Solution.

CS 141 (Closed-book) Midterm Test

Feb. 13, 8:10-9:30am, 2007

Total: 60 points

QUESTION 1. [10 pts] Consider the following algorithm

```
Algorithm Mystery(A[1..n]: integer);
   var  i,j: integer;
   y = 1;
   for i := 1 to n do
       for j := 1 to 2*i do
           y := y + y;
   i := n;
   while i > 1 do
       y := y + 1;
       i : = i/2;
       for j := 1 to n do
           y := y * y;
   print y;
```

Let $T(n)$ denote the (worst-case) time complexity of algorithm Mystery. Analyze the algorithm to obtain a tight asymptotic bound on $T(n)$. You need show the key steps in your analysis.

$$T(n) = \sum_{i=1}^{n} \Theta(i) + \Theta(n \log n)$$

4 pts

$$= \Theta(n^2) + \Theta(n \log n)$$

3 pts

$$= \Theta(n^2)$$

3 pts

QUESTION 2. [10 pts] Consider the following recursive algorithm

```
Algorithm Sillyaverage(var B[1..n]: integer);
   var i,j: integer;
   begin
      if n > 1 then
         for i := 1 to n-1 do
             B[i+1] := (B[i] + B[i+1]) / 2;
         call Sillyaverage(B[1..n-1])
   end;
```

Let $T(n)$ denote the (worst-case) time complexity of algorithm Sillyaverage. Use recurrence relations to obtain a tight asymptotic bound on $T(n)$.

$$T(n) = T(n-1) + \Theta(n) \quad 5 \text{ pts}$$

$$= T(n-1) + \Theta(n-1) + \Theta(n)$$

$$\ldots$$

$$= T(1) + \sum_{i=2}^{n} \Theta(i)$$

$$= 1 + \sum_{i=2}^{n} \Theta(i) \quad 2 \text{ pts}$$

$$= \sum_{i=1}^{n} \Theta(i)$$

$$= \Theta(n^2) \quad 3 \text{ pts}$$

QUESTION 3. [15 pts total] Let $A[1..n]$ be an array of integers. The *prefix sum* problem asks for an array $B[1..n]$, where $B[i] = \sum_{j=1}^{i} A[j] = A[1] + ... + A[i]$. The following algorithm solves the prefix sum problem.

```
Algorithm PrefixSum(A[1..n]: integer);
   var i,j: integer;
   begin
      for i := 1 to n do
          B[i] = A[1];
          for j := 2 to i do
              B[i] := B[i] + A[j];
   end;
```

(a) [5 pts] What is the time complexity of algorithm PrefixSum?

(b) [10 pts] Can you design another algorithm with an improved time complexity? Give the (informal) pseudocode and analyze its time complexity.

(a) $T(n) = \sum_{i=1}^{n} i = \theta(n^2)$ 5 pts

(b) Alg. ImprovedPrefixSum (A[1..n]: integer);

var i, j: integer;

8 pts

begin B[1] := A[1];

for i := 2 to n do

B[i] := B[i-1] + A[i];

end;

Time = $\theta(n)$ 2 pts

QUESTION 4. [10 pts] Let $A[1..8]$ be an array of eight integers. Describe how to find both the maximum and minimum elements in A using at most 10 comparisons.

Hint: Divide-and-conquer. Can you solve the problem for an array of four integers in 4 comparisons?

You may describe your solution by means of an informal pseudocode or a (branching) diagram. Indicate the number of comparisons at each step.

Alg. MinMax$(A[1..n])$

if $n>1$ then

$(min_1, max_1) := MinMax(A[1..\frac{n}{2}])$;

$(min_2, max_2) := MinMax(A[\frac{n}{2}+1..n])$;

10 pts

$min := min\{min_1, min_2\}$;

$max := max\{max_1, max_2\}$;

return (min, max);

OR

$a_1 \; a_2 \quad a_3 \; a_4 \quad a_5 \; a_6 \quad a_7 \; a_8$ — 4 comp.

10 pts

$min_1 \; max_1 \qquad min_2 \; max_2$ — 4 comp.

$min \quad max$ — 2 comp.

QUESTION 5. [15 pts] Recall that in the Simplified Fake Coin problem we are given a set of n coins and a balance scale. Suppose we know that there is a fake coin among the n coins that is *lighter* than the genuine one. Design an algorithm (called the *divide-into-three algorithm* in Levitin) to find the fake coin in at most $\log_3 n$ steps.

(a) [8 pts] Give an (informal) pseudocode of your algorithm. For simplicity, you may assume that $n = 3^k$ for some integer k.

(b) [7 pts] Analyze the time complexity of your algorithm by first setting up a recurrence relation and then solving it for $n = 3^k$.

(a) Alg. FastCoin(S);

// S is a set of n coins //

3 pts

Divide S into S_1, S_2, S_3 with equal sizes;

5 pts

if $W(S_1) = W(S_2)$ then

FastCoin(S_3)

elseif $W(S_1) < W(S_2)$ then

FastCoin(S_1)

else

FastCoin(S_2)

(b) $T(n) = T(n/3) + 1$

$= \log_3 n$ steps

7 pts