CS 577: Introduction to Algorithms — Homework 2

Out: 02/03/16 — **Due: 02/10/16**

Ground Rules

- This homework is for both sections of CS 577.
- The homework consists of two questions. You are required to turn in detailed solutions for both. We will pick one question at random to grade.
- You will be graded on the correctness as well as clarity of your solutions. You are required to prove any claims that you make. In particular, when you are asked to design an algorithm, you must argue its correctness as well as running time. You may use without proof any theorems proven in class or in the textbook, as long as you state and cite them properly. Please refrain from consulting sources other than the text and lecture notes.
- You are required to work and submit solutions in pairs, unless you have prior permission from the instructors.
- Homework is due promptly at the start of class. Late homework will receive zero credit.
- Start working on your homework early. Plan your work in such a way that you have the opportunity to put some problems on the back burner for a while and revisit them later. Good luck!

Problems

1. The bit-wise AND of two k-bit numbers is a k-bit number whose ith bit is the AND of the ith bits of the two given numbers. For example, the bit-wise AND of 1100 and 1010 is 1000. The bit-wise XOR of two k-bit numbers is defined likewise. For example, the bit-wise XOR of 1100 and 1010 is 0110. Using associativity, bit-wise AND and bit-wise XOR can be extended to many k-bit numbers in the natural way.

   In this problem, you are given an array A of n k-bit numbers. For every pair of indices i, j, with $1 \leq i \leq j \leq n$, we can define the bit-wise AND of the subarray $A[i \cdots j]$ as the bit-wise AND of the numbers $A[i]$ through $A[j]$, and likewise for bit-wise XOR. Your goal in this problem is to count the number of subarrays (i.e. the number of pairs of indices i, j, $1 \leq i \leq j \leq n$) such that the bit-wise AND of the subarray is exactly equal to its bit-wise XOR. Note that you are not required to determine which subarrays have the same bit-wise ANDs and XORs; you only need to count the number of such arrays.

   Develop a divide and conquer algorithm for this problem, and aim for an $O(2^{4k}n)$ running time. Prove the correctness of your algorithm and analyze its running time. You may assume that arithmetic operations like addition and multiplication on integers up to n are constant time operations.

   Hint: What are all of the possible pairs of bit-wise ANDs and bit-wise XORs that the subarrays may generate? The algorithm we have in mind is inspired by $O(n)$ algorithms that sort by counting. Intuitively, you should think of k as a small constant, so no matter how long the array is, there are a limited number of values that can be taken by AND and XOR. Take advantage of this.

2. An interval is a contiguous subset of the number line and is defined by its end points (x, y). A interval (x_1, y_1) is said to contain another interval (x_2, y_2) if $x_1 \leq x_2$ and $y_1 \geq y_2$.

   In this problem you are given two lists A and B of n intervals each. Develop an algorithm for finding the number of pairs of intervals (I_i, I_j) such that $I_i \in A$, $I_j \in B$, and I_i contains I_j. You may assume that all of the end points of the intervals in $A \cup B$ are distinct.

   Your algorithm should run in $O(n \log n)$ time. Prove the correctness of your algorithm and analyze its running time.

   Hint: Define an appropriate "median" t and split the lists A and B into sublists with intervals that start before t and those with intervals that start after t. What subproblems do you need to solve recursively?