1 Homework 1 Due April 17, first 5 minutes of class

Properties of Logarithms

Lemma 1

Let x and y be arbitrary positive real numbers, let a be any real number, and let $b > 1$ and $c > 1$ be real numbers.

1. $\log_b$ is a strictly increasing function, and therefore is a one-to-one function.
   That is, if $x > y$, then $\log_b x > \log_b y$; and if $\log_b x = \log_b y$, then $x = y$.
2. $\log_b(x^a) = a \log_b x$.
3. $x^{\log_b y} = y^{\log_b x}$.
4. To convert from one base to another: $\log_c x = (\log_b x)/(\log_b c)$.

Homework 1.01

Prove part 3 of Lemma 1. *Hint*: Other parts of the lemma may be used without proof.

Asymptotic Analysis

Homework 1.02

Let $p(n)$ be a polynomial in n of degree k defined as

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$$

where $a_k > 0$. Show that $p(n)$ is in $\Theta(n^k)$. *Hint*: This is easiest using definitions based on ratios.

Efficiency

Homework 1.03

You have 70 coins that are all supposed to be gold coins of the same weight, but you know that one coin is fake and weighs less than the others.

You have a balance scale; you can put any number of coins on each side of the scale at one time, and it will tell you if the two sides weigh the same, or which side is lighter if they don't weigh the same.

Outline an algorithm for finding the fake coin, trying to do as few weighings as you can. How many weighings will you do in the worst case? Best case?

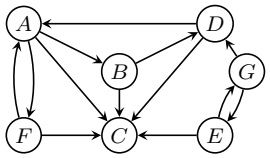

Figure 1: Digraph for homeworks 1.04–1.07

1.1 Depth-First Search

Homework 1.04–1.05

Find the depth-first search tree and the discovery and finishing times for the graph shown in Figure 1 **with E as the starting vertex** under two assumptions about the adjacency-list order:

1.04: Each adjacency list is in alphabetical order (A, B, C, D, E, F, G).

1.05: Each adjacency list is in reverse alphabetical order.

Homework 1.06–1.07

Classify the edges of the graph shown in Figure 1 as one of "tree," "back," "cross," or "descendant" (also called "forward" or "frond"), **with E as the starting vertex** under two assumptions about the adjacency-list order:

1.06: Each adjacency list is in alphabetical order (A, B, C, D, E, F, G).

1.07: Each adjacency list is in reverse alphabetical order.

Hint: Re-use your work on 1.04 and 1.05.

Homework 1.08

If edge vw turns out to be a **back edge** during the DFS of a directed graph, what color(s) can w have when edge vw is checked? Explain your answer.

Homework 1.09

Give an example of a directed graph in which a depth-first search backs up from a vertex v before all the vertices that can be reached from v via one or more edges are discovered.

Homework 1.10

Suppose v and w are distinct vertices in the same directed tree, but have no ancestor/descendant relationship. Show that there is some third vertex c, called their least common ancestor, such that there are tree paths from c to v and from c to w, and these paths have no edges in common. *Hint*: Use the fact that every v vertex in a tree has exactly one path from the root to v.