COT4810.02, Fall 2002 October 15, 2002
S. Lang **Partial Solution Keys to Assignment 1 – 3** (revised on 10/17/2002)

Hw#1, Q6: A website that demonstrates Waltz's edge labeling algorithm is at http://www.cis.ohio-state.edu/~dove/730/lab1/lab1.html.

Hw#2, Q1 (Analysis of Strassen's matrix multiplication algorithm)

Since the time complexity of Strassen's algorithm satisfies the following recurrence:

$$T(n) = 7T(n / 2) + 18(n / 2)^2 = 7T(n / 2) + O(n^2),$$

where n denotes the matrix dimension and is assumed to be a power of 2 (for simplicity reasons), the recurrence can be solved using the master theorem (see below) with $a = 7$, $b = 2$, and $k = 2$. In this case, since $a = 7 > b^k = 2^2 = 4$, the solution becomes (according to the master theorem)

$$T(n) = O(n^{\log_b a}) = O(n^{\log_2 7}) = O(n^{2.81}).$$

The master theorem for solving recurrences: Suppose the running time of a (divide-and-conquer) algorithm $T(n)$ satisfies the following recurrence, where n measures the size of the given problem:

$T(n) = aT(n / b) + O(n^k)$, when n is a power of b, where $a \geq 1$, $b > 1$, and k are constants, and $T(n)$ is non-decreasing. The solution to this recurrence is as follows:

$$T(n) = \begin{cases} O(n^{\log_b a}), \text{ if } a > b^k \\ O(n^k \lg n), \text{ if } a = b^k \\ O(n^k), \text{ if } a < b^k \end{cases}$$

Examples include binary search ($a = 1$, $b = 2$, $k = 0$), and merge-sort ($a = b = 2$, $k = 1$).

Hw#2, Q3 (Transform the Partition Problem to the Subset problem)

The *Subset Problem*: The input is a list of n positive integers $a_1, a_2, \ldots, a_n$, and another integer b. We assume $b \leq sum/2$, where sum = the sum of the n integers $a_1, a_2, \ldots, a_n$. The *Subset Problem* attempts to determine (Yes or No) if there is a sublist of the n integers whose sum equals b. Notice that the Partition problem is a special case of the Subset problem (with $b = sum/2$). Solve the Subset problem and analyze its time complexity assuming there is a function (subroutine) that solves the Partition problem, where the time complexity of solving the Partition problem is given by a function $f(n, sum)$.

Answer: Transform the Subset problem into a Partition problem by adding another integer $c = sum – 2b$ into the list. (Note that the value c is ≥ 0 by the assumption that $b \leq sum/2$; when $c = 0$ the Subset problem becomes the Partition problem.) Run the Partition problem's algorithm with the input consisting of $(n + 1)$ integers $a_1, a_2, \ldots, a_n$, and c, then output Yes or No exactly the same as what the Partition problem outputs. The time complexity is $f(n + 1, 2sum – 2b)$.

Demonstration: Suppose the Subset problem is given 5 integers 2, 5, 7, 3, 9, and $b = 8$. (Note that $sum = 26$, and $b = 8 \leq sum/2$.) Add $c = sum – 2b = 10$ to the list, then solve the Partition problem with 6 integers 2, 5, 7, 3, 9, and 10. Now the new sum is 36; a solution consists of integers 5, 3,10. Taking out the value $c = 10$ yields integers 5 and 3, which gives a solution to the original Subset problem (since $5 + 3 = 8 = b$).

Justification (why this scheme always works): There are two parts to the "proof":

(Part 1) Suppose there is a solution (a sublist A) to the Subset problem. We want to prove that there is a solution to the corresponding Partition problem. First, we note that the sum of the numbers in the Partition problem = $sum + c = sum + (sum – 2b) = (2sum – 2b)$. We claim the union of A and the value c is a solution to the Partition problem. This is true because

sum of the numbers in $A = b$; thus,

sum of the numbers in $A + c = b + (sum - 2b) = sum - b = ½ (2sum - 2b)$.

(Part 2) Suppose there is a solution (a sublist B) to the Partition problem. We want to prove that the original Subset problem also has a solution. We may assume that the sublist B contains the value c (because otherwise, use the complement of B within the list A union $\{c\}$.) We now claim taking the value c out of list B gives a solution to the original Subset problem.

This is true because since B is a solution to the Partition problem, the sum of the values in $B = ½ (sum + c) = ½ (sum + sum - 2b) = sum - b$. Taking the value c out of the list B yields a total of $(sum - b) - c = (sum - b) - (sum - 2b) = b$.

Hw#3, Q3 (The Chinese remainder theorem)

Solve the following system of equations modulus 105:

$x = 2 \pmod 3$

$x = 3 \pmod 5$

$x = 4 \pmod 7$

The Chinese remainder theorem says there is a (unique) solution modulus 105 = 3*5*7, because the moduli 3, 5, and 7 are pairwise co-prime (no common divisors between any two except for value 1). First, should eliminate some inefficient algorithms:

```
/* solution 1 */
for (k = 0; k < 105; k++)
        if (k % 3 == 2 && k % 5 == 3 & k % 7 == 4) {
                printf("x = %d\n", k);
                break;  /* found it */
        }

/* solution 2 */
for (k = 2; k < 105; k += 3)
        if (k % 5 == 3 & k % 7 == 4) {
                printf("x = %d\n", k);
                break;  /* found it */
        }
```

Both algorithms are of an exponential time complexity in terms of the input size, because the input size is $\lg m + \lg n + \lg p$, when the three moduli are m, n, and p, but the above algorithms have a time complexity $O(mnp)$. An efficient, polynomial-time algorithm is based on Euclid's GCD algorithm, which is based on the following theorem:

If $a = bq + r$, where $b > 0$, then $GCD(a, b) = GCD(b, r)$.

Example. The following sequence of division steps demonstrate how to find GCD(2205, 195) using Euclid's GCD algorithm:

Initially, let $a = 2205$, $b = 195$, then divide a by b to obtain the quotient and the remainder:

$2205 = 195 * 11 + 60$ --- (1)

In each of the subsequent steps, the dividend and the divisor are based on the divisor and remainder, respectively, of the previous step. Thus, the subsequent steps are as follows:

$195 = 60 * 3 + 15$ --------(2)

$60 = 15 * 4 + 0$ ---------- (3)

Thus, from (1), GCD(2205, 195) = GCD(195, 60) according to Euclid's theorem. In Step (2), GCD(195, 60) = GCD(60, 15); in Step (3), GCD(60, 15) = GCD(15, 0) = 15. Combining all these results yields GCD(2205, 195) = 15, which is the last divisor. Also, we note that the above division process will always terminate because the remainder of each step is strictly smaller than its divisor (that is how division works in arithmetic), which means smaller than the previous remainder. Further, it can be shown that the number of division steps in computing GCD(a, b) is $\lfloor 2 \lg M \rfloor + 1$, where $M = \max(a, b)$. (For example, $M = \max(2205, 195) = 2205$ in the above example, and $\lfloor 2 \lg M \rfloor + 1 = 22 + 1 = 23$.) This expression says the time complexity of Euclid's GCD algorithm is O(lg max(a, b)), which is a polynomial-time algorithm in terms of the size of the two input integers (i.e., lg a + lg b).

One more useful result out of the GCD process is that for any two positive integers a and b, the above algorithm can be extended to find integers t and u such that $at + bu$ = GCD(a, b). We will demonstrate this extended Euclid's algorithm using the above values of a and b.

Example. From (2) (i.e., the second to the last step in general), we can write

GCD(2205, 195) = 15 = 195 – 60 * 3 ---- (4)

Using (1), 60 = 2205 – 195*11, which can substitute into (4) replacing 60:

GCD(2205, 195) = 195 – (2205 – 195*11) * 3
= 2205 * (–3) + 195 * (1+33)
= 2205 * (–3) + 195 * (34)

In general, the process starts with the second-to-last equation writing the GCD as a "linear combination" of the equation's dividend and divisor. Then, use the previous equation to solve for its remainder, and substitute this into the current result, giving a linear combination of the dividend and the divisor of the new equation. Repeating this process will yield a linear combination of the first equation's dividend and divisor, i.e., of the form $at + bu$, that is equal to the GCD(a, b).

We now show how to solve the system of equations of the homework question. First, find the GCD(5, 3) and write it as a linear combination of 5 and 3, using extended Euclid's algorithm.

5 = 3 * 1 + 2
3 = 2 * 1 + 1
2 = 1* 2 + 0

Thus, GCD(5, 3) = 1 = 3 – 2*1 = 3 – (5 – 3*1)*1 = 5*(–1) + 3*2. Therefore, a solution to the first two equations x = 2 (mod 3) and x = 3 (mod 5), is x = 2*5*(–1) + 3*3*2 = 18 – 10 = 8 (mod 3*5). We then solve the equations: x = 8 (mod 15) and x = 4 (mod 7). Applying extended Euclid's algorithm yields:

15 = 7*2 + 1
7 = 1*7 + 0

Thus, GCD(15, 7) = 1 = 15*1 + 7*(–2). Therefore, a solution (mod 15*7) is

x = 8*7*(–2) + 4*15 = –112 + 60 = –52 = 53 (mod 105)

Note that this process of solving a system of 3 equations is a polynomial time algorithm for arbitrary moduli m, n, and p that are pairwise co-prime, since the complexity is

O(lg max(m, n) + lg max(m, n, p)) = O(lg max(m, n, p)).