CSE 326: Data Structures
Assignment #1
March 31, 1999
Due: Wednesday, April 7

Reading Assignment: Weiss, Chapters 1 – 3

Problems

1. This problem gives an orthogonal view of comparative running times from that given in lecture. Be sure to look at the patterns in your table when you have completed it.

   For each function $f(n)$ and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes $f(n)$ microseconds. For large entries (say, those that warrant scientific notation), an estimate is sufficient.

	1 second	1 minute	1 hour	1 day	1 month	1 year
$150 \log_2 n$						
$15n$						
n^2						
$\frac{1}{4} \cdot 2^n$						

2. Weiss, p. 62, problem 2.1.

3. Weiss, p. 62, problem 2.2.

4. Weiss, p. 62, problem 2.4.

5. Weiss, p. 62, problem 2.7, part a.

6. (a) Prove that $\ln n = O(n^\epsilon)$, for any constant $\epsilon > 0$.

   (b) Let $f(n)$ be an arbitrary polynomial of degree k, i.e. there exist constants $a_0, a_1, \ldots, a_k$, with $a_k \neq 0$ such that $f(n) = a_0 + a_1 \cdot n + a_2 \cdot n^2 + \ldots + a_k \cdot n^k$. Prove that $f(n) = \Theta(n^k)$.

7. You are given a set S of N words. Design a utility that can service anagram queries. Given a word as input, an anagram query returns all anagrams of that word that are in the set S. (For example, if the input is CAT, then the outputs might be ACT and CAT. IF the input is SUBESSENTIAL, then the output might be SUITABLENESS.) Describe concisely and precisely the best algorithm you can for the problem. Consider separately the preprocessing of the set S, which is a one-time cost, and the algorithm for servicing an anagram query. What is the worst-case running time of the preprocessing step (as a function of N, the number of words in the set S, and m, the maximum word

length), and what is the worst-case running time of an execution of the anagram utility (as a function of N, the number of words in the set S, k, the number of letters in the input word, and n, the number of anagrams of the input word)? **Extra credit** will be given if your anagram utility has worst-case running time $O(m \log(N) + k \log k + n)$.