EE4015 Assignment-1 Presentation

Krishna Srikar Durbha (EE18BTECH11014)

25^{th} August 2021

Euclidean Algorithm by Subtraction I

Euclidean Algorithm is a recursive method of finding Greatest Common Divisor of 2 numbers. For some positive integers a and b, it works by repeatedly subtracting the smaller number from the larger one until they become equal. At this point, the value of either term is the greatest common divisor of our inputs.

Algorithm:

Step-1: If $a = b$, then return the value of a

Step-2: Otherwise, if a $>$ b then let a = a - b and return to Step-1

Step-3: Otherwise, if a $<$ b, then let b = b - a and return to Step-1

Proof:

Proof involves proving that, subtracting between a and b doesn't change GCD. Let a, b be 2 positive integers such that $gcd(a, b) = m$ and $a > b$. So, it can be written as,

$$a = a_1 \times m \tag{1}$$

$$b = b_1 \times m \tag{2}$$

$$gcd(a, b) = m \implies gcd(a_1, b_1) = 1 \tag{3}$$

Euclidean Algorithm by Subtraction II

We need to prove that $gcd(a-b,b) = m$. We will prove it by contradiction. Let $gcd(a-b,b) = M$ where $M > m \implies k \neq 1$

$$a - b = (a_1 - b_1) \times m \tag{4}$$

$$b = b_1 \times m \tag{5}$$

$$gcd(a-b,b) = M \implies M = k \times m \text{ (For some integer } k) \tag{6}$$

$$a - b \equiv 0 \pmod{M} \text{ and } b \equiv 0 \pmod{M} \tag{7}$$

$$\implies a - b \equiv 0 \pmod{km} \text{ and } b \equiv 0 \pmod{km} \tag{8}$$

$$\implies a_1 - b_1 \equiv 0 \pmod{k} \text{ and } b_1 \equiv 0 \pmod{k} \tag{9}$$

$$\implies a_1 \equiv 0 \pmod{k} \text{ and } b_1 \equiv 0 \pmod{k} \tag{10}$$

We know that $gcd(a_1, b_1) = 1$, so there doesn't exist a $M \neq m$ such that $gcd(a-b,b) = M$. So, from contradiction, $gcd(a,b) = gcd(a-b,b) = M$ for $a > b$. Worst Case Time-Complexity is $\mathcal{O}(a+b)$.

Complexity Analysis of Euclidean Algorithm by Subtraction

Let $a > b$ and $T(n)$ denote time complexity of $gcd(a, b)$ where $n = a + b$. Then,

$$T(n) = 1 + T(n - b) \tag{11}$$

$$T(n - b) = 1 + T(n - 2b) \text{ if } a > 2b \tag{12}$$

$$T(n - b) = 1 + T(n - a - b) \text{ if } b < a < 2b \tag{13}$$

On assuming $n > (x_1 a + x_2 b)$ for some x_1, x_2, $T(n)$ can be written as:

$$T(n) = k + T(n - x_1 a - x_2 b) \text{ (For } k = x_1 + x_2) \tag{14}$$

No.of steps vary linearly with $n = a + b$. So, in the worst-case scenerio the algorithm performs $a + b$ subtractions. Hence Worst Case Time-Complexity for calculating GCD of a and b using Euclidean Algorithm by Subtraction is $\mathcal{O}(a + b)$.

$$\begin{aligned} f(15, 255) &= f(15, 240) = f(15, 225) = f(15, 210) \\ &= f(15, 195) = f(15, 180) = f(15, 165) = f(15, 150) \\ &= f(15, 135) = f(15, 120) = f(15, 105) = f(15, 90) \\ &= f(15, 75) = f(15, 60) = f(15, 45) = f(15, 30) \\ &= f(15, 15) = 1 \end{aligned}$$

Euclidean Algorithm by Division I

Euclidean Algorithm by Division involves divison rather than subtraction. For some positive integers a and b, $gcd(a, b) = gcd(b, a \bmod b)$. We repeat the procedure until convergence.

Let a, b be 2 positive integers such that $a > b$. By applying Euclid's Algorithm from 0^{th}-step ,

$$a = q_0 b + r_0 \tag{15}$$

$$b = q_1 r_0 + r_1 \tag{16}$$

$$r_0 = q_2 r_1 + r_2 \tag{17}$$

$$r_1 = q_3 r_2 + r_3 ... \tag{18}$$

Here $a > b$, $b > r_0$, $r_0 > r_1$, $r_1 > r_2$.. and so on. So, remainders are decreasing after each step.

Euclidean Algorithm by Division II

Let at n^{th}-step $r_{n-2} = q_n r_{n-1}$ i.e $r_n = 0$.

$$r_{n-2} = q_n r_{n-1} \tag{19}$$

$$r_{n-3} = q_{n-1} r_{n-2} + r_{n-1} \tag{20}$$

$$\implies r_{n-1} \text{ divides } r_{n-2}, r_{n-3}, r_{n-4}, ..., r_1, r_0, b, a \tag{21}$$

$$\implies a \equiv 0 \pmod{r_{n-1}} \text{ and } b \equiv 0 \pmod{r_{n-1}} \tag{22}$$

So, the proof goes as $gcd(a, b) = r_{n-1}$. We will prove it by contraction. Let $gcd(a, b) = M \implies M > r_{n-1}$,

$$a = a_1 \times M \text{ and } b = b_1 \times M \tag{23}$$

$$r_0 = a - q_0 b = M(a_1 - q_0 b_1) \tag{24}$$

$$r_1 = b - q_1 r0 = M(b_1 - a_1 + q_0 b_1) \tag{25}$$

Euclidean Algorithm by Division III

So, M divides $a, b, r_0, r_1, \ldots$ and so on all the following remainders. So, M should divide r_{n-1}, which implies $r_{n-1} \geq M$ which is a contraction from $M > r_{n-1}$.

So, there doesn't exist a $M > r_{n-1}$ which is a divisor of a and b. So, $gcd(a, b) = r_{n-1}$.

$$gcd(15, 255) = gcd(255, 15) = gcd(15, 0) = 15$$

Complexity Analysis of Euclidean Algorithm by Division I

Let f_n denote elements in Pingala Sequence starting from $n = 0$ where $f_0 = 0, f_1 = 1, f_2 = 1, \ldots$ and so on. Elements of the sequence can be written as follows:

$$f_{n+2} = 1 \times f_{n+1} + f_n$$

$$f_{n+1} = 1 \times f_n + f_{n-1}$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$f_4 = 1 \times f_3 + f_2$$

$$f_3 = 2 \times f_2$$

The above equations are similar to equations in Euclidean Algorithm by Division i.e from (15) to (19). Hence it can be proved that $gcd(f_{n+2}, f_{n+1}) = f_2 = 1$ and takes n steps to converge.

Complexity Analysis of Euclidean Algorithm by Division II

If $gcd(a, b)$ where $a > b$ takes n steps to converge by using Euclidean Algorithm by Division, then $a \geq f_{n+2}$ and $b \geq f_{n+1}$.

Proof by Mathematical Induction:

Let $a = 2$ and $b = 1$. Then, $gcd(2, 1) = 1$ takes 1 step to converge. $a \geq f_3 = 2$ and $b \geq f_2 = 1$. Assuming statements hold true at $n - 1^{th}$ step, $gcd(b, a\%b)$ takes $n - 1$ steps to converge.

$$b \geq f_{n+1} \text{ and } a\%b \geq f_n \tag{26}$$

$$a = q_0 b + a\%b \tag{27}$$

$$a \geq b + a\%b \tag{28}$$

$$a \geq f_{n+1} + f_n \implies a \geq f_{n+2} \tag{29}$$

$$\implies a \geq f_{n+2} \text{ and } b \geq f_{n+1} \tag{30}$$

Hence proved.

Complexity Analysis of Euclidean Algorithm by Division III

Let $gcd(a, b)$ takes n steps to converge. Then,

$$a \geq f_{n+2} \tag{31}$$

$$b \geq f_{n+1} \tag{32}$$

$$f_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right) \tag{33}$$

$$\phi = \frac{1+\sqrt{5}}{2} \tag{34}$$

$$f_n \approx \phi^n \tag{35}$$

$$b \approx \phi^{n+1} \tag{36}$$

$$n \approx \log_\phi (min(a, b)) \tag{37}$$

So, in the worst-case scenerio the algorithm performs $n \approx \log_\phi (min(a, b))$ divisons. Hence, Worst Case Time-Complexity for calculating GCD of a and b using Euclidean Algorithm by Division is $\mathcal{O}(\log min(a, b))$.

Another Example:

The following example illustrates the difference in no.of steps between Euclidean Algorithm by Subtraction and Euclidean Algorithm by Divison. To find GCD of two numbers 24 and 92.

By Euclid's Subtraction,

$$f(24, 92) = f(24, 68) = f(24, 44) = f(24, 20)$$
$$= f(4, 20) = f(4, 16) = f(4, 12) = f(4, 8)$$
$$= f(4, 4) = 4$$

By Euclid's Division,

$$f(24, 92) = f(92, 24) = f(24, 20) = f(20, 4)$$
$$= f(4, 0) = 4$$

The difference in no.of steps indicates the performance improvement of Euclidean Algorithm by Division over Subtraction.

Flow Diagrams I

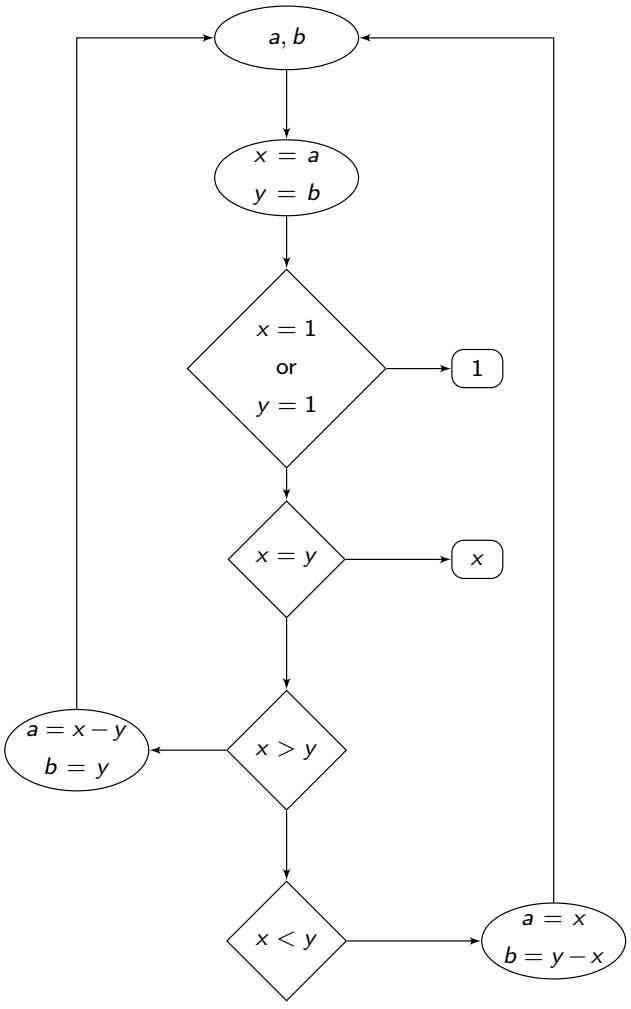

Figure: Flowchart of Euclidean Algorithm by Subtraction

Flow Diagrams II

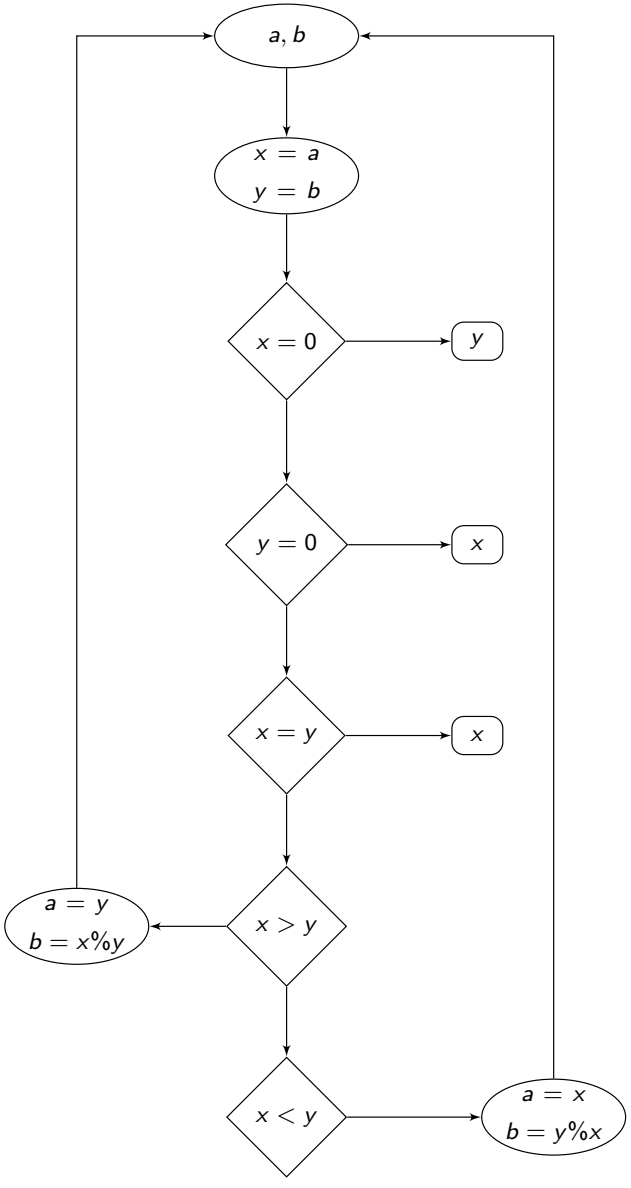

Figure: Flowchart of Euclidean Algorithm by Division