EE4013 Assignment-1

Krishna Srikar Durbha - EE18BTECH11014

Download all codes from

```
https://github.com/dks2000dks/IIT−Hyderabad−
    Semester−Courses/tree/master/EE4013/
    Assignment1/codes
```

and latex-tikz codes from

```
https://github.com/dks2000dks/IIT−Hyderabad−
    Semester−Courses/tree/master/EE4013/
    Assignment1
```

1 Problem

Consider the following ANSI C function:

```c
int SomeFunction(int x, int y){
        if ((x == 1) || (y == 1)) return 1;
        if (x == y) return x;
        if (x > y) return SomeFunction(x−y, y);
        if (x < y) return SomeFunction(x, y−x);
}
```

The value of returned by SomeFunction(15,255) is ——

2 Solution

2.1 Answer

Let **SomeFunction** be represented as f. The recursion goes as follows:

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

One other approach is that knowing or recognising that f is an implementation of Euclidean Algorithm by Subtraction for calculating GCD of positive integers x and y.

$$gcd(15,255) = 15 \text{ (As } 15 \times 17 = 255) \tag{2.1.1}$$

2.2 Euclidean Algorithm by Subtraction

Euclidean Algorithm is a recursive method of finding Greatest Common Divisor of 2 numbers. For some positive integers a and b, Euclidean Algorithm by Subtraction repeatedly subtracts the smaller number from the larger one. $gcd(a,b) = gcd(a-b,b)$ considering that $a > b$. We repeat the procedure till convergence i.e both numbers are equal. At this point, the value of either term is the greatest common divisor of our inputs.

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,

$$a = a_1 \times m \tag{2.2.1}$$
$$b = b_1 \times m \tag{2.2.2}$$
$$gcd(a,b) = m \implies gcd(a_1,b_1) = 1 \tag{2.2.3}$$

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

$$a - b = (a_1 - b_1) \times m \tag{2.2.4}$$
$$b = b_1 \times m \tag{2.2.5}$$
$$gcd(a-b,b) = M \tag{2.2.6}$$
$$M = k \times m \text{ (For some integer } k) \tag{2.2.7}$$
$$a - b \equiv 0 \pmod{M} \text{ and } b \equiv 0 \pmod{M} \tag{2.2.8}$$
$$a - b \equiv 0 \pmod{km} \text{ and } b \equiv 0 \pmod{km} \tag{2.2.9}$$
$$a_1 - b_1 \equiv 0 \pmod{k} \text{ and } b_1 \equiv 0 \pmod{k} \tag{2.2.10}$$
$$a_1 \equiv 0 \pmod{k} \text{ and } b_1 \equiv 0 \pmod{k} \tag{2.2.11}$$

We know that $gcd(a_1,b_1) = 1$, so a and b cannot have a common divisor k. Hence by contradiction, there doesn't exist a $M \neq m$ such that $gcd(a-b,b) = M$. Hence it can be proved that, $gcd(a,b) = gcd(a-b,b) = m$ for $a > b$.

2.2.1 Complexity Analysis:

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

Then,

$$T(n) = 1 + T(n-b) \tag{2.2.12}$$

$$T(n-b) = 1 + T(n-2b) \text{ if } a > 2b \tag{2.2.13}$$

$$T(n-b) = 1 + T(n-a-b) \text{ if } b < a < 2b \tag{2.2.14}$$

On assuming $n > (x_1a + x_2b)$ for some x_1, x_2, $T(n)$ can be written as:

$$T(n) = k + T(n - x_1a - x_2b) \text{ (For } k = x_1 + x_2) \tag{2.2.15}$$

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 $O(a + b)$.

Codes of Euclidean Algorithm by Subtraction:

```
codes/Euclid_Subtraction.py
codes/Euclid_Subtraction.c
```

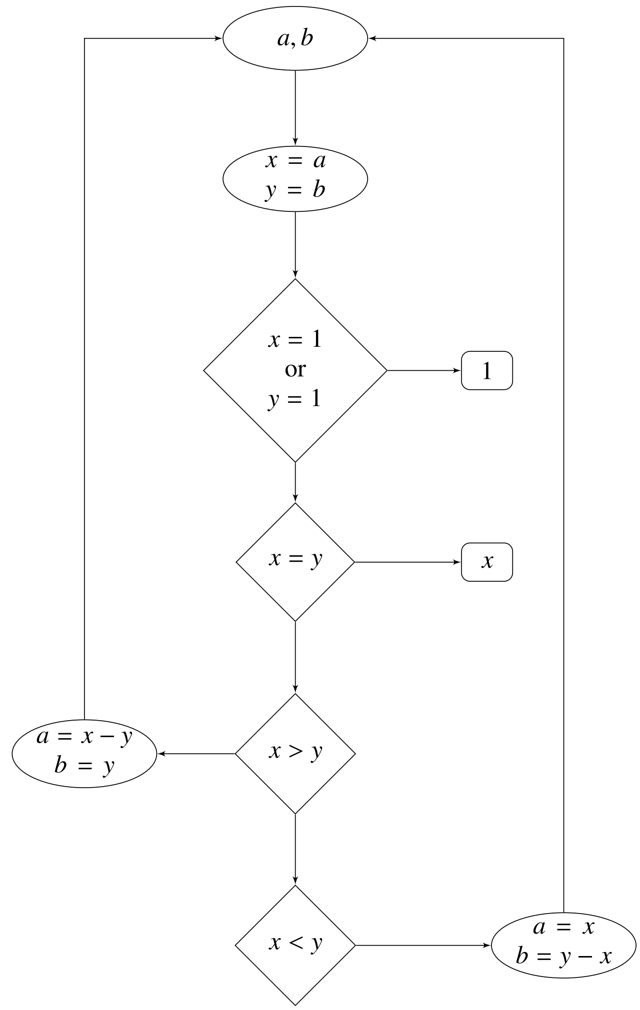

Fig. 0: Flowchart of Euclidean Algorithm by Subtraction

2.3 Euclidean Algorithm by Division

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

$$a = q_0b + r_0 \tag{2.3.1}$$

$$b = q_1r_0 + r_1 \tag{2.3.2}$$

$$r_0 = q_2r_1 + r_2 \tag{2.3.3}$$

$$r_1 = q_3r_2 + r_3... \tag{2.3.4}$$

Here $a > b$, $b > r_0$, $r_0 > r_1$, $r_1 > r_2$.. and so on. So, remainders are decreasing after each step.
Let at n^{th}-step $r_{n-2} = q_nr_{n-1}$ i.e $r_n = 0$.

$$r_{n-2} = q_nr_{n-1} \tag{2.3.5}$$

$$r_{n-3} = q_{n-1}r_{n-2} + r_{n-1} \tag{2.3.6}$$

$$r_{n-1} \text{ divides } r_{n-2}, r_{n-3}, r_{n-4}, ..., r_1, r_0, b, a \tag{2.3.7}$$

$$a \equiv 0 \pmod{r_{n-1}} \text{ and } b \equiv 0 \pmod{r_{n-1}} \tag{2.3.8}$$

So, r_{n-1} is a common divisor of both a and b. Let $gcd(a, b) = M \implies M > r_{n-1}$,

$$a = a_1 \times M \text{ and } b = b_1 \times M \tag{2.3.9}$$

$$r_0 = a - q_0b = M(a_1 - q_0b_1) \tag{2.3.10}$$

$$r_1 = b - q_1r_0 = M(b_1 - q_1a_1 + q_1q_0b_1) \tag{2.3.11}$$

So, M divides $a, b, r_0, r_1, ...$ 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 as $M > r_{n-1}$. Hence by contradiction, there doesn't exist a $M > r_{n-1}$ which is a divisor of a and b. So, $gcd(a, b) = r_{n-1}$.

On using Euclidean Algorithm by Division the recursion goes as follows:

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

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

$$\begin{aligned} f_{n+2} &= 1 \times f_{n+1} + f_n \\ f_{n+1} &= 1 \times f_n + f_{n-1} \\ &.......................... \\ f_4 &= 1 \times f_3 + f_2 \\ f_3 &= 2 \times f_2 \end{aligned}$$

The above equations are similar to equations in Euclidean Algorithm by Division i.e from (2.3.1) to (2.3.5). Hence it can be proved that $gcd(f_{n+2}, f_{n+1}) = f_2 = 1$ and takes n steps to converge.

If $gcd(a,b)$ 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.

$$b \geq f_{n+1} \tag{2.3.12}$$
$$a\%b \geq f_n \tag{2.3.13}$$
$$a = q_0 b + a\%b \tag{2.3.14}$$
$$a \geq b + a\%b \tag{2.3.15}$$
$$a \geq f_{n+1} + f_n \tag{2.3.16}$$
$$a \geq f_{n+2} \tag{2.3.17}$$

Hence proved.

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

$$a \geq f_{n+2} \tag{2.3.18}$$
$$b \geq f_{n+1} \tag{2.3.19}$$
$$f_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right) \tag{2.3.20}$$
$$\phi = \frac{1+\sqrt{5}}{2} \tag{2.3.21}$$
$$f_n \approx \phi^n \tag{2.3.22}$$
$$b \approx \phi^{n+1} \tag{2.3.23}$$
$$n \approx \log_\phi (min(a,b)) \tag{2.3.24}$$

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 $O(\log min(a,b))$.

Codes of Euclidean Algorithm by Division:

```
codes/Euclid_Division.py
codes/Euclid_Division.c
```

2.4 Comparison between Algorithms:

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,

$$\begin{aligned} &f(24,92) = f(24,68) = f(24,44) = f(24,20) \\ &= f(4,20) = f(4,16) = f(4,12) = f(4,8) \\ &= f(4,4) = 4 \end{aligned}$$

By Euclid's Division,

$$\begin{aligned} &f(24,92) = f(92,24) = f(24,20) = f(20,4) \\ &= f(4,0) = 4 \end{aligned}$$

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

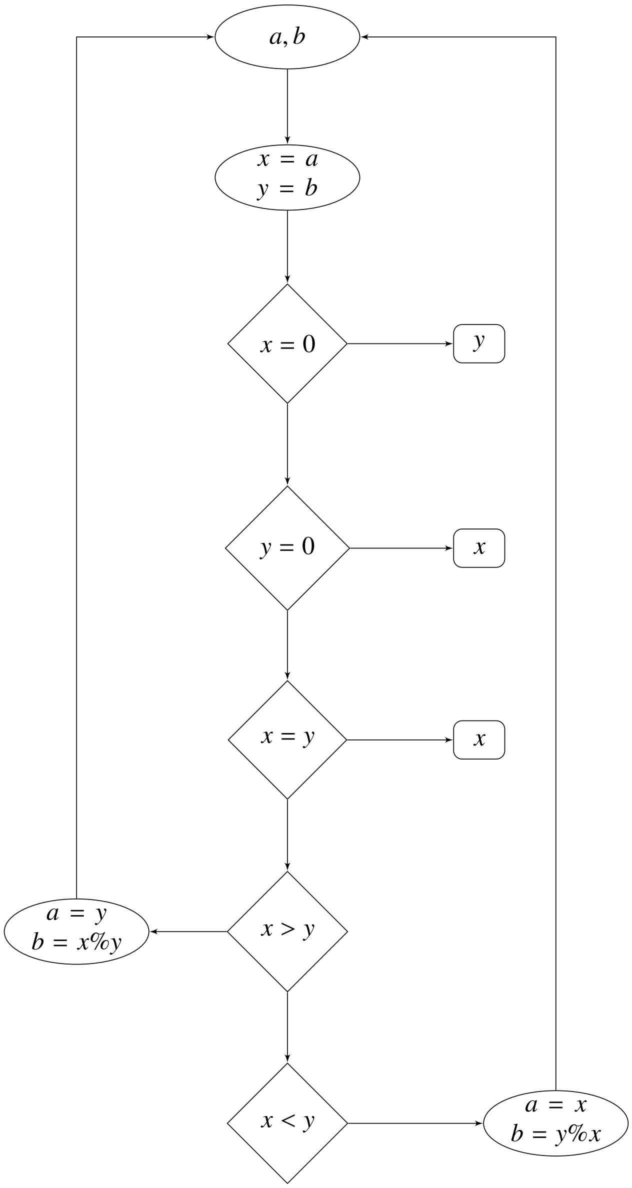

Fig. 0: Flowchart of Euclidean Algorithm by Division

a, b	N_-	$T_-(\mu s)$	$N_\%$	$T_\%(\mu s)$
319, 50	14	1.212	8	0.651
453, 369	14	0.887	7	0.504
263, 810	18	0.886	6	0.517
243, 929	18	0.708	9	0.643
508, 609	41	1.299	6	0.397

TABLE 0: Comparison between Euclidean Algorithm by Division and Euclidean Algorithm by Subtraction

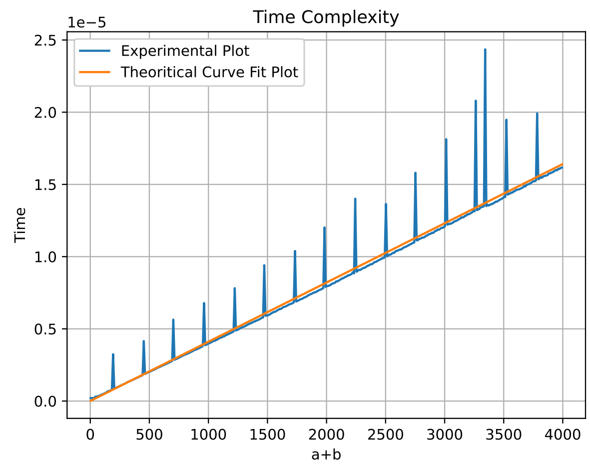

Fig. 0: Plot of Worst-Case Time Complexity of Euclidean Algorithm by Subtraction

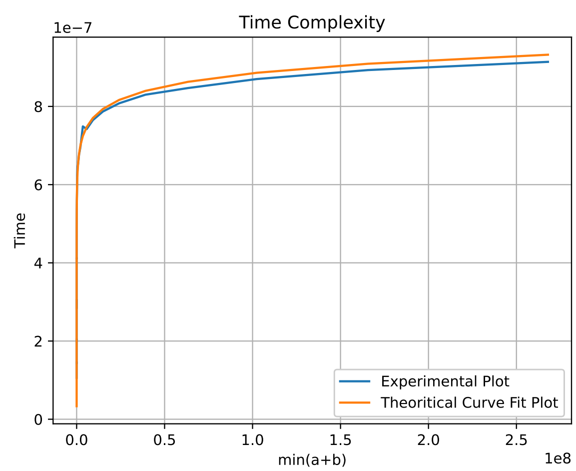

Fig. 0: Plot of Worst-Case Time Complexity of Euclidean Algorithm by Division