Algorithms – CMSC-27200
http://alg15.cs.uchicago.edu
Homework set #1 due January 14, 2015

Read the homework instructions on the website. The instructions that follow here are only an incomplete summary.

Hand in your solutions to problems marked "HW." Do not hand in problems marked "DO." If you hand in homework marked "XC" (extra credit), do so on separate and separately stapled sheets. Print your name on every sheet you submit. We ask you to use LaTeX to typeset your solutions; starting January 25, this will be required. Hand in your solutions on paper, do not email. Carefully study the policy (stated on the website) on collaboration, internet use, and academic integrity.

1.1 DO: Review **asymptotic notation** from Discrete Math.
Recall the notion of *asymptotic equality:* Let a_n and b_n be sequences of real numbers. We say that these sequences are asymptotically equal (notation: $a_n \sim b_n$) if

$$\lim_{n\to\infty} \frac{a_n}{b_n} = 1.$$

Prove:

(a) $n^2 - 3n\cos n + 1000 \sim n^2$
(b) $\sqrt{n+1} - \sqrt{n} \sim 1/(2\sqrt{n})$
(c) $\ln(1 + 1/n) \sim 1/n$

1.2 **HW (2+2 points):** Let $b(n)$ denote the number of binary digits of the positive integer n and let $d(n)$ denote the number of its decimal digits. (To make this definition unique, no initial zeros are allowed in this problem.) (a) Give a simple exact formula for $b(n)$ using the logarithm function and rounding. (b) Prove the asymptotic relation $b(n) = \Theta(d(n))$.

1.3 **HW (2+3 points):** Let a_n and b_n be sequences of real numbers greater than 1. Consider the following two statements:

(A) $a_n = \Theta(b_n)$
(B) $\ln a_n \sim \ln b_n$

Prove:

(i) (B) does not follow from (A). (Give a counterexample.)

(ii) (B) does follow from (A) if we make the stronger assumption that $a_n \to \infty$.

1.4 DO [binary search]: Let $A[1 \ldots n]$ be a $(0,1)$-array (every entry is 0 or 1). Assume $A[1] = 0$ and $A[n] = 1$. Design an algorithm that finds a value i such that $A[i] = 0$ and $A[i+1] = 1$ $(1 \leq i \leq n-1)$. You can access the array by making queries of the form $A[i] \stackrel{?}{=} 0$; each query carries unit cost; all other operations are free. Your goal is to minimize the cost. Your algorithm should work with cost $\log_2 n + O(1)$. Describe your algorithm in clear pseudocode.

1.5 DO: Recall the notion of *divisibility*: we say that the integer d divides the integer n (in notation, $d \mid n$), if there exists an integer x such that $dx = n$. If this is the case, we say d is a *divisor* of n and n is a *multiple* of d.

(a) What are the multiples of (a1) $d = 0$ (a2) $d = 1$ (a3) $d = -1$ (a4) $d = \pm 2$?

(b) What are the divisors of (b1) $n = 1$ (b2) $n = 0$ (b3) $n = -6$?

(c) True or false: $0 \mid 0$.

(d) Prove: if $d \mid a$ and $d \mid b$ then $d \mid a \pm b$.

(e) Prove: if a and b have the same divisors then $a = \pm b$.

(f) Prove: if $a \mid b$ and $b \mid a$ then $a = \pm b$.

(g) Prove: if a and b have the same divisors then $a = \pm b$.

(h) Prove: if a and b have the same multiples then $a = \pm b$.

(i) Prove: the number of positive divisors of the positive integer n is less than $2\sqrt{n}$.

1.6 DO: Let a, b, m be integers. We say that a is congruent to b modulo m (in notation: $a \equiv b \pmod{m}$) if $m \mid a - b$. For instance, $8 \equiv 73 \pmod{13}$.

(a) Prove: day k and day ℓ of a given month fall on the same day of the week if and only if $k \equiv \ell \pmod{7}$. (This is why modular arithmetic, the arithmetic of numbers modulo a given number, is also referred to as *calendar arithmetic*.)

(b) If today is Thursday then what day of the week will be 100 days from now? Note that $100 \equiv 2 \pmod{7}$. How is this observation relevant to the question?

(c) Review the notion of equivalence relations and equivalence classes from Discrete Mathematics.

(d) Prove: for any fixed m, the relation two integers are congruent modulo m is an equivalence relation. The equivalence classes corresponding to congruence modulo m are called *modulo m residue classes*. So two numbers a and b belong to the same modulo m residue class precisely if $a \equiv b \pmod{m}$.

(e) What are the residue classes modulo 2 ? What is the number of residue classes modulo m ?

(f) Prove: if $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ then $a \pm b \equiv c \pm d \pmod{m}$ and $ab \equiv cd \pmod{m}$.

(g) Prove: if $a \equiv b \pmod{m}$ then $\gcd(a, m) = \gcd(b, m)$.

(h) For what values of m is the following statement true for all a and b: "If $ab \equiv 0 \pmod{m}$ then $a \equiv 0 \pmod{m}$ or $b \equiv 0 \pmod{m}$."

1.7 DO: Prove Fermat's little Theorem: If p is a prime number then for every a we have $a^p \equiv a \pmod{p}$.
Hint: Assume $a > 0$ and suppose we have an unlimited supply of beads of a colors. (The beads are identical except for their color.) Count the types of necklaces with p beads we can make. Two necklaces count as being of the same "type" if one is a cyclic rotation of the other.

1.8 DO: Recall the communication complexity problem discussed in class. Use variables where we used specific numbers. When talking about k-bit itegers, we permit initial zeros, so strictly speaking, we are talking about integers with $\leq k$ bits, i. e., integers between 0 and $2^k - 1$.

Here is the setup.

Two processors, Alice and Bob, possess a string of n bits each; Alice's string is X, Bob's string is Y. The problem is to determine whether or not $X = Y$ with minimum number of bits communicated between Alice and Bob. The randomized protocol discussed in class to solve this problem efficiently depends on the choice of a parameter k and proceeds in the following steps:

1. Alice generates a k-bit prime, chosen uniformly at random from among all k-bit prime numbers. (Each k-bit prime has the same probability to be selected.)
2. Alice calculates the quantity $(X \bmod p)$, the remainder of the division of X (an n-bit integer) by p. Note that this remainder has at most k bits.

3. Alice sends p and $(X \bmod p)$ to Bob.
4. Bob calculates $(Y \bmod p)$.
5. If $(X \bmod p) \neq (Y \bmod p)$, i. e., if $X \not\equiv Y \pmod{p}$, then Bob says "NOT EQUAL." Else, Bob says "YES, EQUAL."

The cost of this protocol is at most $2k$ bits of communication (step 3); the cost of local computation by either Alice or Bob is ignored in the "communication complexity" model. We need to show that even for quite small values of k, Bob is not likely to make an error.

(a) Let Z be an n-bit non-zero integer. Let $p(Z)$ denote the number of distinct primes dividing Z. Prove: $p(Z) < n$.

(b) **XC (4 points):** Prove: $p(Z) \lesssim n/\log_2 n$. Here $a_n \lesssim b_n$ means $a_n \sim \min\{a_n, b_n\}$. Use the following fact, which is equivalent to the Prime Number Theorem: Let $s(x) = \sum_{p \leq x} \log p$ where the summation is over all primes $p \leq x$. Then $s(x) \sim x$.

(c) Let R denote the probability that Bob's conclusion is wrong. Use (a) to prove that $R = O(kn/2^k)$.

(d) Use (b) to prove that $R = O(n/2^k)$.

(e) Given an error-tolerance parameter $\epsilon > 0$, use (d) to recommend a value of k as a function of n such that $R \leq \epsilon$. Make k as small as you can.

1.9 **HW (6 points):** Consider the situation that Bob declared "NOT EQUAL." At this point Alice possesses the n-bit integer X, Bob the n-bit integer Y, both of them have the k-bit prime p and the k-bit integer $(X \bmod p)$ and they know that $X \not\equiv Y \pmod{p}$.

Devise a *deterministic* protocol which uses $O(k \log n)$ bits of communication and finds a position i such that $X[i] \neq Y[i]$. ($X[i]$ denotes the i-th bit of X.) Describe your protocol in pseudocode. Prove that it works within the stated amount of communication. Do not make assumptions on the value of k relative to n.

1.10 **HW (8 points):** In class we solved the Knapsack problem with integer *weights* in $O(nW)$ operations where n is the number of items and W is the weight limit. Solve the Knapsack problem under the assumption that the *values* are integers (but the weights are real numbers) in $O(nV)$ operations where V is the total value of all items. Describe your solution is pseudocode. Use dynamic programming. Half the credit

goes for a clear definition of the array of problems you are solving (the "brain" of the solution).

1.11 DO: Recall Strling's formula:

$$n! \sim \left(\frac{n}{\mathrm{e}}\right)^n \sqrt{2\pi n}.$$

Use Stirling's formula to show that $\log(n!) \sim n \log n$. Here "log" refers to base-2 logarithms but the same holds for logarithms to any base.

1.12 DO: Prove that $\log(n!) \sim n \log n$ without using Stirling's formula, taking the following steps.

(a) Prove: $\log(n!) \leq n \log n$ for all $n \geq 1$.

(b) Prove: for all k in the range $1 \leq k \leq n$ we have $n! \geq k^{n-k}$.

(c) Find a sequence of values k_n such that

$$\log(k_n^{n-k_n}) \sim n \log n.$$

1.13 DO: Consider the n-th row of the Pascal triangle: $\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{n}$. Prove that this sequence increases until the middle and then decreases.

1.14 DO: Prove that there exist constants c, d such that

$$\frac{\binom{2n}{n}}{4^n} \sim cn^d.$$

Determine the values of c and d. (Hint: Use Stirling's formula.)

1.15 DO **Exponential growth beats polynomial growth:** Let $c, d > 0$ be constants. Prove: $n^d = o((1+c)^n)$. Infer that for all sufficiently large n we have $n^d < (1+c)^n$.
(Recall the little-oh notation: for two sequences a_n, b_n we say that $a_n = o(b_n)$ if $\lim_{n\to\infty} a_n/b_n = 0$.)

1.16 DO: Recall the Prime Number Theorem (PNT):

$$\pi(x) \sim \frac{x}{\ln x}$$

where $\pi(x)$ denotes the number of primes $\leq x$.
(So, for instance, $\pi(10) = 4$, $\pi(100) = 25$, $\pi(\pi) = 2$.)
Let us write down a random $(0,1)$-string x of length 100. Let us interpret x as a number in binary. Estimate the probability that x is prime, using the PNT. (Assume that 2^{100} is "sufficiently large.")