B08902127 林敬凱

# 1. More from the Class

① Donald Knuth, who was honored as "The Yoda of Silicon Valley", was one of the recipients of ACM Turing Award back in the 70s. He has published a great amount of book, while "The Art of Computer Programming" is an epic in the realm of computer science. He helped with the inventing of computational complexity to evaluate algorithms, developed TeX font and advocated literate programming. He majored in Physics at the first few years in college, but changing to Mathematics and later received his BS and Md. He soon started his career as Assistant Professor at California Institute of Tech, and later joined Stanford to teach Computer Science. The same year, he started his greatest masterpiece – The Art of Computer Programming. Later on, he come up with the computer type system – LaTeX which influenced the way of publication industry worked for a long period of time. While developing LaTeX, he came up with the idea of literate programming, advocating that programmers should think of coding as writing. And now, he has already retired for years, though he sometimes teaches some lectures in Stanford and Oxford. He devoted himself to write "the Art of Computer Programming" – The bible of algorithm. He is a living legend and perhaps the Euler of this era and in the realm of Computer Science.

②

* Claims

Algorithms returns m s.t. $arr[m] \le arr[j]$ for all m, j in range, and for all k if exists which $arr[m] = arr[k]$ $(m \neq k) \Rightarrow m < k$

<Proof>

1° if there's only one element in arr, obviously minpos = 0

2° Supposed that when len = k, minpos = r fulfill the claim

3° for len = k+1, minpos for previous k elements = r, consider the last element arr[k] (r has smallest index for smallest element).

① then if $arr[k] < arr[r]$, we update minpos ← k

② if $arr[k] \ge arr[r]$, minpos remains r since "$arr[k] < arr[r]$" not fulfill.

⇒ it holds for len = k+1

∴ from Mathmetical Induction with 1°, 2°, 3°, the claim is true.

## 2. Sequential Search for GCD

① I claim that upper bound would be $\Rightarrow \min(a, b)$

<Proof>

Without loss of Generality, provided $a \le b$

Set $k = \gcd(a, b)$

$$\therefore \begin{cases} k \mid a \\ k \mid b \end{cases} \Rightarrow \begin{cases} k \le a \\ k \le b \end{cases}$$ should be fulfilled simultaneously

$$\Rightarrow k \le \min(a, b)$$

Thus, with the upper bound $\min(a, b)$ it works.

②

* Claims : Algorithm return the first common devisor which is iterated from large to small number, thus returns gcd.

<proof>

Suppose $\{T\}$ is the set of common devider of $a, b$, $t \in \{T\}$.

$$\Rightarrow \begin{cases} t \mid a \\ t \mid b \end{cases} \Rightarrow \begin{cases} t \le a \\ t \le b \end{cases} \therefore t \le \min(a, b)$$

so gcd is in the range $[1, \min(a, b)]$

1° if $|T| = 1$, then $\gcd = 1$ <=> only 1 devide a and b

$\therefore$ the algorithm returns 1 and fulfill the claim.

2° Suppose $|T| \geq 2$ and $\alpha = gcd \in \{T\}$ , $\beta \in \{T\} \wedge \beta < \alpha$.

Although both $\alpha, \beta$ fulfill the condition, the algorithm stops at $\alpha$ (return) and won't even reach number $\beta$.

$\because \alpha$ is the largest in $\{T\}$ $\therefore \alpha$ is the greatest common devisor.

$\Rightarrow$ it also fulfill the claim. and returns $\alpha$ (gcd)

Thus from 1°, 2°, this algorithm returns $\alpha$.

③ one time ; it occurs when $b$ is the gcd of $a, b$

④ $b$ times ; it occurs when gcd of $a, b$ is "1"

# 3. Binary Algorithm for GCD

① $n = 14$ , $m = 42$

first iteration : $n = 7$ , $m = 14$ , ans = 2

second iteration : $n = 7$ , $m = 0$ , ans = 2

(out of loop) : ans $= 2 \times 7 = 14$

② * Claims

Each iteration reduce at least one of $n, m$ and eventally to 0.

<Proof>

By the following properties of gcd

1. if $a, b$ are even : $\gcd(a, b) = 2 \cdot \gcd(\frac{a}{2}, \frac{b}{2})$
2. if $a$ is even , $b$ is odd : $\gcd(a, b) = \gcd(\frac{a}{2}, b)$
3. if $a \geq b$ , $\gcd(a, b) = \gcd(a-b, b)$
4. Suppose $\gcd(a, 0) = a$

∴ So after each iteration $\gcd(a, b)$ eventually becomes smaller :

| b \ a | odd | even |
|---|---|---|
| odd | $\gcd(a-b, b)$ | $\gcd(\frac{a}{2}-b, b)$ , $\gcd(b-\frac{a}{2}, \frac{a}{2})$ |
| even | $\gcd(\frac{b}{2}-a, a)$ , $\gcd(a-\frac{b}{2}, \frac{b}{2})$ | $\gcd(\frac{a}{2}-\frac{b}{2}, \frac{b}{2}) \times 2$ |

$(a \geq b)$

∵ Natural number including 0 is well-ordered and the algorithm reduces at least one argument of $a, b$ until one reachs 0.

Thus the algorithm is finite.

③ Want to show $k \cdot \gcd(\frac{a}{k}, \frac{b}{k}) = \gcd(a, b)$

Set $a = p_1^{a_1} \cdot p_2^{a_2} \cdot p_3^{a_3} \cdots p_n^{a_n}$

$b = p_1^{b_1} \cdot p_2^{b_2} \cdot p_3^{b_3} \cdots p_n^{b_n}$

$\therefore \gcd(a,b) = p_1^{\min(a_1,b_1)} \cdot p_2^{\min(a_2,b_2)} \cdots p_n^{\min(a_n,b_n)}$

Suppose $k = p_1^{k_1} \cdot p_2^{k_2} \cdots p_n^{k_n}$

$\because k|a \wedge k|b \therefore$ for $i \in [1,k]$, $k_i \le a_i \wedge k_i \le b_i \Rightarrow k_i \le \min(a_i, b_i)$

$\Rightarrow \frac{k_i}{a_i} \ge 1 \wedge \frac{k_i}{b_i} \ge 1$

thus set $m = k \cdot a$, $n = k \cdot b$.

$\gcd(m,n) = \gcd(k \cdot a, k \cdot b) = p_1^{\min(k_1+a_1, k_1+b_1)} \cdots p_n^{\min(k_n+a_n, k_n+b_n)}$

$= (p^{k_1} \cdot p^{k_2} \cdots p^{k_n}) \cdot p^{\min(a_1,b_1)} \cdot p^{\min(a_2,b_2)} \cdots p^{\min(a_n,b_n)}$

$= k \cdot \gcd(a,b)$

Thus, $k \cdot \gcd(\frac{a}{k}, \frac{b}{k}) = \gcd(a,b)$ is proved.

④ Want to show $\gcd(a,b) = \gcd(b, a-b)$.

Set $k = \gcd(a,b)$, $s = \gcd(b, a-b)$

$\because a = k \cdot p > b = k \cdot q \therefore k | a-b = k(p-q)$

$\Rightarrow k|b \wedge k|a-b \Rightarrow k \le b \wedge k \le a-b \Rightarrow k \le s = \gcd(b, a-b)$ — ①

$\because b = s \cdot p'$, $(a-b) = s \cdot q' \Rightarrow a = s \cdot (p'+q')$

$\Rightarrow s|b \wedge s|a \Rightarrow s \le a \wedge s \le b \Rightarrow s \le k = \gcd(a,b)$ — ②

from ①, ② we know $s = k \Rightarrow \gcd(a,b) = \gcd(b, a-b)$

⑤ It will take $3\times\lfloor\log_2 a\rfloor - 4$ for $a > 16$ and $2\times\lfloor\log_2 a\rfloor$ for $a \le 16$

$\Rightarrow$ in general, it takes $\max\{3\lfloor\log_2 a\rfloor - 4,\ 2\cdot\lfloor\log_2 a\rfloor\}$ iterations.

<Proof>

1° for positive number "a", it has $\lfloor\log_2 a\rfloor$ digits.

let's call the "devide by 2" part sec. one.

"minus b'" part sec two.

2° for odd number a in sec one → it won't be modified

for even number a in sec one → last digit 0 was deleted.

$(XX\cdots XX0 \to XX\cdots XX)$.

3° for odd number b in sec two → the more "1" in its binary represent, the more 0 it produce in a after substraction.

from 2°, 3°, we should let b contains as much 0 and a with 1.

4° for $a \le 16$, when $a = 2^n - 1$, $b = 2^{n-1}$ takes most iterations

$$\Rightarrow \begin{cases} a: 11\cdots1 \\ b: 10\cdots0 \end{cases} \xrightarrow[\text{iterations}]{[n-1]} \begin{cases} a: 10\cdots0\ (n\text{ digit}) \\ b: \quad 1 \end{cases} \xrightarrow{[1]} \begin{cases} a: 11\cdots1\ (n-2\text{ digit}) \\ b: 1 \end{cases}$$

$$\xrightarrow{[1]} \begin{cases} a: 11\cdots10\ (n-2\text{ digit}) \\ b: 1 \end{cases} \xrightarrow{[n-3]} \begin{cases} a: 0 \\ b: 1 \end{cases} \circlearrowleft.$$

∴ it takes $2n-2$ which is $2\times$ digit of $a = 2\lfloor\log_2 a\rfloor$

5° for $a > 16$, when $a = 2^n - 2^2$, $b = 2^{n-1}$ takes most iteration.

$$\Rightarrow \begin{cases} a: 11\cdots1011 \\ b: 10\cdots0000 \end{cases} \xrightarrow{[n-3]} \begin{cases} a: 111\cdots11\ (n-1\text{ digit}) \\ b: \quad 100 \end{cases} \xrightarrow{[2]} \begin{cases} a: 11\cdots100\ (n-2) \\ b: \quad 1 \end{cases}$$

$$\xrightarrow{[2]} \begin{cases} a: 111\cdots100\ (n-3\text{ digit}) \\ b: 1 \end{cases} \xrightarrow{2\times[n-5]} \begin{cases} a: 100 \\ b: 1 \end{cases} \xrightarrow{[2]} \begin{cases} a=0 \\ b=1 \end{cases} \circlearrowleft$$

∴ it takes $3n-7$ which is $3\times\lfloor\log_2 a\rfloor - 4$

Process 4° makes sense on ordinary intuition since a is full of 1 and b has as much 0 as it could.

In contrast, Process 5° take away the $2^2$ digit from a which allows a ends with "100" in the process and reduce the frequency of degrading digits.

You could easily use program to prove the following by running through all possibility. for any number $\leq a$

```
albertlin@linxinkaide-MacBook-Pro Homework % ./binary
123 64
000001111011   000001000000
000000100000   000001011011
000000010000   000001001011
000000001000   000001000011
000000000100   000000111111
000000000010   000000111101
000000000001   000000111100
000000000001   000000011101
000000000001   000000011100
000000000001   000000001101
000000000001   000000001100
000000000001   000000000101
000000000001   000000000100
000000000001   000000000001
000000000001   000000000000
Case (123, 64): GCD-By-Binary = 1, taking 14 iterations


127 64
000001111111   000001000000
000000100000   000001011111
000000010000   000001001111
000000001000   000001000111
000000000100   000001000011
000000000010   000001000001
000000000001   000001000000
000000000001   000000011111
000000000001   000000011110
000000000001   000000001110
000000000001   000000000110
000000000001   000000000010
000000000001   000000000000
Case (127, 64): GCD-By-Binary = 1, taking 12 iterations
```

$3\lfloor \log_2 123 \rfloor - 4$

$2\lfloor \log_2 123 \rfloor$

# 4. Euclid's Algorithm for GCD

① $n = 13$, $m = 21$

first iteration : $m = 13$, $n = 8$, $tmp = 13$.

second iteration : $m = 8$, $n = 5$, $tmp = 8$.

third iteration : $m = 5$, $n = 3$, $tmp = 5$

fourth iteration : $m = 3$, $n = 2$, $tmp = 3$.

fifth iteration : $m = 2$, $n = 1$, $tmp = 2$

(out of loop : return 1).

② *Claim :

Each iteration reduce the larger number and eventually makes the larger divisible by the smaller.

<proof>

In every iteration $(a, b)$ becomes $(b, b\%a)$

So consider the following scenario $(a > b)$

① $b \nmid a \Rightarrow$ the next iteration becomes $(b, b\%a)$, the worst scenario $b\%a$ becomes 1.

② $b \mid a \Rightarrow$ doesn't enter next loop and returns $b$

∵ Natural number is well-ordered and the algorithm reduces the larger number until the smaller becomes gcd (worst case : 1).

Thus, the algorithm is finite.

③ * Claim :

It will also takes T iterations for gcd(2a, 2b) as gcd(a, b) does.

<proofs>

* ① gcd(a, b) = gcd(a-b, b)
  ② gcd(k·a, k·b) = k·gcd(a,b). $(a \geq b)$

Without loss of generality, assume $a > b$.

Set k is gcd of a and b, so gcd(2a, 2b) = 2 gcd(a, b) = 2k.

→ Consider the following situation

1. $a \mid b \Rightarrow \gcd(a,b) = k$ (1 step and end)

   ∴ $2a \mid 2b \Rightarrow \gcd(2a, 2b) = 2 \cdot \gcd(a,b) = 2k$ (1 step and end)

2. $a \nmid b \Rightarrow \gcd(a,b) = \gcd(b, a-b)$

   ∴ $2a \nmid 2b \Rightarrow \gcd(2a, 2b) = \gcd(2b, 2a-2b)$

   $= 2 \cdot \gcd(a,b) = 2 \cdot \gcd(b, a-b)$

∴ from 1. 2. ⇒ for every calculation of 2a, 2b, we can extract 2 and its process will be the same as that of a, b.

Thus, if it takes r iterations for $(a, b) \rightarrow (p, q)$
it also takes r iterations for $(2a, 2b) \rightarrow (2p, 2q)$

And since we set (a,b) ends at $(\alpha, \beta)$, $(\beta \mid \alpha)$
⇒ (2a, 2b) will ends at $(2\alpha, 2\beta)$, $(2\beta \mid 2\alpha)$.

In conclusion : (2a, 2b) will also takes T iterations.

④ Want to show that $\gcd(a, b) = \gcd(b, a \bmod b)$

1° for $a < b \Rightarrow a \bmod b = a$

$\therefore \gcd(a, b) = \gcd(b, a) = \gcd(b, a \bmod b)$

2° for $a = b \Rightarrow a \bmod b = 0$

$\therefore \gcd(a, b) = \gcd(a, 0) = \gcd(b, 0) = a = b.$

3° for $a > b \Rightarrow$ Set $a = p \cdot b + q \quad [p, q \in N, b > q]$.

Let $k = \gcd(a, b)$ , $h = \gcd(b, a \bmod b)$.

$\because \begin{cases} k \mid a \\ k \mid b \end{cases} \Rightarrow$ for $q = a - p \cdot b = a \bmod b \quad -①$

devide ① by $k \Rightarrow \frac{q}{k} = \frac{a}{k} - p \cdot \frac{b}{k} \in \mathbb{Z} \Rightarrow k \mid q = (a \bmod b)$

$\rightarrow k \mid b \wedge k \mid q \Rightarrow k \mid h = \gcd(b, a \bmod b) \quad -②$

$\because \begin{cases} h \mid b \\ h \mid q \end{cases} \Rightarrow$ for $a = p \cdot b + q \quad -③$

devide ③ by $h \Rightarrow \frac{a}{h} = p \cdot \frac{b}{h} + \frac{q}{h} \in \mathbb{Z} \Rightarrow h \mid a.$

$\rightarrow h \mid a \wedge h \mid b \Rightarrow h \mid k = \gcd(a, b) \quad -④$

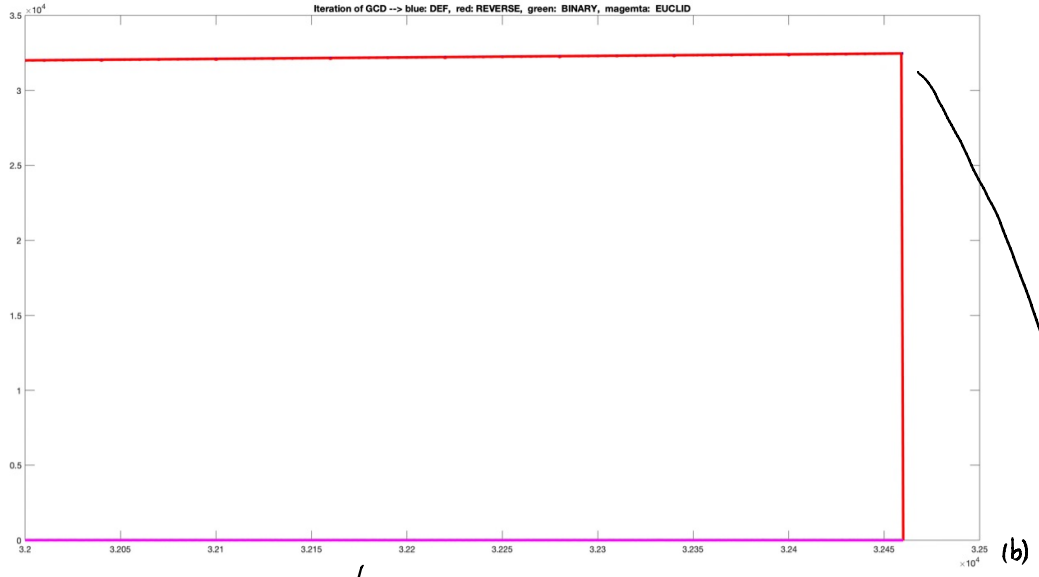
from ②, ④ : $k \mid h \wedge h \mid k \Rightarrow k = h.$

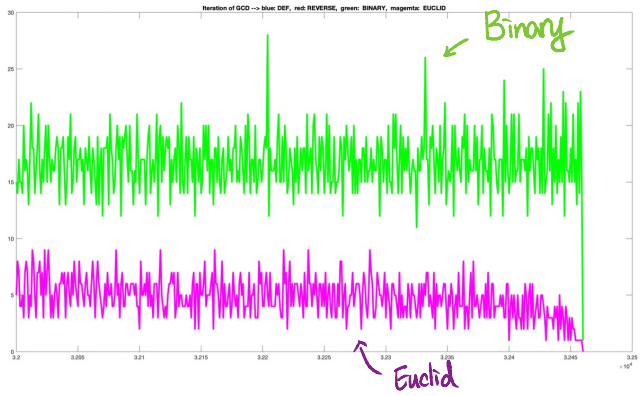
$\Rightarrow \gcd(a, b) = \gcd(b, a \bmod b)$

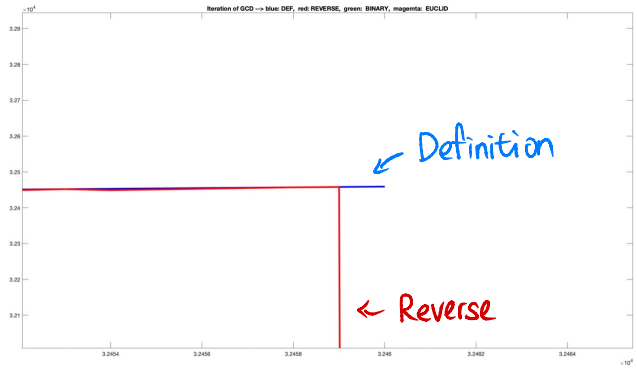
Thus from 1°, 2°, 3° $\Rightarrow$ for $a \gtreqless b$, $\gcd(a, b) = \gcd(b, a \bmod b)$

# 5. Comparison of GCD

(iteration)




As shown in the figure, generally the iterations taken : Euclid ≤ Binary ≪ Reverse ≤ Definition

for Euclid and Binary, you can see that Euclid ≤ Binary, but the best case for Euclid is 0 and Binary is 1

for GCD-BY-DEF, we have to run through all possibility and thus its graph is a line.

for GCD-BY-REV, it can reduce some iterations and the best case is 1. But for most case it still takes almost as more steps at "DEF" does

So the efficiency : Euclid ≥ Binary ≫ Reverse ≥ Definition (consider only by the iterations they take).