DATA STRUCTURES AND ALGORITHMS HOMEWORK #1

B07902143 陳正康

1.1

(1) Donald Ervin Knuth (1938年1月10日 -)，史丹福大學電腦系榮譽退休教授，為現代計算機科學的先驅人物，創造了演算法分析的領域，在計算機科學及數學領域發表了多部具廣泛影響的論文和著作。1974年圖靈獎得主。

Pr. Donald Knuth 所寫的《The Art of Computer Programming》是計算機科學界最受高度敬重的參考書籍之一。在奧斯陸大學當客座教授時，和他的學生開發了 Knuth-Morris-Pratt (KMP) 演算法，使電腦更有效率地在文章中搜尋字串。他在擔任該職務期間開發了TEX排版軟體，成為今天大多數科技書籍使用的排版程式。除此之外，還有在電腦領域作出的多項貢獻，例如LR解析理論 (LR parsing theory)、Knuth-Moris-Pratt pattern 等。

(2) Claim: Let m be the value of minimum element in the array, and
$arr[j_1] = arr[j_2] = \cdots = arr[j_n] = m$, where $j_1 < j_2 < \cdots < j_n$, $n \geq 2$
When getMinIndex() finishes, it must be $minpos = j_1$
Proof: After the end of the loop where $i = j_1$, $minpos$ will remain j_1 since
$arr[i] \geq arr[j_1] = m$ for all $i > j_1$, which doesn't match the if condition and
$minpos$ will not be updated.

1.2

(1) An upper bound $i \leq \dfrac{\min(a,b)}{2}$ but adding extra check $i = \min(a,b)$ will work in general.

I.e. $k \in [2, \dfrac{\min(a,b)}{2}] \cup \min(a,b)$ for all possible common factor k of a and b

Proof: Suppose $a < b$; Assume $k \mid a$ where $k > \dfrac{a}{2}$

$\implies \dfrac{a}{k} < 2 \implies \dfrac{a}{k} = 1 \implies k = a$

In the case where extra check is not allowed, we may use $i \leq \min(a,b)$ as an upper bound.

Proof: Suppose $a < b$, $\forall i > a \quad i \nmid a$, so i cannot be GCD.

(2) Claim: The algorithm will return the greatest common divisor of a and b

a) Proof the result is a common divisor: The algorithm finishes upon the if condition meets: $i \mid a$ and $i \mid b$, so i is a common divisor.

b) Proof the result is the greatest common divisor:

Suppose $i = k$ when the algorithm is about to return

For $i \in (k, \min(a,b)]$, i cannot be common divisor otherwise it would have returned earlier

For $i > \min(a,b)$, $i > a \vee i > b$, so k cannot be a factor of both a and b

(3) There can be down to ONE iteration when $b \mid a$

(4) There can be up to b iterations when $\gcd(a,b) = 1$

1.3

(1)

Iteration	n	m	ans
1	7	14	2
2	7	0	2

(2) Claim: Let n, m at the end of i-th iteration be n_i, m_i , we have $\{n_i + m_i\}$ is strictly decreasing, i.e. $n_i + m_i > n_{i+1} + m_{i+1} \geq 0$ for every i

Proof:

Case 1: n_i is odd, m_i is even

Case 1-1: $m_i/2 < n_i \implies \ n_{i+1} = m_i/2 < m_i \quad ; m_{i+1} = n_i - m_i/2 < n_i$

Case 1-2: $m_i/2 \geq n_i \implies \ n_{i+1} = n_i \quad ; m_{i+1} = m_i/2 - n_i < m_i$

Case 2: n_i is even, m_i is odd

Case 2-1: $m_i < n_i/2 \implies \ n_{i+1} = m_i \quad ; m_{i+1} = n_i/2 - m_i < n_i$

Case 2-2: $m_i \geq n_i/2 \implies \ n_{i+1} = n_i/2 < n_i \quad ; m_{i+1} = m_i - n_i/2 < m_i$

Case 3: n_i is even, m_i is even

Case 3-1: $m_i < n_i \implies \ n_{i+1} = m_i/2 < m_i \quad ; m_{i+1} = n_i/2 - m_i/2 < n_i$

Case 3-2: $m_i \geq n_i \implies \ n_{i+1} = n_i/2 < n_i \quad ; m_{i+1} = m_i/2 - n_i/2 < m_i$

Case 4: n_i is odd, m_i is odd

Case 4-1: $m_i < n_i \implies \ n_{i+1} = m_i \quad ; m_{i+1} = n_i - m_i < n_i$

Case 4-2: $m_i \geq n_i \implies \ n_{i+1} = n_i \quad ; m_{i+1} = m_i - n_i < m_i$

By the claim, n_i and m_i must decrease toward zero, and the loop will finish

(3) Let $s = \gcd(\frac{a}{k}, \frac{b}{k})$, there is A, B such that $\frac{a}{k} = sA$, $\frac{b}{k} = sB$ and $\gcd(A, B) = 1$

$$\implies a = ksA, b = ksB \implies \gcd(a, b) = ks = k \gcd(\frac{a}{k}, \frac{b}{k})$$

(4) Define 2 sets: X is common divisors of a, b; Y is common divisors of $(a - b), b$

$$\forall d \in X \implies \frac{a-b}{d} = \frac{a}{d} - \frac{b}{d} \in Z \implies d \,|\, (a - b) \implies X \subseteq Y$$

$$\forall c \in Y \implies \frac{a}{c} = \frac{a-b}{c} + \frac{b}{c} \in Z \implies c \,|\, a \implies Y \subseteq X$$

so $X = Y$ and $\gcd(a, b) = \max(X) = \max(Y) = \gcd(a - b, b)$

(5) Consider the loop will be repeating some permutation of this order:

{odd(n)even(m) , even(n)odd(m), odd(n)odd(m)}

So a will half every 3 iteration, so will b, until $a = b = 0$

When odd(n)odd(m), nobody will half.

So $\lceil \log_2 a + \log_2 b + \log_2 \max(a, b) \rceil$ will be an upper bound

1.4

(1)

Iteration	n	m	tmp
1	8	13	13
2	5	8	8
3	3	5	5
4	2	3	3
5	1	2	2

(2) Claim: Let n_i, m_i be the value of n, m at the end of the i-th iteration, there is $0 \leq n_{i+1} + m_{i+1} < n_i + m_i$

Proof: At the (i+1)-th iteration,

$n_{i+1} = m_i \mod n_i = m_i - c \cdot n_i \quad$ for some c

$m_{i+1} = n_i$

$\implies n_{i+1} + m_{i+1} < n_i + m_i$

By the claim, we know n_i and m_i will decrease toward zero, and n_i or m_i will finally become zero.

Case 1: $n_k = 0,\ m_k \neq 0$

$n_k = m_{k-1} \mod n_{k-1} = 0 \implies$ the loop finishes after (k-1)-th iteration

Case 2: $n_k \neq 0,\ m_k = 0$

$n_{k+1} = m_k \mod n_k = 0 \mod n_k = 0 \implies$ the loop finishes after k-th iteration

(3) Claim: Let n_i, m_i be the value of n, m at the end of the i-th iteration with input (a, b), and n_i', m_i' be the same definition but with input ($2a$,$2b$), there is:

$$n_i' = 2n_i,\ m_i' = 2m_i \quad \text{for every } i$$

Proof:

In each iteration, $n_{i+1} = m_i \mod n_i,\ m_{i+1} = n_i$

Base case: $n_0' = 2a = 2n_0,\ m_0' = 2b = 2m_0$

Inductive step: Assume $n_i' = 2n_i,\ m_i' = 2m_i$

$n_{i+1}' = m_i' \mod n_i'$

$= m_i' - c \cdot n_i'$

$= 2m_i - c \cdot 2n_i$

$= 2(m_i - c \cdot n_i)$

where c satisfy: $\ m_i' - cn_i' > 0 > m_i' - (c+1)n_i'$

$\implies 2m_i - 2cn_i > 0 > 2m_i - 2(c+1)n_i$

$\implies m_i - cn_i > 0 > m_i - (c+1)n_i$

$\implies m_i - c \cdot n_i = m_i \mod n_i = n_{i+1}$

$\implies n'_{i+1} = 2n_{i+1}$

In addition, $m'_{i+1} = n'_i = 2n_i = 2m_{i+1}$

By mathematical induction, the claim proves.

(4) Let $a \mod b = a - cb$ for some integer c

By 1.3(4), $\gcd(a, b) = \gcd(a - b, a)$

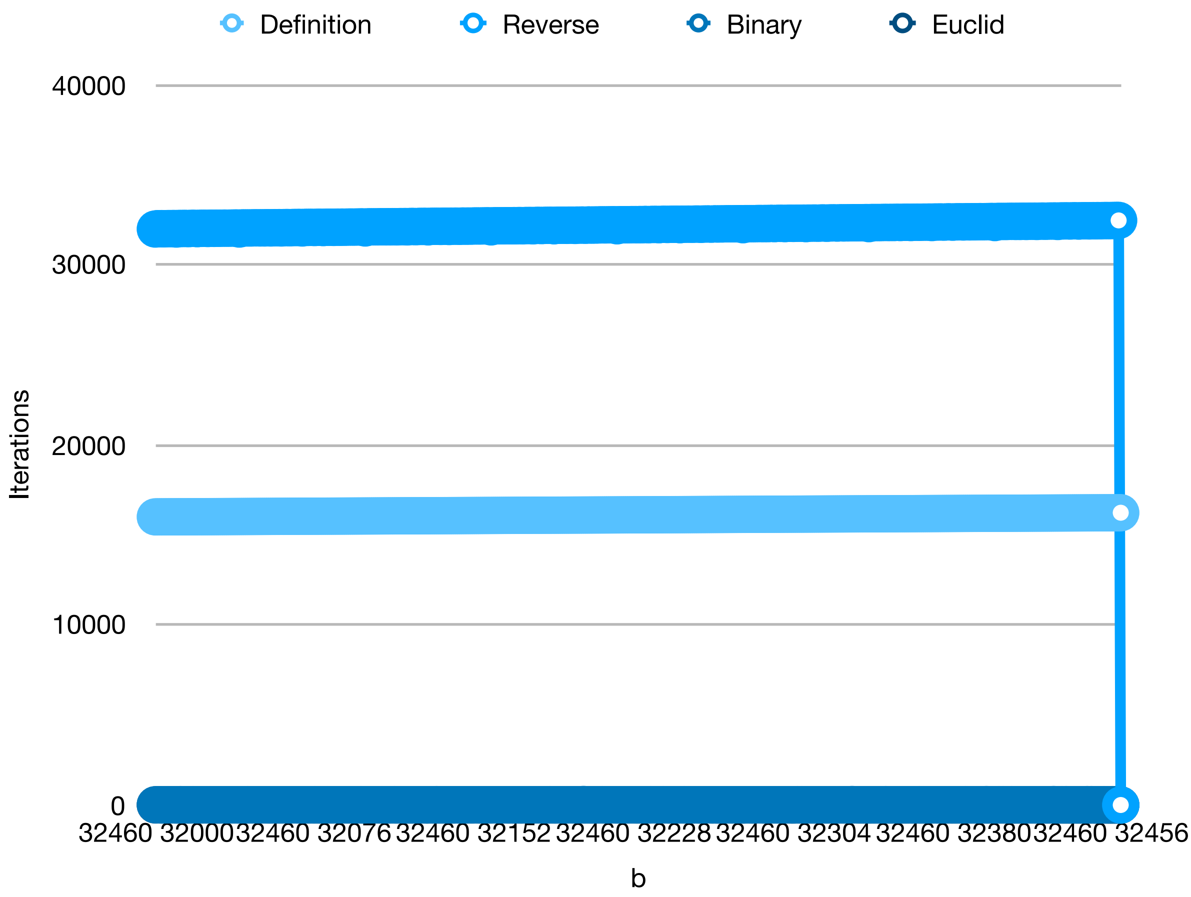
$\implies \gcd(a, b) = \gcd(a - b, b) = \gcd(a - 2b, b) = \cdots = \gcd(a - cb, b)$

$\implies \gcd(a, b) = \gcd(b, a \mod b)$

1.5

(1) Omitted.

(2)



As can be seen, the number of iteration mostly is almost the same when using the same method despite the input, but varies significantly across different method. So within such a range of input, the efficiency of GCD-finding algorithm more depend on the method than input size.