Lecture 2

Euclid's Theorem: There are infinitely many prime numbers.

Proof: Assume by contradiction that $P_1, \dots, P_c$ are the only primes.

Consider: $N := p_1 p_2 \cdots p_t + 1$

It is not a prime by assumption, so it must be a product of primes. But none of the primes $p_1, \dots, p_t$ divides N, a contradiction.

Euclid's division lemma:

$\forall a \in \mathbb{N}^*, \exists!\, q, r \in \mathbb{Z} \ni \ a = qb + r$ and $0 \leq r < b$

Notation: $q = \lfloor a/n \rfloor$ quotient

$r = a \bmod n$ remainder

Ex. $a = 7$ $b = 3$ $q = 2$ $r = 1$ $7 = 2 \times 3 + 1$

Proof for the case $a \geq 0$, $b > 0$:

Start with $q = 0$ and $r = a$.

while $r \geq b$:

$r = r - b$

$q = q + 1$

assert $a == qb + r$

We terminate when $0 \leq r < b$

Ex. $a = 37$, $b = 7$

$(q, r) = (0, 37) \to (1, 30) \to (2, 23) \to (3, 16) \to (4, 9) \to (5, 2)$

$37 = 5 \times 7 + 2$

Uniqueness: Assume $a = q_1 b + r_1 = q_2 b + r_2$

$\rightsquigarrow (q_1 - q_2) b = r_2 - r_1$

Since $-b < r_2 - r_1 < b$, we deduce ...

3. Euclid's Algorithm

The greatest common divisor $\gcd(a,b)$ of two natural numbers a and b is the largest natural number dividing both a and b.

$\gcd(0,0) := 0$

We can calculate gcd via prime factorization

However prime factorization is hard.

More efficient algorithm for computing gcd?

Observation: $\forall a \in \mathbb{N}, b \in \mathbb{N}^*$, $\gcd(a,b) = \gcd(b, a \bmod b)$

Proof: Write $r = a - qb$. Then every common divisor of a and b also divides r.
Write $a = qb + r$. Then every common divisor of b and r also divides a. Thus the set of common divisors of a and b is equal to the set of common divisors of b and r.
Hence $\gcd(a,b) = \gcd(b,r)$

Euclid's algorithm for gcd

```
def gcd(a,b):
    if b==0:
        return a
    else:
        return gcd(b, a mod b)
```

Ex: $\gcd(30,21) = \gcd(21,9) = \gcd(9,3) = \gcd(3,0) = 3$

Complexity analysis (not required)

Fibonacci numbers: $F_0 = 0$, $F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$ for $n \geq 0$

$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 \ldots$

Lamé's Theorem: If $a > b \geq 1$ and $b < F_{k+1}$, then Euclid's algorithm makes fewer than k recursive calls.

Since Fibonacci numbers grow exponentially, the number of recursive calls is $O(\log b)$.

Now, let's extract more information from Euclid's algorithm.

Theorem: For $a, b \in \mathbb{N}$, $\exists x, y \in \mathbb{Z} \ni \gcd(a,b) = ax + by$

Proof by the following extended Euclid's algorithm:

```
def extended_gcd(a, b):
    if b == 0:
        return (a, 1, 0)
    else:
        (d1, x1, y1) = extended_gcd(b, a % b)
        assert d1 == b * x1 + (a % b) * y1
        (d, x, y) = (d1, y1, x1 - (a // b) * y1)
        assert d == a * x + b * y
        return (d, x, y)
```

Corollary: $\forall\, a, b \in \mathbb{Z}$, if $b \mid a$