DP on Strings:

Longest Common Subsequence } we have to find the length of Longest subsequence

s1 = "a dc bc" → Can generate 32 subsequence

s2 = "dcadb"

So, Output is 3

Common Subsequence:

s1		s2
a	Common	d
d		c
c		a
adb		d
b	Common	b
c		dc
ad		dc
ac		.
ab		adb
ac		
.		
.		

Which the Longest Common Subsequence

adb → size/length of "adb" string is 3

Problem Statement :

Common :

match

not match

Problem Statement

Suggest Edit

Given two strings, 'S' and 'T' with lengths 'M' and 'N', find the length of the 'Longest Common Subsequence'.

For a string 'str'(per se) of length K, the subsequences are the strings containing characters in the same relative order as they are present in 'str,' but not necessarily contiguous. Subsequences contain all the strings of length varying from 0 to K.

Example :

```
Subsequences of string "abc" are:  ""(empty string),
a, b, c, ab, bc, ac, abc.
```

Detailed approach of the problem:

let say, we have 2 strings.

s1 = acd
s2 = ced

so longest Common Subsequence is cd = 2

Brute approach would be generate all subsequence of s1 and s2 and find the common between them and then find the longest subsequence

So, by this approach the time Complexity : is $O(2^N) + O(2^N)$ which is exponential

The approach is somewhat current but, either by generate subsequence differently why not simultaneously generate and check common subsequence between subsequence.

So. Now.

We know that we will use recursive as we have to try all possible ways

So. Question arises ~~wh~~ how we generate subsequence

There is a simple concept of take and not take

Here let see,

we have $S1 = acd$ $S2 = ced$

in recursive what we take indx.

Here we have 2 string of different length so, There would be 2 indx

$f(ind1, ind2)$

Question is when we will consider the character and when not ?

When we find common character means we consider it its obvious.

So, let say.

$f(2, 2)$

acd ccd

similar match

$if(s1[ind1] == s2[ind2])$ $1 + f(ind1-1, ind2-1)$

After firstIndx

acd   ced

notmatch

but in future there can be a character which matches

that is concept of subsequence

acd   ced

Once ind1

send ind2

So, there are 2 cases one

1st Case It

move ind1 by

1

2nd Case

we move ind2 by 1

if not match

not match = $0 + \max(f(ind1-1, ind2), f(ind1, ind2-1))$

why? → Because we want longest common subsequence

return max(match, not match),

Base Case:

if (ind1 2 2

Recursive function.

f(ind1, ind2)

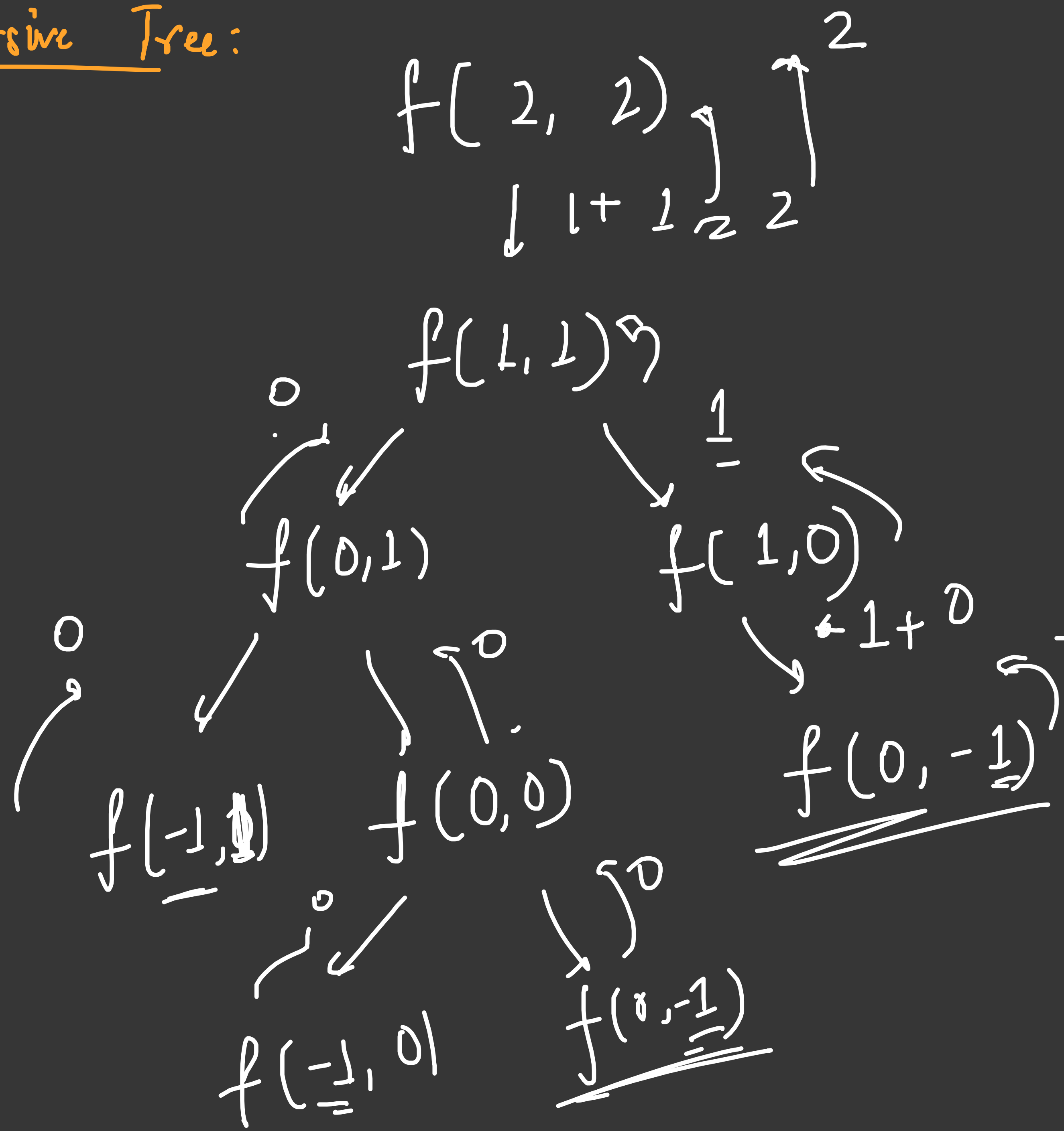
Base case: if (ind1 < 0 || ind2 < 0) return 0;

if (s1[ind] == s2[ind2])

return 1 + f(ind-1, ind2-1);

return max(f(ind1-1, ind2), f(ind1, ind2-1))

Recursive Tree:



a c d

c e d

ans = 2

fono

large input

there would be overlapping subproblem

Memoization.

For memoization we observe changing variable

(ind1, ind2)

```
vector<vector<int>> dp(ind1, vector<int>(ind2, -1))
```

T.C: $O(N \times M)$ S.C: = $O(N \times M) + O(N+M)$

Tabulation:

Base Case: Shifting of index

Base Case changes to if ($\underbrace{i == 0}_{dp[0][j]}$ || $\underbrace{j == 0}_{dp[i][0]}$)

return 0;

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

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