Problem Solving: Homework 3.14

Name: Chen Shaoyuan Student ID: 161240004

December 10, 2017

1 [TJ] Problem 7.3

The ciphertext is seperated into 4 ten-letter groups, with an additional four-letter tail. Since it is a monoalphabetic code, we try frequency analysis. The frequencies of all letters in ciphertext is shown below.

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M
#	2	2	2	1	1	9	1	1	1	1	1	1	1
Letter	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
#	2	1	1	4	1	2	1	1	0	4	1	1	1

Note that the letter 'F' in ciphertext appears 9 times, or 20.5%, which is much higher than the most common letter, 'E', in English, which only occurs about 12% of the time. Also, the average English word length is about 5, so we guess that 'F' is just used to separate words. However, only the last one of consecutive 'F's should be considered as word separator. Hence, the ciphertext can be separated as

```
NBQ FRSMXZ YAWJU HWF ESKGQC WDQA NBQ TILO CWP
```

Note that the three-letter word 'NBQ' appears twice in this text. We know that the word 'the' is the most common word in English, so it is very likely that 'NBQ' means 'THE'. This is consistent with the fact that 'Q' (which means 'E') appears 4 times out of 44 in ciphertext. So 'N' means 'T', 'B' means 'H', 'Q' means 'E'. Also, the separator is often the less commonly used letter, 'X'. Now the code reads

```
the FRSMXZ YAWJU HWx ESKGeC WDeA the TILO CWP
```

Consider the letter 'W' in ciphertext, which appears 4 times. The second common letter in English is 'T', which, however, is encoded as 'N' here. The third common letter is 'A'. If so, the word 'WDeA' has the patter 'a?e?', where $? \neq T, H, E, A$ and the two question marks mean different letters. The only commonly used words satisfying these conditions are 'ages' or 'aged', which does not make sense here. So, we try the fourth common letter, 'O'. Then the word has pattern 'o?e?', which might be 'over', 'open', 'oven' or 'obey'. The third one, 'oven', can be ruled out, for a noun never comes before 'the'. Now the code reads

```
the FRSMXZ YAoJU Hox ESKGeC oDeA the TILO CoP
```

The fourth one, 'obey', can also be ruled out, for no word has the pattern '?yo??' (YyoJU) if 'A' means 'y'. Both words 'over', 'open' are possible, however, the first one is more likely, for the only correspondence of 'YAoJU' is 'snowy' if 'oDeA' is 'open', while there are many choices for 'YorJU' if 'oDeA' is 'over'. Now the code reads

```
the FRSMXZ YroJU Hox ESKGeC over the TILO CoP
```

The remaining part is easy, and the plaintext is

```
the quick brown fox jumped over the lazy dog
```

2 [TJ] Problem 7.7

(a) $y = x^E \bmod n = 31^{629} \bmod 3551 = 2791$

(b) $y = x^E \bmod n = 23^{47} \bmod 2257 = 769$

(c)

$$\begin{aligned}
y_1 &= x_1^E \bmod n = 14^{13251} \bmod 120979 = 112135\\
y_2 &= x_2^E \bmod n = 23^{13251} \bmod 120979 = 25032\\
y_3 &= x_3^E \bmod n = 71^{13251} \bmod 120979 = 442\\
y &= (y_1, y_2, y_3) = (112135, 25032, 442)
\end{aligned}$$

(d)

$$\begin{aligned}
y_1 &= x_1^E \bmod n = 23^{781} \bmod 45629 = 4438\\
y_2 &= x_2^E \bmod n = 15^{781} \bmod 45629 = 16332\\
y_3 &= x_3^E \bmod n = 61^{781} \bmod 45629 = 31594\\
y &= (y_1, y_2, y_3) = (4438, 16332, 31594)
\end{aligned}$$

3 [TJ] Problem 7.9

(a) $x = y^D \bmod n = 2791^{1997} \bmod 3551 = 31$

(b) $x = y^D \bmod n = 34^{81} \bmod 5893 = 2014$

(c) $x = y^D \bmod n = 112135^{27331} \bmod 120979 = 14$

(d) $x = y^D \bmod n = 129381^{671} \bmod 79403 = 21712$

4 [TJ] Problem 7.12

Note that the equation $X^E - X \equiv 0 \pmod{n}$ is equivalent to the equation system

$$\begin{cases} X^E - X \equiv 0 \pmod{p} & (1)\\ X^E - X \equiv 0 \pmod{q} & (2)\end{cases}$$

because $n = pq$ where p and q are distinct primes. Also, every pair of solutions to equation (1) and (2) corresponds to a solution to the original equation, by the Chinese remainder theorem. So, the number of solutions to the original equation is the product of the numbers of solutions to equation (1) and (2).

Without loss of generality, let's consider equation (1). Since p is a prime, it is either the case that $X = 0$ or $X^{E-1} \equiv 1 \pmod{p}$. Let $X = g_p^t \neq 0$ where g is a primitive root modulo p. By discrete logarithm theorem, the second case is equivalent to

$$t(E-1) \equiv 0 \pmod{p-1}.$$

this equation has exactly $\gcd(E-1, p-1)$ solutions. Hence, the equation (1) has $1+\gcd(E-1, p-1)$ solutions, and they are 0 and $g_p^{\frac{k(p-1)}{\gcd(E-1,p-1)}}$, where k is an integer. Likewise the equation (2) has $1+\gcd(E-1, q-1)$ solutions, and they are 0 and $g_q^{\frac{k(q-1)}{\gcd(E-1,q-1)}}$. Let X_1, X_2 denote one pair of solutions to equations (1) and (2), then $X = X_1 q q_p^{-1} + X_2 p p_q^{-1}$ is a solution to the original equation, by the Chinese remainder theorem. There are $(1+\gcd(E-1, p-1))(1+\gcd(E-1, q-1))$ such solutions in all.

This might be a potential problem in the RSA cryptosystem. If $(1+\gcd(E-1, p-1))(1+\gcd(E-1, q-1))$ is large, then the probability that the plaintext contains a fixed point is high. Since RSA modulus is large (usually up to thousands digits), the attacker may find such fixed point (which remains unchanged after encryption) very easily. However, we can avoid such problem by carefully choosing E such that $\gcd(E-1, p-1)$ and $\gcd(E-1, q-1)$ are small.

5 [TC] Problem 31.7-1

$$d = e^{-1} \bmod (p-1)(q-1) = 3^{-1} \bmod 280 = 187$$
$$N = M^3 \bmod n = 100^3 \bmod 319 = 254$$

6 [TC] Problem 31.7-2

Since $ed \equiv 1 \pmod{(p-1)(q-1)}$, we have $ed = k(p-1)(q-1)-1$. Since $e = 3$ and $0 < d < \phi(n)$, we have $k = 1$ or $k = 2$. For $k = 1$, let $m = n+1-(1+ed)/k = p+q$, which can be calculated in polynomial time. Now, we have $m = p+q$, $n = pq$, and we want to solve for p and q. We can rewrite these equations as

$$f(p) = p^2 - mp - n = 0$$

Note that $f(p)$ is a monotonic function on $[0, m/2]$. This enables us to apply binary search to find the zero point of $f(p)$, and the running time is $O(T(\log n)\log m)$, where $T(\alpha)$ is the time of computing $f(x)$ for given α-bit integer x, or equivalently, the running time of multiplying two α-bit integers, which is polynomial. If no solution is found, let $k = 2$ and try again. The total running time is polynomial in the number of bits in n.

7 [TC] Problem 31.2

a. The “paper and pencil” algorithm for division can be decomposed to the following steps:

Step 1 Left shift b until the length of b is greater than or equal to a.

Step 2 Goto Step 5.

Step 3 Compare b and a. If b is larger than or equal to a, subtract a from b and append bit ‘1’ to q as the least significant bit; otherwise, append bit ‘0’ to q.

Step 4 Right shift a by 1 bit.

Step 5 If the length of a is greater than its original length, goto Step 3.

Step 6 Let r be b.

Step 1, 2, 4, 5 can be done in $O(1)$ time. Step 3 can be done in $O(\lg b)$ bit operations, and this step will be executed for $O(\lg q)$ times. Step 6 can be done in $O(\lg b)$ operations. So this method requires $O((1+\lg q)\lg b)$ bit operations.

b. The reduction is to calculate $a \bmod b$. By the conclusion above, such calculation can be done in at most $c(1+\log\lfloor a/b \rfloor)(\log b)$ bit operations, where

$$\begin{aligned} c(\mu(a,b) - \mu(b, a \bmod b)) &= c((1+\lg a)(1+\lg b) - (1+\lg b)(1+\lg(a \bmod b)) \\ &\geq c((1+\lg a)(1+\lg b) - (1+\lg b)(1+\lg b) \\ &\geq c(1+\log\lfloor a/b \rfloor)(\log b) \end{aligned}$$

So it can be done in at most $c(\mu(a,b) - \mu(b, a \bmod b))$ bit operations.

c. The EUCLID requires at most

$$c\mu(a,b) - c\mu(b, a \bmod b) + c\mu(b, a \bmod b) - c\mu(a \bmod b, b \bmod (a \bmod b)) + \cdots < c\mu(a,b) = O(\mu(a,b))$$

bit operations. Since $O(\mu(a,b)) = O(\lg a, \lg b)$ for $a, b > 1$, it requires $O(\beta^2)$ bit operations when applied to two β-bits inputs

8 [TC] Problem 31.3

a. The running time satisfies the following recurrence

$$f(n) = f(n-1) + f(n-2) + O(1)$$

Hence, $f(n) = \Omega(F_n)$. Also, it is easy to verify that $f(n) = O(3^n)$. We know that F_n grows exponentially in n, so the running time is exponential in n.

b. We compute F_i from $i = 2$ to n. For every calculated F_i, we store its value in the memory. Hence, we do not have to recomputer F_{n-1} and F_{n-2}. The running time is therefore $O(n)$.

c. Note that

$$\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_{n-1} \\ F_n \end{pmatrix}$$

So we have

$$\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

We can compute $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n$ by repeated squaring in $O(\log n)$ time. Hence the total running time is $O(\log n)$.

d. For the first method, the recurrence should be changed to

$$f_1(n) = f_1(n-1) + f_1(n-2) + O(\log F_n) = f_1(n-1) + f_1(n-2) + O(n)$$

and we can still verify easily that $f_1(n) = O(3^n)$, so the running time of the first method is still exponential in n.

For the second method, the running time is

$$f_2(n) = O(\sum_{i=1}^{n} \log(F_i)) = O(\sum_{i=1}^{n} i) = O(n^2)$$

For the third method, the running time is

$$f_3(n) = O(\sum_{i=1}^{\log_2 n} \log^2(F_{2^i})) = O(\sum_{i=1}^{\log_2 n} 4^i) = O(n^2)$$