CS 561: Homework 3

Prof. Jared Saia, University of New Mexico

Remember: You are encouraged to work on the homework in groups, but please observe the "Star Trek" rule from the syllabus.

1. On an island, every inhabitant is either a *knight* or a *knave*. Knights always answer truthfully. Knaves may answer either truthfully or falsely; however, a knave always gives the same answer when asked repeatedly about the same person. You may ask only questions of the form, "Person X, is Person Y a knight or a knave?", where $X \neq Y$.

   (a) Suppose there are n inhabitants and strictly more than half are knights. Design an algorithm that identifies every inhabitant's type using $o(n^2)$ questions, and analyze the number of questions it asks.

   Hint: First find one knight. Pairing inhabitants and using recursion may help. Once you know one knight, that person can classify everyone else.

   (b) Prove that if the only promise is that at most half the inhabitants are knights, then no algorithm can be guaranteed to identify every type in every valid instance.

   Hint: Construct two different assignments of types that can produce exactly the same answers to every possible question.

2. A product of matrices is *fully parenthesized* if it is either a single matrix or an expression (PQ), where P and Q are fully parenthesized matrix products. For example,

$$\big(A_1((A_2A_3)A_4)\big)$$

   is fully parenthesized. Prove by induction that every full parenthesization of a product of $n \geq 1$ matrices contains exactly $n-1$ pairs of parentheses.

3. Consider the following recursive algorithm for sorting the array segment $A[\ell \dots r]$. Let $n = r - \ell + 1$ be the length of the segment.

$\textsc{OverlapSort}(A, \ell, r)$
  if $A[\ell] > A[r]$ **then exchange** $A[\ell]$ and $A[r]$
  if $\ell + 1 \geq r$ **then return**
  $k \leftarrow \lfloor (r - \ell + 1)/3 \rfloor$
  $\textsc{OverlapSort}(A, \ell, r-k)$
  $\textsc{OverlapSort}(A, \ell+k, r)$
  $\textsc{OverlapSort}(A, \ell, r-k)$

   (a) Prove by strong induction on n that $\textsc{OverlapSort}(A, \ell, r)$ correctly sorts $A[\ell \dots r]$.

   Hint: Each recursive call sorts a segment of length $\lceil 2n/3 \rceil$. Track where the smallest and largest $\lfloor n/3 \rfloor$ elements must be after the first two calls, and then use the third call.

   (b) Give a recurrence for the worst-case running time and solve it to obtain a tight asymptotic bound. You may ignore floors and ceilings in the recurrence analysis.

   (c) Compare this running time asymptotically with the worst-case running times of insertion sort and merge sort.

4. **Primes and Probability.** In this problem, log means $\log_2$. You may use the following facts:

   (a) every positive integer has a unique prime factorization; and

   (b) if $\pi(m)$ is the number of primes at most m, then $\pi(m) = \Theta(m/\log m)$ for sufficiently large m.

   For integers a, b, the notation $a \mid b$ means that a divides b, and $a \equiv b \pmod{p}$ means that $p \mid (a - b)$.

   (a) Prove that every positive integer x has at most $\log x$ distinct prime factors. *Hint:* If x has t distinct prime factors, compare x with 2^t.

   (b) Let m be sufficiently large, and choose p uniformly at random from the primes at most m. Prove that, for every positive integer x,
   $$\Pr(p \mid x) = O\left(\frac{(\log x)(\log m)}{m}\right).$$

   (c) Let $1 \leq x, y \leq n$ with $x \neq y$, and again choose p uniformly from the primes at most m. Prove that
   $$\Pr(x \equiv y \pmod{p}) = O\left(\frac{(\log n)(\log m)}{m}\right).$$

   (d) Set $m = (\log n)^2$. Simplify the preceding error bound and show that it tends to 0 as $n \to \infty$.

   (e) Alice and Bob hold positive integers x and y, respectively, and both know a common upper bound n such that $x, y \leq n$. They want to test whether $x = y$, but Alice wants to send substantially fewer than $\log n$ bits. Using the preceding parts, give a randomized one-way protocol in which Alice sends Bob a prime and a residue. State what Bob does, prove that the protocol never reports “different” when $x = y$, bound the probability that it reports “equal” when $x \neq y$, and determine the number of bits Alice sends.