CPSC-320 Intermediate Algorithm Design and Analysis Winter 2000

Solutions to Homework # 1

1. Assume all numbers are positive and have no fractions.

   Let $x = x_n x_{n-1} \dots x_1$ and $y = y_n y_{n-1} \dots y_1$ be two n-digit numbers.

   To multiply x and y, form n new numbers as follows. The first new number is x multiplied by y_1. The next new number is x multiplied by y_2, with an additional 0 inserted as the rightmost digit (i.e. the second new number is $x \times y_2 \times 10$). The ith new number is x multiplied by y_i with $i-1$ additional 0's inserted as the rightmost digits. Now, write these new numbers with the rightmost digits aligned, so that the digits of the new numbers are arranged in columns. Repeatedly add all digits in each column plus the carry from the previous column (if any). The carry from the last column becomes the most significant digit(s) of the product.

   Let $add(n)$ be the number of elementary additions needed to add two n-bit binary numbers in the worst case. Then $add(n) = 2(n-1)+1 = 2n-1 = O(n)$ in the worst case because we have 2 elementary additions for each pair of digits, except the first one (right most bit addition) where there is one elementary addition. Hence, $add(n)$ is linear in n.

   Let $mult(n)$ be the number of elementary additions needed to multiply two n-bit binary numbers in the worst case. In what follows, we count elementary additions in which 0 is added to something else. With the above algorithm, the worst case occurs when both numbers are consist of n bits of value 1. In this case, the number of bits in the new numbers is $n, n+1, n+2, \dots 2n$. Hence, if the columns of new numbers are ordered from 1 (rightmost) to $2n-1$ (leftmost), we have the following: n elementary additions are done to add the bits in columns 1 and 2 (note that there is no carry from column 1 to column 2), $n+1$ elementary additions are done to add the carry plus the bits in columns $i, 3 \le i \le n$, and the number of elementary additions done in the remaining columns is $n + (n-1) + \dots + 2$ (because there is one fewer digit in each successive column). Hence, the total number of elementary additions is

$$2n + (n-2)(n+1) + \sum_{i=2}^{n} i = (3/2)(n^2+n) - 3.$$

   Hence, the (worst-case) number of elementary additions needed to multiply two n-bit binary numbers in this way is quadratic in n.

   Comment: Consider the following multiplication algorithm, which is conceptually simpler than the one above: To multiply two numbers, say p and q, use a variable called *sum* and initialize it to p. Then, simply iterate $q-1$ times: $sum \leftarrow sum + p$ and return *sum* as the result. Let $f(n)$ be the (worst case) number of elementary additions done by this algorithm. Then $f(n) = (2n-1)(q-1) = (2n-1)(2^n-2) = \Theta(n2^n)$. Note that in the worst case, q is replaced by $2^n - 1$ since q is an n-bit binary number and the maximum possible decimal value of n-bit binary numbers is $2^n - 1$. This is an exponential time algorithm, which explains why it is not the preferred method!

2. **Claim:** The total number of subsquares in an $n \times n$ grid is $\sum_{k=1}^{n} k^2$, for all $n \ge 1$.

   Proof: by induction.

Basis: Let $n = 1$. It is easy to see that a 1×1 grid has one subsquare, namely itself. Also, $\sum_{k=1}^{1} k^2 = 1$, and so the claim is true for $n = 1$.

Induction Hypothesis: Suppose that for some given n, the total number of subsquares in an $n \times n$ grid is $\sum_{k=1}^{n} k^2$.

Induction Step: We show that the claim is true for an $(n+1) \times (n+1)$ grid. We can think of the total subsquares in such a grid as being the sum of two parts. The first part consists of all subsquares in the $n \times n$ grid forming the top left $n \times n$ subgrid of our $(n+1) \times (n+1)$ grid (see the picture). By the induction hypothesis, the number of subsquares in the first part is $\sum_{k=1}^{n} k^2$.

```
. . . . . . . .|.
. . . . . . . .|.
. . . . . . . .|.
. . . . . . . .|.
. . . . . . . .|.
. . . . . . . .|.
. . . . . . . .|.
. . . . . . . .|.
---------------
. . . . . . . . .
```

The second part consists of all subsquares that include a point from the rightmost (i.e. $(n+1)$st) column or bottom (i.e. $(n+1)$st) row of the grid. If we count all subsquares in the second part, we see that there are $(2(n+1)-1)$ 1×1 subsquares, $(2(n+1-1)-1)$ 2×2 subsquares, and more generally there are $(2(n+1-k+1)-1)$ $k \times k$ subsquares, $1 \le k \le n+1$. Hence the total number of subsquares in the second part is

$$\sum_{k=1}^{n+1}(2(n+1-k+1)-1) = 2n(n+1)-(n+1)(n+2)+3(n+1) = (n+1)(2n-n-2+3) = (n+1)^2.$$

Hence, the total number of subsquares is

$$\sum_{k=1}^{n} k^2 + (n+1)^2 = \sum_{k=1}^{n+1} k^2,$$

and the induction step is completed.

3. Note that if A runs in time $f(n)$ and B runs in time $g(n)$, then A is faster than B on an input of size n if and only if $f(n) < g(n)$.

   (i)

   (a,b) No. for example, let $f(n) = n, g(n) = n \log_2 n$. When $n = 1, f(n) > g(n)$ but when $n > 2, f(n) < g(n)$.

   The answer to (a) and (b) is "No" for relationships (ii) – (v), using the same example.

   (c) Yes.

(i) states that $f(n)\log_2 n = O(g(n))$. So there exist some constants c and n_0, such that for all $n > n_0$

$$f(n)\log_2 n \le cg(n) \Rightarrow f(n) \le (c/\log_2 n)g(n) \text{ for all } n > n_0.$$

Let $n_1 = 2^c$. Then $\log_2 n_1 = c$ and so if $n > n_1$,

$$\log_2 n > c \Rightarrow c/\log_2 n < 1 \Rightarrow (c/\log_2 n)g(n) < g(n) \Rightarrow \text{ for all } n > \max\{n_0, n_1\}, f(n) < g(n).$$

Hence, A is faster than B.

(d)No. Since (c) is correct, (d) must be wrong.

(ii)

(c)Yes: $g(n)\ f(n)\log_2 n \Rightarrow g(n) = \Omega(f(n)\log_2 n) \Rightarrow$ for all n greater than some $c, f(n) < g(n)$.

(d)No.

(iii)

(c)Yes: $g(n) = \Theta(f(n)\log_2 n) \Rightarrow g(n) = \Omega(f(n)\log_2 n)$
$\Rightarrow$ for all n greater than some $c, f(n) < g(n)$.

(d)No.

(iv)

(c)No. For example, suppose $f(n) = g(n) = n$. Then $g(n) = O(f(n)\log_2 n)$ but A is not faster than B for any n.

(d)No. Same counter example used in (c)

(v)

(c,d)No; the same example used in (iv, c) works here too.

4. (a) Let

$$S = \sum_{k=0}^{n} x^k = 1 + x + x^2 + \ldots + x^n$$

$$\Rightarrow S = x \cdot \sum_{k=0}^{n} x^k = x + x^2 + x^3 + \ldots + x^{n+1}$$

$$\Rightarrow (1-x)S = 1 - x^{n+1}$$

because $x \neq 1$ therefore:

$$S = \frac{x^{n+1}}{x-1}$$

We can also use induction to prove this:

Base case: when $n = 0$

$$x^0 = \frac{x-1}{x-1} \Rightarrow 1 = 1$$

Assume the equation is true for n:

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}, (x \neq 1)$$

prove for n+1:

$$\sum_{k=0}^{n+1} x^k = \frac{x^{n+2}-1}{x-1}$$

$$\begin{aligned}\sum_{k=0}^{n+1} x^k &= x^{n+1} + \sum_{k=0}^{n} x^k \\ &= x^{n+1} + \frac{x^{n+1}-1}{x-1} \\ &= \frac{(x+1)x^{n+1} + x^{n+1} - 1}{x-1} \\ &= \frac{x^{n+2} - x^{n+1} + x^{n+1} - 1}{x-1} \\ &= \frac{x^{n+2}-1}{x-1}\end{aligned}$$

Therefore,

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$$

b)Derive a closed form expression for the sum

$$\sum_{k=1}^{n} kx^k, x \neq 1$$

Suggestion: what do you get when you take the derivative with respect to x of the equation in part (a) of this problem?

Verify that your expression is correct for $x = 1/2$ and $n = 4, 5$.

sol:

from part(a), we know

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$$

differentiate both sides

$$\sum_{k=1}^{n} kx^{k-1} = \frac{(x-1)[(n+1)x^n] - (x^{n+1}-1)\cdot 1}{(x-1)^2}$$

$$\sum_{k=1}^{n} kx^k = x \cdot \frac{(x-1)[(n+1)x^n] - (x^{n+1}-1)}{(x-1)^2}$$

$$\sum_{k=1}^{n} kx^k = \frac{(n+1)x^{n+2} - (n+1)x^{n+1} - x^{n+2} + x}{(x-1)^2}$$

therefore,

$$\sum_{k=1}^{n} kx^k = \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(x-1)^2}$$

Check for $x = \frac{1}{2}$, $n = 4$

$$left = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 4 \cdot \frac{1}{16} = \frac{13}{8}$$

$$right = \frac{4(\frac{1}{2})^6 + 5(\frac{1}{2})^5 + \frac{1}{2}}{(-\frac{1}{2})^2} = \frac{\frac{4}{64} - \frac{5}{32} + \frac{1}{2}}{\frac{1}{4}} = \frac{16}{64} - \frac{40}{64} + \frac{128}{64} = \frac{104}{64} = \frac{13}{8}$$

therefore left = right

Check for $x = \frac{1}{2}, n = 5$

$$left = \frac{13}{8} + 5 \cdot \frac{1}{32} = \frac{57}{32}$$

$$right = \frac{5(\frac{1}{2})^7 - 6(\frac{1}{2})^6 + \frac{1}{2}}{(-\frac{1}{2})^2} = frac\frac{5}{128} - \frac{6}{64} + \frac{1}{2}\frac{1}{4} = \frac{20}{128} - \frac{48}{128} + \frac{256}{128} = \frac{228}{128} = \frac{57}{32}$$

therefore left = right