Kent Quanrud's Class 2.1.2025

Problem 5.2 #1

`def common(`i`, `j`):`

""" Given global lists $A[1\dots n]$ and $B[1\dots n]$ determine the length of the longest subsequence of $A[1\dots i]$ that is also a subsequence of $B[1\dots j]$ """

1. If $ij = 0$, return 0.
2. If $A[i] = B[j]$, return $1 +$ `common(`$i-1, j-1$`)`.
3. Otherwise, return $\max($`common(`$i, j-1$`)`, `palindrome(`$i-1, j$`)`$)$.

Then, to answer the question, just evaluate `common(`n, n`)`.

Time Complexity: If the calls to the function are stored in a dictionary, then there are n^2 distinct subproblems, each using $O(1)$ checks, so the time complexity is $O(n^2)$.

Problem 5.2 #2

The answer is $2n -$ `common(`n, n`)`.

Consider any minimal supersequence and place $2n$ markers corresponding to the indices where the sequences A and B are embedded. If we look at just the indices with both an a and a b marker on it, these form a common subsequence of A and B, so there are at most $c =$ `common(`n, n`)` shared locations.

It follows that there are at least $(n-c)+(n-c)+c = 2n-c$ indices with markers on them, so the supersequence must have length at least $2n-c$. It is very easy to greedily construct such a supersequence, so this returns the right answer.

Problem 5.2 #3

`def sharedpal(`L,R,X,Y`):`

""" Given global lists $A[1 \dots n]$ and $B[1 \dots n]$ determine the length of the longest palindromic subsequence of $A[L \dots R]$ that is also a subsequence of $B[X \dots Y]$. """

1. If $R < L$ or $Y < X$, return 0.
2. If $L = R$ and $X = Y$, if $A[L] = B[X]$ return 1, and otherwise, return 0.
3. If $A[L] = A[R] = B[X] = B[Y]$, return $2+$`sharedpal(`$L+1$, $R-1$, $X+1$, $Y-1$`)`.
4. Otherwise, return the maximum of
   - `sharedpal(`$L+1$, R, X, Y`)`,
   - `sharedpal(`L, $R-1$, X, Y`)`,
   - `sharedpal(`L, R, $X+1$, Y`)`, and
   - `sharedpal(`L, R, X, $Y-1$`)`.

The longest palindromic common subsequence of the original A and B is `sharedpal(`1, n, 1, n`)`.

Time Complexity: If the calls to the function are stored in a dictionary, then there are n^4 distinct subproblems, each using $O(1)$ checks, so the time complexity is $O(n^4)$.