# EECS 340: Assignment 1

Shaochen (Henry) ZHONG, sxz517
Zhitao (Robert) CHEN, zxc325

Due on 01/27/2020, submitted on 01/26/2020
EECS 340, Dr. Koyuturk

## 1 Problem 1

### 1.1 (a)

Without loss of generality, assume $x > y$ for $x, y \in \mathbb{Z}^+$. The loop invariant is:

$$Euclidean(x, y) = Euclidean(x - y, y) \tag{1}$$

### 1.2 (b)

Without loss of generality, assume $x > y$ for $x, y \in \mathbb{Z}^+$.

Let $d$ for $d \in \mathbb{Z}^+$ being the greatest common divisor of $x$ and $y$, a.k.a $d = gcd(x, y)$. As $d$ being a divisor of both $x$ and $y$, we may therefore have $x = kd$ and $y = jd$ for $k, j \in \mathbb{Z}^+$. Then we may infer:

$$x - y = dk - dj = d(k - j) \tag{2}$$

From *Equation 2* and the known fact that $y = jd$, we may say that $d$ is also a common divisor of $x - y$ and $y$. By the defination of $gcd$, this means the upper bond of $d$ cannot be greater than $gcd(x - y, y)$. Thus we may conclude:

$$\begin{aligned} gcd(x, y) = d &\leq gcd(x - y, y) \\ \Rightarrow gcd(x, y) &\leq gcd(x - y, y) \end{aligned} \tag{3}$$

Now similarly, Let $e$ for $e \in \mathbb{Z}^+$ being the greatest common divisor of $x - y$ and $y$. We may therefore have $x - y = le$ and $y = me$ for $l, m \in \mathbb{Z}^+$. Then we may infer:

$$x = (x - y) + y = le + me = e(l + m) \tag{4}$$

From *Equation 4* and the known fact that $y = me$, we may say that $e$ is also a common divisor of $x$ and $y$. By the defination of $gcd$, this means the upper bond of $e$ cannot be greater than $gcd(x, y)$. Thus we may conclude:

$$\begin{aligned} gcd(x-y, y) &= e \leq gcd(x, y) \\ \Rightarrow gcd(x-y, y) &\leq gcd(x, y) \end{aligned} \tag{5}$$

By observing *Equation 3* and *Equation 5*, we may have a constraint of $gcd(x, y) = gcd(x-y, y)$. As the given method $Euclidean()$ is a $gcd$ finder, we may promote such constraint to $Euclidean(x, y) = Euclidean(x-y, y)$ due to the equivalency of $Euclidean(a, b)$ and $gcd(a, b)$.

### 1.3 (c)

The `while` loop always terminates as the condition $x = y$ will eventually be reached. Due to the fact that we have $x$ and $y$ for $x, y \in \mathbb{Z}^+$; without loss of generality, we assume $x > y$, therefore we must have $x' = x - y$ for $x' \in \mathbb{Z}^+$.

As we have only a finte amount of $x'$, $x''$, $x'''$... to decrease for $x', x''$, and $x''' \in \mathbb{Z}^+$, the decremental calculation of $x_{k+1} = x_k - y_k$[1] will eventually reaches a condition where $x = y$ due to the "well ordering" nature of the natural numbers. Thus, the `while` loop always terminates.

### 1.4 (d)

In **Section 1.2**, we have proven that $gcd(x, y) = gcd(x-y, y) \Rightarrow Euclidean(x, y) = Euclidean(x-y, y)$ assuming $x > y$ for $x, y \in \mathbb{Z}^+$. Following the principle of induction, we can generalize it as:

$$\begin{gathered} Euclidean(x_k, y_k) = Euclidean(x_{k+1}, y_{k+1}) \\ \text{for } x', y' \in \mathbb{Z}^+ \text{ while} \\ x_{k+1} = x_k - y_k,\ y_{k+1} = y_k \quad \text{if } x_k > y_k \\ y_{k+1} = y_k - x_k,\ x_{k+1} = x_k \quad \text{if } y_k > x_k \end{gathered} \tag{6}$$

Where $Euclidean(x_{k+1}, y_{k+1})$ is the greatest common divisor of $(x, y)$.

As we haven proven in **Section 1.2**, that the `while` loop within the $Euclidean()$ method

[1] for $k$ being the numbers of iteration went through in the `while` loop.

must terminate. Combined such finding with *Equation 6*, we must also have a:

$$\begin{aligned} Euclidean(a, b) = Euclidean(a_k, b_k) = Euclidean(a_{k+1}, b_{k+1}) = ... \\ ... = Euclidean(a_{k+n}, b_{k+n}) = Euclidean(a_{k+n}, a_{k+n}) \end{aligned} \tag{7}$$

As we know by calculation that $Euclidean(a_{k+n}, a_{k+n}) = a_{k+n}$, $a_{k+n}$, this will be the greatest common divisor of $Euclidean(a, b)$.

# 2 Problem 2

## 2.1 (a)

**Algorithm 1** TwoSum(A, B, n, x) with two pointers

**procedure**
    $j \leftarrow n - 1$
    $i \leftarrow 0$
    **while** $i < n$ **and** $j \geq 0$ **do**
        **if** $A[i] + B[j] < x$ **then**
            $i \leftarrow i + 1$
        **else if** $A[i] + B[j] > x$ **then**
            $j \leftarrow j - 1$
        **else**
            **return** $i, j$
    **return** $False$

## 2.2 (b)

The following is graphical demenstration of *TwoSum(A, B, 5, 12)*:

| A | | | | | | B | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0($i$) | 1 | 2 | 3 | 4 | | 0 | 1 | 2 | 3 | 4($j$) |
| 1 | 3 | 5 | 7 | 9 | | 2 | 4 | 6 | 7 | 10 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1($i$) | 2 | 3 | 4 | | 0 | 1 | 2 | 3 | 4($j$) |
| 1 | 3 | 5 | 7 | 9 | | 2 | 4 | 6 | 7 | 10 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1($i$) | 2 | 3 | 4 | | 0 | 1 | 2 | 3($j$) | 4 |
| 1 | 3 | 5 | 7 | 9 | | 2 | 4 | 6 | 7 | 10 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2($i$) | 3 | 4 | | 0 | 1 | 2 | 3($j$) | 4 |
| 1 | 3 | 5 | 7 | 9 | | 2 | 4 | 6 | 7 | 10 |

Thus we have $A[i] + B[j] = 12$ for $(i, j)$ being $(2, 3)$.

## 2.3 (c)

If $A[i] + B[j] \neq x$, then it is impossible to have any combination $(a, b)$ where $a \in \{A[0], A[1], ..., A[i-1], A[i]\}$ and $b \in \{B[j], B[j+1], B[j+2], ..., B[n-1]\}$ to be $a + b = x$ for all **iterarted** $i, j$ within the `while` loop.

Assume we get $A[i] + B[j] = k$ within an interation of the `while` loop, we either have $k < x$ or $k > x$ (assume $A[i] + B[j] \neq x$ because otherwise the loop will be terminated and the problem will be solved). Due to the sorted nature of array $A, B$, we must have:

$$\underbrace{A[0] \leq A[1] \leq ... \leq A[i-1]}_{a} \leq A[i] \quad \Rightarrow a \leq A[i] \tag{8}$$

$$\underbrace{B[n-1] \geq ... \geq B[j+2] \geq B[j+1]}_{b} \geq B[j] \quad \Rightarrow b \geq B[j] \tag{9}$$

Thus, we may infer the followings from the above two *Equations*:

$$\text{if } k < x: \ a + B[j] \leq A[i] + B[j] \quad \Rightarrow a + B[j] < x \tag{10}$$

$$\text{if } k > x: \ A[i] + b \geq A[i] + B[j] \quad \Rightarrow A[i] + b > x \tag{11}$$

It is a bit unintuitive about the case of $a' + b'$ for $a' \in \{A[0], A[1], ..., A[i-1]\}$ and

$b' \in \{B[j+1], B[j+2], ..., B[n-1]\}$. But as the value of index $i$ ($j$) travels in an incremental (decremental) fashion in the bond of $\mathbb{Z}^+ \in [0, n)$; the index of $a'$ must be smaller than $i$, and the index of $b'$ must be greater than $j$. Thus, if there is any $a' + b' = x$, such loop would have been terminated immediately and we should not be able to reach to indexes $i, j$ in the `while` loop. Therefore, as we are now at the $i, j$ iteration of the `while` loop, this suggests there is no $(a', b')$ combination where $a' + b' = x$.

This together with *Equation 10, 11*, we have proven the loop invariant as there will be no combination of $(a, b)$ where $a \in \{A[0], A[1], ..., A[i-1], A[i]\}$ and $b \in \{B[j], B[j+1], B[j+2], ..., B[n-1]\}$ to be $a + b = x$ for all **iterarted** $i, j$ within the `while` loop.

### 2.4 (d)

The worst case of this algorithm is when it returns `False`. In such case we wil have $i$ and $j$ travel the entire bond of $\mathbb{Z}^+ \in [0, n)$. The loop will execute $2n$ times, thus the run time is $O(n)$.

## 3 Problem 3

### 3.1 Describe the process using a loop: Pseudocode

**Algorithm 2** Kepler442b(m, n)

**procedure**
    $l \leftarrow m$
    $p \leftarrow n$
    **while** $(l + p) > 1$ **do**
        Let two individuals fight each other.
        **if** Both individuals are Lydians **then**
            $l \leftarrow l - 1$
        **else if** Both individuals are Pisidians **then**
            $p \leftarrow p - 2$
            $l \leftarrow l + 1$
        **else**
            $l \leftarrow l - 1$
    **return** $l, p$

## 3.2 Define loop invariant and termination

At the start of the $k^{th}$ iteration of the `while` loop, we must have

$$l + p = m + n - (k - 1) \tag{12}$$

$$p \bmod 2 = n \bmod 2 \tag{13}$$

When the `while` loop terminates at $l + p = 1$ there can only be two cases:

$$l = 1,\ p = 0 \ \ (n \bmod 2 = 0) \tag{14}$$

$$l = 0,\ p = 1 \ \ (n \bmod 2 = 1) \tag{15}$$

Thus, we may infer that if $n$ is odd, a *Pisidian* shall remain. Otherwise if $n$ is even, a *Lydian* shall remain.

## 3.3 Proof of loop invariant

### 3.3.1 Initialization

At the beginning it is known that $k = 1$ and $b = n$, thus we may have:

$$l + p = m + n - (k - 1) = l + p = m + n - (1 - 1) \Rightarrow l + p = m + n \tag{16}$$

$$p \bmod 2 = n \bmod 2 \tag{17}$$

As the above two *Equations* satisfy the assumption of the loop invariant, we may say that the scenario of $k = 1$ is true.

### 3.3.2 Maintenance

**Case 1** As it is known that $l+p = (l-1)+p$ and $k = k+1$, we may conclude that $l+p+k$ must remain the same.

**Case 2** As it is known that $l + p = (l + 1) + (p - 2)$ and $k = k + 1$, we may conclude that $l + p + k$ must remain the same.

**Case 3** As it is known that $l+p = (l-1)+p$ and $k = k+1$, we may conclude that $l+p+k$ must remain the same.

As all three cases satisfy the loop invariant of the algorithm, the induction is therefore proven to be valid.