Agenda

1) Maximum Sum w/o Adjacent elements

2) Count Unique Paths → All cells are unblocked
   → Cells are blocked

3) Dungeon Princess

Find maximum subsequence sum from a given integer array of size N. where you cannot select adjacent elements

$A = \{ \overset{0}{9}, 4, \overset{2}{13} \}$ Ans = 22

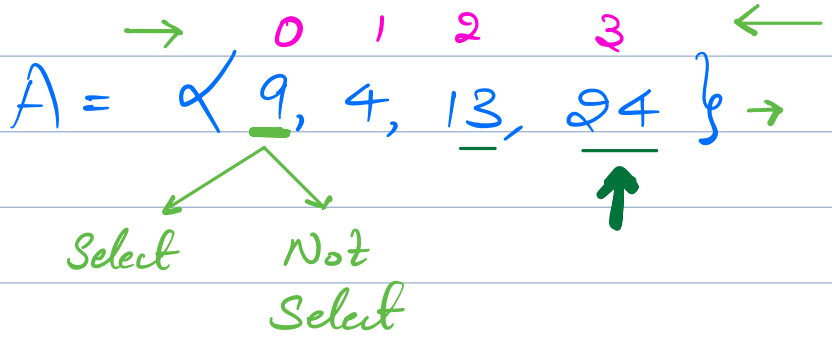
$A = \{ \overset{0}{9}, \overset{1}{4}, \overset{2}{13}, \overset{3}{24} \}$ Ans = 33

⇒ $A = \{ \overset{0}{10}, \overset{1}{20}, \overset{2}{30}, \overset{3}{40} \}$ Ans = 60

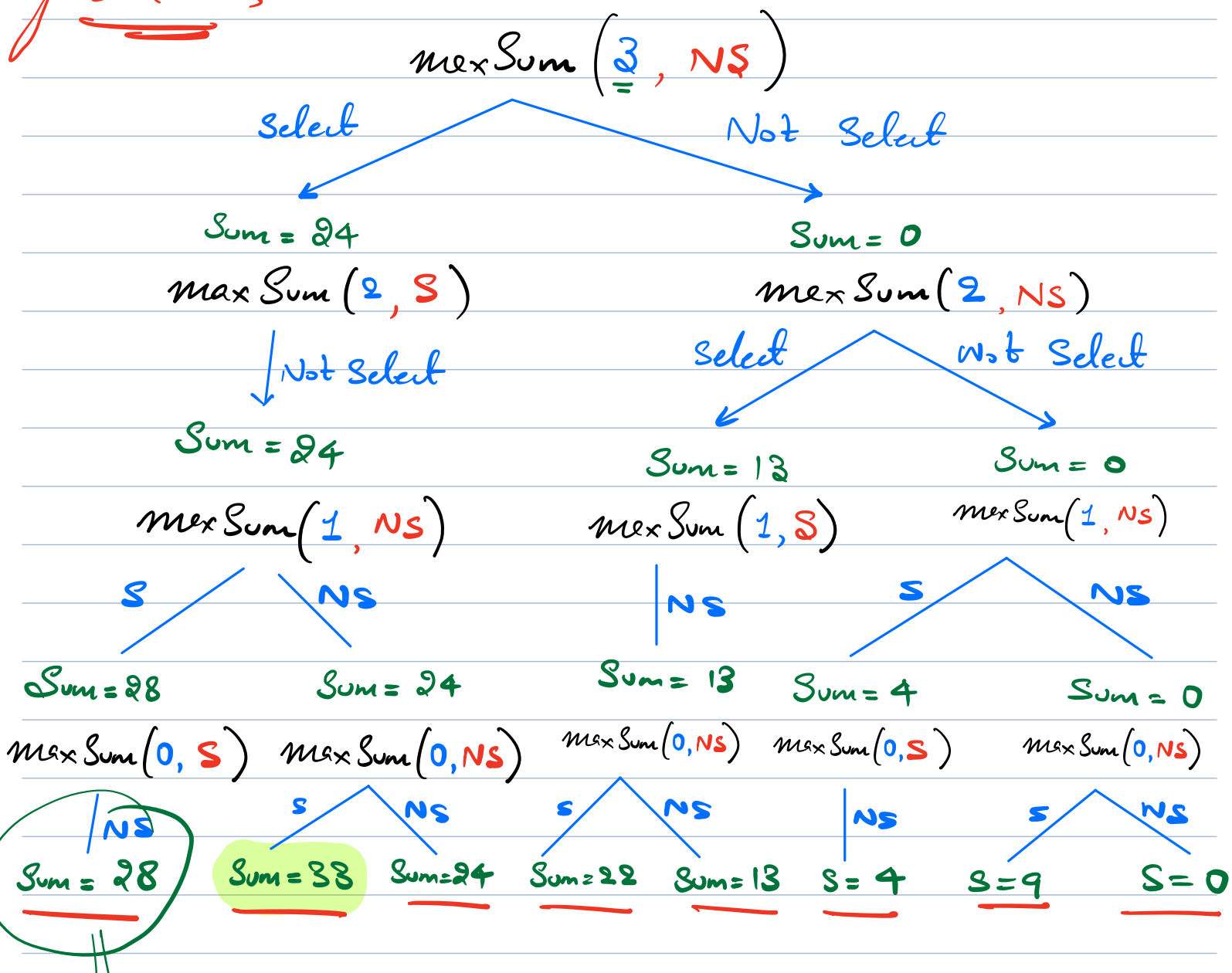
$A = \{ 10, 102, 100 \}$

Sol^n 1) Brute force

⇒ Check sum for all valid subsequences.



globSol as



↓

Answers are at Base Case

Code

```
ans = 0;                                   N-1          false        0
                                            ↑            ↑           ↑
void maxSum(A, index, bool lastSelect, sum){

    if (index == -1){
        ans = max(ans, sum);
        return;
    }
    maxSum(A, index-1, false, sum); //NS

    if (lastselect == false){
        maxSum(A, index-1, True, sum+A[index]);
    }
}
```

2) Using Recursion.

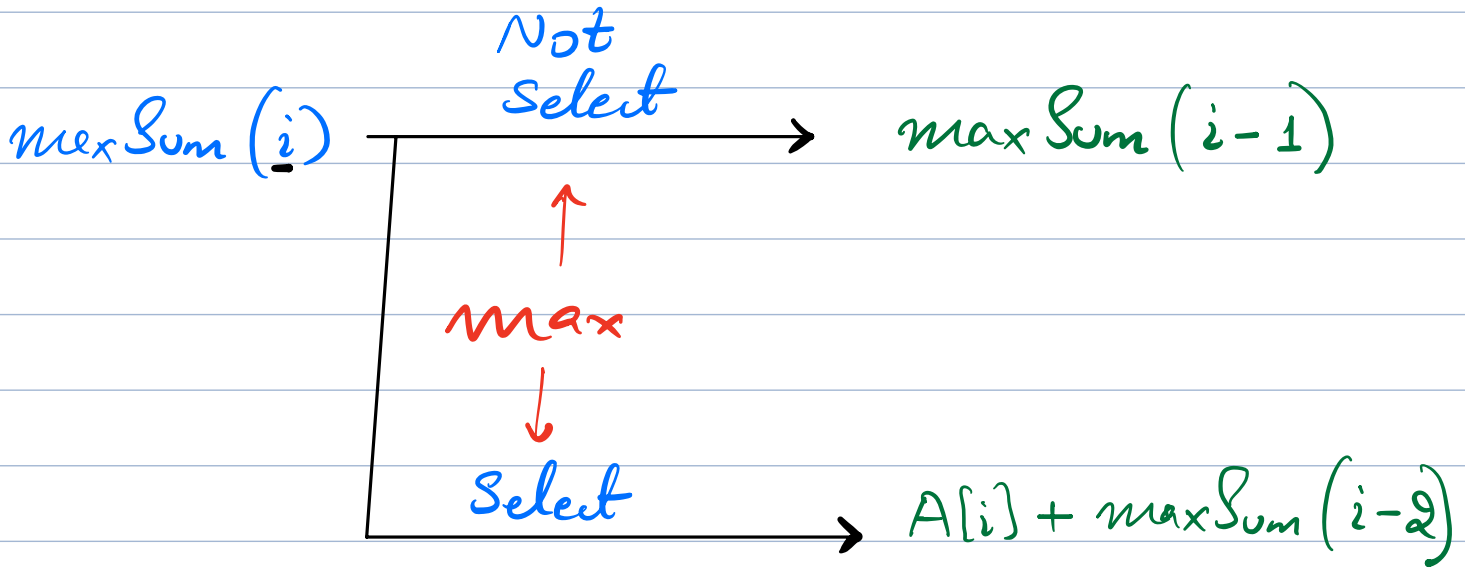
$$A = [\underset{0}{9}, \underset{1}{4}, \underset{2}{13}, \underset{3}{24}]$$

1) Elements of Choice ⇒ Select or Not Select

2) What does my state represents ??

maxSum(i) = Maximum subsequence sum from A[0, i] without selecting consecutive elements.

3) Recurrance relationship.

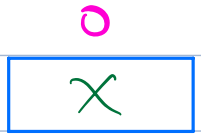


$$maxSum(i) = max(maxSum(i-1),\ A[i] + maxSum(i-2));$$

Base Case ⇒ Transition in RR.

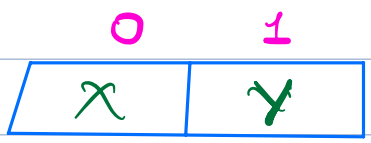
i	i − 1	i − 2
0 ✓	−1 ✗	−2 ✗
1 ✓	0 ✓	−1 ✗
2 ✗	1 ✓	0 ✓

$i = 0$



⇒ A[0]

$i = 1$



⇒ max(A[0], A[1]);

4) Which state is the answer ??

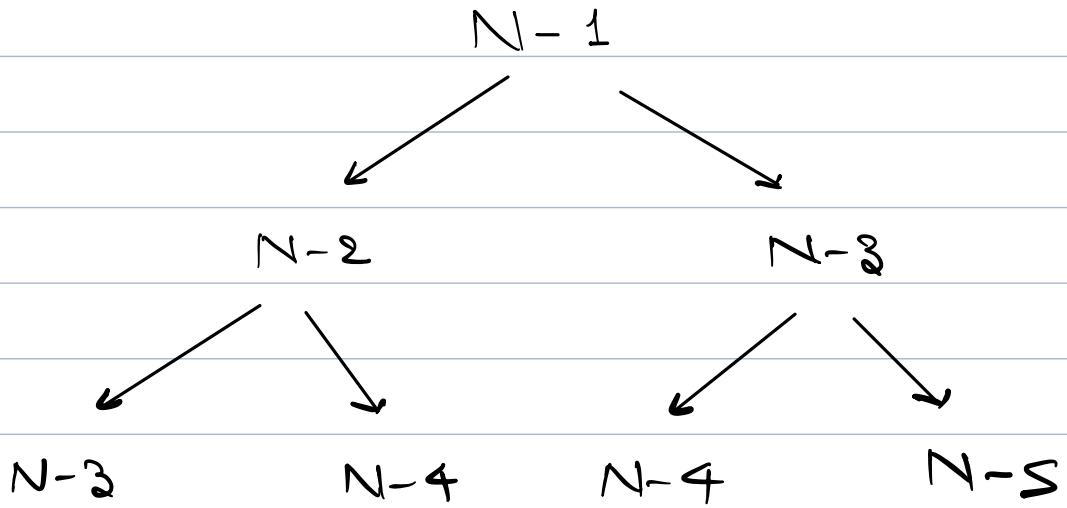
maxSum (N − 1).

Code

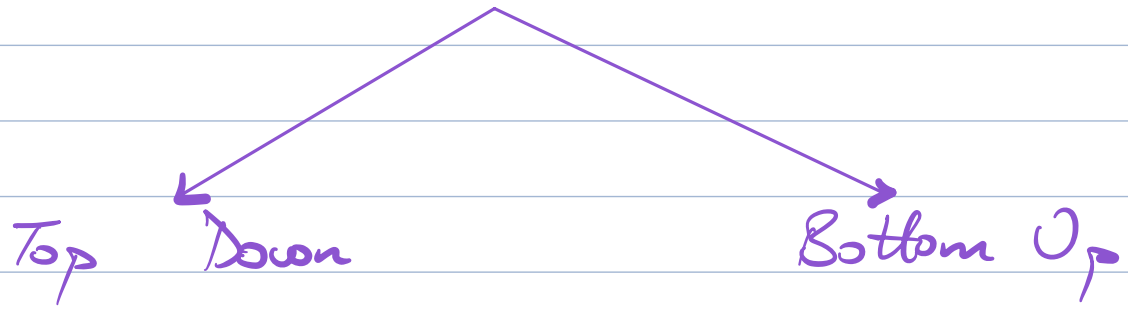
```
int maxSum(A, i) {
    if (i==0) { return A[0]; }
    if (i== 1) { return max(A[0], A[1]);
    return max(maxSum(i-1), A[i] +
                maxSum(i-2));
}
```

$i \Rightarrow i-1, i-2$

[Fibonacci DP]



We can use DP Here



1) Top Down

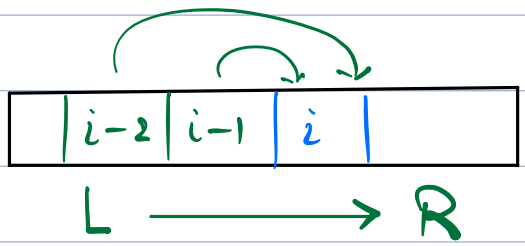
```
int DP[N-1] = {-1}

int maxSum(A, i) {

    if (i==0) { return A[0]; }

    if (i== 1) { return max(A[0], A[1]);

    if (DP[i] != -1) { return DP[i]; }

    DP[i]= max( maxSum (i-1), A[i] +
                maxSum (i-2));

    return DP[i];

}
```

T.C. = O(N)

S.C. = O(N) + O(N) = O(N)

---

2) Bottom Up




```
DP[N];
DP[0] = A[0];
DP[1] = max(A[0], A[1]);

for (i=2; i<N; i++) {

    DP[i] = max(DP[i-1], A[i]+DP[i-2]);
}

return DP[N-1];
```

T.C. = O(N)

S.C. = O(N)

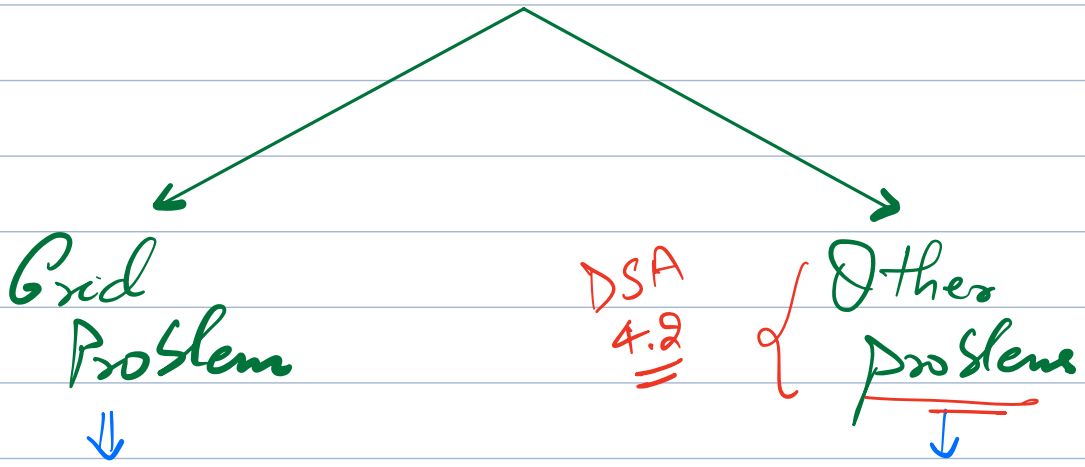
⇓

S.C. = O(1)

2D DP

Recursive State ⇒ (i, j)

DP[max of i] [max value of j]

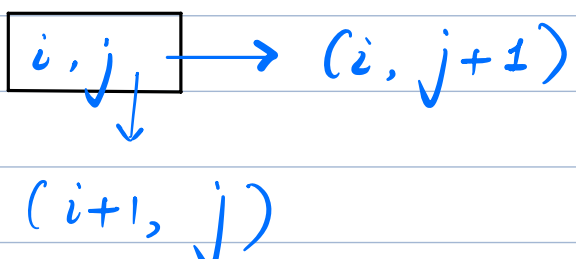


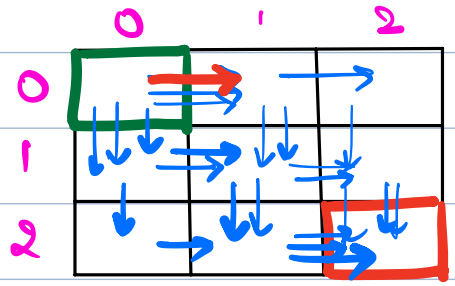
Given a matrix of size N×M.

Find the total no. of ways to go from (0, 0) to (N-1, M-1).

Movement ⇒ 1 step horizontally right

1 step vertically down





3×3

h h v v
h v h v
h v v h
v h h v
v h v h
v v h h

} 6

Sol^n Notion of Choice ⇒ Recursion

1) Elements of Choice ⇒ Go Right
or
Go Down.

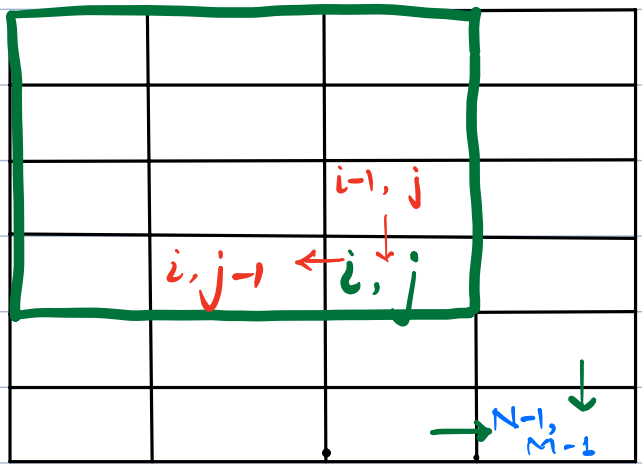
2) What does the recursive state calculates.

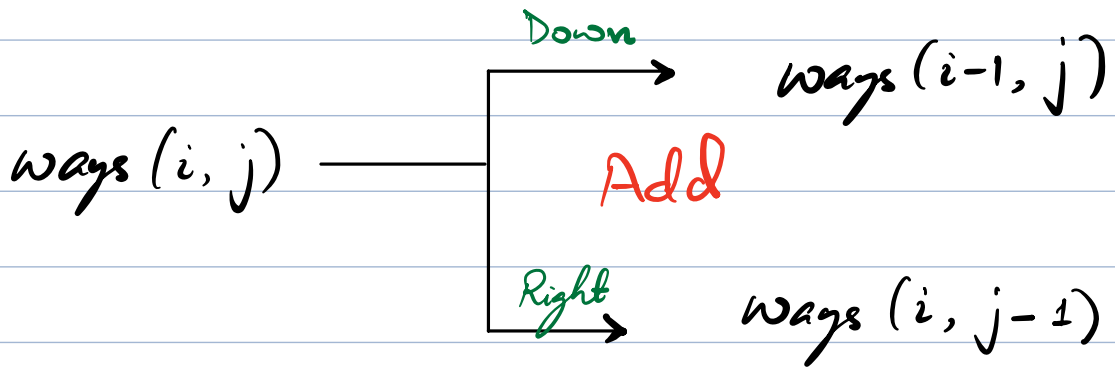
ways (N-1, M-1)

⇓

ways (i, j) = no. of ways to reach the cell (i, j) from (0, 0).

3) Recurrance relationship.





$$ways(i,j) = ways(i-1,j) + ways(i,j-1)$$

Base Cases

i	$i-1$
0 ✓	-1 ✗
1 ✗	0 ✓

j ⟶ j - 1

0 ✓ → -1 ✗

1 ✗ → 0 ✓

	0	1	2	3
0	1	1	1	1
1	1			
2	1			
3	1			
4	1			
5	1			

4) Which state is answer ??

ways (N-1, M-1)

Code

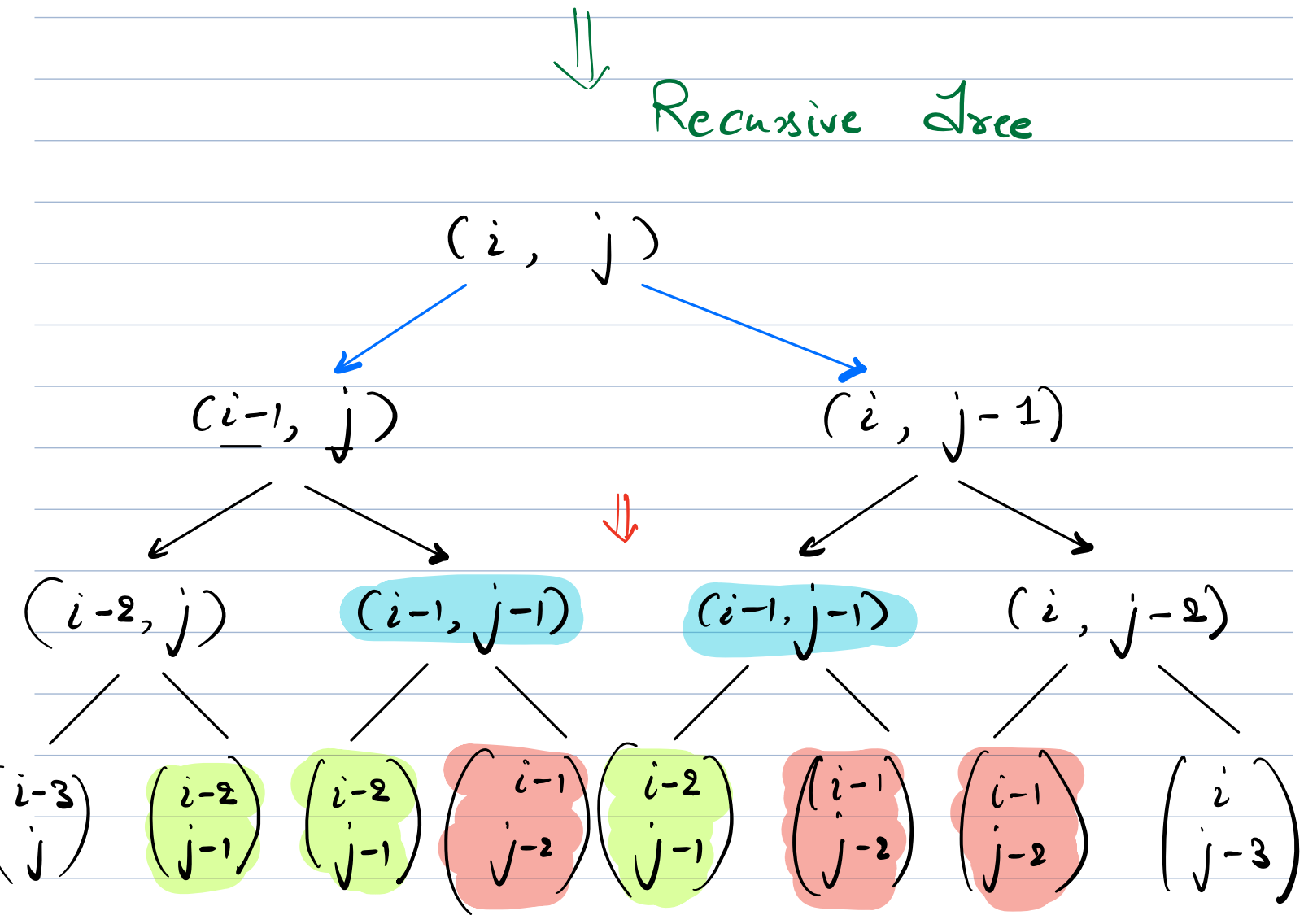
```
int ways (i, j) {
    if (i==0 || j==0) {
```

return 1;

}

return ways(i-1, j) + ways(i, j-1);

}



Overlapping Subproblems ⇒ DP

1) Memoization (Top-Down)

Code

```
DP[N][M] = {-1};

int ways(i, j) {
    if (i==0 || j==0) {
        return 1;
    }
    if (DP[i][j] != -1) {
        return DP[i][j];
    }
    DP[i][j] = ways(i-1, j) + ways(i, j-1);
    return DP[i][j];
}
```

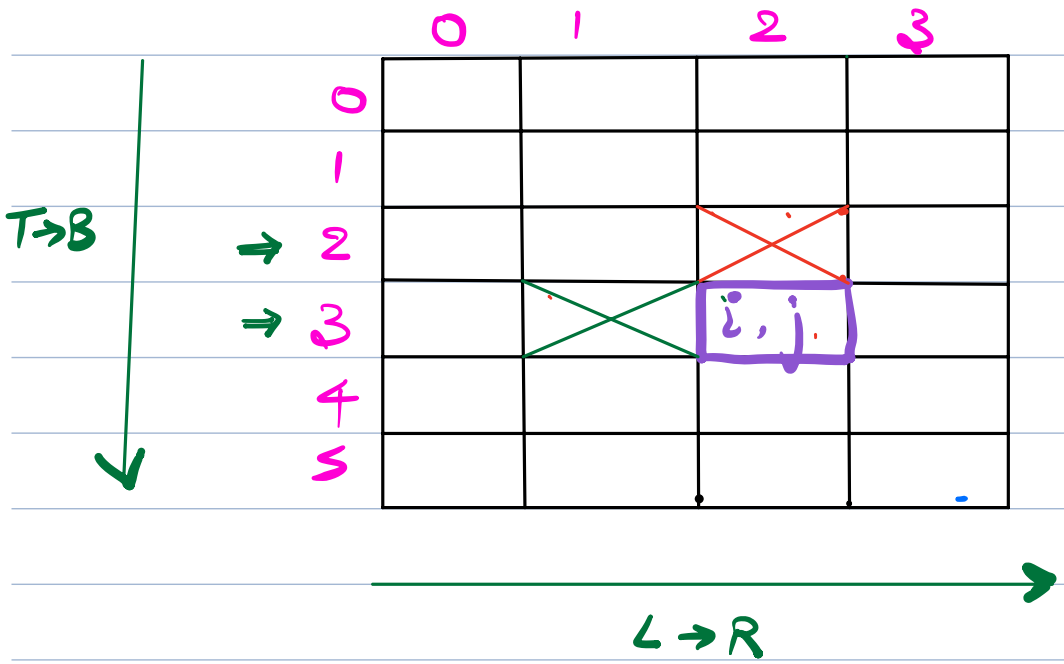
T.C. = O(N x M)

S.C. = O(N x M)

---

2) Bottom Up

ways(i, j) = ways(i-1, j) + ways(i, j-1)



Code

```
DP[N][M]

for (j = 0; j < M; j++) {
        DP[0][j] = 1;
}
for (i = 0; i < N; i++) {
        DP[i][0] = 1
}

for (i = 1; i < N; i++) {
        for (j = 1; j < M; j++) {
```

Base Case

DP[i][j] = DP[i-1][j] + DP[i][j-1];

}

}

T.C. = O(N x M)

S.C. = O(N x M)

⇓ H.W.

O(2 x M)

---

1 month { DSA 4.2 (< 5%)

⇓

Maths ⇒ Inverse Modulo

Trie

DP on Strings

2D DP

Graphs → Bi-partite
→ Floyd Warshall
→ Bellman Ford

Interview Problems

DSA Mock Interviews

DSA 4

&

Content

18th March (DSA 4.2

SQL

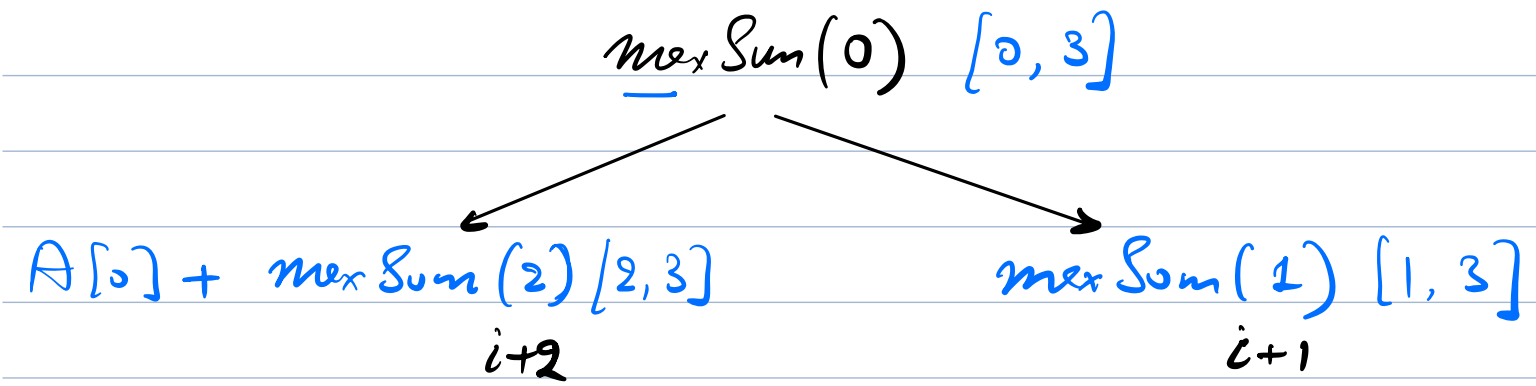
LLD

HLD

Project

$A = [9, 4, 13, 24]$ (indices 0, 1, 2, 3)

i



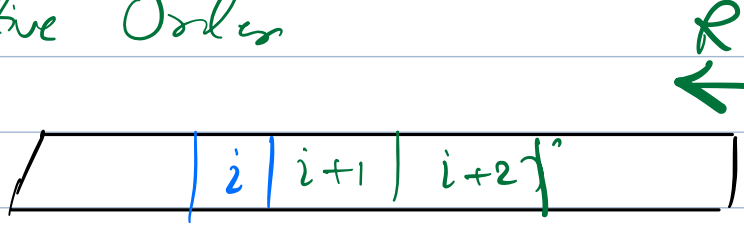
$$maxSum(i) = max\Big(maxSum(i+1),\ A[i] + maxSum(i+2)\Big);$$

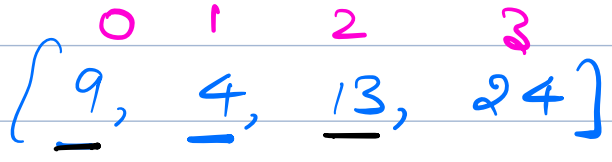
i $\quad$ $i+1$ $\quad$ $i+2$

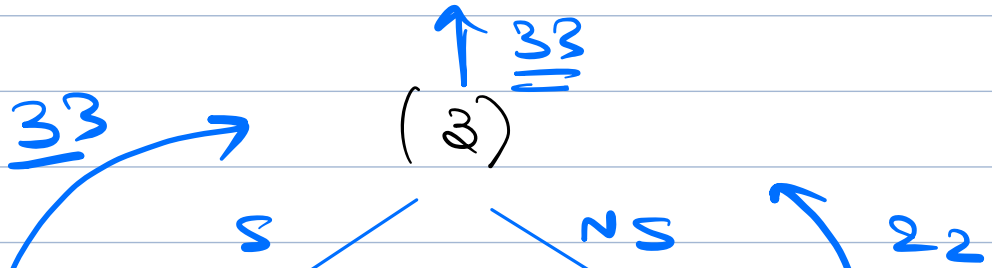
$N-1 \Rightarrow A[N-1]$

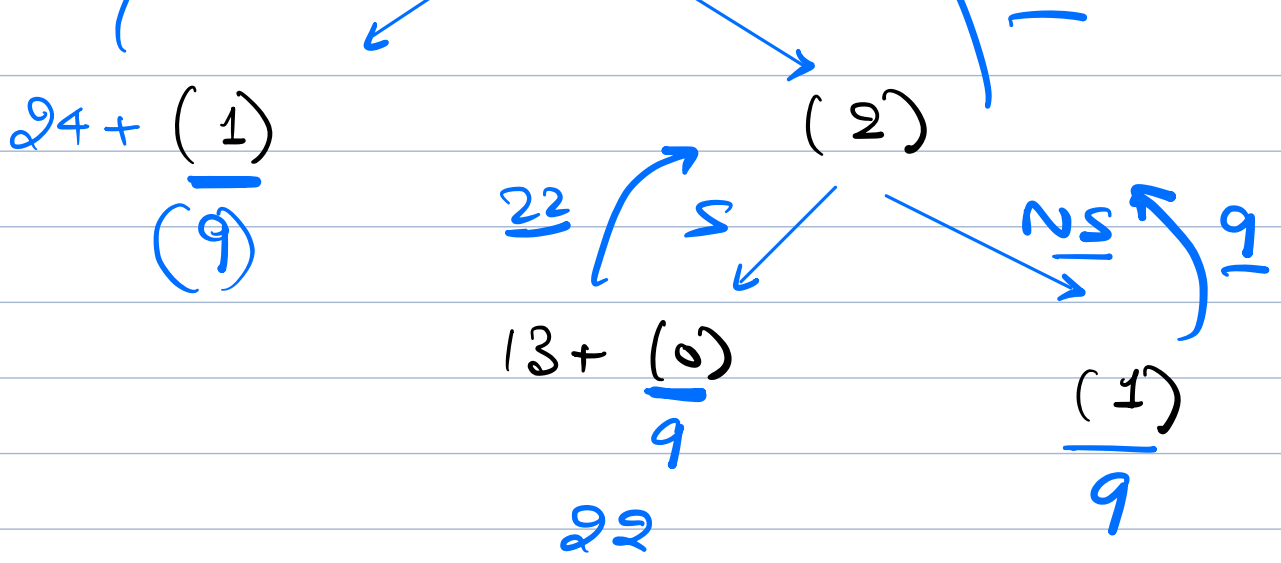
$N-2 \Rightarrow max(A[N-1], A[N-2])$

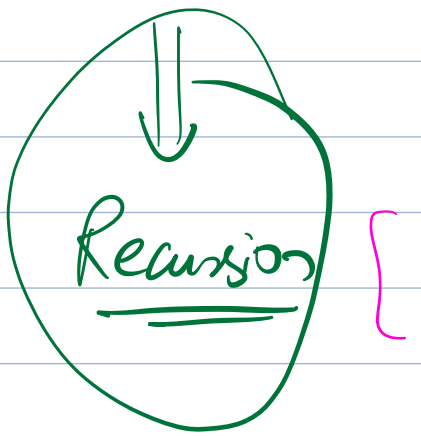
Iterative Order

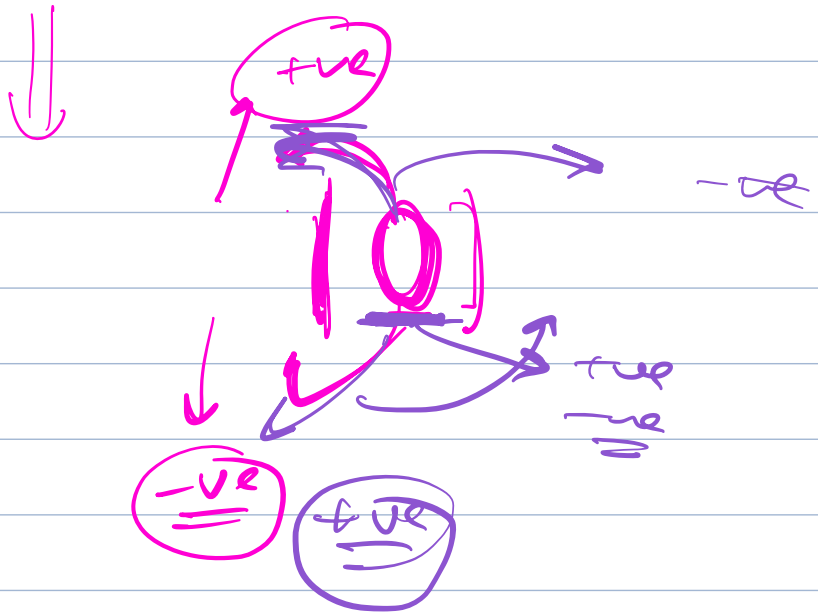







24+ (1)
9
(2)
22
5
13+ (0)
9
22
25
9
(1)
9


Recursion


+ve
-ve
+ve
-ve
-ve
+ve