EE4013 Assignment-1

Krishna Srikar Durbha - EE18BTECH11014

Download all python 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 $a > b$.

Worst Case Time-Complexity of Euclidean Algorithm by Subtraction is $O(a+b)$.

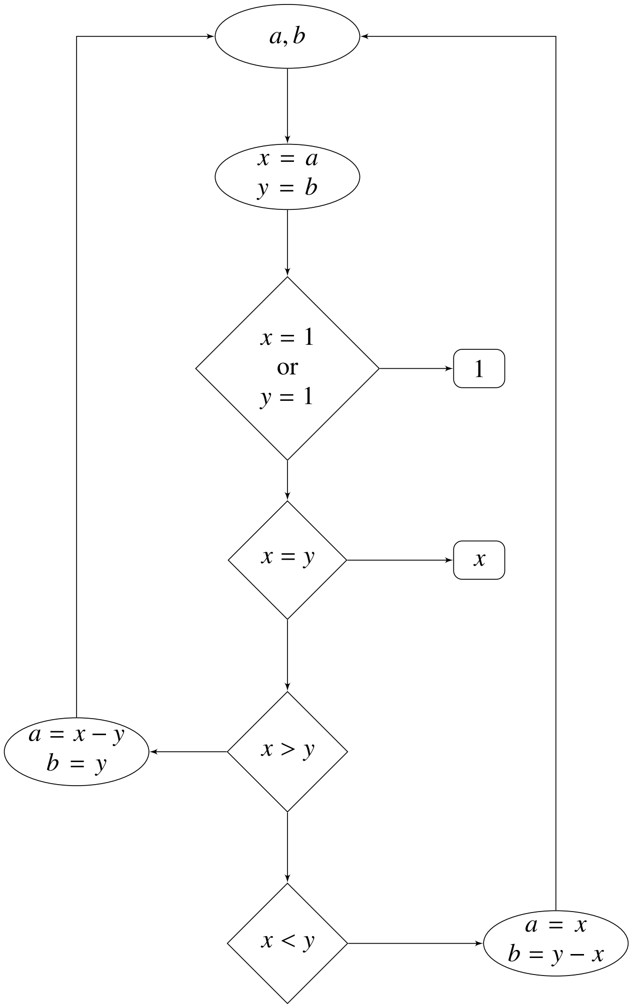

Fig. 0: Flowchart of Euclidean Algorithm by Subtraction

Codes of Euclidean Algorithm by Subtraction:

codes/Euclid_Subtraction.py codes/Euclid_Subtraction.c

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,

$$a = q_0 b + r_0 \tag{2.3.1}$$
$$b = q_1 r_0 + r_1 \tag{2.3.2}$$
$$r_0 = q_2 r_1 + r_2 \tag{2.3.3}$$
$$r_1 = q_3 r_2 + r_3 ... \tag{2.3.4}$$

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 $r_{n-2} = q_n r_{n-1}$ i.e $r_n = 0$.

$$r_{n-2} = q_n r_{n-1} \tag{2.3.5}$$
$$r_{n-3} = q_{n-1} r_{n-2} + r_{n-1} \tag{2.3.6}$$
$$r_{n-1} \text{ divides } r_{n-2}, r_{n-3}, r_{n-4}, ..., r_1, r_0, b, a \tag{2.3.7}$$
$$a \equiv 0 \pmod{r_{n-1}} \text{ and } b \equiv 0 \pmod{r_{n-1}} \tag{2.3.8}$$

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

$$a = a_1 \times M \text{ and } b = b_1 \times M \tag{2.3.9}$$
$$r_0 = a - q_0 b = M(a_1 - q_0 b_1) \tag{2.3.10}$$
$$r_1 = b - q_1 r_0 = M(b_1 - q_1 a_1 + q_1 q_0 b_1) \tag{2.3.11}$$

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, $gcd(a, b) = r_{n-1}$.

Worst Case Time-Complexity of Euclidean Algorithm by Division is $O(\log min(a, b))$.

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

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

Codes of Euclidean Algorithm by Division:

codes/Euclid_Division.py codes/Euclid_Division.c

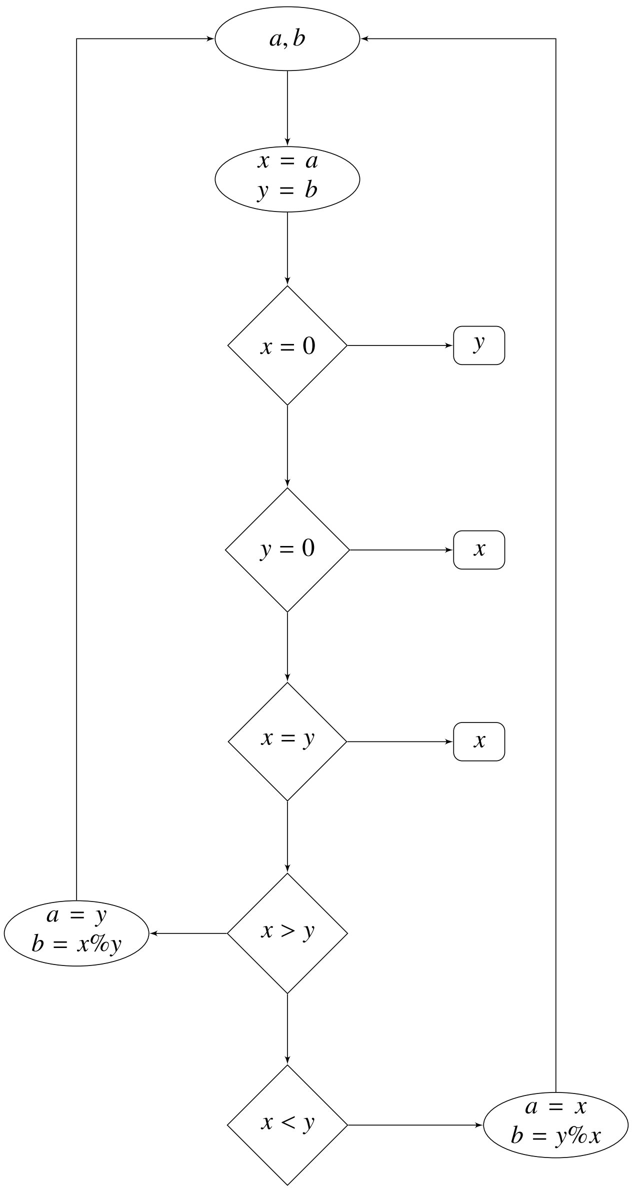

Fig. 0: Flowchart of Euclidean Algorithm by Division