Longest common subsequence(lcs)

`lcs(A, m, B, n)` = length of longest common subsequence in $A[0..m), B[0..n)$.

Example: $A = \{1, 2, 3, 6, 5, 4, 7\}, B = \{7, 1, 2, 6, 5, 4, 3\}$. An lcs is $\{1, 2, 6, 4\}$ of length 4.

Recursive solution:

$$lcs(m,n) = \begin{cases} 0, & \text{if } m == 0 \text{ or } n == 0, \\ 1 + lcs(A, m-1, B, n-1), & \text{if } (A[m-1] = B[n-1], \\ max \begin{pmatrix} lcs(A, m-1, B, n), \\ lcs(A, m, B, n-1) \end{pmatrix}, & \text{otherwise.} \end{cases}$$

This works because if the last item in A and the last in B are equal then there is always a longest common subsequence ending at this item. (proof left to reader). Thus the second clause is correct by induction when the last items are equal. When the last items of A and B are unequal, at least one of them will not be in the longest common subsequence, so induction implies the third clause is correct. (We are inducting on m + n. So the hypothesis of inductive correctness applies to m-1 + n-1, to m-1 + n, and to m + n-1.)

Dynamic programming on the variables m, n gives a $\Theta(mn)$ time and memory algorithm. An implementation is in lcs.C, attached.