Midterm

Note

This is a closed-book exam. Each problem accounts for 10 points, unless otherwise marked.

Problems

1. Prove *by induction* that the regions formed by n circles in the plane can be colored with two colors such that any neighboring regions (which share an arc, not just a point) are colored differently.

2. Prove *by induction* that, for complete binary trees with three nodes or more, one of the two subtrees under the root is a full binary tree and the other is a complete binary tree. (Note: full binary trees are special cases of complete binary trees.)

3. Consider the following variant of Euclid's algorithm for computing the greatest common divisor of two positive integers.

```
Algorithm Euclid_Simplified ($m, n$);
begin
    // assume that $m > 0 \wedge n > 0$
    $x := m$;
    $y := n$;
    while $x \neq 0 \wedge y \neq 0$ do
        if $x < y$ then swap($x$,$y$);
        $x := x - y$;
    od
    ...
end
```

   where swap(x,y) exchanges the values of x and y.

   (a) (5 points) To speak about the values of a variable at different times during an execution, let m', n', x', and y' denote respectively the new values of m, n, x, and y after the next iteration of the while loop (m, n, x, and y themselves denote the current values of these variables at the start of the next iteration). Please give a precise relation between m', n', x', and y' and m, n, x, and y.

   (b) (10 points) Prove *by induction* that the following is a loop invariant of the while loop:

$$x \geq 0 \wedge y \geq 0 \wedge (x \neq 0 \vee y \neq 0) \wedge \gcd(x, y) = \gcd(m, n).$$

   (Note: by convention, $\gcd(0, z) = \gcd(z, 0) = z$ for $z > 0$.)

4. Consider the Knapsack Problem: Given a set S of n items, where the i-th item has an integer size $S[i]$, and an integer K, find a subset of the items whose sizes sum to exactly K or determine that no such subset exists.

   We have discussed in class two approaches to implementing a solution that we designed by induction: one uses dynamic programming (see the Appendix), while the other uses recursive function calls.

   Suppose there are 5 items, with sizes $2, 3, 5, 7, 8$, and we are looking for a subset whose sizes sum to 15. Assuming recursive function calls are used, please give the two-dimension table P whose entries are filled with -, O, I, or left blank when the algorithm terminates. Which entries of $P[n, K]$ are visited/computed more than once? Please mark those entries in the table.

5. Show all intermediate and the final AVL trees formed by inserting the numbers 6, 5, 3, 1, 2, and 4 (in this order) into an empty tree. Please use the following ordering convention: the key of an internal node is larger than that of its left child and smaller than that of its right child. If re-balancing operations are performed, please also show the tree before re-balancing and indicate what type of rotation is used in the re-balancing.

6. The input is a set S with n real numbers. Design an $O(n)$ time algorithm to find a number that is *not* in the set. Prove that $\Omega(n)$ is a lower bound on the number of steps required to solve this problem.

7. Design an efficient algorithm that, given an array A of n integers and an integer x, determine whether A contains two integers whose sum is exactly x. Please present your algorithm in an adequate pseudo code and make assumptions wherever necessary. Give an analysis of its time complexity. The more efficient your algorithm is, the more points you will be credited for this problem.

8. Consider rearranging the following array into a max heap using the *bottom-up* approach.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	3	7	1	2	8	14	6	11	4	10	13	12	9	15

   Please show the result (i.e., the contents of the array) after a new element is added to the current collection of heaps (at the bottom) until the entire array has become a heap.

9. (15 points) Below is a variation of the n-coins problem.

   > You are given a set of n coins $\{c_1, c_2, \ldots, c_n\}$, among which at least $n-1$ are identical "true" coins and at most one coin is "false". A false coin is either *lighter* or *heavier* than a true coin. Also, you are given a balance scale, which you may use to compare the total weight of any m coins with that of any other m coins. The problem is to find the "false" coin, or show that there is no such coin, by making some sequence of comparisons using the balance scale.

(a) For the case of $n = 12$, design a scheme to find the false coin (if there is one) with only three comparisons using the balance. Please use c_1, c_2, …, c_{12} to identify the coins in your scheme.

(b) Prove that, when $n = 12$, it is not possible to find the false coin with just two comparisons, implying that using just three comparisons is optimal. (Hint: think about decision trees and how many possible outcomes there can be.)

(c) To generalize the preceding result, prove that in the worst case it is impossible to solve the n-coins problem with k comparisons (for any n and k) if $n > \frac{(3^k-1)}{2}$.

Appendix

- Below is an algorithm for determining whether a solution to the Knapsack Problem exists.

```
Algorithm Knapsack ($S, K$);
begin
    $P[0,0].exist := true$;
    for $k := 1$ to $K$ do
        $P[0,k].exist := false$;
    for $i := 1$ to $n$ do
        for $k := 0$ to $K$ do
            $P[i,k].exist := false$;
            if $P[i-1,k].exist$ then
               $P[i,k].exist := true$;
               $P[i,k].belong := false$
            else if $k - S[i] \geq 0$ then
                    if $P[i-1,k-S[i]].exist$ then
                       $P[i,k].exist := true$;
                       $P[i,k].belong := true$
end
```