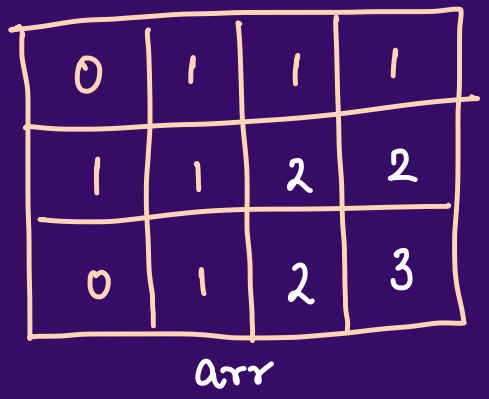
[Leetcode 63]

Ques:

PW SKILLS

Q : Count Square Submatrices with all Ones (Tabulation)



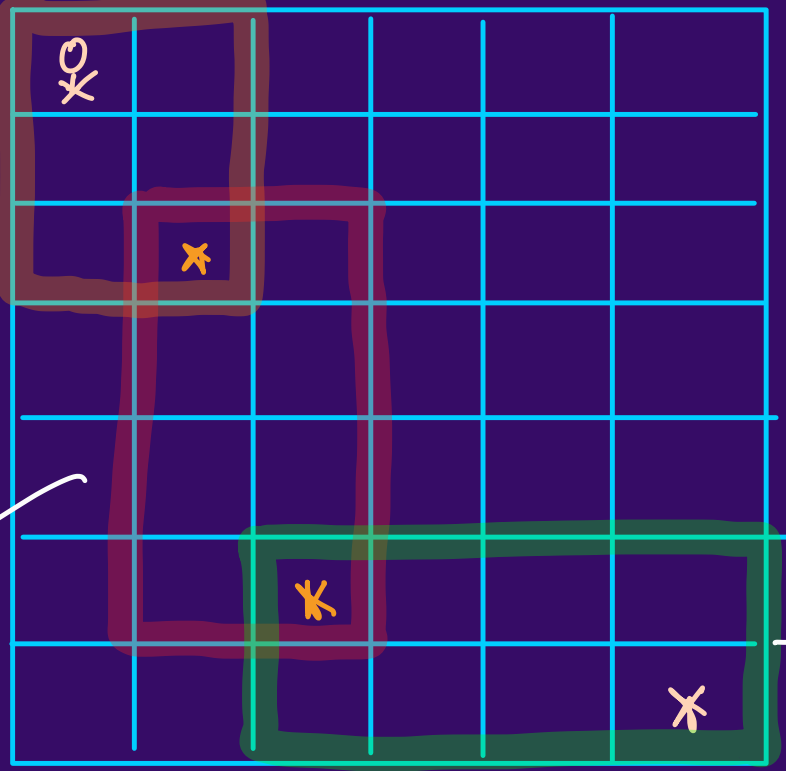


Count = sum of this matrix

arr[i][j] = the side of square whose bottom right corner is i,j

[Leetcode 1277]

Paths which passes through certain checkpoints



Right & Down

No. of paths = $a * b * c$

Space Optimisation

PW SKILLS

Abhi tak,

1D DP → 'n' size array (dp) Space O(n) → O(1)

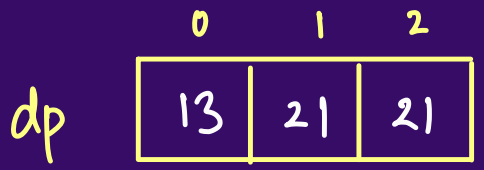
2D DP → 'm*n' size 2D array (dp) Space O(m*n) → O(m) / O(n)

Ques:

PW SKILLS

Q : Fibonacci Number (Tabulation) (Space Optimization)

n = 8 (21)



$dp[2] = dp[0] + dp[1]$

$dp[0] = dp[1]$

$dp[1] = dp[2]$

T.C. = O(n)

S.C. = O(1)

[Leetcode 509]

Ques:

Q : Fibonacci Number (Tabulation)

[Leetcode 509]

Ques:

Q : Unique Paths (Tabulation)

🚶	1	1	1	1	1
1	2	3	4	5	6
1	3	6	10	15	21
1	4	10	20	35	56
1	5	15	35	70	126

dp

5 x 6
m x n

0	1	5	15	35	70	126
1	1	4	10	20	35	56

2 x n

m-1 times

loop

$$dp[i][j] = dp[i][j-1] + dp[0][j]$$

[Leetcode 62]

Ques:

Q : Unique Paths

```java
public int uniquePaths(int m, int n) {
    int[][] dp = new int[2][n];
    for(int j=0;j<n;j++){
        dp[0][j] = 1;
        dp[1][j] = 1;
    }
    for(int i=1;i<=m-1;i++){ // m-1 times
        // DP wala kaam
        for(int j=1;j<n;j++){
            dp[1][j] = dp[1][j-1] + dp[0][j];
        }
        // copy the 1st row to 0th row
        for(int j=1;j<n;j++){
            dp[0][j] = dp[1][j];
        }
    }
    return dp[1][n-1];
}
```

2nd Best

Best

```java
public int uniquePaths(int m, int n) {
        int[][] dp = new int[2][n];
        for(int j=0;j<n;j++){
            dp[0][j] = 1;
            dp[1][j] = 1;
        }
        for(int i=1;i<=m-1;i++){ // m-1 times
            if(i%2==1){
                for(int j=1;j<n;j++){
                    dp[1][j] = dp[1][j-1] + dp[0][j];
                }
            }
            else{
                for(int j=1;j<n;j++){
                    dp[0][j] = dp[0][j-1] + dp[1][j];
                }
            }
        }
        return Math.max(dp[1][n-1],dp[0][n-1]);
    }
```

[Leetcode 62]

Ques:

Q : Perfect Squares

$n = a + b + c \cdots$

$\downarrow \quad \downarrow \quad \downarrow$

perfect squares

ans = min count of these

n	
$3 = 1+1+1$	3
$4 = 4$	1
$5 = 1+4$	2
$6 = 1+1+4$	3
$7 = 1+1+1+4$	4
$8 = 4+4$	2
$9 = 9$	1

PW SKILLS

[Leetcode 279]

Ques:

11 = 9 + 1 + 1

Q : Perfect Squares

12 → 1 + 11 → 1 + 3 = 4

2 + 10

3 + 9

4 + 8

5 + 7

6 + 6

14 → 3 → 9

numSquares(n) = numSquares(1) + numSquares(n-1)

numSq(2) + numSq(n-2)

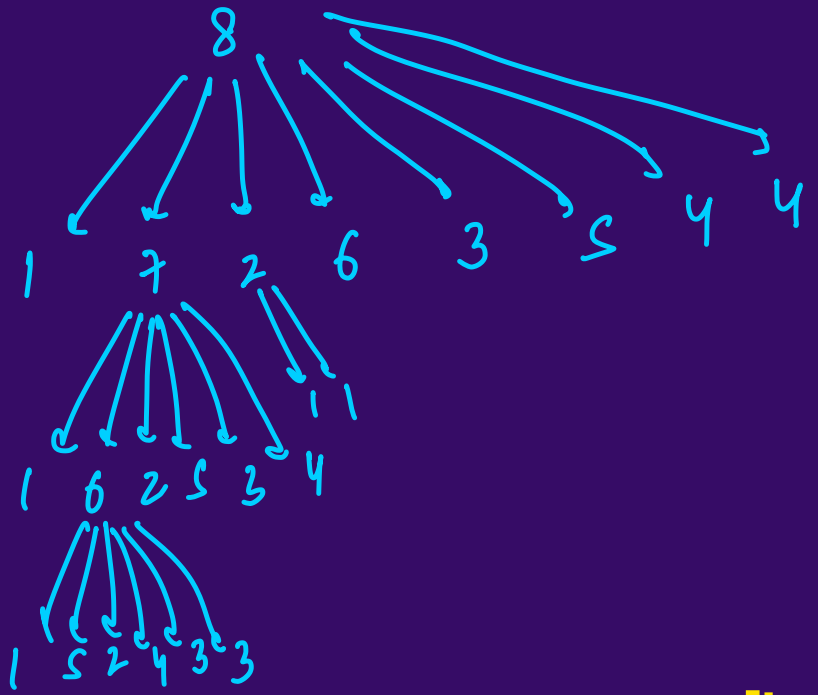
[Leetcode 279]

Ques:

Q : Perfect Squares

$n = 8$

$T.C. <\cong O(n^n)$



[Leetcode 279]

Ques:

$i < \sqrt{n} \rightarrow i^2 < n$

Q : Perfect Squares (Memoization)

```
public int minSquares(int n, int[] dp) {
    if(isPerfect(n)) return 1;
    if(dp[n]!=-1) return dp[n];
    int min = Integer.MAX_VALUE;
    for(int i=1;i<=n/2;i++){
        int count = minSquares(i,dp) + minSquares(n-i,dp);
        min = Math.min(min,count);
    }
    return dp[n] = min;
}
```

T.C. = $O(n^2)$

↓

TLE Error

S.C. = $O(n)$

'n' calls

but in every call → 'n/2' times the for loop

[Leetcode 279]

Ques:

Q : Perfect Squares

$n = 12 \rightarrow$

$1 + 11$

$2 + 10$

$3 + 9$

$4 + 8$

$5 + 7$

$6 + 6$

$minS(n) = minS(i) + minS(n-i)$

$1, 2, 3$

$n = 12 \rightarrow 1 + 11$

$2^2 + 12 - 2^2$ or $4 + 8$

$3^2 + 12 - 3^2$ or $9 + 3$

$minS(n) = minS(i^2) + minS(n-i^2)$

[Leetcode 279]

Ques:

Q : Perfect Squares

```java
public int minSquares(int n, int[] dp) {
    if(isPerfect(n)) return 1;
    if(dp[n]!=-1) return dp[n];
    int min = Integer.MAX_VALUE;
    for(int i=1;i*i<=n;i++){
        int count = minSquares(i*i,dp) + minSquares(n-i*i,dp);
        min = Math.min(min,count);
    }
    return dp[n] = min;
}
```

T.C. = $O(n\sqrt{n})$

S.C. = $O(n)$

[Leetcode 279]

Ques:

Q : Perfect Squares (Tabulation)

```java
public int numSquares(int n) {
    int[] dp = new int[n+1];
    for(int i=1;i<=n;i++){
        if(isPerfect(i)) dp[i] = 1;
        else{
            int min = Integer.MAX_VALUE;
            for(int j=1;j<=i/2;j++){
                int count = dp[j] + dp[i-j];
                min = Math.min(min,count);
            }
            dp[i] = min;
        }
    }
    return dp[n];
}
```

T.C. = $O(n^2)$
S.C. = $O(n)$

gt just works

```java
public int numSquares(int n) {
        int[] dp = new int[n+1];
        for(int i=1;i<=n;i++){
            if(isPerfect(i)) dp[i] = 1;
            else{
                int min = Integer.MAX_VALUE;
                for(int j=1;j*j<=i;j++){
                    int count = dp[j*j] + dp[i-j*j];
                    min = Math.min(min,count);
                }
                dp[i] = min;
            }
        }
        return dp[n];
    }
```

T.C. = $O(n\sqrt{n})$
S.C. = $O(n)$

gt obviously works

[Leetcode 279]

Ques:

Try out all possible combination

PW SKILLS

Q : 0/1 Knapsack

	Gold	iPhone	Table	Painting
price	5	3	9	16
weight	1	2	8	10
	5	1.5	1.1	1.6

C = 8

ans = 9

Objective : To get maximum amount (profit)

$\frac{price}{weight}$ → only in Fractional Knapsack

[Hackerrank]

Ques:

Q : O/1 Knapsack

```
public static int profit(int i, int[] wt, int[] val, int C){
    if(i==wt.length) return 0;
    int skip = profit(i+1,wt,val,C);
    if(wt[i]>C) return skip;
    int pick = val[i] + profit(i+1,wt,val,C-wt[i]);
    return Math.max(pick,skip);
}
```

$T.C. = O(2^n)$

$A.S. = O(n*C)$

[Hackerrank]

Ques:

Q : O/1 Knapsack (Recursion + Memoization)

```java
public static int profit(int i, int[] wt, int[] val, int C, int[][] dp){
    if(i==wt.length) return 0;
    if(dp[i][C]!=-1) return dp[i][C];
    int skip = profit(i+1,wt,val,C,dp);
    if(wt[i]>C) return dp[i][C] = skip;
    int pick = val[i] + profit(i+1,wt,val,C-wt[i],dp);
    return dp[i][C] = Math.max(pick,skip);
}

public static void main(String[] args) {
    int[] val = {5,3,7,16};
    int[] wt = {1,2,8,10};
    int C = 8;
    int n = wt.length;
    // i = 0 to n-1 | C = C to 0
    int[][] dp = new int[n][C+1];
    for(int i=0;i<dp.length;i++)
        for(int j=0;j<dp[0].length;j++) dp[i][j] = -1;
    System.out.println(profit(0,wt,val,C,dp));
}
```

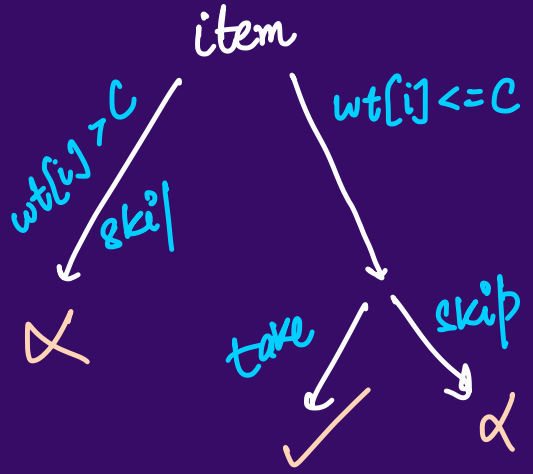
$T.C. = O(n*C)$

$A.S. = O(n*C)$

[Hackerrank]

Ques:

Q : O/1 Knapsack



[Hackerrank]

Ques:

0/1 knapsack

Q : [redacted] Subset Sum

arr = {0, 8, 5, 2, 4} target = 9

tell if there exists a subset of the array with sum == target

Try recursion & try all possible subsets

pick or skip the element

[Leetcode 416]

Ques:

PW SKILLS

Q : [redacted] Subset Sum

```
private static boolean subset(int i, int[] arr, int target) {
    if(i==arr.length){
        if(target==0) return true;
        else return false;
    }
    boolean skip = subset(i+1,arr,target);
    if(target-arr[i]<0) return skip; // Only valid for +ve numbers
    boolean pick = subset(i+1,arr,target-arr[i]);
    return pick || skip;
}


public static void main(String[] args) {
    int[] arr = {8,-1,2,4};
    int target = 7;
    System.out.println(subset(0,arr,target));
}
```

$T.C. = O(2^n)$

$A.S. = O(n*t)$

[Leetcode 416]

Ques:

Q : [redacted] Subset Sum

```
private static boolean subset(int i, int[] arr, int target, int[][] dp) {
    if(i==arr.length){
        if(target==0) return true;
        else return false;
    }
    if(dp[i][target]!=-1) return (dp[i][target]==1);
    boolean ans = false;
    boolean skip = subset(i+1,arr,target,dp);
    if(target-arr[i]<0) ans = skip;
    else{
        boolean pick = subset(i+1,arr,target-arr[i],dp);
        ans = pick || skip;
    }
    if(ans) dp[i][target] = 1;
    else dp[i][target] = 0;
    return ans;
}
```

T.C. = $O(n*t)$

A.S. = $O(n*t)$

[Leetcode 416]

Ques:

Q : Partition equal Subset Sum

↓

0/1 Knapsack

[Leetcode 416]

Ques: → Try out all possible ways

Q : Unbounded Knapsack → Every item has infinite quantity

	Gold	iPhone	Table	Painting
price	6	3	9	25
weight	2	2	8	9
	3	1.5	1.1	2.7

C = 9

maxProfit = 25

	0/1 Knapsack	Unbounded
pick →	i+1	i

Ques:

Q : Coin Change (Very famous question)

(Unbounded Knapsack)

amount = 11

coins = {1, 2, 5}

[Leetcode 322]

Ques:

Q : Coin Change → try out all possible combinations

amount = 8

coins = {1, 2, 4, 5}

8 = 5 + 2 + 1 → 3 coins

8 = 4 + 4 → 2 coins

[Leetcode 322]

Ques:

Q : ~~Coin Change~~

Knapsack general code :

'i+1'

```
if (amt - arr[i] < 0)    skip

else {
      skip & pick
3
```

'i' in unbounded

'i+1' in 0/1

[Leetcode 322]

Homework:

PW SKILLS

Q : Length of the longest subsequence that sums to target.

arr = { 1, 2, 5, 9, 3, 4, 3 } target = 9

ans = 4

[Leetcode 2915]

Ques:

Q : Target Sum

arr = {1, 1, 1, 1, 1} target = 3 ways = 5

st = 2 → ways = 0

+1 +1 +1 +1 -1 = 3

+1 +1 +1 -1 +1 = 3

+1 +1 -1 +1 +1 = 3

+1 -1 +1 +1 +1 = 3

-1 +1 +1 +1 +1 = 3

[Leetcode 494]

Ques:

Q : Target Sum

Knapsack → pick or skip strategy

↓ ↓

\+ −

[Leetcode 494]

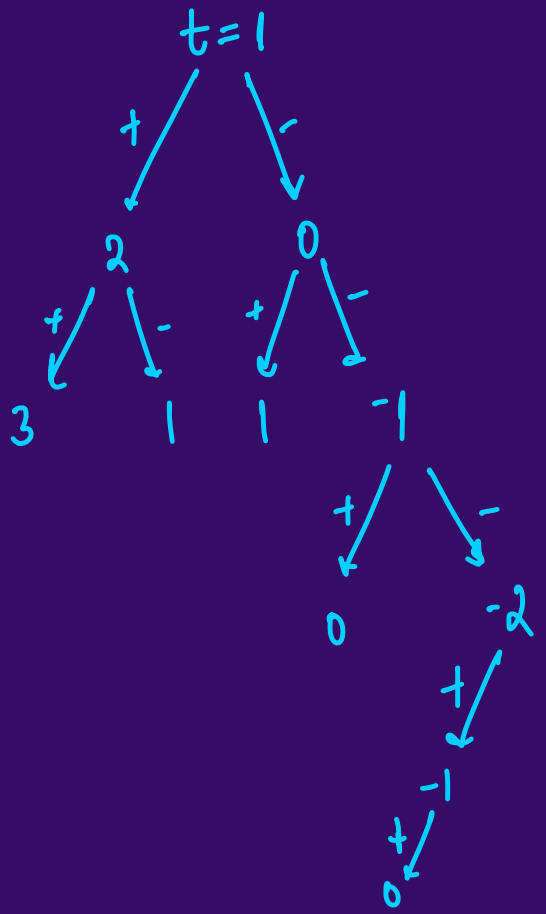
Ques:

arr = { 1, 1, 1, 1, 1 }

target = 1

PW SKILLS

Q : Target Sum



i = 0 to 4 target = 0 to t

dp[i][j]

↓

Stores the result of a call in which i & j

dp[5][t+1]

target = -s to s

↓ transform

0 to 2^*s

[Leetcode 494]

Ques:

Q : Target Sum

S = 5

t → -S to S

target = -5 -4 -3 -2 -1 0 1 2 3 4 5

0 1 2 3 4 5 6 7 8 9 10

[Leetcode 494]

Ques:

Q : Target Sum

- We can use hashmap for negative indexing

HashMap < Integer, Hashmap < int , value > >

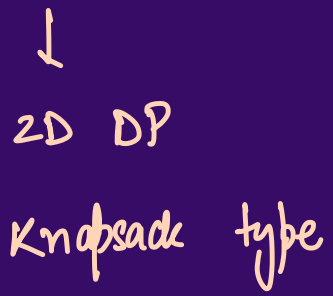
Integer → row number

int → -ve as well

[Leetcode 494]

Homework:

Q : Visit Array positions to maximise score



[Leetcode 2786]

Tabulation in 'Knapsack Problem'

PW SKILLS

1) Thinking of tabulation

2) Thinking of Recursion → Top-Down → convert → Bottom Up

Steps :

1) idx → n-1 to 0 instead of 0 to n-1

2) Make formula

3) P/S make sense of the base case

Tabulation

```
public static int profit(int i, int[] wt, int[] val, int C, int[][] dp){
    if(i<0) return 0;
    if(dp[i][C]!=-1) return dp[i][C];
    int skip = profit(i-1,wt,val,C,dp);
    if(wt[i]>C) return dp[i][C] = skip;
    int pick = val[i] + profit(i-1,wt,val,C-wt[i],dp);
    return dp[i][C] = Math.max(pick,skip);
}
```

if(wt[i] > C) profit(i, C) = profit(i-1, C)

else

profit(i, C) = max(val[i] + profit(i-1, c-wt[i]), profit(i-1, C))

dp[i][c] = max(val[i] + dp[i-1][c-wt[i]], dp[i-1][c])

Tabulation

Bottom Up

```
for(int i=0;i<n;i++){
    for(int j=0;j<C+1;j++){
        int skip = (i>0) ? dp[i-1][j] : 0;
        if(wt[i]>j) dp[i][j] = skip;
        else{
            int pick = val[i];
            pick += ((i>0) ? dp[i-1][j-wt[i]] : 0);
            dp[i][j] = Math.max(pick,skip);
        }
    }
}
```

```
public static int profit(int i, int[] wt, int[] val, int C, int[][] dp){
    if(i<0) return 0;
    if(dp[i][C]!=-1) return dp[i][C];
    int skip = profit(i-1,wt,val,C,dp);
    if(wt[i]>C) return dp[i][C] = skip;
    else{
        int pick = val[i] + profit(i-1,wt,val,C-wt[i],dp);
        return dp[i][C] = Math.max(pick,skip);
    }
}
```

Top - Down

Tabulation

```
long[][] dp = new long[n][amount+1];
        for(int i=0;i<dp.length;i++){
            for(int j=0;j<dp[0].length;j++){
                long skip = (i>0) ? dp[i-1][j] : ((j==0) ? 0 : Integer.MAX_VALUE);
                if(j-coins[i]<0) dp[i][j] = skip;
                else{
                    long pick = 1 + dp[i][j-coins[i]];
                    dp[i][j] = Math.min(skip,pick);
                }
            }
        }
```

```
long[][] dp = new long[2][amount+1]; // i-1 : 0 & i = 1 -> dp
        for(int i=0;i<n;i++){
            for(int j=0;j<amount+1;j++){
                long skip = (i>0) ? dp[0][j] : ((j==0) ? 0 : Integer.MAX_VALUE);
                if(j-coins[i]<0) dp[1][j] = skip;
                else{
                    long pick = 1 + dp[1][j-coins[i]];
                    dp[1][j] = Math.min(skip,pick);
                }
            }
            // copy paste 1st row to 0th row
            for(int j=0;j<amount+1;j++){
                dp[0][j] = dp[1][j];
            }
        }
```

Space Optimisation.

Ques:

PW SKILLS

Q : Longest Common Subsequence [length]

m → s1 = b a c d e f

n → s2 = a g h c i f

ans = 3

→ a c f

if (s1[m-1] == s2[n-1]) lcs = 1 + lcs(0 to m-2 of s1 & 0 to n-2 of s2)

else

[Leetcode 1143]

Ques:

PW SKILLS

Q : Longest Common Subsequence (length)

s1 = b a c d e f k

s2 = a g h c i f b

bacf ≠ a c f b

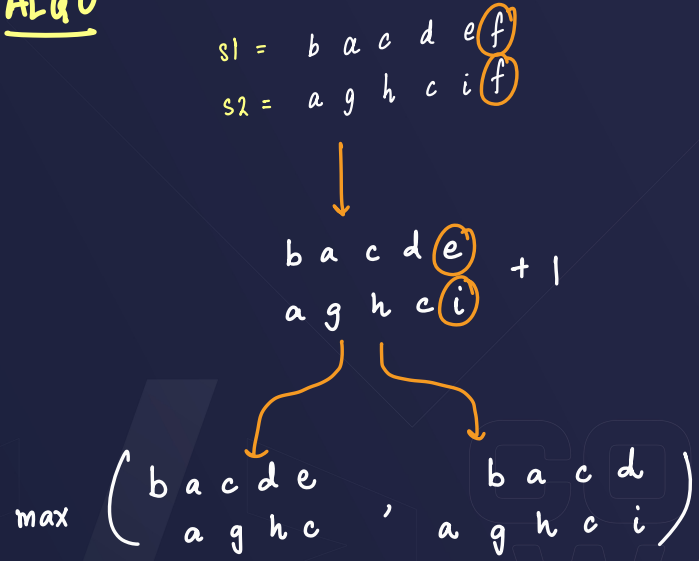
[Leetcode 1143]

Ques:

Q : Longest Common Subsequence (print)

[Leetcode 1143]

ALGO


s1 = b a c d e f
s2 = a g h c i f
b a c d e
a g h c i
+1
max
b a c d e
a g h c
b a c d
a g h c i


b a c d e
a g h c
b a c d e
a g h
b a c d
a g h c
b a c d
a g h
b a c
a g h c
1+
ba
agh

LCS (Recursion + Memoization)

```
public int lcs(int i, int j, StringBuilder a, StringBuilder b, int[][] dp) {
    if(i<0 || j<0) return 0;
    if(dp[i][j]!=-1) return dp[i][j];
    if(a.charAt(i)==b.charAt(j))
        return dp[i][j] = 1 + lcs(i-1,j-1,a,b,dp);
    else
        return dp[i][j] = Math.max(lcs(i-1,j,a,b,dp),lcs(i,j-1,a,b,dp));
}
public int longestCommonSubsequence(String text1, String text2) {
    StringBuilder a = new StringBuilder(text1);
    StringBuilder b = new StringBuilder(text2);
    int m = a.length(), n = b.length();
    // i = m-1 to 0 | j = n-1 to 0
    int[][] dp = new int[m][n];
    for(int i=0;i<dp.length;i++)
        for(int j=0;j<dp[0].length;j++) dp[i][j] = -1;
    return lcs(m-1,n-1,a,b,dp);
}
```

T.C. = $O(m*n)$ A.S. = $O(m*n)$

Ques:

Derivative of LCS

PW SKILLS

Q : Longest Palindromic Subsequence (length)

Hint :

1) LCS
2) Reverse of given string

Str	=	a	b	c	d	b	f	g	a	g
rev	=	g	a	g	f	b	d	c	b	a

] LCS

$LPS(str) = LCS(str, rev(str))$

$n \quad n$

$\Rightarrow$ T.C. $= O(n^2)$

A.S. $= O(n^2)$

~~[Leetcode 516]~~

Ques:

$s + rev(s)$ is always a palindrome

PW SKILLS

Q : Minimum Insertion steps to make string Palindrome

$s =$ m b a d m $\rightarrow$ ans = 2

1) m b d a d b m $\Rightarrow$ m b d a d b m (2)

2) m d b a b d m $\Rightarrow$ m d b a b d m (2)

Hint-1 : 1) LPS

2) This is a Easy Question

[Leetcode 1312]

Ques:

Deletion

Q : Minimum Insertion steps to make string Palindrome

a b c d b f g a g

steps = s.length() - LPS(s)

g a gf b c d c b fg a g

ans = 4

[Leetcode 1312]

Homework:

Q : Delete Operation for 2 strings

[Leetcode 583]

Ques:

Q : Edit Distance

a = movie

b = love

Delete
Insert
Replace

1) Delete i

move
love

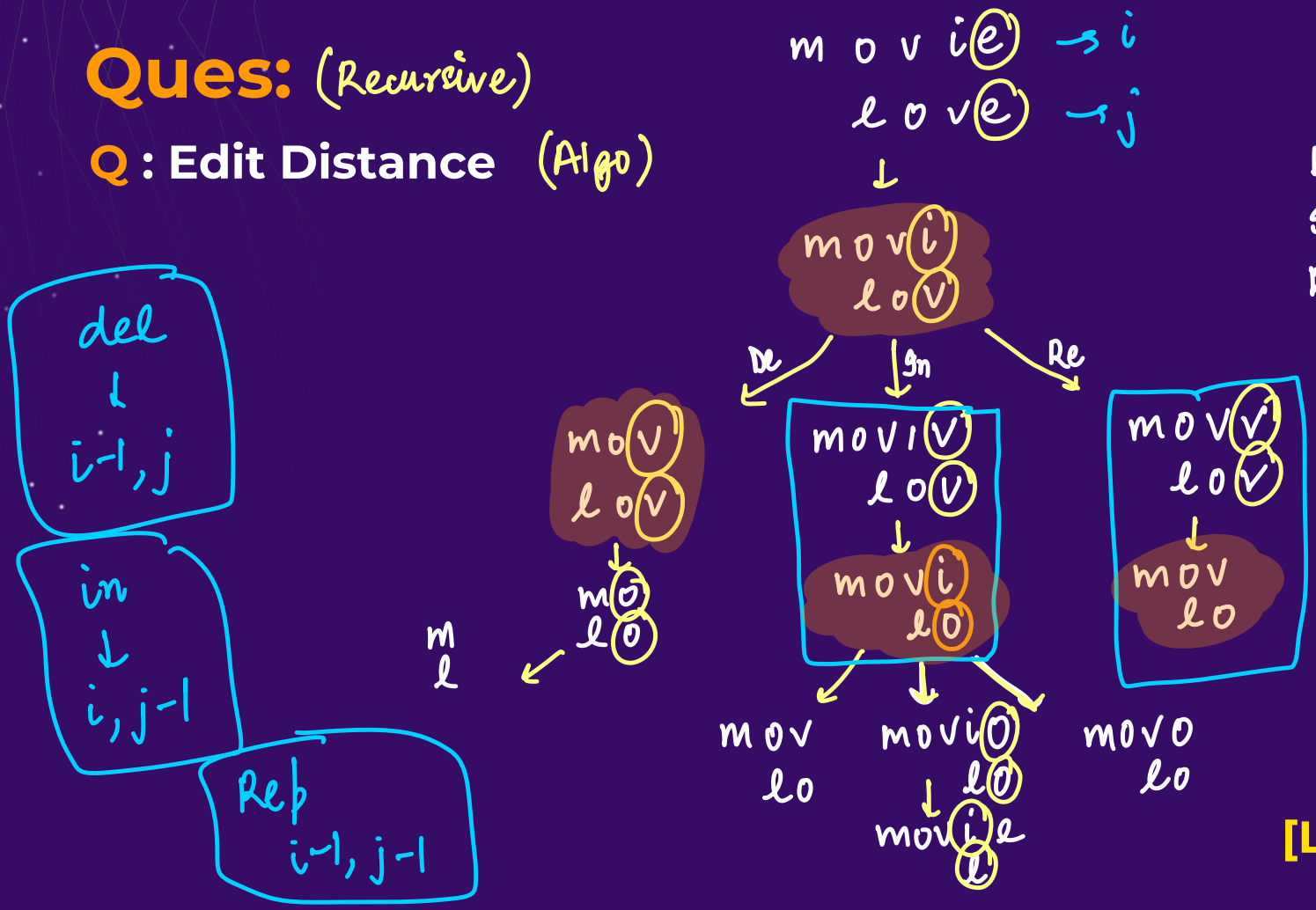
2) Replace 'm' with 'l'

love
love

[Leetcode 72]

Ques: (Recursive)

Q : Edit Distance (Algo)



[Leetcode 72]

Ques:

Q : Edit Distance

```java
public int minDistance(String a, String b) {
        int m = a.length(), n = b.length();
        if(m==0 || n==0) return Math.max(m,n);
        int[][] dp = new int[m][n];
        for(int i=0;i<m;i++){
            for(int j=0;j<n;j++){
                int p = (i>=1 && j>=1) ? dp[i-1][j-1] : (i==0 ? j : i);
                int q = (j>=1) ? dp[i][j-1] : i;
                int r = (i>=1) ? dp[i-1][j] : j;
                if(a.charAt(i)==b.charAt(j)) dp[i][j] = p;
                else dp[i][j] = 1 + Math.min(r,Math.min(q,p));
            }
        }
        return dp[m-1][n-1];
    }
```

[Leetcode 72]

Ques:

Q : Edit Distance

a = ""

b = "raghav"

[Leetcode 72]

Ques:

PW SKILLS

Q : Longest Common Subsequence (Tabulation)

We will understand tabulation in 2 ways:

1) Just convert memoization to tabulation

2) Thinking via tabulation.

to avoid the base case conversion step, you can make 'dp' larger

→ m+1 & n+1

[Leetcode 1143]

Ques:

Q : Longest Common Subsequence (Tabulation)

```java
public int longestCommonSubsequence(String a, String b) {
    int m = a.length(), n = b.length();
    int[][] dp = new int[m][n]; // i = m-1 to 0 | j = n-1 to 0
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            int p = (i>=1 && j>=1)? dp[i-1][j-1] : 0;
            int q = (j>=1)? dp[i][j-1] : 0;
            int r = (i>=1)? dp[i-1][j] : 0;
            if(a.charAt(i)==b.charAt(j)) dp[i][j] = 1 + p;
            else dp[i][j] = Math.max(q,r);
        }
    }
    return dp[m-1][n-1];
}
```

T.C. = $O(m*n)$

A.S. = $O(m*n)$

→ Stores LCS of substr(a, 0, m-1) & substr(b, 0, n-1)

- dp[i][j] will store the LCS of substring of a from 0 to i & substring of 'b' from 0 to j.

[Leetcode 1143]

Ques:

Q : Longest Common Subsequence (Tabulation)

```java
public int longestCommonSubsequence(String a, String b) {
    int m = a.length(), n = b.length();
    int[][] dp = new int[m+1][n+1];
    for(int i=1;i<=m;i++){
        for(int j=1;j<=n;j++){
            if(a.charAt(i-1)==b.charAt(j-1))
                dp[i][j] = 1 + dp[i-1][j-1];
            else
                dp[i][j] = Math.max(dp[i][j-1],dp[i-1][j]);
        }
    }
    return dp[m][n];
    // dp[i][j] will store LCS of substr(a,0,i-1) & substr(b,0,j-1)
}
```

Extra Row & Extra Col

T.C. = O(m*n)

A.S. = O(m*n) ⟶ O(n) [Space optimization]

[Leetcode 1143]

Ques:

Q : Longest Common Subsequence (Tabulation)

[Leetcode 1143]

Ques:

s1 = b a c d e f

s2 = a g h c i f

SKILLS

Q : Longest Common Subsequence (Tabulation)

dp

	a	g	h	c	i	f
b	0	0	0	0	0	0
a	1	1	1	1	1	1
c	1	1	1	2	2	2
d	1	1	1	2	2	2
e	1	1	1	2	2	2
f	1	1	1	2	2	3

m x n

dp[i][j]

LCS [s1 → 0 to i substr
s2 → 0 to j substr

[Leetcode 1143]