NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 1, Lecture 3

Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Algorithm for `gcd(m,n)`

- To find the largest common factor, start at the end and work backwards
- Let `i` run from `min(m,n)` to 1
- First common factor that we find will be gcd!

Euclid's algorithm

- Suppose `d` divides both `m` and `n`, and `m > n`
- Then `m = ad`, `n = bd`
- So `m-n = ad - bd = (a-b)d`
- `d` divides `m-n` as well!
- So `gcd(m,n)` = `gcd(n,m-n)`

Euclid's algorithm

- Consider `gcd(m,n)` with $m > n$
- If `n` divides `m`, return `n`
- Otherwise, compute `gcd(n,m-n)` and return that value

Euclid’s algorithm

```
def gcd(m,n):
  # Assume m >= n
  if m < n:
    (m,n) = (n,m)

  if (m%n) == 0:
    return(n)
  else:
    diff = m-n
    # diff > n? Possible!
    return(gcd(max(n,diff),min(n,diff))
```

Euclid’s algorithm, again

```
def gcd(m,n):

  if m < n:  # Assume m >= n
    (m,n) = (n,m)

  while (m%n) != 0:
    diff = m-n
    # diff > n? Possible!
    (m,n) = (max(n,diff),min(n,diff))

  return(n)
```

Even better

- Suppose n does not divide m
- Then $m = qn + r$, where q is the quotient, r is the remainder when we divide m by n
- Assume d divides both m and n
- Then $m = ad$, $n = bd$
- So $ad = q(bd) + r$
- It follows that $r = cd$, so d divides r as well

Euclid’s algorithm

- Consider `gcd(m,n)` with $m > n$
- If `n` divides `m`, return `n`
- Otherwise, let `r = m%n`
- Return `gcd(n,r)`

Euclid’s algorithm

```
def gcd(m,n):

  if m < n:  # Assume m >= n
    (m,n) = (n,m)

  if (m%n) == 0:
    return(n)
  else:
    return(gcd(n,m%n))  # m%n < n, always!
```

Euclid’s algorithm, revisited

```
def gcd(m,n):

  if m < n:  # Assume m >= n
    (m,n) = (n,m)

  while (m%n) != 0:
    (m,n) = (n,m%n)  # m%n < n, always!

  return(n)
```

Efficiency

- Can show that the second version of Euclid's algorithm takes time proportional to the number of digits in `m`

- If `m` is 1 billion (10^9), the naive algorithm takes billions of steps, but this algorithm takes tens of steps