Sample Solution

Fall 2011, Test 1, Introduction to Algorithms

Name:
Section:
Email id:

3rd October, 2011

Answer all Six questions. You have 90 minutes to complete the exam.
You can use a calculator - You are allowed to use your text booka and notes..

1. **Recursion and Analysis**

(a) Write an algorithm (C++ like program) to find the largest integer less than equal to $n^{\frac{1}{3}}$ of a given input positive integer, n,. (You are not allowed to use built in mathematical libraries.. You are allowed to use any of the standard (+,-*,/) arithmetical operators) For example if $n = 3$, the output should be 1, if $n = 16$, the output should be 2 and if $n = 69$, the output should be 4. What is the running time of your algorithm in O notation. [**5 points**]

```
i = 1;
while (i*i*i <= n)
    i = i+1;
return i-1;
```

Complexity $O(n^{\frac{1}{3}})$

Binary search is also acceptable.

(b) You are given an unsorted integer array A of size n. Your output will be 1 if every element in the even position of A is smaller than every element in the odd position of A and 0 otherwise. (i.e., each element A[0], A[2], A[4],.... is smaller than each element A[1],A[3],A[5],...) Write a linear time algorithm (pseudo code will suffice) to do this. (A nonlinear algorithm will get you only one point). [**5 points**]

```
max = A[0];
for (i=0; i < n; i = i+2)
{ if A[i] > max
        max = A[i]
}
min = A[1]
for (i=1; i < n; i = i+2)
   {if A[i] < min
        min = A[i]

}
if max < min  return 1
        else return 0
```

2. **Big Oh Notations**

(a) For the expressions on the left hand side, are the Asymptotic Notations (in Big Oh or Ω) correct - State either True or False. [**5 points**]

i. $n\log(n) = O(n\sqrt{n})$ True

ii. $\sqrt{(n)} = O(\log(n))$ False

iii. $\log(n) + \sqrt{n} = O(n)$ True

iv. $\frac{1}{n} = \Omega(\log(n))$ False

v. $3n^2 + \sqrt{n} = \Omega(n\log(n))$ True

(b) What will be printed by the folllowing program when n = 2 and n = 3. Write a recurrence relation (not solve) for the number of times print statements are executed..

```
void csci2300(int n)
{
  if (n==1) cout << ''Happy'' << endl;
  else
  {
    csci2300(n-1);
    cout << ''Semester'' << endl;
    csci2300(n-1);
  }
return;
}
```

n = 2

Happy
semester
Happy

n = 3

Happy
semester
Happy
semester
Happy
semester
Happy

$T(n) = 2T(n-1) + 1$ 3

$T(1) = 1$

3. **Divide and Conquer and Design of Algorithm**

Suppose you are choosing between the following two algorithms:

- Algorithm A solves problems of size n by dividing them into 6 subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
- Algorithm B solves problems of size n by dividing them into twelve subproblems of size $n/3$, recursively solving each subproblem, and then comining the solutions in $O(n^2)$ time.

(a) What are the running times of each the algorithms in big Oh notation? (Please write a recurrence relations and solve (using any method, including master theorem) [**6 Points**]

(b) Which would you choose and Why? [**4 Points**]

a. $T(n) = 6T(\frac{n}{2}) + O(n)$

$a = 6$, $b = 2$, $d = 1$

$\frac{a}{b^d} = \frac{6}{2} > 1$

$O(n^{\log_2^6}) = O(n^{2.585})$

$\log_2^6 = x$

$6 = 2^x$

$12 = 3^y$

$2^{x+1} = 3^y$

b. $T(n) = 12T(\frac{n}{3}) + O(n^2)$

$a = 12$, $b = 3$, $d = 2$

$\frac{a}{b^d} > 1$

$\frac{12}{9} > 1$

$O(n^{\log_3^{12}}) \quad O(n^{2.262})$

(b) Prefer 2nd Algorithm as it is faster.

4. **Algorithms with Numbers**

(a) Find x and y such that $71 \times x + 19 \times y = gcd(71, 19)$ [**6 points**]

x = -23
y = 86

71	19	3	14
19	14	1	5
14	5	2	4
5	4	1	1
4	1	4	0
1	0		

$1 = 1 - 0$
$= 1 - (4 - 4 \cdot 1)$
$= 5 \cdot 1 - 4$
$= 5 \cdot (5 - 1 \cdot 4) - 4$
$= 5 \cdot 5 - 6 \cdot 4$
$= 5 \cdot 5 - 6 \cdot (14 - 2 \cdot 5)$
$= 17 \cdot 5 - 6 \cdot 14$
$= 17 \cdot (19 - 1 \cdot 14) - 6 \cdot 14$
$= 17 \cdot 19 - 23 \cdot 14$

$17 \cdot 19 - 23(71 - 3 \cdot 19)$
$86 \cdot 19 - 23 \cdot 71$

(b) what is the value of 2^{128} *mod* 17 (Hint: Use Fermat's little Theorem and 17 is a prime number). [**4 points**]

$2^{16} \bmod 17 = 1$ (FLT)

$2^{128} \bmod 17 = (2^{16})^8 \bmod 7$

$= 1^8 = 1$

5. **Recursive Ternary Trees**

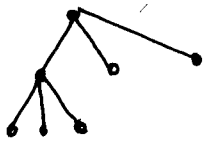
A full ternary tree is a tree that has 0 or 3 children (left, middle and right) (This is **a Full** Ternary). Let T_n denote the number of full ternary trees with n vertices. Naturally T_1=1. By definition $T_2 = 0$. T_3=0, T_5=0;T_6=0

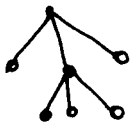
(a) Draw ternary trees with 4 and 7 vertices, determine the exact values of T_4, T_7 [**6 points**].

(b) For general n, derive a recurrence relation for T_n [**4 points**] (Hint: There are i nodes in the left subtree,j nodes in the middle subtree)

$T_1 = 1$

$T_4 = 1$





$T_7 = 3$

$$T(n) = \sum_{j=1}^{n-1} \sum_{i=1}^{n-1} T(i)\,T(j)\,T(n-i-j)$$

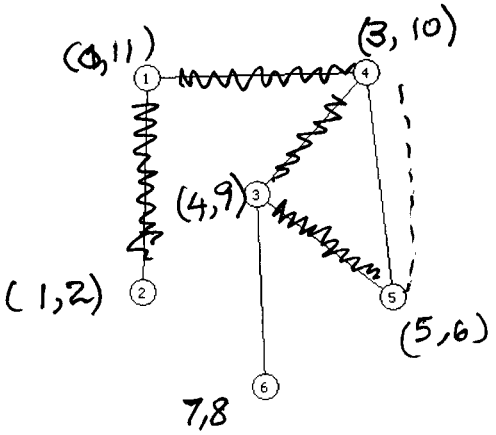
$T(i) = 0$ for all $i =$ ~~$3k+2, 3k+3,$~~ 1, 2, 3, 5, 6, 8, 9 . . .

$i = 3k+2$ or $3k$

$k = 0, \ldots$

6. **Graphs and Graph Algorithms**

(a) Give the pre and post numbering of vertices (i.e., the number you assign when you first visit a vertex and the number you assign when you leave a vertex) of a Depth First Traversal of a graph G (shown below) starting at vertex 1 [**6 points**].



(b) Construct a graph G, with 5 nodes such that the pre and post numbers for 4 of the nodes are successive (i.e. for nodes n=2 to 5, post(n) = pre(n)+1) [**4 points**].