CS 341, Fall 1999 due Friday Oct. 1

A. Lubiw, D. Schuurmans

ASSIGNMENT 1

Questions are worth 10 points each. This assignment will be worth slightly less than the others.

The bonus question is worth 5 points. In general, the bonus questions are for those who want a challenge. They are not easy marks.

Do not copy from others, and acknowledge your sources.

1. Use the relations "$\subset$" and "$=$" to put the orders of the following functions into a sequence, e.g. $O(n) \subset O(n^2)$.

$\log\log n$	n	$n!$	$(n+1)!$
$\log^2 n$	$n^{\log n}$	n^3	$(\log n)!$
2^{n-1}	$2^n + \log n$	2^n	$\sqrt{n}$

2. Give a proof or counter-example to the following claim: for all functions $f(n)$ and $g(n)$ from the natural numbers to the non-negative reals, either $f(n) \in O(g(n))$ or $g(n) \in O(f(n))$.

3. Give asymptotic upper 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. You may use Example 4.7.16 in the text.
   (a) $T(n) = 2T(n/2) + n^3$
   (b) $T(n) = 2T(n/4) + \sqrt{n}$
   (c) $T(n) = T(n-1) + n$

4. Suppose you need to generate a random permutation of the first n integers. You may assume a random number generator `rand(i,j)` that generates integers between i and j with equal probability. Here are two algorithms:

   A. Fill the array from `a[1]` to `a[n]`. To fill `a[i]`, generate random numbers `rand(1,n)` until you get one that is not already in `a[1], ..., a[i-1]`. To check if a number is already in the array, use an auxiliary array `used[1..n]` initially set to `False` and set `used[a[j]]` to `True` when `a[j]` is filled.

   B. Fill the array such that `a[i] = i`. Then

   ```
   for i = 2 to n, swap(a[i], a[rand[1,i]])
   ```

   (i) Analyze the expected running time of each algorithm. Hint: For A, you need the expected number of calls to `rand` to fill `a[i]`. And check out the Harmonic series—see Section 1.7.2.

   (ii) Analyze the worst case running time of each algorithm.

5. Suppose you are given a list of n distinct sorted numbers that has been circularly shifted k positions to the right. For example $(35, 42, 5, 15, 27, 29)$ is a sorted list circularly shifted $k = 2$ positions to the right. Suppose that you *do not* know k. Give an $O(\lg n)$ algorithm to find the maximum number in such a list. Justify.

6. BONUS QUESTION. The city skyline problem. Suppose you are given n rectangles in the plane all with their bottom edges lying on a horizontal line. Each rectangle is given by its corner coordinates. The rectangles represent the buildings in a city. You want to find the skyline—i.e. the shape representing the union of all the rectangles. Your output should be a list of coordinates as you travel along the skyline. Give an $O(n \log n)$ time algorithm for this problem. Justify.