CS41 Lab 8: Divide and Conquer

October 27, 2022

In typical labs this semester, you’ll be working on a number of problems in groups of 3-4 students. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.

1. **Recurrence Relation.** Solve the following recurrence relation using Partial Substitution:

   \[ A(n) = 5A(n/3) + 2n, \]
   \[ A(1) = 6 \]

2. **Database Queries** (K&T 5.1) You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains \( n \) numerical values (so there are \( 2n \) values total). You’d like to determine the median of this set of \( 2n \) values, defined as the \( n \)-th smallest value.

   The only way you can access these values is through queries to the databases. In a single query, you can specify a value \( k \) to one of the two databases, and the chosen database will return the \( k \)-th smallest value it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

   - Design an algorithm that finds the median value using at most \( O(\log n) \) queries. Full pseudocode is not necessary, but you must clearly explain how it works, and you must handle all edge cases; e.g., do not assume that \( n \) is even.
   - Show that your algorithm correctly returns the median.
   - Prove that your algorithm uses only \( O(\log n) \) queries.

3. **Counting significant inversions** (K&T 5.2)

   Recall the problem of finding the number of inversions between two rankings. As we saw, we are given a sequence of \( n \) numbers \( a_1, a_2, \ldots, a_n \), which we assume are all distinct, and we define an inversion to be a pair of indices \( i < j \) such that \( a_i > a_j \).

   We previously used counting inversions as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let’s call a pair a **significant inversion** if \( i < j \) and \( a_i > 2a_j \). Give an \( O(n \log n) \) algorithm to count the number of significant inversions.