

Homework 1

CSCI 5870

Due date: 22 September 2008

- 1 (1.24) Write an algorithm to find the second-largest element in a set containing n entries. How many comparisons of elements does your algorithm do in the worst case?
- 2 (1.25) Write an algorithm to find both the smallest and largest elements in a set of n entries. Try to find a method that does at most roughly $1.5n$ comparisons.
- 3 (1.28) Let $p(n) = a_k n^k + a_{k-1} n^{k-1} + \dots + a_1 n + a_0$ be a polynomial in n of degree k with $a_k > 0$. Prove that $p(n) \in \Theta(n^k)$.
- 4 (1.30) List the functions below from lowest asymptotic order to highest asymptotic order. If any two or more are of the same asymptotic order, indicate which.

n	3^n	$n \lg n$	n^3
n^2	$\lg n$	$n - n^3 + 7n^5$	$n^2 \lg n$
e^n	\sqrt{n}	3^{n-1}	$\lg \lg n$
$\ln n$	$\lg^2 n$	$n!$	$n^{1+\epsilon}$

- 5 (1.32) Prove or give a counterexample: For every positive constant c and every function f from nonnegative integers to nonnegative reals, $f(cn) \in \Theta(f(n))$. *Hint*: Consider some of the fast-growing functions listed in problem 5.
- 6 (1.37) Prove Theorem 1.12: $n^k \in o(c^n)$ for any $c > 1$.
- 7 (1.40) Prove or disprove:

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

- 8 (3.1) Show that every 2-tree with n internal nodes has $n+1$ external nodes. A 2-tree is a binary tree where every node has either zero or two children.

- 9 (3.3) Show that the external path length epl in a 2-tree with m external nodes satisfies $epl \leq \frac{1}{2}(m^2 + m - 2)$. Show also that $epl \leq \frac{1}{2}n(n + 3)$ for a 2-tree with n internal nodes.
- 10 (3.6) In this exercise all integers are considered to be nonnegative, for simplicity. A *divisor* of an integer k is any integer $d \neq 0$ such that $k \equiv 0 \pmod{d}$. A *common divisor* for a set of integers is an integer that is a divisor for each integer in the set. Euclid's algorithm for finding the greatest common divisor (GCD) of two nonnegative integers, m and n , can be written without using division as follows:

```
int gcd(int m, int n) {
    int ans;
    if (m == 0)
        ans = n;
    else if (m > n)
        ans = gcd(n,m);
    else {
        nLess = n - m;
        ans = gcd(n,nLess);
    }
    return ans;
}
```

The preconditions are that $m \geq 0$, $n \geq 0$ and $m + n > 0$. The following facts will come in handy:

1. If $a > b$, then $a - c > b - c$
2. If d is a divisor of k , then d is a divisor of $k \pm d$
3. If d is a divisor of k , then $d \leq k$ or $k = 0$

Given this information, first prove that if the preconditions are satisfied, then the algorithm returns *some* common factor of m and n . Then, prove that the algorithm returns the *greatest* common factor.