CSE 101 Homework 2
Due Oct. 22
Divide-and-Conquer
100 points total = 10 %

Short answers: p.496-501 of text 24 b, 25 f (5 points each)

Order: 10 points Show that for any constants \( a, b > 1, \log_a n \in \Theta(\log_b n) \).
Does it follow that for any \( a, b > 1, 2^{\log_a n} \in \Theta(2^{\log_b n}) \)?

Divide and Conquer Recurrence: 10 points Give and solve up to order a recurrence relation for the time taken by the following divide-and-conquer algorithm.

We are given an array of real numbers \( V[1..n] \). We wish to find a subset of array positions, \( S \subseteq [1..n] \) that maximizes \( \sum_{i \in S} V[i] \) subject to no two consecutive array positions being in \( S \). For example, say \( V = [10, 14, 12, 6, 13, 4] \), the best solution is to take elements 1, 3, 5 to get a total of \( 10 + 12 + 13 = 35 \). If instead, we try to take the 14 in position 2, we must exclude the 10 and 12 in positions 1 and 3, leaving us with the second best choice 2, 5 giving a total of \( 14 + 13 = 27 \). The recursive procedure is based on a case analysis, do we pick position \( n/2 \) or not? The algorithm just finds the best sum, not the set of positions, but it would be easy to modify. RMNCS stands for Recursive Maximum Non-consecutive Sum.

\( \text{RMNCS}[V[1..n]] = \text{array of positive reals]; real number} \;
1. IF \( n = 0 \) return 0;
2. IF \( n = 1 \) return \( V[1] \);
3. IF \( n = 2 \) return \( \max(V[1], V[2]) \);
4. IF \( n = 3 \) return \( \max(V[2], V[1] + V[3]) \);
5. \( k \leftarrow \lfloor n/2 \rfloor \);
6. \( \text{Sum1} \leftarrow \text{RMNCS}[V[1..k-2]] + V[k] + \text{RMNCS}[V[k+2..n]] \) \{Case 1: If we include \( V[k] \), we cannot include \( V[k-1] \) or \( V[k+1] \)\}
7. \( \text{Sum2} \leftarrow \text{RMNCS}[V[1..k-1]] + \text{RMNCS}[V[k+1..n]] \) \{Case 2: If we do not include \( V[k] \), we can include any other positions.\}
8. Return \( \max(\text{Sum1}, \text{Sum2}) \).
Polynomial multiplication - 15 points. Describe and analyze a divide-and-conquer algorithm for polynomial multiplication: Given \( p(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_0 \) and \( q(x) = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \ldots + b_0 \) compute \( p(x)q(x) \). (Assume \( p \) and \( q \) are given as arrays \( a[0..n-1] \), \( b[0..m-1] \) of their coefficients, and output \( p(x)q(x) \) as an array \( c[0..2n-2] \) of coefficients.) Can you use the ideas for integer multiplication, as in prod2 in the text, to get a faster algorithm?

Depths in a tree (20 points) Consider the following recursive algorithm: We are given a tree \( T \), each node having a parent \( p[x] \) pointer and a list of children \( Children(x) \). We wish to find the depth of each node in the tree, where depth is defined to be the number of intermediate nodes between it and the root. Don't assume \( T \) is a binary tree or that it is balanced.

The algorithm to do so is as follows:
\[ DepthNumber(root) = Depths(root, 0) \] where:
1. \( Depths(x, I) \) {compute all depths in the sub-tree rooted at \( x \), assuming \( depth(x) = I \).}
2. begin;
3. \( D[x] := I; \)
4. For each \( y \in Children(x), Depths(y, I + 1); \)
5. end;

Give a (probably, inductive) proof to show that this algorithm is time \( O(n) \) for any tree.

Implementing Multiplication - 15 points Implement both a naive integer multiplication algorithm and \texttt{prod2} from the text. Compare their performance on integers of length powers of 2, say between \( 2^6 \) and \( 2^{12} \). Plot the performance on a log - log scale. Which is better for the lengths in your range? What is the slope of the curve for both methods? Where do you expect \texttt{prod2} to be better, if anywhere?