CS 473: Fundamental Algorithms, Fall 2014
HW 4 (due Tuesday, at noon, October 7, 2014)

Version: 1.11.

1. (35 PTS.) Longest calculation sequence.

A sequence of numbers $x_1, x_2, \ldots, x_u$ is a ***calculation sequence*** if for any $i \geq 3$, we have that x_i is equal to one of the following values:

(i) $x_{i-1} + x_{i-2}$,
(ii) $x_{i-1} - x_{i-2}$,
(iii) $x_{i-1} * x_{i-2}$, or
(iv) x_{i-1}/x_{i-2}.

You are given a sequence of numbers $y_1, \ldots, y_n$.

(A) (20 PTS.) Describe a recursive algorithm that finds the longest subsequence of the numbers that is a calculation sequence. What is the running time of the algorithm?

(B) (15 PTS.) Modify the above algorithm so that it becomes more efficient. What is the running time of the modified algorithm? (The faster, the better, naturally.)

2. (30 PTS.) LA heavy path is not hard to find.

You are given a binary tree T over n nodes, with weights on the edges, and a parameter k. Describe an algorithm, as fast as possible, that computes the path with k edges in T of maximum weight. How fast is your algorithm? (The faster, the better, naturally.)

3. (35 PTS.) The Tetris strikes back.

You are given a Tetris like board of width k and height n (here k is a small constant, say 5); that is, a grid of size $k \times n$. The board is initially empty. You are also given a sequence of pieces. You have to place the pieces on the board, one after the other.

Each piece is made out of four connected squares (intuitively, a piece is formed by gluing together four squares along common edges). Each piece appears in the top row of the board (i.e., you are given for each piece the location where exactly it is going to appear). You are allowed to move each piece either left, right or downward by one step (but not upward), as long as the grid squares used by its new locations are empty of other pieces. Furthermore, you can do this repeatedly till the piece arrives to its desired location. A ***legal*** final location for a piece is a location where it can not be moved downward immediately. Once a piece is placed in a final location, the next piece appears, and it has to be placed. This is repeated till all the pieces in the sequence are handled.

To make the problem simpler, assume you are not able to rotate the pieces, and furthermore, you can not use a cell if it is occupied, or any cell above it is occupied. Unlike the real Tetris game, no rows disappear if they are full.

The player wins the game if there is a way to place all pieces such that in no time a cell of the board is occupied by two pieces (i.e., usually the game ends when a new piece appears and one of its cells is already occupied).

(A) (5 PTS.) Given an $n \times k$ array B where $B[i, j] = 1$ if the cell in row i and column j is occupied and $B[i, j] = 0$ if not (so B is basically a configuration of the board), four cells $(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4)$ that represent a piece in its start location, and four cells $(a'_1, b'_1), (a'_2, b'_2), (a'_3, b'_3), (a'_4, b'_4)$ that represent the same piece in its end location, describe an algorithm that decides if one can move the piece (using allowable moves) from its start location to its end location. How fast is your algorithm?

(B) (30 PTS.) Describe an algorithm, as fast as possible, such that given such a sequence of m pieces with their start location (each represented by four cells), and an initially empty board, it decides whether the game is winnable. What is the running time of your algorithm? (The faster the better.)