Algorithms Lab 1

The in-lab problems are to be solved during the lab time. Work with your team, but write your solutions individually. You do not need to turn in your answers, only discuss them with me. Ask me for feedback on how you write your answers as well.

The homework problem set is due on Friday, 9/18. Work with your team, but write your solutions individually. List the people with whom you discussed the problems.

1 In lab exercises

1. Algorithm A uses $10n \lg n$ operations, while algorithm B uses n^2 operations. Determine the value n_0 such that A is better than B for $n \geq n_0$.

2. Let $f(n) = \lg n$ and assume that we have an algorithm whose running time is $f(n)$ microseconds. Determine the largest size of a problem that can be solved by the algorithm in: (a) 1 second; (b) 1 hour; (c) 1 month; (d) 1 century.

   Same problem for $f(n) = n$ and $f(n) = 2^n$.

3. Show that $an^2 + bn + c$ is $O(n^2)$ (where a, b, and c are constants). Assume that $a > 0$; b and c, however, might be 0 or negative. Hint: Use absolute values.

4. Order the following expressions from fastest to slowest:

$$\sqrt{2}^{\lg n}, n^2, (\frac{3}{2})^n, n^3, \lg n^2, \lg^2 n, 2^n, \lg \lg n, n \lg n$$

5. Suppose you have algorithms with these five running times:

   (a) n^2
   (b) $100n^2$
   (c) n^3
   (d) $n \lg n$
   (e) 2^n

   How much slower do each of these algorithms get when you increase the input size by one?

2 Homework problems

Your assignment will be evaluated based not only on the final answer, but also on clarity, neatness and attention to details. For example, you'll want to leave plenty of space in between problems so that we can give you feedback.

1. Describe a method for finding both the minimum and the maximum of n numbers with fewer than $3n/2$ comparisons.

2. Give an example of a positive function $f(n)$ such that $f(n)$ is neither $O(n)$ nor $\Omega(n)$.

3. Arrange the following functions in ascending order of growth rate. For each pair of consecutive functions, give a brief justification on why they are in this order. For e.g., if you ordered A, B, C, you need to justify that 1. $A = O(B)$; and 2. $B = O(C)$.

$$2^{\sqrt{\log n}}, 2^n, n^{4/3}, n(\log n)^3, n^{\log n}, 2^{2^n}, 2^{n^2}$$

4. Suppose each row of an $n \times n$ array A consists of 1's and 0's such that, in any row i of A, all the 1's come before any 0's. Assuming A is already in memory, describe a method running in $O(n)$ time (*not* $O(n^2)$ time) for finding the row of A that contains the most 1's.

5. Suppose you have algorithms with these five running times:

   (a) n^2
   (b) $100n^2$
   (c) n^3
   (d) $n \lg n$
   (e) 2^n

   How much slower do each of these algorithms get when you double the input size?