G22.1170: Fundamental Algorithms I
Problem Set 2
(Due Thursday, March, 8 2007)

The problems in this problem set involve various techniques used in the analysis of algorithms. You may consult the Notes on Recurrence Equations on the class website, which discusses these materials in depth. Try to give the exact solutions to the recurrence equations wherever possible.

Problems from Cormen, Leiserson and Rivest: (pp. 72–73)
4-1 *Recurrence Examples* & 4-4 *More Recurrence Examples*
4-1 Recurrence examples

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

a. $T(n) = 2T(n/2) + n^3$.
b. $T(n) = T(9n/10) + n$.
c. $T(n) = 16T(n/4) + n^2$.
d. $T(n) = 7T(n/3) + n^2$.
e. $T(n) = 7T(n/2) + n^2$.
f. $T(n) = 2T(n/4) + \sqrt{n}$.
g. $T(n) = T(n-1) + n$.
h. $T(n) = T(\sqrt{n}) + 1$.

4-4 More recurrence examples

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

a. $T(n) = 3T(n/2) + n \lg n$.
b. $T(n) = 3T(n/3 + 5) + n/2$.
c. $T(n) = 2T(n/2) + n/\lg n$.
d. $T(n) = T(n-1) + 1/n$.
e. $T(n) = T(n-1) + \lg n$.
f. $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

Problem 1.1 Consider the following recursive function:

```
function TOP(N: integer): integer;
begin
        if (N <= 0) then TOP := 1
                    else TOP := TOP(N-1) + TOP(N-1)
end
```

1. Give an explicit formula for TOP(N).
2. To within a constant factor, state how many additions would be executed by a program that computes TOP(N), *as written above.*
3. State the running time of an *efficient* procedure to compute the same function. The only arithmetic operations your procedure may use are addition and multiplication.

Problem 1.2 Karatsuba presented a divide-and-conquer algorithm to multiply two arbitrary n-bit numbers using the following facts

- *For every $n > 3$, it is possible to multiply two n-bit numbers by doing three multiplications of $\frac{n}{2}$-bit or smaller numbers and $O(n)$ additional work.*
- *For $n \leq 3$, it is possible to multiply two n-bit numbers with $O(1)$ work.*

1. Write down the recurrence relation for the time complexity of Karatsuba's algorithm.
2. Solve the recurrence equation as exactly as possible.