**Pattern: DP on strings**

**Approach: matching- not matching**

**DP25:** LCS [Longrest Common Subsequence](https://leetcode.com/problems/longest-common-subsequence/)

**Subsequence**: it may or may not consecutive but has the same order of characters.

**Recursion:**

* f(ind1, ind2, text1, text2)
* Matching- not matching
* return the length of the LCS

=> start from the back: text1= abcde text2= bcde

* Matching- one match we have add that and move to the prev index in both the strings and try finding the matches.

**1 + f(ind-1, ind2-1)**

* Not-matching- from this point we are not sure moving which pointer is beneficial, so get the max of both of these possibilities-

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

=> Base case: we are matching the characters in the string (a-z) but what if any of the string get exhausted then in that case we can not get more natching characters.

if(ind1<0 || ind2<0) return 0;

**Tabulation Approach**:

* Use base cases to form the base of the table
* Use the recurrence relation of matching and not matching to form the entire dp table.
* Return the LCS that is stored at dp[ind1-1][ind2-1]

=> Shifting of the index- we will use base case to build the base of our dp table, in the base case we have seen if (ind1<0 || ind2<0) return 0; but in the table we don’t have -1 so

* to consider -1 in the table we will make a table of size m+1, n+1 so that index 0 can be treated as -1 and 1 can be treated as 0 and so on.
* Ind1/ ind2 in the dp table means ind1-1/ ind2-1 in the text1/ text2

=> base case: fill 0th row & 0th col with zeros (acc to base case ind1<0 || ind2<0)

=> forming the table: fill table from 1st row and 1st col-

* Matching: (text1[ind1-1] == text2[ind2-1]) dp[i][j] = 1+ dp[i-1]][j-1]
* Not- matching: dp[i][j] = max(dp[i-1][j], dp[i][j-1])

**Example-**

**text1 = abcde, text2 = bdgek**

**o/p: 3 (bde)**

b d g e k

| - | 0 | 0 | 0 | 0 | 0 | 0 |
| --- | --- | --- | --- | --- | --- | --- |
| a | **0** | 0 | 0 | 0 | 0 | 0 |
| b | 0 | **1** | 1 | 1 | 1 | 1 |
| c | 0 | 1 | 1 | 1 | 1 | 1 |
| d | 0 | 1 | 2 | 2 | 2 | 2 |
| e | 0 | 1 | 2 | 2 | 3 | 3 (LCS) |