Due Time: 7:00pm, Wednesday January 18th **Due at Canvas**

EECS 376, Foundations of Computer Science Winter 2017, University of Michigan, Ann Arbor

EECS 376 Week 2 Problem Set

Problem 2.1 Show how to find the prime factorization of an integer n in polynomial time if the input is given in unary (i.e. as a sequence of n 1s).

Problem 2.2 Euclid's original algorithm calculated $a \bmod b$ by repeatedly subtracting b from a until the remainder is less than b. If a and b have n or fewer digits, show that this algorithm can take exponential time as a function of n.

Problem 2.3 Another sense in which P is robust is that it allows one program to use another as a subroutine. Let B be a polynomial-time algorithm, and suppose we have an algorithm A that executes a polynomial number of instructions, some of which may be calls to B as a subroutine. Show that A runs in polynomial time.

Note: The inputs A sends to B are not necessarily the original input, but rather some subproblem that A is interested in, which could be polynomially larger.

Problem 2.4 (Problem 3.14 in the textbook.)

In this problem, we will generalize the running time of the Fast Fourier Transform to values of n other than powers of 2.

(a) Suppose n has a factor p and we divide the list into p sublists of size n/p. Show that the running time satisfies

$$T(n) = pT(n/p) + pn.$$

(For purposes of elegance, we've omitted the Θ on the right-hand side.)

(b) Let $n = p_1 p_2 \cdots p_k$ be a prime factorization of n (note that the p_is are not necessarily distinct). Taking the base case $T(p) = p^2$ for prime p, show that the running time is

$$T(n) = n \sum_i p_i.$$

(c) We say that a number is *q-smooth* if all of its prime factors $\leq q$. Show that if n is q-smooth for some constant q, the running time is $\Theta(n \log n)$.

Problem 2.5 Suppose you're working for the national bank in some country and you're asked to investigate a potential case of counterfeiting. The bank has seized a very large collection of n paper bills, suspecting that some of them may be counterfeits. Genuine bills have a unique serial number intricately woven into the fabric. For a counterfeit bill, though, its embedded serial number will likely be the same as that of another bill. (We're talking about good counterfeits here—the rest have already been weeded out by the bank.)

It's very difficult to read the serial number by examining the fabric directly, but the bank has a high-tech machine that, given two bills, will examine them both and then output whether or not the serials numbers on the bills are the same or different.

Your question is the following: Among the collection of n bills, is there a subset of more than $n/2$ of them that all have the same serial number as each other? Assume that the only feasible thing you can do with the bills is to insert two of them into the machine to test if they have the same serial number. Give an algorithm that requires only $O(n \log n)$ invocations of the machine.

Hint: Divide the n bills into two (roughly equal) halves. Show that if there is a set of more than $n/2$ bills with the same serial number, then at least one of the two halves must contain more than $(n/2)/2$ of them.

Problem 2.6 (Exercise 3.2 from the textbook.)

Prove that $Q \cdot Q^* = \mathbb{1}$, where Q^* denotes the element-wise complex conjugate of Q and $\mathbb{1}$ denotes the identity matrix. We define the complex conjugate of a complex number $(a + bi)$ to be $(a + bi)^* = (a - bi)$ where $a, b \in \mathbb{R}$.

Hint: First notice that the n-th roots of unity are roots of $x^n - 1$. Next notice that $x^n - 1 = (x - 1)(1 + x + x^2 + \cdots + x^{n-1})$ Conclude that

$$\sum_{j=0}^{n} \omega_n^{a \cdot j} = 0$$

for any integer a that is not a multiple of n and

$$\sum_{j=0}^{n} \omega_n^{a \cdot j} = n$$

if a is a multiple of n.

Second notice that the complex conjugate of ω_n^k is just ω_n^{-k}. To show this write out $\omega_n^k = e^{ki2\pi/n} = \cos(2\pi k/n) + i \sin(2\pi k/n)$ and use some well known trigonometric identities.