Problem Solving: Homework 3.13

Name: Chen Shaoyuan Student ID: 161240004

November 28, 2017

1 [TC] Problem 31.1-12

```
DIVIDE(a, b, β)
b = b ≪ β
Let r be a β-bit integer
Let ⟨r_{β−1}, ···, r_0⟩ be the binary representation of r
i = β
while i > 0
    i = i − 1
    b = b ≫ 1
    if a ≥ b
        r_i = 1
        a = a − b
    else r_i = 0
return r, a
```

where $\ll$, $\gg$ means logical left and right shift, respectively. The returned r is the quotient, a is the remainder.

Each elementary operation (assignment, subtraction, comparison, logical shift) on β-bit integer(s) takes $\Theta(\beta)$ time, and there are $\Theta(\beta)$ operations in total, so this algorithm runs in $\Theta(\beta^2)$ time.

2 [TC] Problem 31.1-13

```
CONVERT(a)
Make the length of a a power of 2 by zero padding
l = a.length
p[0] = 2
i = 2
while 2^i < l
    p[log_2 i] = p[log_2 i − 1] × p[log_2 i − 1]
    i = i + 1
return CONVERT-REC(a)
```

```
CONVERT-REC(a)
l = a.length
if l == 1
    return a
let a_1, a_2 be the higher and lower l/2 bits
b_1 = CONVERT-REC(a_1)
b_2 = CONVERT-REC(a_2)
return b_1 × P[log_2 l − 1] + b_2
```

The precalculation of the decimal representations of 2^{2^k} takes $\Theta(M(\beta)\log\beta)$ time. Since $\Theta(n^2)$ brute-force multiplication algorithm is known, we may assume that $M(\beta) = O(\beta^2) = \Omega(\beta)$. The running time of the recursion satisfies the following recurrence

$$f(n) = 2f(n/2) + M(n)$$

since $kM(n/k) = O(M(n))$ for positive integer k, we may conclude that $f(n) = O(M(\beta)\log\beta)$. Hence the total running time if $\Theta(M(\beta)\log\beta)$.

3 [TC] Problem 31.2-4

```
EUCLID(a, b)
while b ≠ 0
    t = a mod b
    a = b
    b = t
return a
```

4 [TC] Problem 31.2-5

By Lamé's theorem, the call EUCLID(a,b) makes at most k recursive calls, where $b < F_{k+1}$. Since $F_{k+1} > \phi^{k-1}$ when $k > 1$, $b < \phi^{k-1}$ implies $b < F_{k+1}$, so we can take $k = 1 + \log_\phi b$ when $b > 1$. If $b = 1$, it only makes one recursive call. Hence EUCLID(a,b) makes at most $1 + \log_\phi b$ recursive calls.

The executions of EUCLID(a,b) and EUCLID$(a/\gcd(a,b), b/\gcd(a,b))$ are same except that the numbers in the former one is $\gcd(a,b)$ times larger than those in the latter one. Hence they make the same number of recursive calls. This improves the bound to $1 + \log_\phi(b/\gcd(a,b))$.

5 [TC] Problem 31.2-6

It returns $(1, (-1)^k F_{k-1}, (-1)^{k+1} F_k)$. We prove this by induction.

For the base case, $k = 1$, the algorithm returns $(1,1,0) = (1, (-1)^k F_{k-1}, (-1)^{k+1} F_k)$.

For the induction space, let $k' = k+1$, the algorithm returns

$$\begin{aligned}&(1,(-1)^{k+1}F_k,(-1)^kF_{k-1}-\lfloor F_{k+2}/F_{k+1}\rfloor(-1)^{k+1}F_k)\\=&(1,(-1)^{k'}F_{k'-1},(-1)^k(F_{k-1}+F_k))\\=&(1,(-1)^{k'}F_{k'-1},(-1)^{k'+1}F_{k'})\end{aligned}$$

By mathematical induction, the conclusion is correct.

6 [TC] Problem 31.2-9

$\gcd(n_1n_2,n_3n_4)=1$ implies n_i and n_j are relatively prime, where $i=1,2,\ j=3,4$. $\gcd(n_1n_3,n_2n_4)=1$ implies n_i and n_j are relatively prime, where $i=1,3,\ j=2,4$. So n_1,n_2,n_3,n_4 are pairwise relatively prime.

Let $p_i(1\le i\le\lceil\lg k\rceil)$ (q_i, resp.) denote the product of n_j, where the i-th bit of the binary representation of $j-1$ is 1 (0, resp.). Then $n_1,n_2,\cdots,n_k$ are pairwise relatively prime if and only if $p_i,q_i(1\le i\le\lceil\lg k\rceil)$ are relatively prime:

If: for every pair of numbers in $n_1,n_2,\cdots,n_k$, let n_i,n_j denote them. Since $i\neq j$, the binary representation of i and j must differ at at least one bit. Let a denote the index of the bit, then n_i is a term of p_a (or q_a, resp.), n_j is a term of q_a (or p_a, resp.). Since p_a and q_a are relatively prime, n_i and n_j are also relatively prime. Hence $n_1,n_2,\cdots,n_k$ are pairwise relatively prime.

Only if: suppose to the contrary that p_i and q_i are not relatively prime. Let r be any prime factor of $\gcd(p_i,q_i)$. Then, at lest one term in p_i (and q_i) is a multiple of r. Let n_x be the term in p_i, n_y be the term in q_i. Since the the binary representation of x and y differ at the i-th bit, we have $x\neq y$, but n_x and n_y are multiples of r, which means they are not relatively prime, leading to contradiction. So $p_i,q_i(1\le i\le\lceil\lg k\rceil)$ are relatively prime.

7 [TC] Problem 31.3-5

If $f_a(x)=f_a(y)$, i.e. $ax\equiv ay \pmod n$. Multiplying a^{-1} on both sides gives $x\equiv y \pmod n$, i.e. $x=y$, so $f_a(x)$ is one-to-one. For every $y\in\mathbb{Z}_n^*$, $f(a^{-1}y)=y$, so the function is onto. Therefore, $f_a(x)$ is a bijection from $y\in\mathbb{Z}_n^*$ to $y\in\mathbb{Z}_n^*$, so it is a permutation of $\mathbb{Z}_n^*$.

8 [TC] Problem 31.4-2

Since $\gcd(a,n)=1$, the multiplicative inverse of a modulo n exists. Multiplying a^{-1} on both sides of $ax\equiv ay \pmod n$ gives $x\equiv y \pmod n$.

$2\times3\equiv2\times8 \pmod{10}$, while $3\not\equiv8 \pmod{10}$, because $\gcd(2,10)>1$.

9 [TC] Problem 31.4-3

This will work. Let $x_0'=x'(b/d) \bmod (n/d)$, then $x_0'=x_0 \bmod (n/d)=x_0+k(n/d)$, which is also a solution. By Theorem 31.24, this will work.

10 [TC] Problem 31.5-2

The equation system is

$$\begin{aligned}x&\equiv1 \pmod 9\\x&\equiv2 \pmod 8\\x&\equiv3 \pmod 7\end{aligned}$$

then we have $a_1=1,n_1=9,m_1=56,c_1=m_1(m_1^{-1} \bmod n_1)=56\times5=280,\ a_2=2,n_2=8,m_2=63,c_2=m_2(m_2^{-1} \bmod n_2)=63\times7=441,\ a_3=3,n_3=7,m_3=72,c_3=m_3(m_3^{-1} \bmod n_3)=72\times4=288$. By the Chinese remainder theorem, we have

$$\begin{aligned}x&\equiv a_1c_1+a_2c_2+a_3c_3 &&\pmod{504}\\x&\equiv1\times280+2\times441+3\times288 &&\pmod{504}\\x&\equiv2026 &&\pmod{504}\\x&\equiv10 &&\pmod{504}\end{aligned}$$

So the solutions are $x=10+504k$, where k is an integer.

11 [TC] Problem 31.5-3

Since $\gcd(a,n)=1$, we have $\gcd(a,n_i)=1$, so the multiplicative inverse of a modulo n_i exists. Because $a_i=a \bmod n_i$, we have $a_i^{-1}\equiv a^{-1} \pmod{n_i}$. Hence we have the correspondence

$$(a^{-1} \bmod n)\leftrightarrow((a_1^{-1} \bmod n_1),\cdots,(a_k^{-1} \bmod n_k)).$$

12 [TC] Problem 31.6-2

MODULAR-EXPONENTIATION(a,b,n)

```
c = a
d = 1
let ⟨b_k, b_{k-1}, ··· b_0⟩ be the binary representation of b
for i = 0 to k
    if b_i == 1
        d = (d · c) mod n
    c = (c · c) mod n
return d
```

13 [TC] Problem 31.6-2

By Euler's theorem, $a^{\phi(n)-1} \cdot a \equiv 1 \pmod{n}$, so $a^{\phi(n)-1} \bmod n$ is the multiplicative inverse of a modulo n, i.e. $a^{-1} \bmod n = a^{\phi(n)-1} \bmod n$, which can be calculated by MODULAR-EXPONENTIATION.