Assignment 4

Foundations of Algorithms

1. [20 pts, data structures]
   Solve problem 10-1 (page 249)

2. [20 pts, AVL trees]
   Recall an AVL binary search tree is a BST where, for each node in the tree, the height of its children differs by no more than 1. For this problem, assume we have a team of biologists that keep information about DNA sequences in an AVL binary search tree using the specific weight (an integer) of the structure as the key. The biologists routinely ask questions of the type, "Are there any structures in the tree with specific weight between *a* and *b*, inclusive?" and they hope to get an answer as soon as possible. Design an efficient algorithm that, given integers *a* and *b*, returns True if there exists a key *x* in the tree such that $a \le x \le b$, and False if no such key exists in the tree. Describe your algorithm in English and pseudocode. What is the time complexity of your algorithm? Explain.

3. [30 pts, dynamic programming]
   In dynamic programming, a recurrence equation is required in order to represent the solution to the current problem using solutions from previous subproblems. This relation is a bottom-up dynamic programming algorithm, when we fill a table, such that all needed subproblems are solved before solving the current problem. (Hint: Boundary conditions can be set covering complete rows or columns, more than once)
   For each one of the following, determine and explain a valid traversal order, if one is possible. Otherwise, explain why it is not possible

   a) `A(i,j) = F( A(i,j-1), A(i-1,j-1), A(i-1,j+1) )`
   b) `A(i,j) = F( A(min{i,j}-1,min{i,j}-1), A(max{i,j}-1,max{i,j}-1) )`
   c) `A(i,j) = F( A(i-2,j-2), A(i+2,j+2) )`

4. [30 pts, Optimal BST]
   Given a search problem where some elements are searched more than others, it is more important to minimize the total cost of several searches rather than the worst-case cost of a single search. If x is a more frequent search target than y, building a tree where the depth of x is smaller than the depth of y will work better, even if that means increasing the overall depth of the tree. A perfectly balanced tree is not the best choice if some items are significantly more popular than others.
   Suppose we are given a sorted array of keys A[1..n] and an array of corresponding access frequencies f[1..n]. Build the binary search tree that minimizes the total search time, assuming that there will be exactly f[i] searches for each key A[i]. Suggest a recursive definition of the cost function, such that Cost(T, f[1..n]]=..., where T is the tree.