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 a planar graph all of whose vertices have even degrees can be colored with two colors such that no two adjacent regions have the same color.

2. For each of the following pairs of functions, decide if $f(n) = O(g(n))$ holds and if $f(n) = \Omega(g(n))$ holds? Justify your answers.

	$f(n)$	$g(n)$
(a)	$(\log n)^{\log n}$	$\frac{n}{\log n}$
(b)	$n^3 \cdot 2^n$	3^n

3. In the towers of Hanoi puzzle, there are three pegs A, B, and C, with n (generalizing the original eight) disks of different sizes stacked in decreasing order on peg A. The objective is to transfer all the disks on peg A to peg B, moving one disk at a time (from one peg to one of the other two) and never having a larger disk stacked upon a smaller one.
   (a) Give an algorithm to solve the puzzle. Explain how induction works here. (10 points)
   (b) Compute the total number of moves in the algorithm. Show the details of your calculation. (5 points)

4. Construct a gray code of length $\lceil \log_2 12 \rceil$ $(= 4)$ for 12 objects. Show how the gray code is constructed from gray codes of smaller lengths. Your construction should be systematic.

5. Show all intermediate and the final AVL trees formed by inserting the numbers 0, 1, 2, 3, 4, 9, 8, 7, 6, and 5 (in this order).

6. Apply the quicksort algorithm to the following array. Show the result after each partition operation.

7	1	5	11	14	12	2	15	8	3	13	4	10	9	16	6

7. Rearrange the following array into a heap using the buttom-up approach.

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

Show the result after each element is added to the part of array that already satisfies the heap property.

8. Write a program (or modify the following code) to recover the solution to a knapsack problem using the *belong* flag. You should make your solution as efficient as possible.

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

9. Compute the *next* table as in the KMP algorithm for the string *ababaababab*. Show the details of your calculation.

10. Explain why the time complexity of the KMP algorithm is $O(n)$, where n is the length of string A. (5 points)