Advanced Algorithms (高等演算法)

Date: 2019/11/5 Class: ID: Name:

1. (10%) The n-th Fibonacci number can be calculated with the following recursive function. Assume the time costs of all integer arithmetic (addition, subtraction, multiplication, division) operations are equal. First, find the time complexity of the function. Next, rewrite the function for sub-linear time complexity and show your complexity analysis.

```
Fibo( n )
{
    if (n = 1) or (n = 2) then
        return 1
    else
        return Fibo(n – 1) + Fibo(n – 2)
}
```

2. (10%) Consider the following program segment. Let n be a large positive integer.

```
i ← 3
x ← 4
while ( i <= n ) do
{
    x ← x + 5
    i ← i × i × i
}
```

   (a) Find the final value of x as a function of n. (5%)
   (b) Find the final value of i as a function of n. (5%)

3. (10%) A linear consecutive-k-out-of-n system ($k < n$) consists of n nodes arranged in a line, where the system fails if and only if some k consecutive nodes fail. Suppose the nodes are statistically independent and the reliability of node i is p_i for any $i \in \{1, 2, \ldots, n\}$ (node i functions with probability p_i and fails with probability $1 - p_i$). Let $R(i, j, k)$ denote the reliability of the linear consecutive-k-out-of-n subsystem consisting of nodes $i, i+1, i+2, \ldots, j$.
   (a) Express $R(1, n, k)$ with $R(1, n-1, k)$, $R(1, n-2, k)$, $R(1, n-3, k)$, ..., and $R(1, n-k, k)$.
   (b) Express $R(1, n, k)$ with $R(1, n-1, k)$ and $R(1, n-k-1, k)$.

4. (15%) Use Master Theorem to solve the following **recurrences** for $T(n)$.
   (a) $T(n) = 3T(n/9) + n$
   (b) $T(n) = 3T(n/9) + n^{0.5}\log_2(n)$
   (c) $T(n) = 3T(n/9) + \log_2(n)$

5. (10%) Given a subroutine Unbiased_Rand( ) that outputs 1 with probability 1/2 and 0 with probability 1/2, do the following tasks.
   (a) Design an algorithm Biased_Rand( ) that returns 1 with probability 4/5 and 0 with probability 1/5. (5%)
   (b) Prove that your algorithm is correct and find its expected running time. (5%)

6. (10%) Consider the following recursive function where the global variable *count* is initialized to 0 and input n is a positive integer.
   (a) Find the asymptotic time complexity in Θ-notation. (5%)
   (b) Find the final value of *count* as a function of n. (5%)

```
Rec-x( n )
{
   if ( n = 1 ) or ( n = 2 ) then
      count ← count + 1
   else
   {
      Rec-x( n–2 )
      Rec-x( n–2 )
      Rec-x( n–1 )
      count ← count + 1
   }
}
```

7. (10%) Given 65 distinct numbers, please answer the following questions.
   (a) In the worst case, how many comparisons at least do you need to find both the maximum and minimum numbers? Describe your algorithm with a brief proof. (5%)
   (b) In the worst case, how many comparisons at least do you need to find the second largest number? Describe your algorithm with a brief proof. (5%)

8. (15%) Given n distinct unsigned integers where each integer has b bits, we want to sort these n integers into increasing order by Counting Sort or Radix Sort.
   (a) What is the (worst-case) time complexity of Counting Sort? (5%)
   (b) What is the (worst-case) time complexity of Radix Sort? (5%)
   (c) How to sort n integers in the range 0 to $n^3 - 1$ in $O(n)$ time? (5%)

9. (10%) Use dynamic programming to find a Longest Common Subsequence of the two sequences X = (A, B, C, A, B, D, A, B) and Y = (B, D, C, A, B, A).