

Longest Common Subsequence (Recursion) :-

(LCS)

Given two strings, find the length of longest subsequence present in both of them. Both the strings are in uppercase latin alphabets.

ex 1)

A = G ; B = G

str1 = ABCDGH

str2 = AEDFHR

[output : 3]

explanation: LCS for input strings "ABCDGH" and "AEDFHR" is "ADH" of length 3.

ex 2)

x: a b c d g h

y: a b e d f h r

in subseq → abdh → common

Recursion:-

- 1) Base Condition
- 2) choice diagram.
- 3) Ip- small.

3) Ip small

x: a b c d g h

y: a b e d f h r

last index pr decision lenge lena hoga

nahi.

after that small size pr call krge

Small

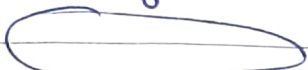
x: a b c d g

y: a b e d f h

case

Base condition

think of smallest valid input:

x:  n=6y:  m=5.

Small string & small string empty string

n → 0

m → 0

Box

x: empty

y: empty

CS = empty

if (n == 0 || m == 0)

return 0;

}

choice diagram:

[int] fun (Tp)

{ [Base condition]

[choice diagram]

}

→ case 1 when both last element is equal.like:

x: abcdgh

y: abedfh

if (x[n-1] == y[m-1])

x → n-1;

y → m-1;

{

case 2

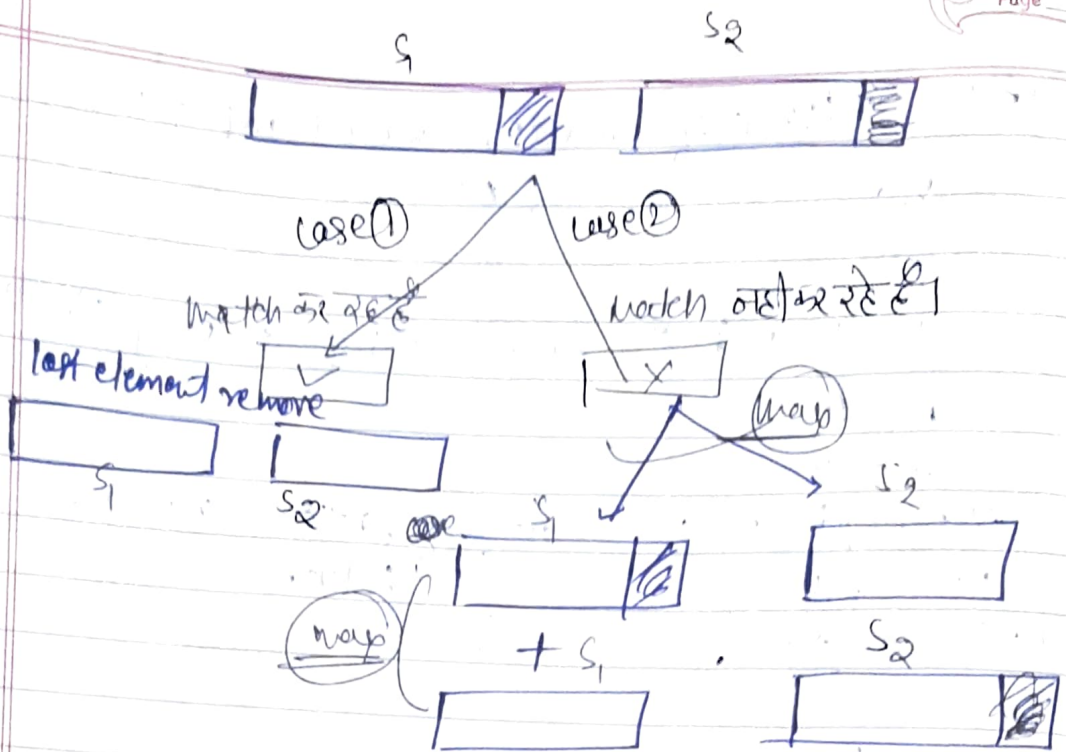
when last element is not equal:-

x: abcdgh

y: abedfh

thendon't string is small
or don't is maximum

max(abcdg, abedfh), (abcdgh, abedfh)



int lcs (string X, string Y, int n, int m)

Base Condition \rightarrow if $(n=0 \text{ or } m=0)$
 $\{$ return 0;
 $\}$

case 1: if $(X[n-1] == Y[m-1])$

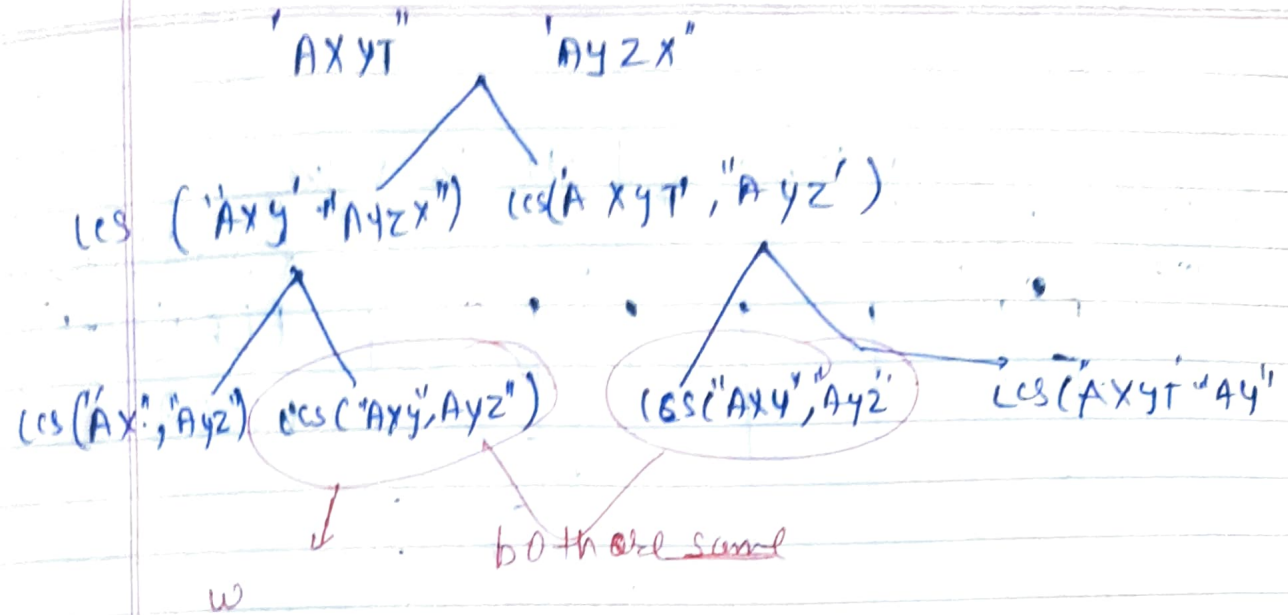
return $1 + \text{lcs}(X, Y, n-1, m-1);$

case 2: else

return $\max(\text{lcs}(X, Y, n, m-1), \text{lcs}(X, Y, n-1, m))$

}

LCS memoized (Bottom up dp)



⇒ m, n change & it's only Recursion H.
 don't pass copy → pass with address

```
int LCS(string x, string y, int m, int n)
```

```
{
```

```
    if (n == 0 || m == 0) return 0;
```

```
    if (dp[m][n] != -1) return dp[m][n];
```

```
    if (x[m-1] == y[n-1])
```

```
        return dp[m][n] = 1 + LCS(x, y, m-1, n-1);
```

```
    else
```

```
        return dp[m][n] = max(LCS(x, y, m, n-1),
```

```
                                LCS(x, y, m-1, n));
```

```
}
```

```
int main()
```

```
{
```

```
    int dp[m+1][n+1];
```

```
    memset(dp, -1, sizeof(dp));
```

```
    return LCS(x, y, m, n);
```


Top-down DP LCS

We take a table and store the value:-

X = abef $\rightarrow m=4$
Y = abdaf $\rightarrow n=6$

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

f(m+1)(n+1)

f(m)(n+1)

f(m+1)(n)

X: abef

Y: abdaf

ans = 4

X: ab

Y: abcd

LCS len $\Rightarrow 2$

X: abef

Y: ab

LCS len $\Rightarrow 2$

Base condition

if (i==0 || j==0)
return 0;

m \rightarrow i
n \rightarrow j
change

Choice diagram

if (X[n-1] == Y[m-1])

return 1 + LCS(X, Y, n-1, m-1);

else
return max (LCS(X, Y, n, m-1),
LCS(X, Y, n-1, m));

for (int i=0; i<n+1; i++)
for (int j=0; j<m+1; j++)

if (i==0 || j==0) dp[i][j] = 0;

for (int i=1; i<n+1; i++)
for (int j=1; j<m+1; j++)

if (X[i-1] == Y[j-1])

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

else
dp[i][j] = max(dp[i-1][j],
dp[i][j-1]);

return dp[n][m];