The GCD Algorithm
Algorithms

GCD Algorithm

```
Algorithm GCD(m,n)
  Input nonnegative integers m, n, not both 0
  Output gcd(m,n)
  if n=0 then return m
  else
     return GCD(n, m mod n)
```

Observations

1. If `m = 0` and `n > 0`, then the first self-call of the algorithm has the effect of switching m and n. Generally, if `m < n`, the first step of the algorithm causes the first argument to be the larger one.
2. The algorithm works even if `m < 0` but not when `n < 0` because "`mod n`" is not defined for negative values of n. Because $\gcd(m,n) = \gcd(-m,n) = \gcd(m,-n) = \gcd(-m,-n)$, usually the GCD algorithm is formulated in this way, restricting m and n to be nonnegative. With this restriction, in a Java implementation, "mod" can be replaced by "`%`". Adding only a bit of O(1) overhead allows us to compute the gcd of any pair of integers (not both 0):

```
Algorithm GeneralGCD(m,n)
  Input integers m, n, not both 0
  Output gcd(m,n)
  m1 ← (m < 0 ?  -m :  m)
  n1 ← (n < 0 ?  -n :  n)
  return GCD(m1, n1)
```

Correctness of the GCD Algorithm

Correctness of the algorithm is established by using the following lemma:

Lemma 1 Whenever m, n are integers with $m \geq n \geq 1$, we have

$$\gcd(m,n) = \gcd(n, m \bmod n)$$

For the proof, one shows that the pair m, n have exactly the same divisors as the pair $n, m \bmod n$, using the fact that, whenever $m \geq n \geq 1$,

$$m = n \cdot \lfloor \frac{m}{n} \rfloor + m \bmod n.$$

Running time of GCD

Clearly, the running time of GCD is proportional to the number of self-calls of GCD. The following represents a sequence of GCD self-calls:

```
GCD(m,n)
GCD(n,m % n)  let r0 = m % n
GCD(r0,n % r0)  let r1 = n % r0
GCD(r1, r0 % r1)
  •
  •
  •
GCD(rk,0)=r
```

The first argument inputs are reduced by more than half in *every other* self-call. That is, although it is NOT true that passing from `GCD(a,b)` to `GCD(b, a % b)`, cuts the value of the first argument by $\frac{1}{2}$, it *is* true that the first argument of the next self call `GCD(a % b, r)`, namely a % b, is more than twice as small as `a`. This observation also applies to the *second* argument of successive calls to `GCD`. The conclusion is that the sequence of self-calls can be no longer than twice the length of descending sequences obtained by repeated cutting input size in half — i.e.

$$\text{running time of } \texttt{GCD(m,n)} \leq 2(2 + \log n) \text{ in } \mathbf{O}(\log n).$$

This observation is based on the following lemma:

Lemma 2. Whenever $m \geq n \geq 2$ are integers,

$$m\%n < \frac{m}{2}.$$

This is shown by considering two cases: When $n > m/2$, use the fact that $m\%n = m - n$. When $n \leq m/2$, notice that $m\%n < n \leq m/2$.

Since only a few extra steps are needed to run `GeneralGCD(m,n)`, we also may conclude that

$$\text{running time of } \texttt{GeneralGCD(m,n)} \text{ is } \mathbf{O}(\log n).$$