**Q1** Given an array. Find max subsequence sum.

arr[] = [4 -3 2 -5 6 10]

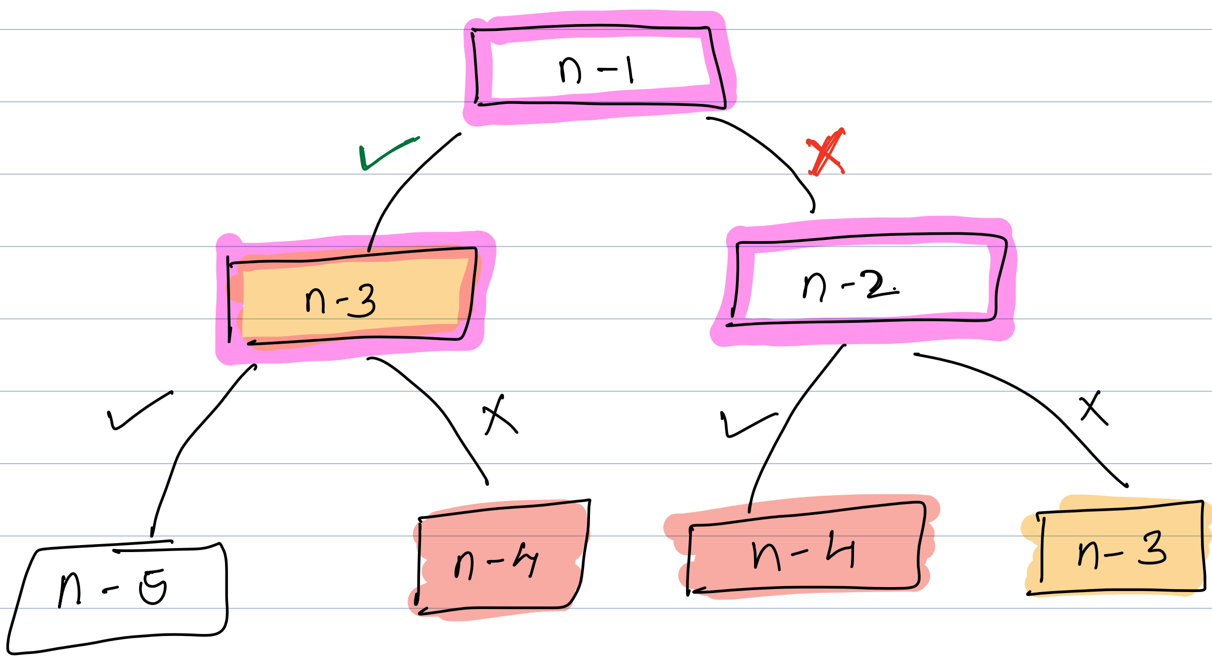
**Q1** Given an array. Find max subsequence sum. You are not allowed to pick adjacent elements. All elements are positive integers.

**Ex1** arr[] = [9, 4, 13]

$\Rightarrow$ 22

**Ex2** arr[] = [50 100 60]

**Ex3** arr[] = [100 20 40 100]



1) Element of Choice.

$$\begin{tikzcd}[row sep=small] & n-1 \\ n \arrow[ur, no head] \arrow[dr, no head] & \\ & n-2 \end{tikzcd}$$

2) State

$dp[i] \Rightarrow$ Maximum sum from ($0^{th}$ – $i^{th}$)

$func(i) \Rightarrow$ "

3) Recurrence Relation

$$dp[i] \Rightarrow \max\left(arr[i] + dp[i-2],\ dp[i-1]\right)$$

$$func(i) \Rightarrow \max\left(arr[i] + func(i-2),\ func(i-1)\right)$$

4) Base Case :

$$func(0) = arr[0]$$

$$func(1) = \max\left(arr[0],\ arr[1]\right)$$

dp[n] = {-1}, dp[0] = arr[0], dp[1] = max(arr[0], arr[1]);

```
int maxSum(int i) {
    if (dp[i] != -1)
        return dp[i];

    dp[i] = max( maxSum(i-2) + arr[i],
                 maxSum(i-1) );

    return dp[i];
}
```

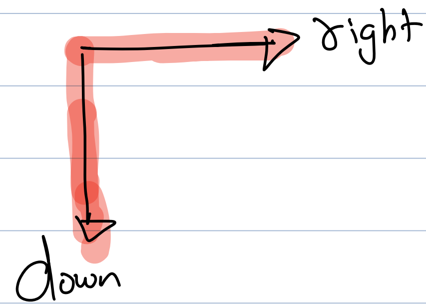
TC : O(n)
SC : O(n)

Bottom Up:

TC: O(n)
SC: O(1) ⇒ using 2 variables.

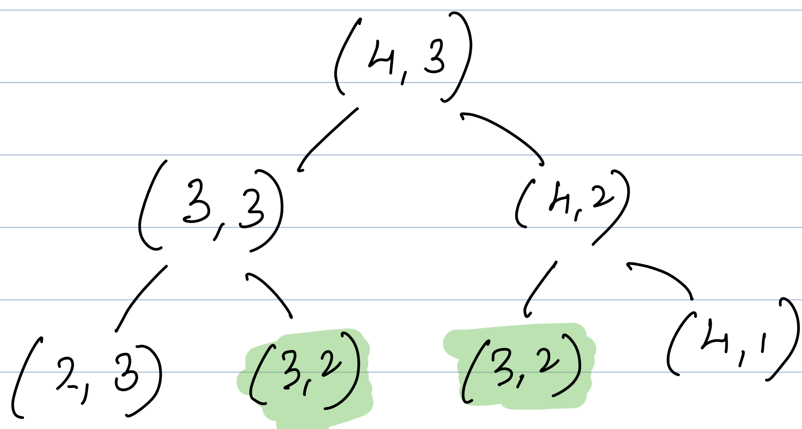
**Q2** How many ways to reach to $(n-1, m-1)$ from $(0,0)$

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 3 | 4 |
| 2 | 1 | 3 | 6 | 10 |
| 3 | 1 | 4 | 10 | 20 |
| 4 | 1 | 5 | 15 | 35 |




$$Sum(i,j) = Sum(i+1, j) + Sum(i, j+1)$$

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



1) func(i,j) $\Rightarrow$ ways to reach i,j from (0,0)

2) Element of choices :

func(i,j) — func(i-1, j)
func(i,j) — func(i, j-1)

3) Recurrence relation

$$func(i,j) \Rightarrow func(i-1,j) + func(i,j-1)$$

4) Base Cases

```
if (i==0 || j==0)
    return 1
```

```
dp[n][m] = {-1}

for (int i=0; i<n; i++)
    dp[i][0] => 1;

for (int j=0; j<m; j++)
    dp[0][j] = 1;

int ways (int i, int j) {

    if (dp[i][j] != -1)
        return dp[i][j];

    dp[i][j] => ways(i-1, j) + ways(i, j-1);

    return dp[i][j];

}
```

TC: $O(m*n)$
SC: $O(m*n)$

Bottom Up: TC: $O(m*n)$
SC: $O(\min(m,n))$

10:24pm

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 2 | 0 | 1 | 1 | 1 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |

```
int ways (int i, int j) {

    if (dp[i][j] != -1)
        return dp[i][j];

    if (mat[i][j] == 0) {
        dp[i][j] = 0
        return dp[i][j];
    }
    dp[i][j] => ways (i-1, j) + ways (i, j-1);

    return dp[i][j].

}
```

TC: O(m*n)

SC: O(m*n)

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 1 | 1 |

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 2 |
| 4 | 0 | 0 | 1 | 3 |

Min Cost

Path

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 2 | 1 | 3 | 7 | 3 |
| 1 | 3 | 3 | 4 | 8 | 3 |
| 2 | 0 | 5 | 1 | 2 | 1 |
| 3 | 0 | 2 | 3 | 1 | 3 |
| 4 | 4 | 1 | 5 | 2 | 3 |

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

Dungeon Princess