Euclid Algo

If one number is 0, gcd is other number.

$$gcd(a,b) = \begin{cases} a, & \text{if } b=0 \\ gcd(b, a\%b), & \text{else} \end{cases}$$

($a\%b < b$)

Correctness proof:

* In each iteration second argument decreases strictly. So, it will terminate.

To prove: $gcd(a,b) = gcd(b, a \bmod b)$

for all $a \geq 0, b > 0$.

Method: If left side divides, right side and right side divides left side, both are equal.

$d = gcd(a,b)$

$\Rightarrow$ d divides a $\rightarrow d|a$

d divides b $\rightarrow d|b$

$$a \bmod b = a - b \cdot \left\lfloor \frac{a}{b} \right\rfloor$$

$\Rightarrow d | a \bmod b$ and $| b$

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

Time complexity:

Lame's Theorem – Establish a cn b/w fibonacci and Euclid's theorem.

if $a > b \geq 1$, and $b < F_n$, gcd perform $n-2$ calls (recursive).

Worst case: 2 consecutive fibonacci numbers

→ Binary Gcd:

* optimization to normal gcd.
* Slow part in normal gcd are modulo.
* modulo operations (looks like O(1)), but are slower than add", sub", bitwise operation. Refer to arnold.

'Avoid Modulo Ops'

$gcd(2a, 2b) = gcd(a,b) * 2$ — both even

$gcd(2a, b) = gcd(a,b)$ — one even

$gcd(a,b) = gcd(b, a-b)$ — No even

* in built gcd's are already optimized.

$\Rightarrow$ `__builtin_ctz(unsigned int);`

↑ count trailing zeroes.

`__builtin_clz(unsigned)`

↑ count leading zeroes.

`__builtin_clzll((long long));`

Extended Euclid Algo:

* Represent gcd interms of a and b.

$$a \cdot x + b \cdot y = gcd(a,b) \quad -(1)$$

Ex: $gcd(55,80) = 5$

$$5 = 55 \times 3 + 80(-2) \checkmark$$

Now, I wanna see, how will values of (x,y) will change, if I keep $(b, a \bmod b)$ in place of (a,b)

Let x_1, y_1 be points satisfy.

$$b \cdot x_1 + (a \bmod b) y_1 = g$$

$$bx_1 + \left(a - b\left\lfloor \frac{a}{b} \right\rfloor\right) y_1 = g$$

on rearrag.

$$a(y_1) + b\left(x_1 - \left\lfloor \frac{a}{b} \right\rfloor y_1\right) = g$$

on comparing

$$\boxed{\begin{aligned} x &= y_1 \\ y &= x_1 - \left\lfloor \frac{a}{b} \right\rfloor y_1 \end{aligned}}$$

Note: Extended Euclid Algo, not only gives gcd of (a,b), but also returns x and y s.t

$$\boxed{ax + by = gcd(a,b)}$$

$\Rightarrow$ for $x_1, y_1 \Rightarrow$ we can recursively go in a s.t.n s.t until

$$a \cdot x_1 + b y_1 = gcd(a,b)$$

$\hookrightarrow \Rightarrow a = gcd(a,b), \; b = 0$

$\Rightarrow x_1 = 1$ and $y_1 = 0$

Code

Euclid

```
int gcd ( int a, int b) {
        if ( b == 0 ) return a;
        return (b, b%a);
}
```

However use __gcd(a,b) for practical purpose.

Extended Euclid

```
int ex_gcd (int a, int b, int &x, int &y) {
   if ( b == 0 ) {
            x = 1, y = 0;
            return a;
      }
        int x1, y1;
        int d = ex_gcd(b, a%b, x1, y1);
        x = y1;
        y = x1 - y1*(a/b);
        return d;
}
```