Problem Set 1

This problem set is due at the beginning of class on Thursday, February 13, 2003.

Each problem is to be done on a separate sheet (or sheets) of 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.

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.

\[
\begin{array}{cccccccc}
  n^2 & n^{2\log n} & n^2 + 2^{100}n & |n| & n^n & 2^{2n} \\
  (\frac{3}{2})^n & (\frac{1}{2})^n & n^{\log_2 n} & (\lg n)! & 100^{100} & (1/n)^{1/\lg n} \\
  \ln \ln n & 2^{\lg^* n} & n \cdot 2^n & 3(n!) & \ln n & 1 \\
  2^{\lg n} & (\lg n)^{\lg n} & e^n & \sum_{k=1}^{n} k \cdot (n + 1)! & \sqrt{\lg n} & 2^{2n+1} \\
  \lg(\lg^* n) & \lg^*(\lg n) & n & n^{\lg n} & 2^n & n \lg n
\end{array}
\]

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) \( T(n) = 6T(\frac{n}{3}) + n^3 \)
(b) \( T(n) = 6T(\frac{n}{4}) + n \)
(c) \( T(n) = 9T(\frac{n}{3}) + n^2 \)
(d) \( T(n) = 8T(\frac{n}{2}) + n^3 \log^2 n \)
(e) \( T(n) = 10T(\frac{n}{5}) + n^2 \sqrt{n} \)
(f) \( T(n) = T(\frac{n}{3}) + 2T(\frac{n}{4}) + n \)
(g) \( T(n) = T(n^{1/3}) + \lg n \)
(h) \( T(n) = 3T(n - 1) + n^3 \)
(i) \( T(n) = T(\lg n) + 1 \)
(j) \( T(n) = T(\frac{n}{4}) + \sqrt{n} \)
(k) \( T(n) = T\left(\frac{n}{2} + \sqrt{n}\right) + 1 \)

Problem 1-3. Product Finder

Design an algorithm, which, given an input array \( A[1], \ldots A[n] \) of different integers, and a target integer value \( x \), prints ALL pairs \((p_i, p_j)\) such that \( A[p_i] \cdot A[p_j] = x\). Your algorithm should have a running time of \( O(n \log n) \).

Gentle Reminder: Recall that for all algorithm solutions, we expect more than just the algorithm. We want you to prove that the algorithm is correct, and provide runtime analysis.

Problem 1-4. Coinage

Long John Platinum has a treasure trove of \( n \) gold coins. BUT sneaky Jim Larkins has taken one of the gold coins and replaced it with a similar looking but slightly lighter coin. Poor John only has a balance, which can tell him which of two piles of coins \( a \) or \( b \) is heavier.

Design an algorithm that will help John figure out which is the fake coin. Your algorithm should have a running time of \( \Theta(\log n) \) (where each use of the balance is an “instruction”).

Extra Credit: Will be given to valid solutions that have the best actual (non-asymptotic) running time (the fewest actual uses of the balance).