CSE 101 Homework 4
Dynamic Programming
Due June 5, 2 PM
100 points total = 10 %

Longest Increasing Sequence Let $A[1..n]$ be an array of integers. An *increasing sub-sequence* of A is a subset of positions in the array $I = \{i_1 < i_2 < ... < i_k\}$ with $A[i_1] < A[i_2] < .. < A[i_k]$, i.e., looking at these positions only, the resulting subarray is sorted in increasing order. The problem we are considering is to find the length of the largest increasing sub-sequence that contains the first position, i.e. with $i_1 = 1$

Last assignment, you were asked to give a back-tracking algorithm for this problem. Here is a recursive, back-tracking algorithm to find the size of the largest increasing sub-sequence that has the additional constraint that it contains the first element. It is based on a case analysis: Which is the next element in the sub-sequence? In other words, we are given $i_1 = 1$, now what is i_2? (If there is a second position.)

LIS[A[1..n]:array of integers]: integer

1. IF $n = 1$ return 1.
2. $MaxLength \leftarrow 1$. {If the sequence only contains the first element}
3. FOR I=2 TO n do:
4. begin;
5. $TempLength \leftarrow 0$
6. IF $A[I] > A[1]$ THEN $TempLength \leftarrow 1 + LIS[A[I..n]]$. {If $A[I]$ can be the second element in an array, find the length of the best sequence in the case $i_2 = I$.}
7. IF $TempLength > MaxLength$ THEN $MaxLength \leftarrow TempLength$.
8. end;
9. Return MaxLength.

Part 1: 2 points Show the recursion tree of the above algorithm on the following input: $A[1...2] = [1, 7, 2, 3, 5]$.

Part 2: 3 points Give a bound on the worst-case number of recursive calls the recursive algorithm could make in terms of n.

Part 3: 10 points Give a dynamic programming version of the recurrence.

Part 4: 3 points Give a time analysis of this dynamic programming algorithm.

Part 5: 2 points Show the array that your dynamic programming algorithm produces on the same example.

Descending partitions-20pts A descending partition of positive integer N is a sequence of positive integers $A_1 > A_2 > ... > A_k$ with $\sum_{i=1}^{i=K} A_i = N$. Give an efficient (poly-time in N) algorithm that, given N, computes the NUMBER of decreasing partitions of N. For example, if N=6, the decreasing partitions are: $(6); (5,1); (4,2); (3,2,1)$ so your algorithm, on input 6 should return 4.

Non-negative paths -20pts Let G be an n-node directed graph whose edges are labelled either -1 or 1. A path $e_1, ..e_t$ in G is *always non-negative* if the sum of the weights of e_1, up to e_i is non-negative for all $1 \leq i \leq t$. A source s and a destination s' are given as inputs, as well as G. Give a polynomial-time algorithm to decide whether there is an always non-negative path from s to s'. (Hint: solve the more general problem: What is the maximum sum of a non-negative path from s to t in at most l steps?) (7pts. correctness, 3 pts. efficiency.)

Library storage, 20 pts A library has n books that must be stored in alphabetical order on adjustable height shelves. Each book has a height and a thickness. The width of the shelf is fixed at W, and the sum of the thicknesses of books on a single shelf must be at most W. The next shelf will be placed on top, at a height equal to the maximum height of a book in the shelf. Give a dynamic programing algorithm that minimizes the total height of shelves used to store all the books. As always, include an informal correctness argument and a time analysis. You are given the list of books in alphabetical order, $b_i = (h_i.t_i)$, where h_i is the height and t_i is the thickness, and must organize the books in that order.

Implementation: 20 pts Implement, time and compare the running time of the back-tracking and dynamic programming versions of the Longest Increasing Sequence algorithm. Plot them on a log-log curve. For n as many powers of 2 as you can get to run, use n element arrays of random integers from 1 to n as your instances.