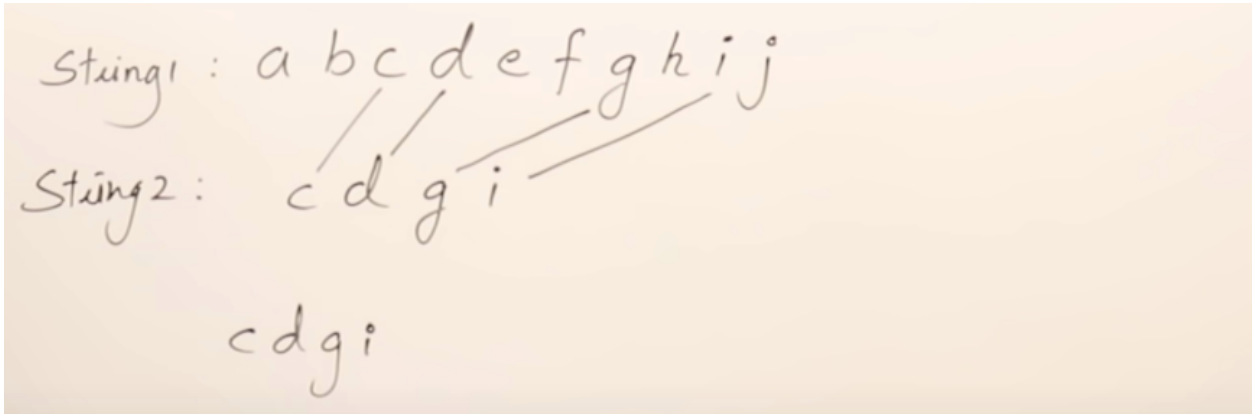
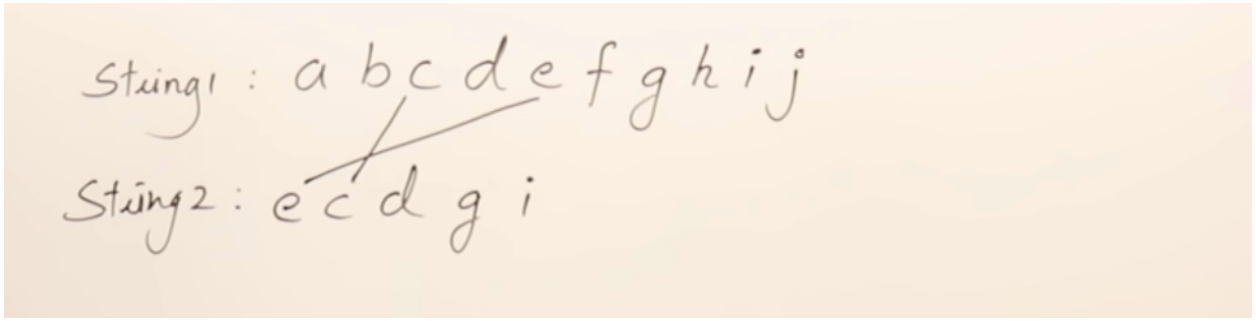
Longest Common Subsequence(LCS)

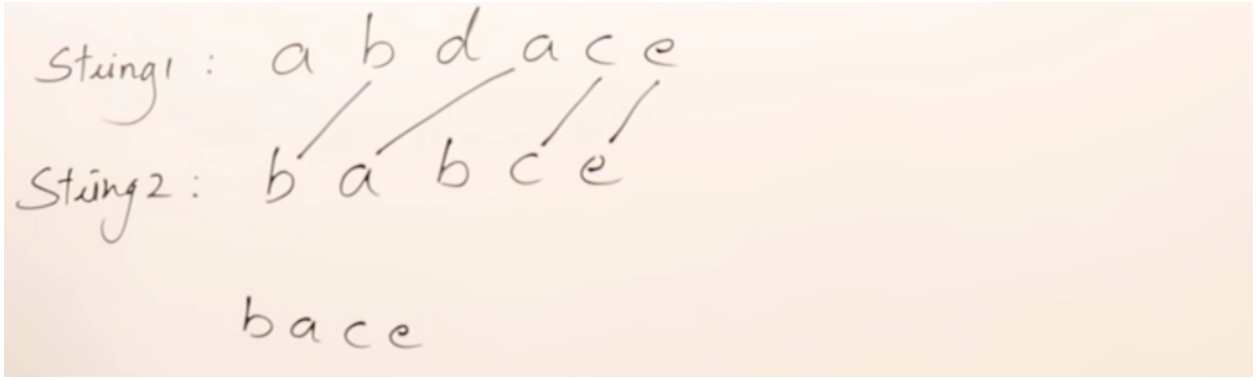
*subsequence mean Order wise but not contiguous



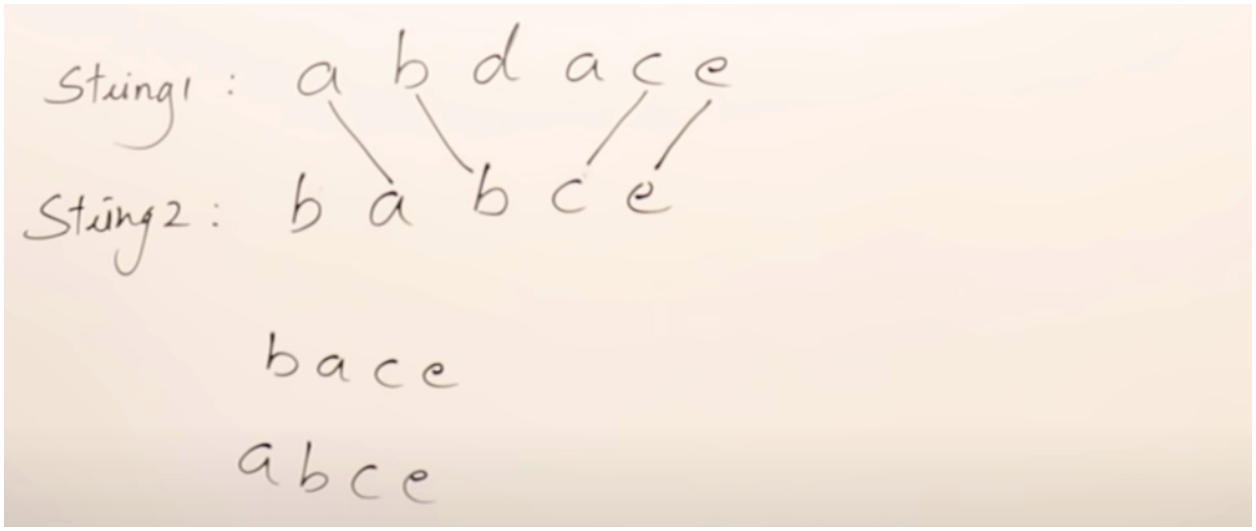
***Intersection line not allowed**



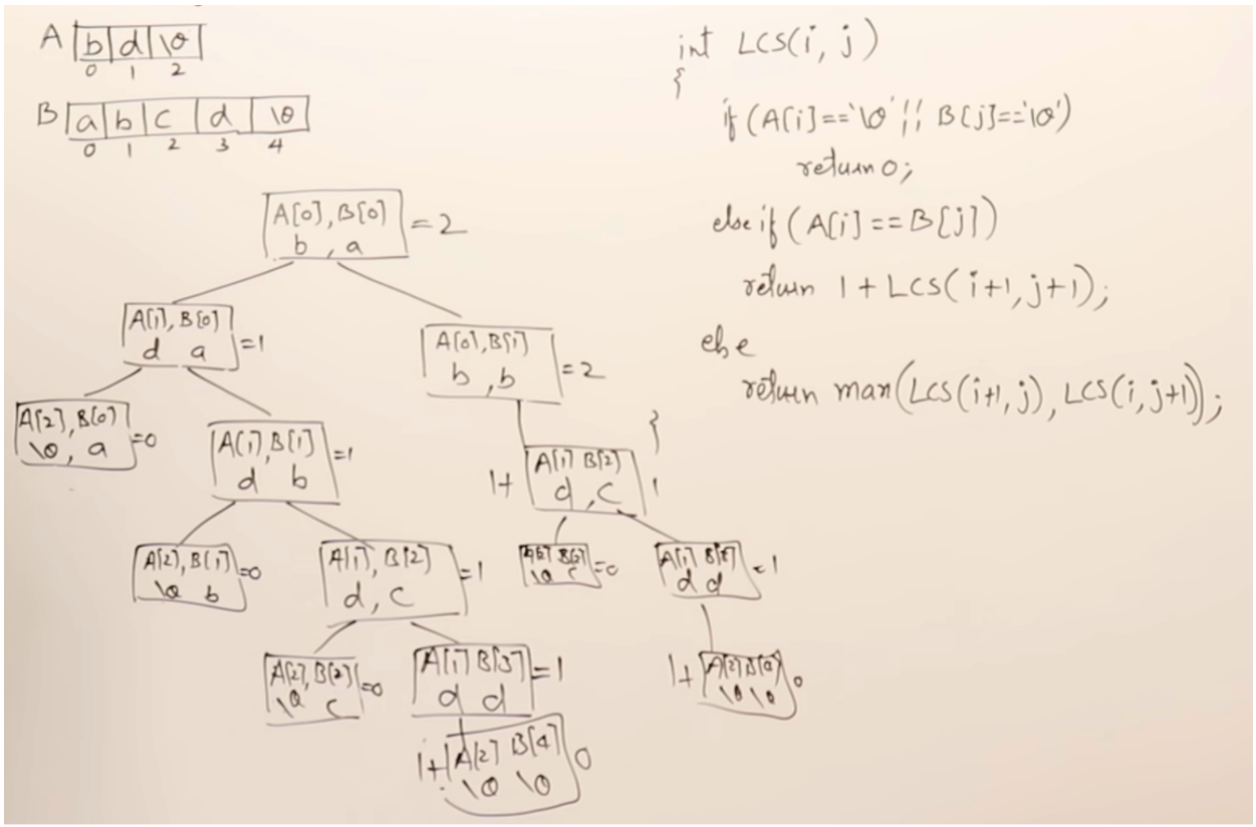
Example 1:



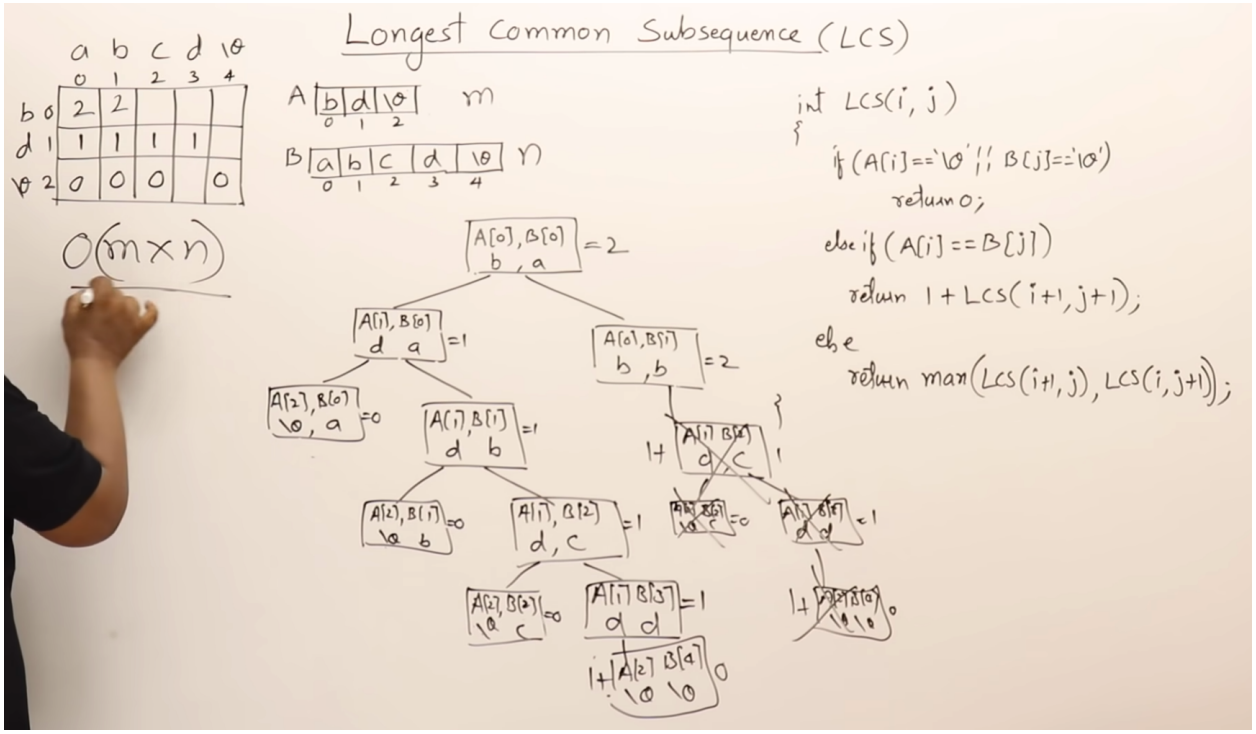
Example 2:



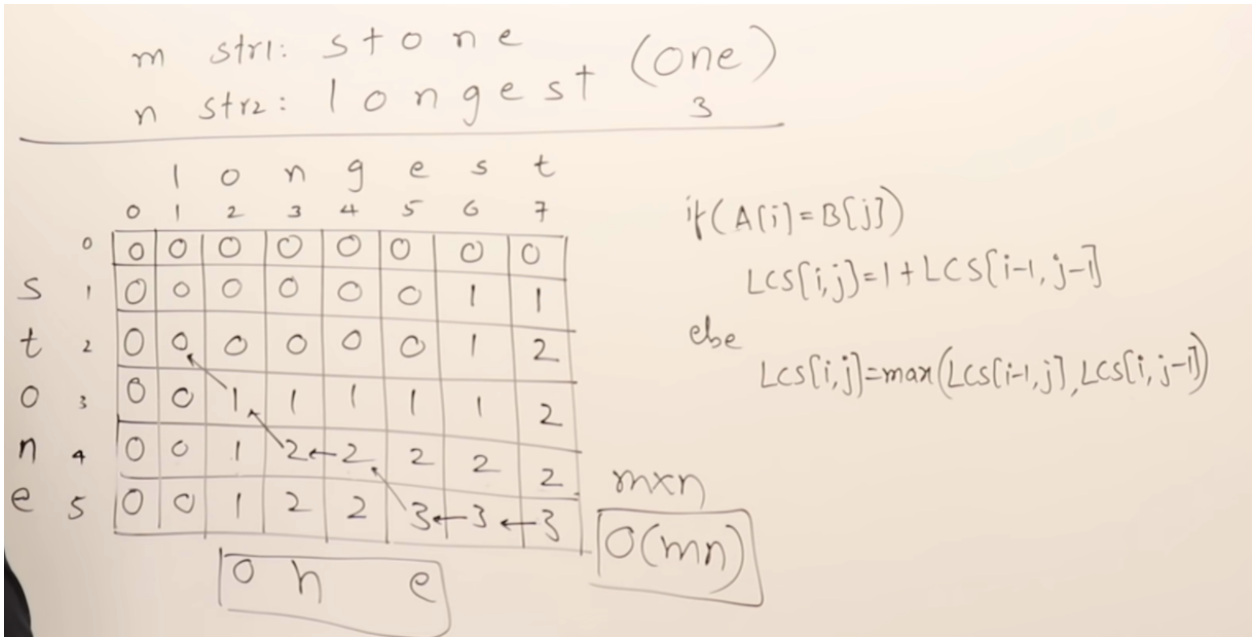
Longest Common Subsequence(LSC) Recursion Method:



LSC - Using Dynamic Programming (Memoization) Method:



Example 1:



LSC - Using Dynamic Programming Implementation:

```
/* Returns length of LCS for X[0..m-1], Y[0..n-1] */
   int lcs(char[] X, char[] Y, int m, int n)
   {
       int L[][] = new int[m + 1][n + 1];

       /* Following steps build L[m+1][n+1] in bottom up fashion. Note
        that L[i][j] contains length of LCS of X[0..i-1] and Y[0..j-1] */
       for (int i = 0; i <= m; i++) {
           for (int j = 0; j <= n; j++) {
               if (i == 0 || j == 0)
                   L[i][j] = 0;
               else if (X[i - 1] == Y[j - 1])
                   L[i][j] = L[i - 1][j - 1] + 1;
               else
                   L[i][j] = max(L[i - 1][j], L[i][j - 1]);
           }
       }
       return L[m][n];
   }

   /* Utility function to get max of 2 integers */
   int max(int a, int b)
   {
       return (a > b) ? a : b;
   }
```

```
public static void main(String[] args)
   {
       String s1 = "AGGTAB";
       String s2 = "GXTXAYB";

       char[] X = s1.toCharArray();
       char[] Y = s2.toCharArray();
       int m = X.length;
       int n = Y.length;

       System.out.println("Length of LCS is"
                          + " " +lcs(X, Y, m, n));
   }
```

Output: Length of LCS is 4