Massachusetts Institute of Technology
6.046J/18.410J: Introduction to Algorithms
Professors Michel Goemans and Piotr Indyk

Handout 3
February 7, 2002

# Problem Set 1

This problem set is due in class on *Thursday, February 14, 2002.*

Both exercises and problems should be solved, but *only the problems* should be turned in. Exercises are intended to help you master the course material and will be useful in solving the assessed problems. You are responsible for material covered by the exercises. Each problem is to be done on a separate sheet (or sheets) of three-hole punched paper. Mark the top of each sheet with your name, 6.046J/18.410J, the problem number, your recitation section, the date, and the names of any students with whom you collaborated.

---

**Exercise 1-1.** Do Problem 3-4 on page 59 in CLRS.

**Exercise 1-2.** Do Problem 3-6 on page 60 in CLRS.

**Exercise 1-3.** Do Exercise 2.3-7 on page 37 in CLRS.

---

**Problem 1-1. Asymptotic notation**

Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, \ldots, g_{30}$ of the functions satisfying $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $\ldots$, $g_{29} = \Omega(g_{30})$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. You don't need to show your work for this problem only, just give us the list.

$$\begin{array}{cccccc}
\lg(\lg^* n) & n^{lglgn} & n^{\log_6 5} & n^2 & n! & 2^{2n} \\
(\frac{4}{3})^n & n^2+n & (\frac{3}{4})^n & \lg(n!) & 2^{2^n} & n^{1/\lg n} \\
\ln\ln n & 2^{\lg^* n} & n \cdot 2^n & n^n & \ln n & 1 \\
2^{\lg n} & (\lg n)^{\lg n} & e^n & \sum_{k=1}^n k & (n+1)! & \sqrt{\lg n} \\
\lg^*(\lg n) & n^2 & n & 2^n & n\lg n & 2^{2^{n+1}}
\end{array}$$

The function $log^* n$ is discussed on pages 55-56 of CLRS.

**Problem 1-2. Recurrences**

Give asymptotic upper and lower bounds for $T(n)$ which are as tight as possible. Assume that $T(n)$ is constant for $n \leq n_0$, where $n_0$ is a constant. Justify your bounds.

**(a)** $T(n) = 6T(\frac{n}{2}) + n^3$

**(b)** $T(n) = 9T(\frac{n}{3}) + n^2 \lg^3 n$

**(c)** $T(n) = T(n^{1/3}) + 1$

**(d)** $T(n) = 5T(\frac{n}{3}) + n\sqrt{n}$

**(e)** $T(n) = T(\frac{n}{3}) + T(\frac{n}{2}) + n$

**(f)** $T(n) = 2T(\frac{n}{8}) + \sqrt[3]{n}$

**(g)** $T(n) = T(\frac{n}{2} + \lg n) + 1$

**(h)** $T(n) = 2T(n-1) + n^7$

**Problem 1-3. Inversions**

Do Problem 2-4 on page 39 in CLRS.

**Problem 1-4. Stable Sorting**

Given an input array $A[1], \ldots, A[n]$, a sorting algorithm has to produce a permutation $\sigma$ of the input which is in non-decreasing order. We say a sorting algorithm is *stable* if for any $i < j$ such that $A[i] = A[j]$, then $\sigma(i) < \sigma(j)$. Describe a technique for converting any unstable sorting algorithm into a stable sorting algorithm without changing its asymptotic running time.