Advanced Algorithms (高等演算法)

Date: 2021/11/18 Class: ID: 410785008 Name:

1. (10%) Consider the following program segment, where input n is a positive integer.

```
a ← 2
b ← 3
c ← 4
while ( a < n ) do
{
    a ← a × a × a
    b ← b × 2
    c ← c + 1
}
```

2~7

3 1 2, ...

[0 ~ 0, ...] = 0

$2 \sim 2^3-1$

$2^3 \sim 2^9-1$

$2^9 \sim 2^{27}-1$

n: 3 4 5 6 7 8 9

8 8

8 < 3 ~ 8 x

8 < 9 ✓

→ 512 < 9 X

(a) Find the final value of a as a function of n. (4%)

(b) Find the final value of b as a function of n. (2%)

(c) Find the final value of c as a function of n. (2%)

(d) Find the time complexity (in Θ notation) of this program segment. (2%)

2. (10%) Consider the following recursive function where the global variable *count* is initially 0 and input n is a positive integer.

(a) Find the final value of *count* as a function of n. (5%)

(b) Find the asymptotic time complexity in Θ-notation. (5%)

```
RecX( n )
{
  if ( n = 1 ) or ( n = 2 ) then
     count ← count + 3
  else
  {
     RecX( n–1 )
     RecX( n–2 )
     RecX( n–2 )
     count ← count + 2
  }
}
```

$C(3) = C(2) + 2C(1) + 2$

$9 + 2 = 11$

$\frac{32}{3} + \frac{4}{3} - 1$

3. (15%) 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. A **circular** consecutive-k-out-of-n system ($k < n$) consists of n nodes arranged on a circle, where the system fails if and only if some k consecutive nodes fail. Suppose all 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 $R_L(i, j, k)$ denote the reliability of the **linear** consecutive-k-out-of-n subsystem consisting of nodes $i, i+1, i+2, \ldots, j$, and $R_C(i, j, k)$ denote the reliability of the **circular** consecutive-k-out-

of-n subsystem consisting of nodes i, i+1, i+2, …, j.

(a) Express $R_L(1, n, k)$ with $R_L(1, n-1, k)$, $R_L(1, n-2, k)$, $R_L(1, n-3, k)$, ..., and $R_L(1, n-k, k)$.

(b) Express $R_L(1, n, k)$ with only $R_L(1, n-1, k)$ and $R_L(1, n-k-1, k)$.

(c) Express $R_C(1, n, k)$ with some terms of $R_L(?, ?, k)$.

4. (10%) The Fibonacci function F(n) is defined as F(0) = 0, F(1) = 1, and F(n) = F(n-1) + F(n-2) for $n > 1$.

   (a) Design a $\Theta(n)$-time algorithm to compute F(n).

   (b) Design a $\Theta(\lg n)$-time algorithm to compute F(n).

5. (10%) Given a randomized procedure Biased_Rand( ) that outputs 1 with probability p and 0 with probability $1 - p$, where $0 < p < 1$ and p is unknown, do the following tasks.

   (a) Design a randomized algorithm Unbiased_Rand( ) that returns 1 or 0 with equal probability 1/2. (4%)

   (b) Prove that your algorithm is correct. (3%)

   (c) What is the expected running time of your algorithm, as a function of p? (3%)

6. (10%) Given a randomized procedure Unbiased_Rand( ) that outputs 1 or 0 with equal probability 1/2, do the following tasks.

   (a) Design a randomized algorithm Biased_Rand( ) that returns 1 with probability 2/5 and 0 with probability 3/5. (4%)

   (b) Prove that your algorithm is correct. (3%)

   (c) What is the expected running time of your algorithm? (3%)

7. (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 if every r bits are considered as one digit? (5%)

   (c) How to sort n integers in the range from n^3 to $n^5 - 1$ in O(n) time? (5%)

8. (10%) Given 97 distinct numbers, please answer the following questions. 96

   (a) In the worst case, how many comparisons are necessary to find the maximum number? 48
   Describe your algorithm with a brief proof. (2%) 144

   (b) In the worst case, how many comparisons at least is necessary to find both the maximum and minimum numbers? Describe your algorithm with a brief proof. (4%)

   (c) In the worst case, how many comparisons at least is necessary to find the second largest number? Describe your algorithm with a brief proof. (4%)

9. (10%) Given an unsorted list of n distinct numbers, how to find the rank i (the i-th smallest) element in $O(n)$ worst-case time complexity? Describe the algorithm and give an analysis for its worst-case time complexity.