① Determine LCS of 10010101 and 010110110

→ We create 8×8 array giving C[m,n], the length of first 'm' of 1st sequence and 'n' of second sequence.

-	-	0	1	0	1	1	0	1	1	0
-	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1	1
0	0	1	1	2	2	2	2	2	2	2
0	0	1	1	2	2	2	3	3	3	3
1	0	1	2	2	3	3	3	4	4	4
0	0	1	2	3	3	3	4	4	4	5
1	0	1	2	3	4	4	4	5	5	5
0	0	1	2	3	4	4	5	5	5	6
1	0	1	2	3	4	5	5	6	6	6

→ Length is 6. Start at bottom right and walk until hitting the edge. At (i,j) go diagonal left if

$C[i,j] = C[i-1, j-1] + 1$.

If not go left on up whichever is C[i,j].

		0	1	0	1	1	0	1	1	0
	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1	1
0	0	1	1	2	2	2	2	2	2	2
0	0	1	1	2	2	2	3	3	3	3
1	0	0	2	2	3	3	3	4	4	4
0	0	1	2	3	3	3	4	4	4	5
1	0	1	2	3	4	4	4	5	5	5
0	0	1	2	3	4	4	5	5	5	6
1	0	1	2	3	4	5	5	6	6	6

→ The places where we go diagnolly left are the same in both sequences and these give the common sequence 010101.

---

(3)

```
PRINTLCS( c,x,Y, i,j)
{
    if C[i,j] == 0
        return;
    if x[i] == y[j]
        PRINTLCS (C,x,Y, i-1,j-1);
        print (x[i])
    else if C[i-1,j] > C[i,j-1]
        PRINTLCS (C,x,Y, i-1,j)
    else
        PRINTLCS(C,x,Y, i,j-1)
}
```

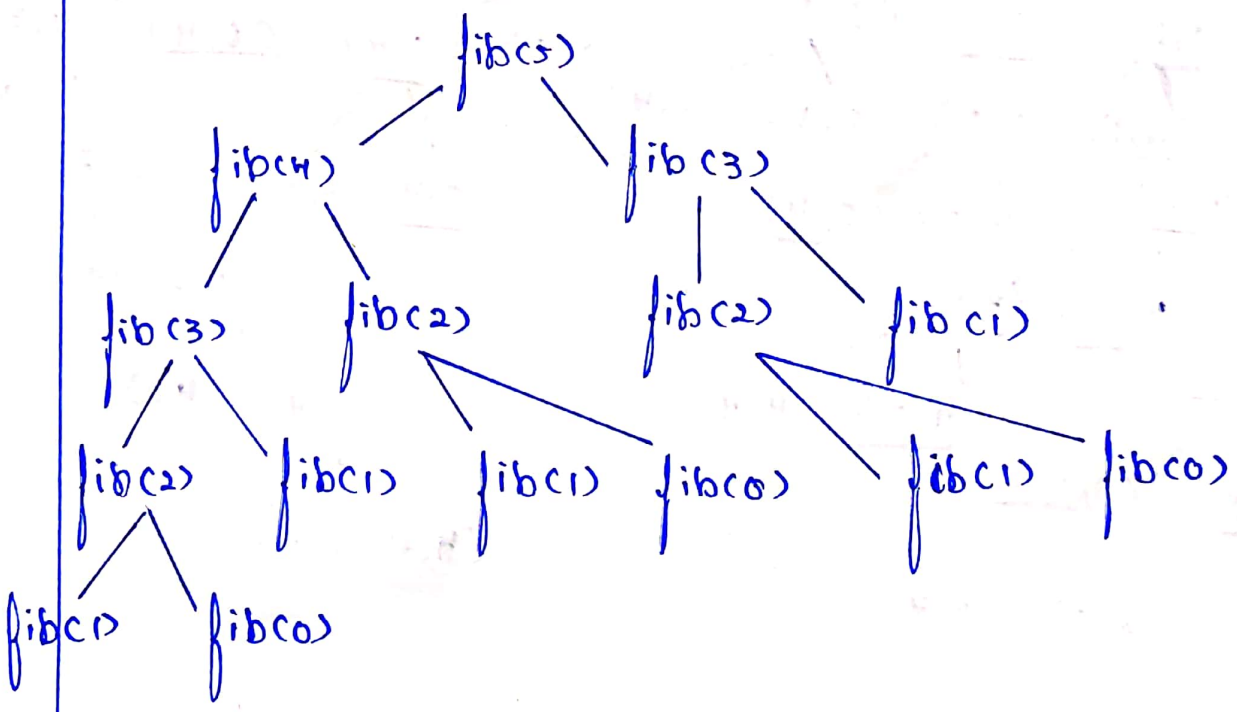
(5) Overlapping subproblem using DP.

→ Used when solutions of same sub problems are needed again and again.

Eg: Fibonacci Numbers

```
fib( int n)
{
    if (n<=1)
        return n;
    return fib(n-1) + fib(n-2);
}
```

Recursion tree of fib(5)



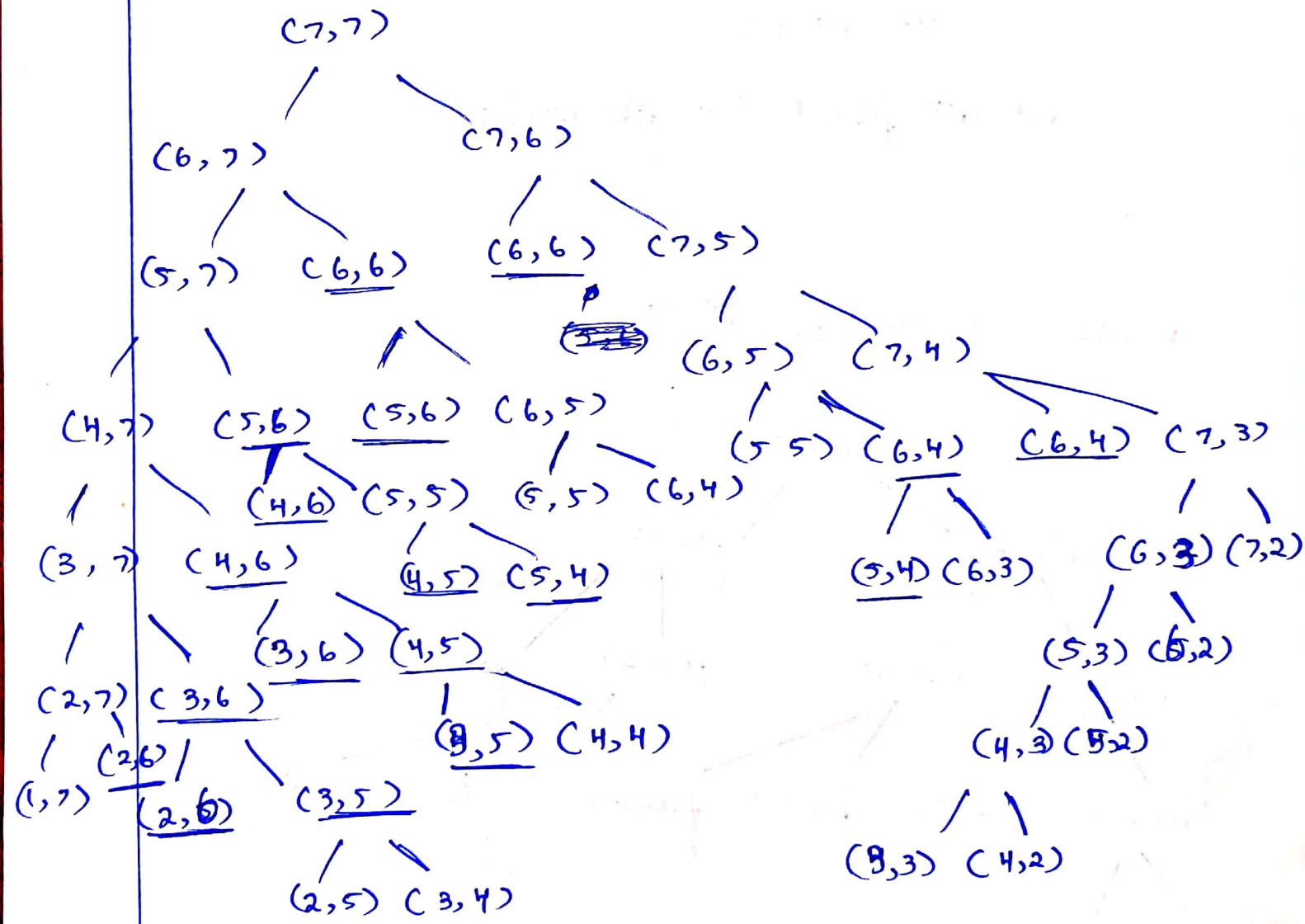
fib(3) called 2 times. So if we store fib(3) once, it can be used without computing it again.

(6) LCS:

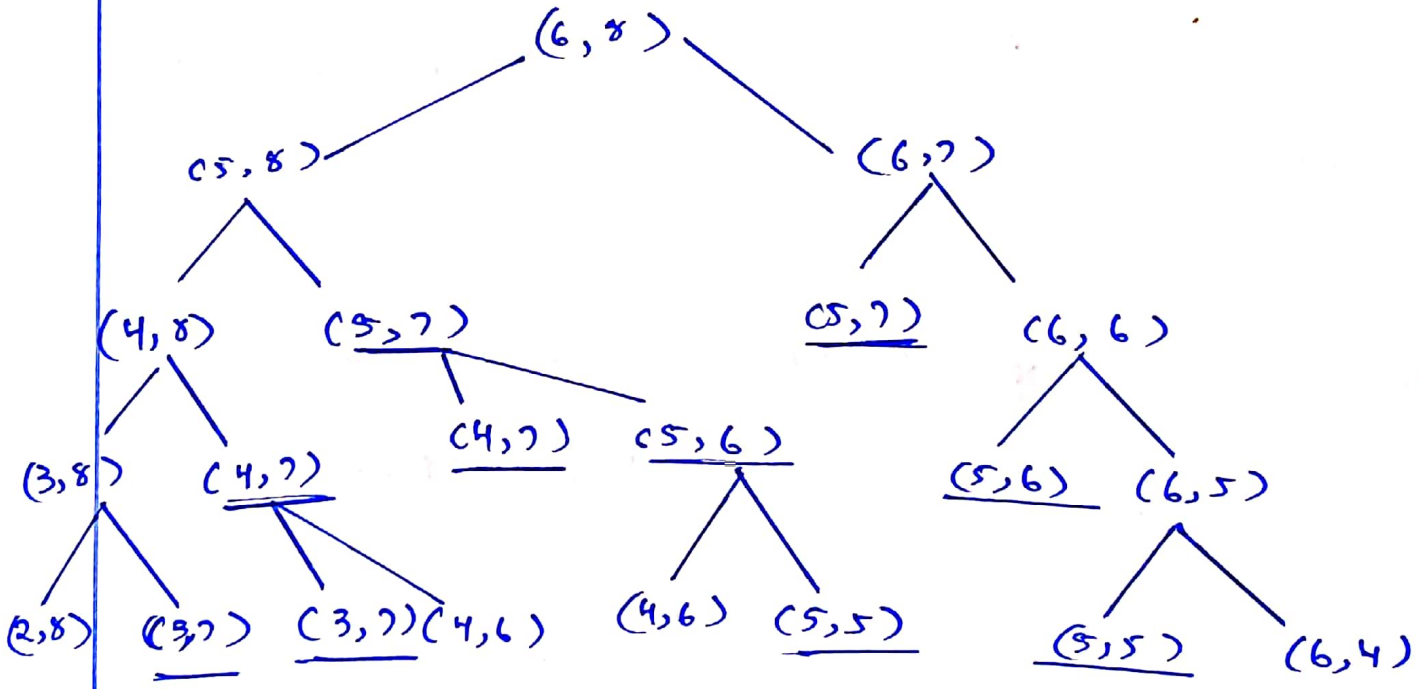
→ Find longgest subsequence that is present in given two sequence in same order.

Eg : XMJYAUZ , MZJAWXU.

X = XMJ YAUZ Y = MZJAWXU.



		0 ϕ	1 m	2 Z	3 J	4 A	5 W	6 X	7 U
0	ϕ	0	0	0	0	0	0	0	0
1	X	0	0	0	0	0	0	1	1
2	m	0	1	1	1	1	1	1	1
3	J	0	1	1	2	2	2	2	2
4	Y	0	1	1	2	2	2	2	2
5	A	0	1	1	2	3	3	3	3
6	U	0	1	1	2	3	3	3	4
7	Z	0	1	2	2	3	3	3	4



LCS : MJAU

length of LCS = 4

(7) Difference between Memorization and Tabulation.

Memorization	Tabulation
Using recursion to solve the subproblem and storing it, so that next time when we need to actually compute that again, we can take it from our stored cache.	Using iterative approach to solve the problem by solving the smaller subproblem first and then using it during the execution of bigger problem.
→ It is a Top-Down approach.	→ It is a bottom top approach.