Midterm

(April 27, 2000)

Note

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

Problems

1. Find a gray code of length $\lceil \log_2 14 \rceil$ $(= 4)$ for 14 objects. Show how the gray code is constructed systematically from gray codes of smaller lengths.

2. Below is a theorem from Manber's book:

   > For all constants $c > 0$ and $a > 1$, and for all monotonically increasing functions $f(n)$, we have $(f(n))^c = O(a^{f(n)})$.

   Prove, by using the above theorem, that $n^2(\log n)^2 = O(n^{2.25})$. (5 points)

3. 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). If a rotation is performed during an insertion, please also show the tree before the rotation.

4. Write a program (in pseudo code) to merge two skylines.
   An example skyline: (1,**9.2**,5.5,**11**,9,**0**,12.8,**6.6**,18,**15**,22.9).

5. The Knapsack Problem is defined as follows: Given a set S of n items, where the ith 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.

   Below is an algorithm for determining whether a solution to the 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] ≥ 0 then
                    if P[i-1,k-S[i]].exist then
                       P[i,k].exist := true;
                       P[i,k].belong := true
end
```

(a) Modify the algorithm to solve a variation of the knapsack problem where each item has an unlimited supply.

(b) Design an algorithm to recover the solution recorded in the array P of the preceding algorithm.

6. Below is the algorithm that we studied in class for determining, given a sorted array A of distinct integers, whether there exists an index i such that $A[i] = i$. The idea of the algorithm is good, but its pseudo code has a bug. Please identify the error and give an example input for which the code produces an incorrect output or fails to terminate.

```
Algorithm Special_Binary_Search (A,n);
begin
    Position := Special_Find(1,n);
end

function Special_Find (Left,Right) : integer;
begin
    if Left = Right then
       if A[Left] = Left then Special_Find := Left
       else Special_Find := 0
    else
        Middle := ⌈(Left+Right)/2⌉;
        if X[Middle] < Middle then
```

$Special\_Find := Special\_Find(Middle + 1, Right)$

else

$Special\_Find := Special\_Find(Left, Middle)$

end

(5 points)

7. Given as input a sorted array A of n numbers and another number x, design an algorithm with running time $O(n)$ to determine whether there are two elements of A whose sum is exactly x. (Hint: Recall the ideas of the $O(n)$ soluton to the Celebrity Problem discussed in class.)

8. Rearrange the following array into a heap using the bottom-up approach.

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

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

9. A variation of the n-coins problem is defined as follows.

> 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 lighter than the other true coins. 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.

Show that in the worst case it is impossible to solve the n-coins problem with k comparisons (for any n and k) if $n > (2^k - 1)$.

10. Design an algorithm as efficient as possible to determine whether two sets of numbers (represented as arrays) are disjoint. State the time complexity of your algorithm in terms of the sizes m and n of the given sets.

11. Draw a Huffman tree for a text that contains eight characters A, B, C, D, E, F, G, and H with frequencies 6, 2, 3, 5, 14, 8, 4, and 2, respectively.