Design and Analysis of Algorithms 2012 (Home work 1)

September 10, 2012

- Due on Monday, September 24, before 10 a.m.
- Late home works will not be accepted.
- Please give precise arguments for all statements that you write.
- Please do not hesitate to contact me if you do not understand the problems.
- Collaboration is discouraged, but not prohibited. It is recommended that you try to solve the problems on your own. You can discuss the questions with your colleagues but you should not copy solutions. Always write down your own answers. If copying is detected that may immediately lead to a grade less than 7. (**This would be followed strictly**)
- Credits would be given to partial solutions also.
- The answers should be typed or written clearly and a hard copy is to be submitted.

The Fibonacci Numbers: [$5 \times 2 = 10$ **points**]

1. Prove the following. In all the exercises of these question f_n denotes the n-th Fibonacci number,i.e., $f_0 = 0$, $f_1 = 1$ and $f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$.

   (a) $f_n^2 - f_{n+1}f_{n-1} = (-1)^{n-1}$, $n \geq 1$

   (b) $\sum_{i=0}^{n} f_i = f_{n+2} - 1$

Practice with solving recurrence relations: [$5 \times 2 = 10$ **points**]

2. Solve the following recurrence relations.

   (a) $a_0 = 1$, $a_1 = 2$, and $a_n = 4a_{n-2}$, for $n \geq 2$

   (b) $a_0 = 0$, $a_1 = a_2 = 1$, $a_n = 6a_{n-1} - 12a_{n-2} + 8a_{n-3}$, for $n \geq 3$

Basic Algorithms: [$4 \times 10 = 40$ **points**]

3. Write the pseudo-code of an algorithm to find the maximum of n given numbers. What is the best case and worst case running time of your algorithm.

4. Write the pseudo-code of an algorithm to find both the maximum and minimum of an array of n numbers. Try to make it as efficient as possible. (Hint: The best algorithm would only use around $\frac{3}{2}n$ comparisons).

5. Consider the problem of evaluating a polynomial at a point. Given n integer coefficients $a_0, a_1, \ldots, a_{n-1}$ and and integer x, we want to compute $\sum_{i=0}^{n-1} a_i x^i$. Write the pseudo-code of an algorithm to do this. How many additions and multiplications are required to do this. Try to make your algorithm as efficient as possible. (Hint: The best algorithm would use n multiplications and n additions)

6. Let A be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an *inversion* of A.

   (a) List the inversions of the array $< 2, 3, 8, 6, 1 >$.

   (b) Which array containing the elements from the set $S = \{1, 2, 3, \ldots, n\}$ contains the maximum number of inversions. How many does it have?

   (c) What is the relationship between the running time of *insertion sort* and the number of inversions present in the array being sorted? Justify your answer.

Growth of functions: [$10 + 10 + 15 = 35$ **points**]

7. Here is a list of functions in one variable, n.

$$\frac{17n^5}{100}, 8, \frac{n^2}{n+1}, 2^{6\lg n}, n^2 \lg n, n^2 + 27n, 2^n + n$$

   Place these functions in a list such that $f(n)$ is any function in the list and $g(n)$ is in right of it then $f(n) = O(g(n)$. You do not need to prove your answer.

8. Find the smallest integer c such that $f(x) = O(n^c)$ for the following functions (do not have to justify your answer):

   (a) $f(x) = 3n^2 + 2$

   (b) $f(x) = n^2 + 3n \log n$

   (c) $f(x) = n(\log n)^3$

   (d) $f(x) = \sum_{i=1}^{n} i^3$

   (e) $f(x) = \sum_{i=0}^{\lceil \lg n \rceil} \frac{\lg n}{2^i}$

9. Prove or disprove the following:

(a) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$.

(b) $f_1(n) = o(g_1(n))$ and $f_2(n) = o(g_2(n)$, then $f_1(n) + f_2(n) = o(g_1(n) + g_2(n))$.

(c) $\log(n!) = \Theta(n \log n)$