# CS 70 Spring 2005 — Discrete Mathematics for CS, Clancy/Wagner — HW 6

## Due Thursday, March 10

**Coverage:** This assignment involves topics from the lectures of February 22 and 24 and March 1, and from Rosen sections 2.2, 2.4, 2.5, and pages 181-185 in section 2.6.

**Administrative reminders:** We will accept only unformatted text files or PDF files for homework submission. Include your name, login name, section number, and partner list in your submission. Give the command `submit hw6` to submit your answers to this assignment.

*Also:* It is time to switch partners!

**Homework exercises:**

1. **(14 pts.) Radix –2**
   *Radix* $-2$ *representation* is analogous to radix 2 (binary) representation. An integer $n$ is represented as $d_k\, d_{k-1} \cdots d_1\, d_0$ if $n = d_k(-2)^k + d_{k-1}(-2)^{k-1} + \ldots d_1(-2)^1 + d_0(-2)^0$ and all the $d_j$ are 0 or 1.

   (a) What's the radix $-2$ representation of 9?

   (b) Prove that every integer $n > 0$ can be represented in radix $-2$.

   (c) Is the representation unique? That is, can every integer $n > 0$ be represented in *exactly one way* in radix $-2$? Prove the claim, or display a counterexample.

2. **(14 pts.) Big-O notation**
   The purpose of this problem is to teach you Big-**O** notation in a careful way. First, study the following.

   *Formally:* If $f(n), g(n)$ are two non-negative functions of a single integer variable, the statement $f(n) \in \mathbf{O}(g(n))$ means that

   $$\exists N_0 \in \mathbf{N}. \quad \exists C \in \mathbf{N}. \quad \forall x \in \mathbf{N}. \quad x \geq N_0 \implies 0 \leq f(x) \leq C \cdot g(x).$$

   In other words, $\mathbf{O}(g(n))$ is the set of functions $\{f_i(n) : \exists N_0 \in \mathbf{N}.\, \exists C \in \mathbf{N}.\, \forall x \in \mathbf{N}.\, x \geq N_0 \implies f_i(x) \leq C \cdot g(x)\}$. This is the definition of Big-**O** notation.

   *Informally:* $f(n) \in \mathbf{O}(g(n))$ means, roughly, that $f(n)$ grows "no faster than" $g(n)$ (except possibly for a constant factor), as $n$ gets large. For instance, $n^2 \in \mathbf{O}(n^2)$, $n(n+1)/2 \in \mathbf{O}(n^2)$, and $10000n^2 \in \mathbf{O}(n^2)$, because these functions all grow at asymptotically the same rate (ignoring constant factors). Also, $n^2 \in \mathbf{O}(n^3)$, because $n^2$ grows more slowly than $n^3$ does, as $n$ gets large.

   *Some basic facts:* If $f(n) \in \mathbf{O}(g(n))$ and $f'(n) \in \mathbf{O}(g'(n))$, then $f(n) + f'(n) \in \mathbf{O}(g(n) + g'(n))$. If $f(n) \in \mathbf{O}(g(n))$ and $f'(n) \in \mathbf{O}(g'(n))$, then $f(n) \times f'(n) \in \mathbf{O}(g(n) \times g'(n))$.

   *Common notation:* Instead of writing $f(n) \in \mathbf{O}(g(n))$, almost everyone instead writes $f(n) = \mathbf{O}(g(n))$. Strictly speaking, this is a sloppy abuse of notation, but this practice is widespread; you are guaranteed to see it throughout your studies of computer science, so be prepared. Also, we often write something like $n^2$ as a shorthand for the function $f(n) = n^2$, just to make our life easier.

Now, with that background established, do the following problems:

(a) Prove that $n^2 + 2003 \in \mathbf{O}(n^3)$.
*Hint:* One possible approach is to give an example of constants $N_0, C$ that satisfy the definition.

(b) Prove that $100n^2 \lg n \in \mathbf{O}(n^3)$.

(c) True or false: There exists $e \in \mathbf{N}$ such that $2^n \in \mathbf{O}(n^e)$. Briefly justify your answer.

(d) Prove that if $f(n) \in \mathbf{O}(g(n))$ and $g(n) \in \mathbf{O}(h(n))$, then $f(n) \in \mathbf{O}(h(n))$.

(e) Critique the following argument. Is the reasoning valid? If not, why not? If there is an error, identify the erroneous step and explain what's wrong with it.

> We have $n^2 = \mathbf{O}(n^4)$.
> Also, we have $n^2 = \mathbf{O}(n^3)$.
> By transitivity, it follows that $\mathbf{O}(n^4) = \mathbf{O}(n^3)$.
> This means that $n^4 = \mathbf{O}(n^3)$.

**3. (10 pts.) Power detection**

(a) Design an efficient algorithm for a function named *ispower?* that, given positive integers $n$ and $k$ with $k < n$, tests whether $n$ is a perfect $k$-th power. That is, *ispower?*$(n, k)$ should return true if and only if $\exists x \in \mathbf{N} \,.\, x^k = n$. Your algorithm should run, in the worst case, in time $\mathbf{O}((\lg n)^c)$ for some constant $c$.

(b) Prove the running time bound on your algorithm from part (a).

**4. (12 pts.) Binary gcd**

(a) Prove that the following statements are true for all $m, n \in \mathbf{N}$.

| | |
|---|---|
| If $m$ is even and $n$ is even, | $\gcd(m,n) = 2\gcd(m/2, n/2)$. |
| If $m$ is even and $n$ is odd, | $\gcd(m,n) = \gcd(m/2, n)$. |
| If $m, n$ are both odd and $m \geq n$, | $\gcd(m,n) = \gcd((m-n)/2, n)$. |

(b) Give an algorithm that computes $\gcd(m,n)$ using at most $\mathbf{O}(\lg m + \lg n)$ subtractions, halvings, doublings, and odd/even tests.

**5. (10 pts.) Diophantine equations**
Given positive integers $a, b, c$, you are to find an integer solution (for $x, y, z$) to the equation $ax + by + cz = 1$.

(a) Design an efficient algorithm to find such a solution, assuming that $\gcd(a,b) = \gcd(a,c) = \gcd(b,c) = 1$.

(b) Design an efficient algorithm to find such a solution, assuming only that $\gcd(a,b,c) = 1$.