GCD Algorithm Report

Implement Euclidean GCD Algorithm by Python 3 & C.

University of Ulsan IT Convergence

20152262 Hong Geun Ji

Euclidean GCD algorithm is fastest way to find greatest common divisor. There are many samples about calculating GCD using PL(Programming Language), but most of them really do not use this algorithm. As our professor's assignment requires, this report will introduce about Calculating GCD using Euclidean GCD algorithm.

This algorithm has been introduced in Discrete math Number theory. The basic concept is as follows.

- $a = bq + r, \quad 0 \le r < b$
- Let c be a common divisor of a and b
  $\Rightarrow c \mid bq$
  $\Rightarrow c \mid a$ and $c \mid bq$
  $\Rightarrow c \mid (a-bq)\ (=r)$
  $\Rightarrow c$ is a common divisor of b and r
- If c is a common divisor of b and r
  $\Rightarrow c \mid bq$ and $c \mid bq + r\ (=a)$
  $\Rightarrow c$ is a common divisor of a and b

$\Rightarrow \gcd(a, b) = \gcd(b, r)$

By using this concept, we can write pseudo code about this algorithm. And we also can make Python and C program by using the pseudo code. The assignment requires just implement by using Python or Scratch, but I think it is good to introduce not only Python 3 version but also C version. (There are some reasons why I do this.) So at first, I will introduce Python 3 version and then C version.

Pseudo code

```
/////////////////////////////////////////////////////////////////////

 if m < n, swap(m,n)
 while n does not equal 0
         r = m mod n
         m = n
         n = r
 end while
 output m

/////////////////////////////////////////////////////////////////////
```

Python 3 version

```
//////////////////////////////////////////////////////////////////////////
#
#  Euclid-Algo.py
#  An efficient algorithm for finding the greatest common divisor of two integers.
#
#  Created by Ji Hong Guen on 01/11/18.
#  Copyright © 2018 Ji Hong Geun. All rights reserved.
#

class Number:                  ## The reson why I use Number class is described on the report.
    def __init__(self, x):     ## In short, Python 3 does not support things like pointer.
        self.num = x


def euclideanGCD(x, y):        ## Return a value of two value's Greatest Common Divisor.
    if x.num < y.num:          ## For a convenience, if y is bigger, exchange them.
        swap(x, y)
    if not y.num:              ## The basis level of this recursion.
        return x.num
    tmp = Number(x.num % y.num)
    return euclideanGCD(y, tmp)        ## Do this recursively.


def swap(x, y):                ## This function exchanges original's value.
    tmp = x.num
    x.num = y.num
    y.num = tmp
```

```
def main():
    x = Number(int(input("Type the First Number : ")))
    y = Number(int(input("Type the Second Number : ")))
    print("The GCD : " + str(euclideanGCD(x, y)))

main()

//////////////////////////////////////////////////////////////////////////
```

As you can see, there is a code using "Number" class for produce Number object.
The reason why I use this class is that the Python does not support things like pointer.
So if we have to swap the original value of two integers, we have to use object as parameter so that the "swap" function can refer original value of two integers. (Python only support call by reference for array and object.) And I used recursive function to check time complex more easier.

C version

```
//////////////////////////////////////////////////////////////////////

//
//  Euclidean-Algo.c
//  An efficient algorithm for finding the greatest common divisor of two integers.
//
//  Created by Ji Hong Guen on 01/11/18.
//  Copyright © 2018 Ji Hong Geun. All rights reserved.
//

#include <stdio.h>

int euclideanGCD(int x, int y);     // Get a value of two value's Greatest Common Divisor.
void swap(int *x, int *y);          // This function exchanges original's value.
int tmp;                            // This will only used at once.

main()
{
  int x, y;

  printf("Type the First Number : ");
  scanf("%d", &x);
  printf("Type the Second Number : ");
  scanf("%d", &y);
  printf("The GCD : %d\n", euclideanGCD(x, y));

}
```

```
int euclideanGCD(int x, int y)
{
  if (x < y) swap(&x, &y);                    // For a convenience, if y is bigger, exchange them.
  if (!y) return x;                           // The basis level of this recursion.
  return euclideanGCD(y, x % y);              // Do this recursively.
}

void swap(int *x, int *y)
{
  tmp = *x;
  *x = *y;
  *y = tmp;
}
```

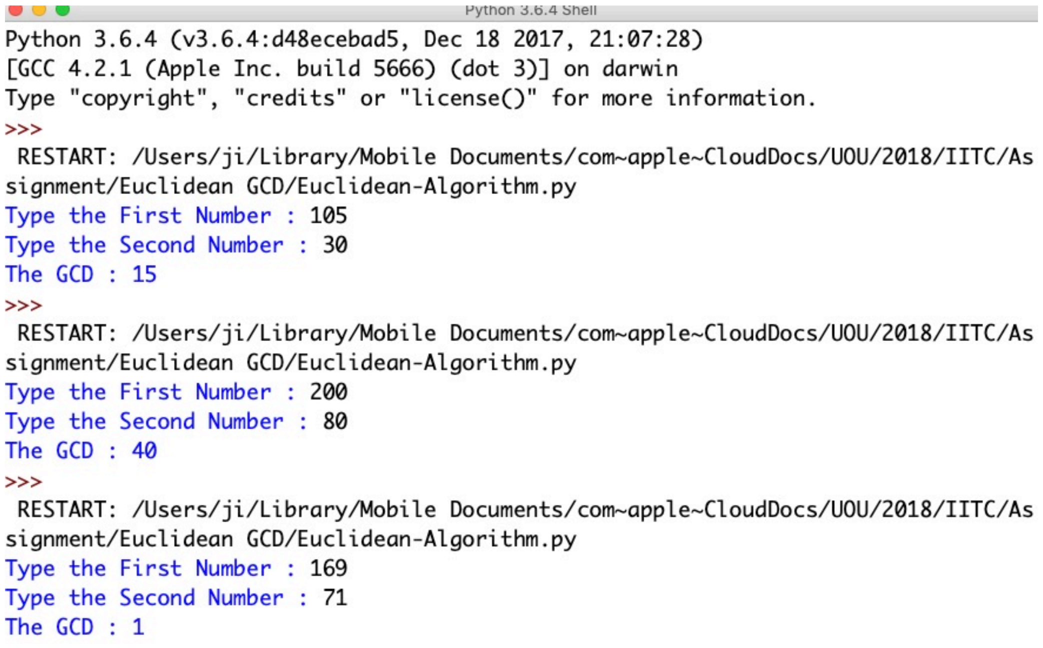
//

This is C version of Euclidean GCD Algorithm. As the same with Python 3 version, I used recursive function to check time complex more easily and clearly. There is a swap function using pointer as parameter. This will swap original value.
Maybe "swap" function will be called only just once since x is bigger than the (x % y).

As you can see, the C version of "euclideanGCD" function looks more simple and elegant than the Python 3 version. And exchanging two original values can be done by using pointer. So this would be great advantage of using C. Also I do not have to make temp value always while using "euclideanGCD" function since the result of modular operate can be used as parameter. (Python have to make object to use "call by reference".)

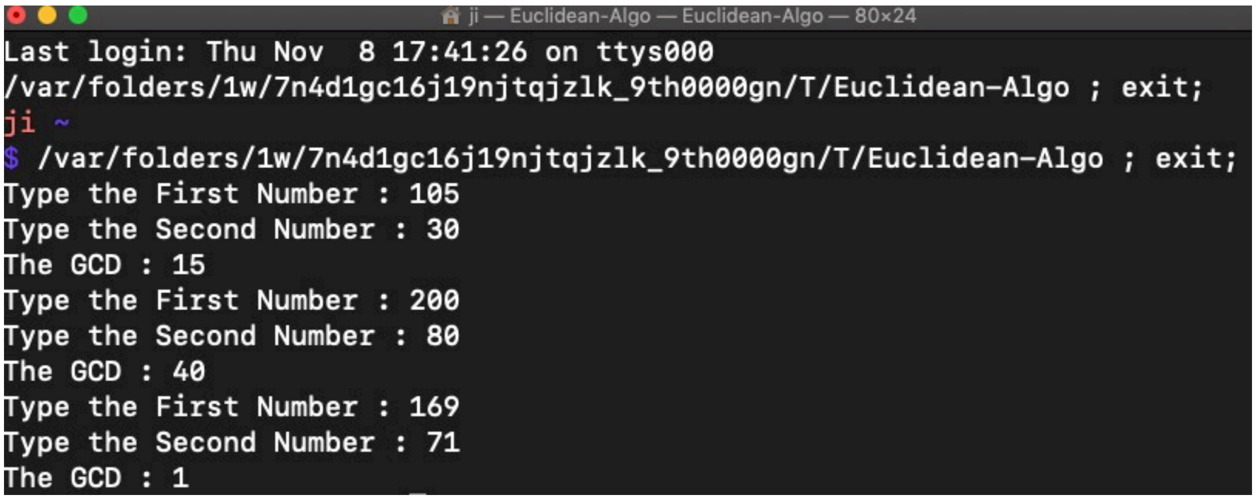
Result

The result of Python 3 version.



```
Python 3.6.4 (v3.6.4:d48ecebad5, Dec 18 2017, 21:07:28)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: /Users/ji/Library/Mobile Documents/com~apple~CloudDocs/UOU/2018/IITC/As
signment/Euclidean GCD/Euclidean-Algorithm.py
Type the First Number : 105
Type the Second Number : 30
The GCD : 15
>>>
 RESTART: /Users/ji/Library/Mobile Documents/com~apple~CloudDocs/UOU/2018/IITC/As
signment/Euclidean GCD/Euclidean-Algorithm.py
Type the First Number : 200
Type the Second Number : 80
The GCD : 40
>>>
 RESTART: /Users/ji/Library/Mobile Documents/com~apple~CloudDocs/UOU/2018/IITC/As
signment/Euclidean GCD/Euclidean-Algorithm.py
Type the First Number : 169
Type the Second Number : 71
The GCD : 1
```

The result of C version.



```
Last login: Thu Nov  8 17:41:26 on ttys000
/var/folders/1w/7n4d1gc16j19njtqjzlk_9th0000gn/T/Euclidean-Algo ; exit;
ji ~
$ /var/folders/1w/7n4d1gc16j19njtqjzlk_9th0000gn/T/Euclidean-Algo ; exit;
Type the First Number : 105
Type the Second Number : 30
The GCD : 15
Type the First Number : 200
Type the Second Number : 80
The GCD : 40
Type the First Number : 169
Type the Second Number : 71
The GCD : 1
```