CSC110 Lecture 16: Greatest Common Divisor

Hisbaan Noorani

October 20, 2020

Contents

1 Ex 1: Property of the greatest common divisor

For your reference, here is the definition of greatest common divisor.

> Let $x, y, d \in \mathbb{Z}$. We say that d is a **common divisor** of x and y when d divides x and d divides y.
> We say that d is the **greatest common divisor** of x and y when it the largest number that is a common divisor of x and y, or 0 when x and y are both 0.
>
> We can define the function $\gcd : \mathbb{Z} \times \mathbb{Z} \to \mathbb{N}$ as the function which takes numbers x and y, and returns their greatest common divisor.

And here is the Quotient-Remainder Theorem, slightly modified to handle negative integers.

> *Quotient-Remainder Theorem.* Let $n \in \mathbb{Z}$ and $d \in \mathbb{Z}$. If $d \neq 0$ then there exist $q \in \mathbb{Z}$ and $r \in \mathbb{N}$ such that $n = qd + r$ and $0 \leq r < d$. Moreover, these q and r are *unique* for a given n and d.
>
> We say that q is the **quotient** when n is divided by d, and that r is the **remainder** when n is divided by d. We use $n \ \% \ d$ to denote the remainder of n divided by d.

And finally, here is the Divisibility of Linear Combinations Theorem that we covered in lecture today.

> *Divisibility of Linear Combinations.* Let $m, n, d \in \mathbb{Z}$. If d divides m and d divides n, then d divides every linear combination of m and n.

1. Prove the following statement: For all $a, b, c \in \mathbb{Z}$, $b \neq 0$ and $c \mid a$ and $c \mid b$, then $c \mid a \ \% \ b$.

   $\forall a, b, c \in \mathbb{Z}, b \neq 0 \land c \mid a \land c \mid b \implies c \mid a \ \% \ b$

   Tip: introduce variables q and r using the Quotient-Remainder theorem in your proof.

   Let $a, b, c \in \mathbb{Z}$

   Assume $b \neq 0, \ c \mid a, \ c \mid b$

   By QRT, $\exists q \in \mathbb{Z}, a = qb + r \implies r = a - qb$

   r is a linear combination of a and b

   Since $c \mid a$ and $c \mid b$, then by DLC, $c \mid r$ which means $c|a \ \% \ b$ $\blacksquare$

2. Prove the following statement: For all $a, b \in \mathbb{Z}$, if $b \neq 0$ then $\gcd(b, a \ \% \ b) \mid a$.

   $\forall a, b \in \mathbb{Z}, b \neq 0 \implies \gcd(b, a \ \% \ b) \mid a$

   Tip: introduce variables q and r using the Quotient-Remainder theorem in your proof.

   Let $a, b \in \mathbb{Z}$

   Assume $b \neq 0$

   Let $r = a \ \% \ b$, $d = \gcd(b, r)$

   By QRT, $\exists q \in \mathbb{Z}, a = qb + r$

   Since $d = \gcd(b, r)$, then by definition, $d \mid b \wedge d \mid r$

   Since $d \mid b \wedge d \mid r$, then by DLC $d \mid qb + r$

   Therefore $d \mid a$ as needed. ■

3. Suppose we have three numbers $a, b, c \in \mathbb{Z}$, and we know that $b \neq 0$. If we want to prove that $c = \gcd(a, b)$, what would we need to prove, using the above definition of gcd?

   (a) $c \mid a$
   (b) $c \mid b$
   (c) $\forall e \in \mathbb{N}, \ (e \mid a \wedge e \mid b) \implies e \leq c$

2 Ex 2: The Extended Euclidean Algorithm

We left off our discussion of the *Extended Euclidean Algorithm* in lecture with the following code:

```python
def gcd_extended(a: int, b: int) -> Tuple[int, int, int]:
    """Return the gcd of a and b, and integers p and q such that

    gcd(a, b) == p * a + b * q.

    >>> gcd_extended(10, 3)
    (1, 1, -3)
    """
    x, y = a, b

    # Initialize px, qx, py, and qy
    px, qx = 1, 0
    py, qy = 0, 1

    while y != 0:
        # Loop invariants
        assert math.gcd(a, b) == math.gcd(x, y)  # L.I. 1
        assert x == px * a + qx * b              # L.I. 2
        assert y == py * a + qy * b              # L.I. 3

        q, r = divmod(x, y)  # quotient and remainder when a is divided by b

        # Update x and y
        x, y = y, r

        # Update px, qx, py, and qy
        px, qx, py, qy = py, qy, px - q * py, qx -q * qy
```

```python

    return (x, px, qx)
```

1. Recall that the loop invariants must hold for the initial values of the loop variables. Given this, how should we initialize `px`, `qx`, `py`, and `qy`? (*Hint*: use the simplest possible numbers.)

   Write your answer below, or directly in the `gcd_extended` code.

```python
px, qx = 1, 0
py, qy = 0, 1
```

2. Inside the loop body, we need to figure out how to update the new variables `px`, `qx`, `py`, and `qy`. Now in order to derive the updates for `px`, `qx`, `py`, and `qy`, we need to do a bit of math.

   For an arbitrary iteration of the while loop:

   - *Let*x_0, y_0, px_0, qx_0, px_0, py_0 be the values of the variables `x`, `y`, `px`, `qx`, `py`, `qx`, and `qy` at the *start* of the iteration.
   - x_1, y_1, px_1, qx_1, py_1, qy_1 be the values of these variables at the *end* of the iteration.
   - q and r be the values of the variables `q` and `r` during the iteration.

   (a) What do the loop invariants 2 and 3 tell you about the relationships between x_0, y_0, px_0, qx_0, py_0, qy_0? (This is what we can assume to be True at the start of the loop iteration.)
   $x_0 = (px_0)(a) + (qx_0)(b)$
   $y_0 = (py_0)(a) + (qy_0)(b)$

   (b) What do the loop invariants tell you about the desired relationships between x_1, y_1, px_1, qx_1, py_1, qy_1? (This is what we want to be True at the end of the loop iteration.)
   $x_1 = (px_1)(a) + (qx_1)(b)$
   $y_1 = (py_1)(a) + (qy_1)(b)$

   (c) From the Quotient-Remainder Theorem, what is the relationship between x_0, y_0, q, and r?
   $x_0 = q \cdot y_0 + r$

   (d) What are the values of x_1 and y_1 in terms of x_0, y_0, q, and/or r?
   $x_1 = y_0$
   $y_1 = r$

   (e) What should px_1 and qx_1 equal to satisfy the invariant for x_1, using only the "0" variables?
   $px_1 = py_1$
   $qx_1 = qy_1$

   (f) (This is the hardest part) Given your answers from earlier parts, what should py_1 and qy_1 equal in order to satisfy the invariant for y_1, using only the "0" variables?
   Hint: You'll need to do a calculation to start with your invariant in (b) and replace all of the "1" variables with "0" variables.
   $py_1 = px_0 - q \cdot py_0$
   $qy_1 = qx_0 - p \cdot qy_0$

3. Using your answers to 2(e) and 2(f), complete the code for the Euclidean algorithm to update the variables `px`, `qx`, `py`, and `qy` inside the loop body. You should then be able to run the code with all of the loop invariants passing.

```python
px, qx, py, qy = py, qy, px - q * py, qx -q * qy
```

Congratulations, you've just completed a derivation of the Extended Euclidean Algorithm!