Data Structures and Algorithms Homework # 1

Tim Liou (b03902028)

March 24, 2015

Question 1.1

(1) Professor Donald Knuth has been called "father of analysis of algorithms". He is also the creator of TeX computer typesetting system. He found the galley proof of his great book, The Art of Computer Programming, awful. Meanwhile, he saw for the first time the output of high-quality digital typesetting system, and decided to design his own typesetting system, TeX, to solve this problem. He used to pay $2.56 to those who find mistakes in his books because "256 pennies is one hexadecimal dollar".

(2) Proof by induction.

Base Case

If string array's length is 0, i will not go into the while loop because i initialize to 0 and $str[i]$(now is $str[0]$) is already the end of the string. This algorithm then return $i = 0$, which is the length of the string.

Induction Hypothesis

Claim: At the end of while loop, string array whose length is k return $i = k$.

Induction Step

Assume Claim is correct at the case that string's length is M. We denote this string A. When string's length is $M + 1$ (we denote this string B), i comes to M which is A's end. However, $str[M]$ isn't the end of B. Therefore, i becomes to $M + 1$, and now $str[M + 1]$ is the end of B. Finally, this algorithm return $i = M + 1$ when the length of string is $M + 1$.

Question 1.2

(1) Without loss of generality, let $a \geq b$. We know that any number bigger than n cannot divide n without a remainder. Therefore, by definition, $1 \leq gcd(a, b) \leq min(a, b)$. Since we choose i from $min(a, b)$ to 1, the first i satisfing the if statement will be the largest one which can divide a and b without remainder. This shows that i is $gcd(a, b)$.

(2) This value is 1.

When $a = b \times k, k > 1, k \in N$, we have the best case.

Question 1.3

(1) $a = l \times k \Rightarrow k = \frac{a}{l}$
$b = l \times j \Rightarrow j = \frac{b}{l}$

By definition,

$k = gcd(k, j) \times r, j = gcd(k, j) \times w,\ which\ gcd(r, w) = 1.$
$a = l \times gcd(\frac{a}{l}, \frac{b}{l}) \times r$
$b = l \times gcd(\frac{a}{l}, \frac{b}{l}) \times w$

Claim: $gcd(a, b) = l \times gcd(\frac{a}{l}, \frac{b}{l})$

1. We can easily observe that $a\ mod\ (l \times gcd(\frac{a}{l}, \frac{b}{l})) = 0$ and $b\ mod\ (l \times gcd(\frac{a}{l}, \frac{b}{l})) = 0$
2. Suppose $gcd(a, b)$ is greater than $gcd(a, b) = l \times gcd(\frac{a}{l}, \frac{b}{l})$ and denote it $(l \times gcd(\frac{a}{l}, \frac{b}{l}) \times c)$ which $c \geq 2$.
$\Rightarrow \begin{cases} a = l \times gcd(\frac{a}{l}, \frac{b}{l}) \times r = l \times gcd(\frac{a}{l}, \frac{b}{l}) \times r \times c \times A \\ b = l \times gcd(\frac{a}{l}, \frac{b}{l}) \times w = l \times gcd(\frac{a}{l}, \frac{b}{l}) \times r \times c \times B \end{cases}$
$\Rightarrow \begin{cases} r = c \times A \\ w = c \times B \end{cases}$
$\Rightarrow gcd(r, w) \geq c \geq 2$

   This is a contradiction. Therefore,

$$gcd(a, b) = l \times gcd(\frac{a}{l}, \frac{b}{l})$$

(2) This value is 1, If the pseudo code enter that line, this means that the number from $min(a, b)$ to 2 couldn't divide a, b without remainder. By 1.2(1), we know that $1 \leq gcd(a, b) \leq min(a, b)$. Therefore 1 is the only value can return.

(3) This value must be 2. If we choose 1, the if statement will always be true, this means that we will always return $1 \times gcd(a, b)$ and never compute the answer. If we choose the number greater than 2, we will miss if a $mod\ 2 = 0$ and b $mod\ 2 = 0$ because for loop start from the number greater than 2.

Question 1.4

(1)

```
n = 14     m = 56     ans =  1
-----------------------------
n =  7     m = 21     ans =  2

n =  7     m = 14     ans =  2

n =  7     m =  0     ans =  2
-----------------------------
n =  7     m =  0     ans = 14
```

(2) Claim: m will finally become 0.

1. At the end of each iteration, m is getting smaller and smaller.
   Case 1: if $n > m$, n will swap with m. Therefore m becomes smaller.
   Case 2: if $n = m$, m will become 0. m becomes smaller and is 0.
   Case 3: if $n < m$, m will become $(m - n)$. m becomes smaller.
2. We can easily observe that n will never become to 0.
3. If we enter the line $m \leftarrow (m - n)$, $m \geq n$, $m - n \geq 0$, this shows that m will getting smaller but still greater than 0 if $m \neq n$. We notice that m will not have a chance to become 0 before enter this statement, this means that $m \geq n > 0$.
   Case 1: if $n = m \neq 1$, m will become 0.
   Case 2: Since m is getting smaller, m will finally become 1. $m \geq n > 0$, n will become 1, too. $m = m - n = 0$.

This prove that m will finally become 0, that is, GCD-By-Binary satisfies the finiteness property.

(3) By definition,
$a = gcd(a, b) \times k, b = gcd(a, b) \times r, gcd(k, r) = 1, k > r \geq 1$
$\Rightarrow a - b = gcd(a, b) \times (k - r)$

Case 1: r = 1
$a - b = gcd(a, b) \times (k - 1)$
$gcd(a - b, b) = gcd(a, b) \times gcd(k - 1, 1) = gcd(a, b)$, by 1.3(1).

Case 2: r > 1
$a - b = gcd(a, b) \times (k - r)$
$gcd(a - b, b) = gcd(a, b) \times gcd(k - r, r)$
if $gcd(k - r, r) = c \geq 2 . k - r = c \times A, r = c \times B$
$\Rightarrow k = c \times (A + B), r = c \times B$
$\Rightarrow gcd(k, r) \geq c \geq 2$ This is a contradiction. Therefore, $gcd(k - r, r) = 1$
$\Rightarrow gcd(a - b, b) = gcd(a, b)$.

(4) $4 \times \lceil \log_2 a \rceil + 1$

The worst case is that a take 2 iterations to become half of itself, and so does b. This implies that it take at most $4 \times \lceil \log_2 a \rceil$ iterations. If they both become 1, it take one more iteration. Therefore, the tightest upper bound I can think of is $4 \times \lceil \log_2 a \rceil + 1$.

Question 1.5

(1)

```
m = 56     n = 14     tmp =  *
------------------------------

------------------------------
m = 56     n = 14     tmp =  *
```

(2) Claim1: $3a \ mod \ 3b = 0$ if $a \ mod \ b = 0$
$a \ mod \ b = 0 \Rightarrow a = b \times r, r > 0$
$\Rightarrow 3a = 3b \times r$
$\Rightarrow 3a \ mod \ 3b = 0$

Claim2: $3a \ mod \ 3b = 3k \ if \ a \ mod \ b = k$
$a \ mod \ b = k \Rightarrow a = b \times r + k, 0 < k < b$
$\Rightarrow 3a = 3b \times r + 3k, 0 < 3k < 3b$
$\Rightarrow 3a \ mod \ 3b = 3k$

In while loop, we check $m \ mod \ n$ is 0 or not
Case 1: $m \ mod \ n = 0$, by Claim1, we know that $3m \ mod \ 3n = 0$, which will be out of while loop.
Case 2: $m \ mod \ n = k \neq 0$, m will become n, and n will become k. By Claim2, we know that $3m \ mod \ 3n = 3k$, then $3m$ will become $3n$, and $3n$ will become $3k$.

This imply that gcd-by-euclid$(3a, 3b)$ and gcd-by-euclid(a, b) take the same iterations.

(3) Proof by induction.

Base Case

If $T = 0$, this means that $a \ mod \ b = 0$, and we know that $a > b$, this implies that $a > b \geq 1$. We have $a > b \geq 1 = F_2 = F_1$

Induction Hypothesis

Claim: If applying GCD-By-Euclid(a, b) takes T iterations, we can show that $a \geq F_{T+2}$ and $b \geq F_{T+1}$

Induction Step

Assume Claim is correct when $T = k$, this means that $a_k \geq F_{T+2}$ and $b_k \geq F_{T+1}$
When $T = k + 1$, We can easily find that

$$a_{k+1} = b_{k+1} \times l + (a_{k+1} \ mod \ b_{k+1}) \geq b_{k+1} + (a_{k+1} \ mod \ b_{k+1})$$

Notice that $gcd(a_{k+1}, b_{k+1}) = gcd(b_{k+1}, a_{k+1}\ mod\ b_{k+1})$, and we now remain k iterations, therefore, we have

$$b_{k+1} \geq F_{k+2} = F_{(k+1)+1}$$

$$(a_{k+1}\ mod\ b_{k+1}) \geq F_{k+1}$$

Then, we also get

$$a_{k+1} \geq b_{k+1} + (a_{k+1}\ mod\ b_{k+1}) \geq F_{k+2} + F_{k+1} = F_{k+3} = F_{(k+1)+2}$$

We conclude that the Claim is correct.

Question 1.6

(1) code done.

(2) Result:

```
Average-GCD-By-Reverse-Search : 11254
Average-GCD-By-Filter         : 6491
Average-GCD-By-Filter-Faster  : 6491
Average-GCD-By-Binary         : 18
Average-GCD-By-Euclid         : 8
```

GCD-By-Euclid and GCD-By-Binary take fewer iterations than the others. This means they are much faster. On average, GCD-By-Filter-Faster is not much faster than GCD-By-Filter in this case because $11260 = 2^2 \times 5 \times 563$, which GCD-By-Filter-Faster cannot save lots of iterations.