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: \(\text{GCD}(a, b) = 4\)

\[
\begin{align*}
\text{GCD}(a, b) &= \text{GCD}(a-b, b) \\
\text{GCD}(64, 44) &= \text{GCD}(20, 44) \\
\text{GCD}(20, 44) &= \text{GCD}(44, 20) \\
\text{GCD}(44, 20) &= \text{GCD}(24, 20) \\
\text{GCD}(24, 20) &= \text{GCD}(4, 20) \\
\text{GCD}(4, 20) &= \text{GCD}(20, 4) \\
\text{GCD}(20, 4) &= \text{GCD}(16, 4) \\
\text{GCD}(16, 4) &= \text{GCD}(12, 4) \\
\text{GCD}(12, 4) &= \text{GCD}(8, 4) \\
\text{GCD}(8, 4) &= \text{GCD}(4, 4) \\
\text{GCD}(4, 4) &= \text{GCD}(0, 4)
\end{align*}
\]

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)

\[ \langle x, y \rangle \Rightarrow \langle x - y, y \rangle \]
\[ \Rightarrow \langle x - 2y, y \rangle \]
\[ \Rightarrow \langle x - 3y, y \rangle \]
\[ \Rightarrow \langle x - 4y, y \rangle \]
\[ \Rightarrow \langle x - iy, y \rangle \]
\[ \Rightarrow \langle x \mod y, y \rangle \]
\[ = \langle x \mod y, y \rangle \]
\[ \Rightarrow \langle y, x \mod y \rangle \]

But \( x \mod y < y \)
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: \((a, b) = (10000000000001, 9999999999999)\)
\((x, y) = (10000000000001, 9999999999999)\)
\(= (9999999999999, 2)\)
\(= (2, 1)\)
\(= (1, 0)\)

\(\text{GCD}(a,b) = \text{GCD}(x,y) = 1\)

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: \(O(\log(a) + \log(b)) = O(n)\)
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
        $x_{new} = y$  $y_{new} = x \mod y$
        $x = x_{new}$
        $y = y_{new}$
    end loop
    return($x$)
end algorithm