COSC413 Examination on Advanced Algorithms

14 October 2002
Lecturer: Prof. T. Takaoka
Open book, calculators are allowed

**NP-complete problems**

(1) transform the following satisfiability problem into the corresponding clique problem, and discuss all corresponding solutions.

\[ F = (x_1 + x_2)(x_1 + x_2' + x_3)(x_1' + x_2') \]

Note. \(x'\) is the negation of \(x\), multiplication is ‘and’, and addition is ‘or’.

(2) Transform the following 2-vertex cover problem into the corresponding clique edge set problem and discuss corresponding solutions.

---

**Number theory**

(3) Let a difference equation be defined by

\[
x(0) = 0, \quad x(1) = 1 \\
x(n+2) = x(n+1) + 2x(n) \quad (n=0, 1, ...)
\]

(3.1) Obtain the solution for this equation.

(3.2) Transform the equation into a vector-matrix form as follows:

\[
(x(1), x(0)) = (0, 1) \\
(x(n+1), x(n)) = (x(n), x(n-1)) \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = (x(n), x(n-1)) A
\]

Obtain the matrix \(A\). Then we have \((x(n+1), x(n)) = (x(1), x(0)) A^n\). Obtain \(x(8)\) by repeated use of the equation 7 times and the repeated squaring of \(A\) three times, and confirm that you have the same result.

Note. Any linear recurrence of the above type is computable in \(O(\log n)\) time.
(4) The Euclidean algorithm for greatest common divisors is given as follows:
For \( a > b > 0 \) such that \( a = bq + r \), let \( a = r(0), b = r(1), q = q(1) \) and \( r = r(2) \). We repeat division as follows:

\[
\begin{align*}
a &= b \cdot q(1) + r(2), & 0 \leq r(2) < b \\
b &= r(2) \cdot q(2) + r(3), & 0 \leq r(3) < r(2) \\
\vdots \\
r(i-1) &= r(i)q(i) + r(i+1), & 0 \leq r(i+1) < r(i) \\
\vdots \\
r(n) &= r(n)q(n) + r(n+1), & r(n+1) = 0
\end{align*}
\]

\[
gcd(a, b) = r(n)
\]

(4.1) Trace this algorithm with \( a = 98 \) and \( b = 63 \).

(4.2) Define sequences \( c \) and \( d \) by

\[
\begin{align*}
c(0) &= 0, & c(1) &= 1, & c(i) &= c(i-2) - q(i-1)c(i-1) \\
d(0) &= 1, & d(1) &= 0, & d(i) &= d(i-2) - q(i-1)d(i-1).
\end{align*}
\]

Then we have \( a \cdot d(i) + b \cdot c(i) = r(i) \) for \( i = 0, \ldots, n \). By tracing sequences \( c \) and \( d \), compute \( 9^{-1} \) mod 14 in the range of \( \{1, \ldots, 13\} \).

(5) Let \( m_1 \) and \( m_2 \) be positive integers mutually prime to each other, that is, \( \gcd(m_1, m_2) = 1 \). Let \( m_1' \) be the inverse of \( m_1 \) (mod \( m_2 \)) and \( m_2' \) be the inverse of \( m_2 \) (mod \( m_1 \)). For arbitrarily given two positive integers \( x_1 \) and \( x_2 \), let \( x \) be defined by

\[
x = x_1 + (x_2 - x_1)m_1m_2' \quad (*)
\]

(5.1) Prove that \( x \equiv x_1 \) (mod \( m_1 \)) and \( x \equiv x_2 \) (mod \( m_2 \)), and that such \( x \) is unique within (mod \( m_1m_2 \)). Thus we can express integer \( x \) by the pair of remainders \( (x_1, x_2) \).

(5.2) Let \( a \) and \( b \) be positive integers. Let

\[
\begin{align*}
a_1 &= a \mod m_1, & a_2 &= a \mod m_2 \\
b_1 &= b \mod m_1, & b_2 &= b \mod m_2
\end{align*}
\]

Then we have \( (a + b) \mod m_1 = a_1 + b_1 \), and \( (a + b) \mod m_2 = a_2 + b_2 \). We can use this property to add two large numbers by adding the remainders and using the formula (*)

Example. \( m_1 = 7, m_2 = 15, a = 20, \) and \( b = 34 \). Then we have \( 7' \) (mod \( 15 \)) = 13.

\[
\begin{align*}
20 \mod 7 &= 6, & 20 \mod 15 &= 5 \\
34 \mod 7 &= 6, & 34 \mod 15 &= 4
\end{align*}
\]
By adding remainders, we have pair (12, 9). Using the formula (*), we obtain

\[ a + b = 12 + (9 - 12)(7 \times 13) = 12 - 273 \equiv 54 \pmod{105} \]

A similar calculation is possible for other operations by changing addition to subtraction or multiplication. Following the above example, perform the multiplication of \( a=5 \) and \( b=17 \).

**Note.** If we use moduli \( m_1, m_2, \ldots, m_n \), we can deal with very large numbers, and this method can be an alternative to multi-precision arithmetic. Unfortunately division does not work well in this method. The above (5.1) in this general setting is called the Chinese remaindering theorem. The advantage of this method is that we need not worry about carry propagation. When we add, subtract, or multiply many numbers, we can perform those operations on small remainders independently, and apply the formula (*) only at the end.