College of Computer & Information Science
Northeastern University
CS5800: Algorithms

Spring 2011
Handout 3
26 January 2011

Sample Solution to Problem Set 1

1. (10 points) On the toughness of glass beakers

You are given the task of determining the toughness of a collection of identical glass beakers that are going to be used in space experiments by NASA. The question you need to answer is simple: what is the highest floor of the Empire State Building from which the glass beaker can be dropped safely without shattering it?

Of course, in order to answer this question satisfactorily, you should be allowed to break at least one beaker. One simple solution is to start from floor 1, and for each floor going up, repeatedly drop the beaker to the ground from the floor, until the beaker breaks. Let us call the drop from a floor as a *test*. If the Empire State Building has n floors, then this could require as many as n tests.

Suppose you are given a budget of k beakers. You would like to determine the highest floor from which it is safe to drop the beaker, using as few tests as possible, while breaking at most k beakers. Let $f_k(n)$ denote the maximum number of tests you need to do with a budget of k beakers. In our solution above with $k = 1$, we obtained $f_1(n) = n$.

(a) Describe your best possible strategy for $k = 2$ that aims to minimize the number of tests. What is $f_2(100)$? Derive an expression for $f_2(n)$.

(b) Can you generalize your approach to arbitrary k? Your strategy must have the property that $f_k(n)$ grows asymptotically slower than $f_{k-1}(n)$; i.e. $\lim_{n\to\infty} f_k(n)/f_{k-1}(n) = 0$, for all $k \geq 2$.

Answer:

(a) If we have two beakers, suppose the maximum number of tests we want to incur is w. Then, it makes sense to have the first test at floor w. If beaker breaks, we have to start from floor 1, because we have only one chance remaining, we can't skip any floors. So in the worst case, we will test from 1 to $(w-1)$. The total tests in worst case will be w.

If beaker doesn't break, we can go up $(w-1)$ floors, if then the second test fails, it will still need to test floor $(w+1)$ to $(2w-2)$, again yielding a total of w tests.

If the second test succeeds, we go to $(w-2)$, and so on $(w-3)$, $(w-4)$, ... For any additional test we do using the first beaker, we subtract a test from the maximum we would do with the second beaker, so that the total number of tests is minimized.

Thus, what we need to solve is the equation

$$w + (w-1) + (w-2) + ... + 1 = w(w+1)/2 = 100 \Rightarrow w = 14.$$

More generally, we have

$$w + (w-1) + (w-2) + ... + 1 = w(w+1)/2 = n,$$

from which we can solve for w. Roughly speaking, $w = \Theta(\sqrt{n})$.

A simpler strategy that has the same number of tests (under the big-Oh notation) is to test at evenly spaced floors $\lfloor\sqrt{n}\rfloor$, $2\lfloor\sqrt{n}\rfloor$, ..., $\lfloor\sqrt{n}\rfloor \cdot \lfloor n/\lfloor\sqrt{n}\rfloor\rfloor$. The number of tests in the worst-case is at most $2\sqrt{n} + 1 = \Theta(\sqrt{n})$.

(b) Let us consider the case where $k = 3$. Suppose we wanted to achieve a maximum of w tests. Then, where should we have the first test? Suppose that we do the first test from floor ℓ; if the beaker breaks, then the number of tests we need for the first $\ell - 1$ floors using 2 beakers is $f_2(\ell - 1)$. So we want ℓ such that $1 + f_2(\ell - 1) = w$. Thus, the first testing floor ℓ and the number w of tests are approximately related as $\Theta(\sqrt{\ell}) = w$.

Suppose we test at evenly-spaced floors $\ell, 2\ell, \ldots, \lfloor n/\ell \rfloor \cdot \ell$, then the maximum number of tests is at most $n/\ell + \Theta(\sqrt{\ell}) + 1$. As we increase ℓ, the first term decreases, but the second increases. To reach a minimum, we need to set the two terms to be the same; that is, $n/\ell = \Theta(\sqrt{\ell})$, implying that $\ell = \Theta(n^{2/3})$, and the number of tests being $\Theta(n^{1/3})$. Thus $f_3(n) = \Theta(n^{1/3})$

With 4 beakers, we can use the same approach as above, except that if the first test floor is ℓ, then we have $\Theta(\ell^{1/3})$ tests. Again, the maximum number of tests is $n/\ell + \Theta(\ell^{1/3}) + 1$, yielding the best solution at $\ell = \Theta(n^{3/4})$, implying that $f_4(n) = \Theta(n^{1/4})$.

Generalizing the above approach to k beakers, we get $f_k(n) = \Theta(n^{1/k})$ (for fixed k). Note that

$$\lim_{n\to\infty} \frac{f_{k+1}(n)}{f_k(n)} = \lim_{n\to\infty} \frac{n^{1/(k+1)}}{n^{1/k}} = \lim_{n\to\infty} \frac{1}{n^{1/(k(k+1))}} = 0.$$

2. (10 points) Comparison of functions

Arrange the following functions in order from the slowest growing function to the fastest growing function. Briefly justify your answer. (*Hint:* It may help to express each function as a power of 2 and then compare. It may also help to plot the functions and obtain an estimate of their relative growth rates.)

$$(\lg n)^{\lg n} \quad (\frac{3}{2})^{2n} \quad n^{100} \quad 2^{\sqrt{\lg n}}$$

Answer: The order from slowest growing to the fastest growing function is: $2^{\sqrt{\lg n}}$, n^{100}, $(\lg n)^{\lg n}$, $(\frac{3}{2})^{2n}$. We prove this using three steps: (i) $2^{\sqrt{\lg n}} = o(n^{100})$, (ii) $n^{100} = o((\lg n)^{\lg n})$, and (iii) $n^{100} = o((\frac{3}{2})^{2n})$.

(i) We use the connection between asymptotic notation and limits.

$$\begin{aligned}
\lim_{n\to\infty} \frac{2^{\sqrt{\lg n}}}{n^{100}} &= \lim_{n\to\infty} \frac{2^{\sqrt{\lg n}}}{2^{100 \lg n}} \\
&= \lim_{n\to\infty} \frac{1}{2^{100 \lg n - \sqrt{\lg n}}} \\
&= 0,
\end{aligned}$$

since $100 \lg n - \sqrt{\lg n}$ tends to ∞ as n tends to ∞.

(ii) We again use the connection between asymptotic notation and limits.

$$\begin{aligned}\lim_{n\to\infty} \frac{n^{100}}{(\lg n)^{\lg n}} &= \lim_{n\to\infty} \frac{2^{100\lg n}}{2^{\lg n \lg\lg n}} \\ &= \lim_{n\to\infty} \frac{1}{2^{\lg n(\lg\lg n - 100)}} \\ &= 0,\end{aligned}$$

since $\lg\lg n - 100$ tends to ∞ as n tends to ∞.

(iii) We again use the connection between asymptotic notation and limits.

$$\begin{aligned}\lim_{n\to\infty} \frac{(\lg n)^{\lg n}}{(\frac{3}{2})^{2n}} &\leq \lim_{n\to\infty} \frac{(\lg n)^{\lg n}}{(2^n} \\ &= \lim_{n\to\infty} \frac{2^{\lg n \lg\lg n}}{2^n} \\ &\leq \lim_{n\to\infty} \frac{1}{2^{\lg n(n - \lg n \lg\lg n)}} \\ &= 0,\end{aligned}$$

since $n - \lg n \lg\lg n$ tends to ∞ as n tends to ∞.

3. (10 points) An alternative gcd algorithm

Euclid's algorithm computes the gcd of two positive numbers a and b ($a \geq b$) by reducing the problem to that of computing the gcd of b and ($a \bmod b$), where $a \bmod b$ is often much smaller than a. Thus the algorithm rapidly converges to the final result. While the calculation of $a \bmod b$ is not hard, it requires division and may be more expensive to do on some very simple computing devices. In the world of binary arithmetic, division by 2 is much easier to compute. Consider the following alternative algorithm for gcd using subtraction and division by 2, developed below through a series of exercises.

(a) Show that if a and b are both even, then $\gcd(a, b) = 2 \cdot \gcd(a/2, b/2)$.

Answer: Any even number is divisible by 2. If a and b are both even numbers, then 2 is a common divisor for a and b:

$\gcd(a, b) = \gcd(2 \times \frac{a}{2}, 2 \times \frac{b}{2}) = 2 \cdot \gcd(a/2, b/2)$.

(b) Show that if a is even and b is odd, then $\gcd(a, b) = \gcd(a/2, b)$. (Similarly, if a is odd and b is even, then $\gcd(a, b) = \gcd(a, b/2)$.)

Answer: If a is even and b is odd, then 2 is a divisor of a, but not a divisor of b; therefore 2 is not a common divisor of a and b:

$\gcd(a, b) = \gcd(a/2, b)$.

(c) Show that if a and b are both odd, then $\gcd(a, b) = \gcd((a - b)/2, b)$.

(*Hint:* Show that $\gcd(a, b) = \gcd(a - b, b)$; for inspiration, see the proof of correctness for the Euclidean Algorithm. If a and b are both odd, what is true about $a - b$? Now apply your result from part (b) above.)

Answer: Let us consider that $a \geq b$.

Since $\gcd(a, b) \mid a$ and $\gcd(a, b) \mid b$, from the definition of divides there exist integers m and n such that:

$a = m \times \gcd(a, b)$

$b = n \times \gcd(a, b)$

Subtracting the left and right hand sides of the two equations above, we obtain:

$a - b = m \times \gcd(a, b) - n \times \gcd(a, b) = (m - n) \times \gcd(a, b)$

So, by definition of divides:

$\gcd(a, b) \mid (a - b)$

Since $\gcd(a, b)$ divides both b and $a - b$, then $\gcd(a, b) \mid \gcd((a - b), b)$. Similarly, since $\gcd((a - b), b)$ divides both a and b, then $\gcd((a - b), b) \mid \gcd(a, b)$. Hence $\gcd(a, b) = \gcd((a - b), b)$.

(Observation: This is one step of the Euclidean algorithm, which uses subtraction at each step.)

Since a and b are both odd numbers, their difference is an even number. To show this, we can write a as: $a = 2 \times m + 1$ and b as: $b = 2 \times n + 1$, which is a general form of odd numbers.

$a - b = 2 \times m + 1 - 2 \times n - 1 = 2 \times (m - n)$, which is even.

Since $(a - b)$ is even and b is odd, we can apply (b) discussed before:

$\gcd(a, b) = \gcd((a - b), b) = \gcd((a - b)/2, b)$

(d) Apply the above three claims repeatedly to compute the following: $\gcd(323, 76)$, $\gcd(693, 378)$. Show your work.

Answer:

$$\begin{aligned}
\gcd(323, 76) &= \gcd(323, 38) \\
&= \gcd(323, 19) \\
&= \gcd(152, 19) \\
&= \gcd(76, 19) \\
&= \gcd(38, 19) \\
&= \gcd(19, 19) \\
&= \gcd(19, 0) \\
&= 19.
\end{aligned}$$

$$\begin{aligned}
\gcd(693, 378) &= \gcd(693, 189) \\
&= \gcd(252, 189) \\
&= \gcd(189, 126) \\
&= \gcd(189, 63) \\
&= \gcd(63, 63) \\
&= \gcd(63, 0) \\
&= 63.
\end{aligned}$$

4. (10 points) An addition circuit

Exercise 1.30 of text.

Answer:

(a) We divide the n numbers into $n/2$ groups. Add all these groups, we get $n/2$ numbers, repeat it until we get the result. The depth is $\log n$.

We can parallelize between each groups. In each group, the cost is $\log m$.

So the final depth is $O(\log n \log m)$.

(b) One of the two numbers can be the bitwise sum, completely ignoring the carry. The other can be just the carry bits. Each of these can be calculated using bitwise operations in depth 1. We should note the fact that the carries have to be shifted one left. We define r and s like the following.

$$\begin{aligned} r &= x \oplus y \oplus z \\ s' &= (x \wedge y) \vee (y \wedge z) \vee (z \wedge x) \\ s &= s' \ll 1 \end{aligned}$$

Then we can re-express $(x + y + z)$ as the sum of just two binary numbers $(r + s)$.

(c) We implement the standard grade school multiplication of n-bit numbers x and y. This involves the addition of n numbers each with at most $2n$ bits (after adding the least significant 0s wherever necessary).

How do we add these n numbers in $O(\log n)$ depth. We will divide them into groups of 3, and apply the idea of part (b) to obtain 2 new numbers (the r and s) for each group. Thus, with one depth, we go from n numbers to $2n/3$ numbers. We repeat this process in a tree-like fashion. After 1 more depth, we have $(2/3)(2n/3) = 4n/9$ numbers. And so on.

After at most $\log_{3/2} n = O(\log n)$ depth, we are left with the addition of at most two $O(n)$ bit numbers. By the carry lookahead addition circuit, that can be done in $O(\log n)$ depth.

So the total depth of multiplication will be $O(\log n)$.