CSE214 Data Structures

Euclidean Algorithm

YoungMin Kwon

Euclidean Algorithm

- Subtraction based
  - Let g = gcd(a, b) and a > b, then gcd(a, b) = gcd(a – b, b)
    - a – b = g · a’ – g · b’ = g · (a’ – b’)

    -
      ```
        gcd(a - b          , b)
      = gcd(g · (a’ – b’) , g · b’)
      = g
      = gcd(a, b)
      ```
  - Example

    -
      ```
        gcd(24          , 15)
      = gcd(24 – 15 = 9 , 15)
      = gcd(15 – 9  = 6 , 9)
      = gcd(9  – 6  = 3 , 6)
      = gcd(6  – 3  = 3 , 3)
      = 3
      ```

Euclidean Algorithm

- Modulo based
  - Let g = gcd(a, b) and a > b, then gcd(a, b) = gcd(a % b, b)
    - $a = q \cdot b + r$
      $\;\; = q \cdot g \cdot b' + g \cdot r'$ ← because g divides a
    - gcd(a, b) = gcd(a % b, b) ← because g divides r

  - Example

    -
      ```
        gcd(39          , 15)
      = gcd(39 % 15 = 9 , 15)
      = gcd(15 % 9  = 6 , 9)
      = gcd(9  % 6  = 3 , 6)
      = gcd(6  % 3  = 0 , 3)
      = 3
      ```

```java
//
// 0. Download Euclidean.java
// 1. Implement GCDBySub class
// 2. Implement GCDByMod class
//

public class Euclidean_Sol {
    public static interface GCD {
        public int gcdImpl(int a, int b);

        //default methods can have implementations
        public default int gcd(int a, int b) {
            return gcdImpl(abs(a), abs(b));
        }
        public default int abs(int a) {
            return a < 0 ? -a : a;
        }
    }
    //TODO: implement GCD using subtraction
    public static class GCDBySub implements GCD {
    }

    //TODO: implement GCD using modulo
    //public static class GCDByMod implements GCD {
    //}

    public static void test(GCD gcd, int a, int b) {
        System.out.println("GCD: " + gcd.gcd(a, b));
    }
    public static void main(String[] args) {
        final int a = 39;
        final int b = -15;

        test(new GCDBySub(), a, b);
        //test(new GCDByMod(), a, b);
    }
}
```