GCD: iterative algorithms

Recall the definition of GCD(a,b). Recall also the high-school technique for computing GCD(a,b).

Key observation: if (a>b) GCD(a,b) = GCD(a – b, b)

How do you prove this?

Any divisor of a,b divides a-b!

Try the new idea

Input: <a,b> = <64,44>
Output: GCD(a,b) = 4

GCD(a,b) = GCD(a-b,b)

GCD(64,44) = GCD(20,44)

GCD(20,44) = GCD(44,20)

GCD(44,20) = GCD(24,20)

GCD(24,20) = GCD(4,20)

GCD(4,20) = GCD(20,4)

GCD(20,4) = GCD(16,4)

GCD(16,4) = GCD(12,4)

GCD(12,4) = GCD(8,4)

GCD(8,4) = GCD(4,4)

GCD(4,4) = GCD(0,4)

What is the running time?

Running time for GCD(a,b)

Input: $<a,b> \;= <9999999999999,2>$

$<x,y> \;= <9999999999999,2>$

$= <9999999999997,2>$

$= <9999999999995,2>$

$= <9999999999993,2>$

$= <9999999999991,2>$

Time $= O(a) \;= 2^{O(n)}$

Size $= n = O(\log(a))$

A faster algorithm for GCD(a,b)

$<x,y> \Rightarrow <x-y,y>$

$\Rightarrow <x-2y,y>$

$\Rightarrow <x-3y,y>$

$\Rightarrow <x-4y,y>$

$\Rightarrow <x-iy,y>$

$\Rightarrow <x \text{ rem } y,y>$

$= <x \bmod y,y>$

$\Rightarrow <y,x \bmod y>$

But $x \bmod y < y$

Try the improvement

GCD(a,b) = GCD(b,a mod b)

Input: <a,b> = <44,64>

<x,y> = <44,64>

= <64,44>

= <44,20>

= <20, 4>

= < 4, 0>

GCD(a,b) = 4

A bad example

Input: <a,b> = <10000000000001,9999999999999>

<x,y> = <10000000000001,9999999999999>

= <9999999999999,2>

= <2,1>

= <1,0>

GCD(a,b) = GCD(x,y) = 1

Every two iterations:

the value x decreases by at least a factor of 2.

the size of x decreases by at least one bit.

Running time: O(log(a)+log(b)) = O(n)

GCD(a,b)

algorithm $GCD(a,b)$

$\langle$***pre−cond***$\rangle$**:** a and b are integers.

$\langle$***post−cond***$\rangle$**:** Returns $GCD(a,b)$.

begin
    int x,y
    $x = a$
    $y = b$
    loop
        $\langle$***loop−invariant***$\rangle$**:** GCD(x,y) = GCD(a,b).
        if($y = 0$) exit
        $x_{new} = y \quad y_{new} = x \bmod y$
        $x = x_{new}$
        $y = y_{new}$
    end loop
    return(x)
end algorithm