Algorithms Lab 1

The in-lab problems are to be solved during the lab. Work with your team, but write your solutions individually. You need to finish these problems during lab time, and check them with me.

The homework problem set is due next Friday. Use the lab period to get started and talk to your team. Work with your team, but write your solutions individually. List the people with whom you discussed the problems.

Appearance: You are **strongly** encouraged to type your solutions. Please leave plenty of open space for us to write feedback.

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. 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?

5. (CLRS 3-4.a) Prove or disprove: $f = O(g)$ implies that $g = O(f)$.

6. Prove or disprove: $f = O(g)$ implies that $g = \Omega(f)$.

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

   What does the running time of these algorithms become when you double the input size?

3 Additional problems

interview question Close all notes, pick a language of your choice, and implement: (a) bubble sort; (b) insertion sort; (c) binary search. Include tester functions (generate random arrays etc).

Do not open your notes! The goal is to be able to go on your own from the high level description of the algorithms (which you should remember), to the low level details, which you need to figure out on the spot. To make it more fun, imagien an interviewer is looking at your screen.

interview question You are given a set of n points on a circle in the plane. Write a function that determines if there exists a pair of points that are antipodal (two points are antipodal if they are diametrically opposite). Analyze running time.

interview question You are presented with 9 marbles. All of the marbles look identical i.e. same shape, color, and dimensions(except for weight). However, 8 of the 9 marbles have exactly the same weight; the last marble is heavier. The only tool you have to measure weights is an old fashioned balance scale. You are only allowed to use the scale 2 times. How do you find the one marble that is not the same weight as the others?

interview question Given an unsorted array and a number k. Find two elements in the array whose sum is k, or report if no such set exists. Analyze running time.

Generalize to 3-sum: Find if there exist 3 elements in the array whose sum is k, or report that no such subset exists. Analyze running time.