Lesson 22: Iterative Algorithms #2 (W06D4)

Balboa High School

Michael Ferraro

September 24, 2015

Do Now

```
int i = 0;

while ( ??? ) {
    System.out.println("Hello!");
    i++;
}
```

Consider the code snippet above.

1. Write down the condition that should be used for the `while()` loop so that "Hello!" is printed 4 times.
2. What about the condition that causes "Hello!" to print 100 times?
3. Write a short program that runs the code and verify your answers to the first two questions.

Aim

Students will continue working with iterative algorithms for the *Sum of Squares* and *Euclid's GCF*[1] *method*.

[1] Greatest Common Factor

SumOfSquares from Last Class

- With every iteration of the `while()` loop, increase the base of the square to be added to the sum
- Stop looping once we enough terms are added
- Return the sum

```
public static int findSumOfSquares(int n) {
    int sum = 0; //accumulator
    int i = 1;   //term to be squared

    while ( i <= n ) {
        sum += i*i;
        i++;
    }

    return sum;
}
```

`SumOfSquares`: Take Two!

- $\sum_{a=1}^{n} a^2 = 1^2 + 2^2 + 3^2 + \ldots + (n-1)^2 + n^2$

`SumOfSquares`: Take Two!

- $\sum_{a=1}^{n} a^2 = 1^2 + 2^2 + 3^2 + \ldots + (n-1)^2 + n^2$
- What if we were to count *backward*?

`SumOfSquares`: Take Two!

- $\sum_{a=1}^{n} a^2 = 1^2 + 2^2 + 3^2 + \ldots + (n-1)^2 + n^2$
- What if we were to count *backward*?
- For example, if $n = 4$, return value of

  $$4^2 + 3^2 + 2^2 + 1^2$$

  instead of

  $$1^2 + 2^2 + 3^2 + 4^2$$

SumOfSquares: Take Two!

- $\sum_{a=1}^{n} a^2 = 1^2 + 2^2 + 3^2 + \ldots + (n-1)^2 + n^2$
- What if we were to count *backward*?
- For example, if $n = 4$, return value of

  $$4^2 + 3^2 + 2^2 + 1^2$$

  instead of

  $$1^2 + 2^2 + 3^2 + 4^2$$

- $\sum_{a=1}^{n} a^2 = n^2 + (n-1)^2 + \ldots + 3^2 + 2^2 + 1^2$

`SumOfSquares`: Take Two!

- $\sum_{a=1}^{n} a^2 = n^2 + (n-1)^2 + \ldots + 3^2 + 2^2 + 1^2$
- With a partner — **on paper** — figure out how to write a `while()` loop to accomplish this task.

  Hints:
  - as before, have an *accumulator* variable like `sum`
  - consider what *changes* with every term
  - **then** figure out the stopping condition
- Once you and your partner agree on a solution — on paper — implement the algorithm in Java.

SumOfSquares, Backward

```
public static int findSumOfSquares(int n) {
    int sum = 0;

    while ( n > 0 ) {
        sum += n*n;
        n--;
    }

    return sum;
}
```

Which implementation of `findSumOfSquares()` do you think is better — the one that counts up or the new one that counts down? Which is easier to write? Which is easier to understand?

Euclid's GCF Method

- What is the GCF of two numbers? Consider how you reduce fractions!

Euclid's GCF Method

- What is the GCF of two numbers? Consider how you reduce fractions!

$$\frac{18}{6} = \frac{18 \div \mathbf{6}}{6 \div \mathbf{6}} = \frac{3}{1} = 3$$

Euclid's GCF Method

- What is the GCF of two numbers? Consider how you reduce fractions!

$$\frac{18}{6} = \frac{18 \div \mathbf{6}}{6 \div \mathbf{6}} = \frac{3}{1} = 3$$

$\therefore$ `GCF(18, 6) =` $\mathbf{6}$

Euclid's GCF Method

- What is the GCF of two numbers? Consider how you reduce fractions!

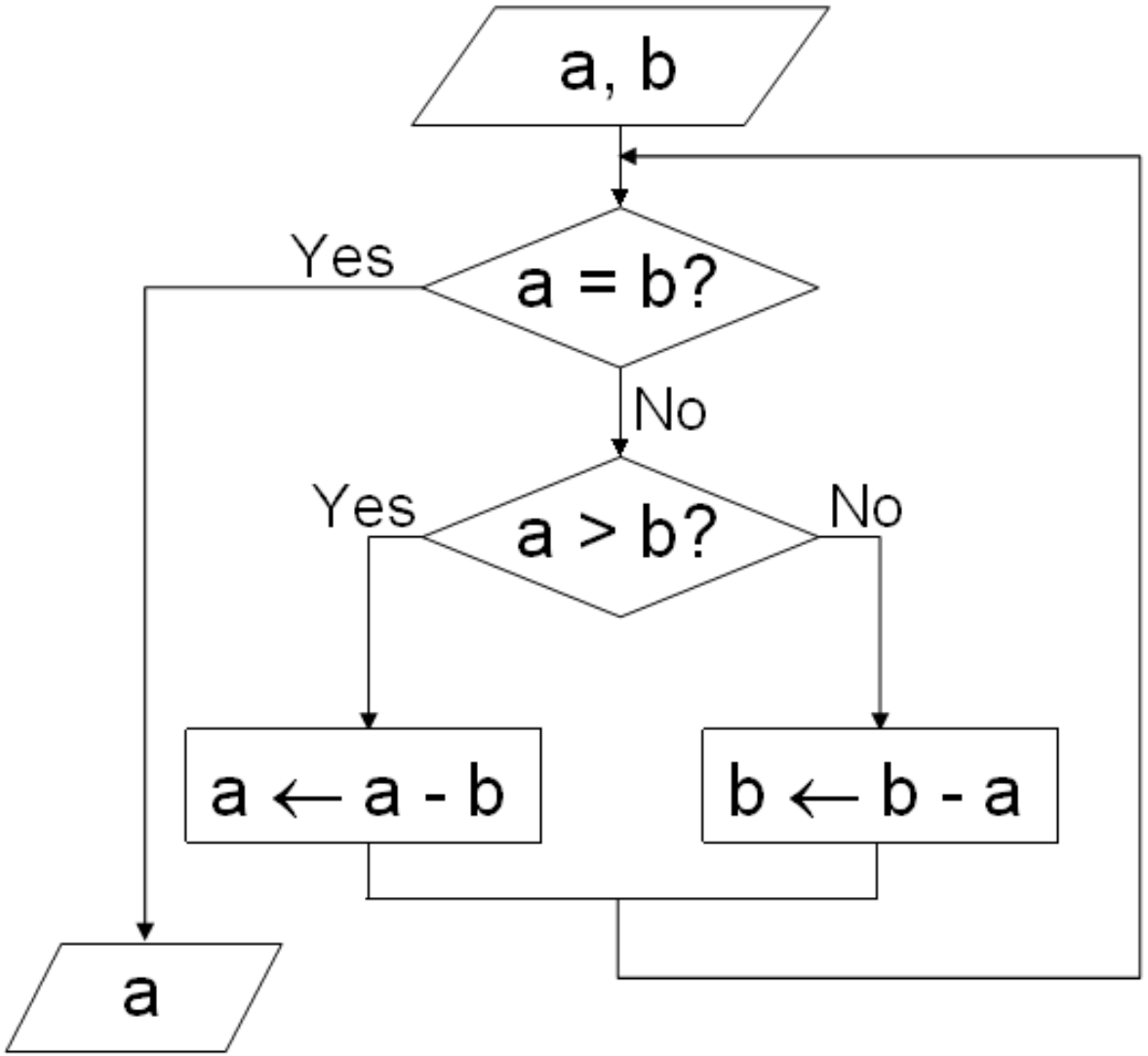
  $\frac{18}{6} = \frac{18 \div \mathbf{6}}{6 \div \mathbf{6}} = \frac{3}{1} = 3$

  $\therefore$ `GCF(18, 6) =` **6**

- Euclid had a method (algorithm) for finding the GCF of two positive integers (you copied the flowchart for this algorithm into PS #4a for HW)

Euclid's GCF Method

Consider `GCF(18,6)`, tracking *variable state*:

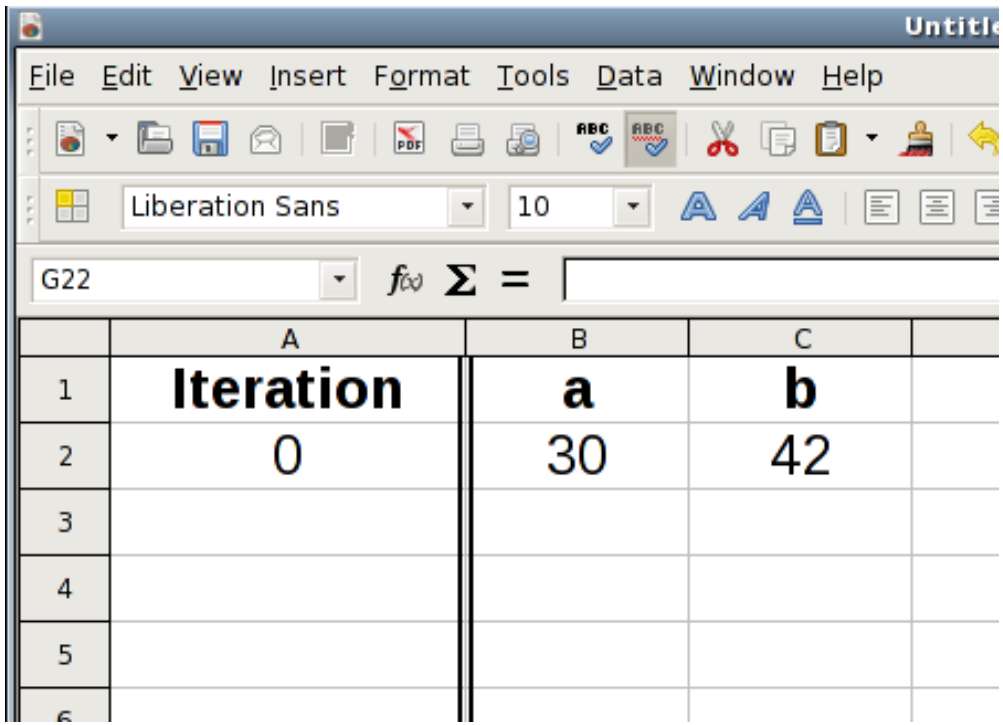


2

[2] chart from Litvin's Ch04 PPT

Tracing the Algorithm

Guided Example: Finding `GCF(30, 42)`, tracking *variable state* in a spreadsheet...

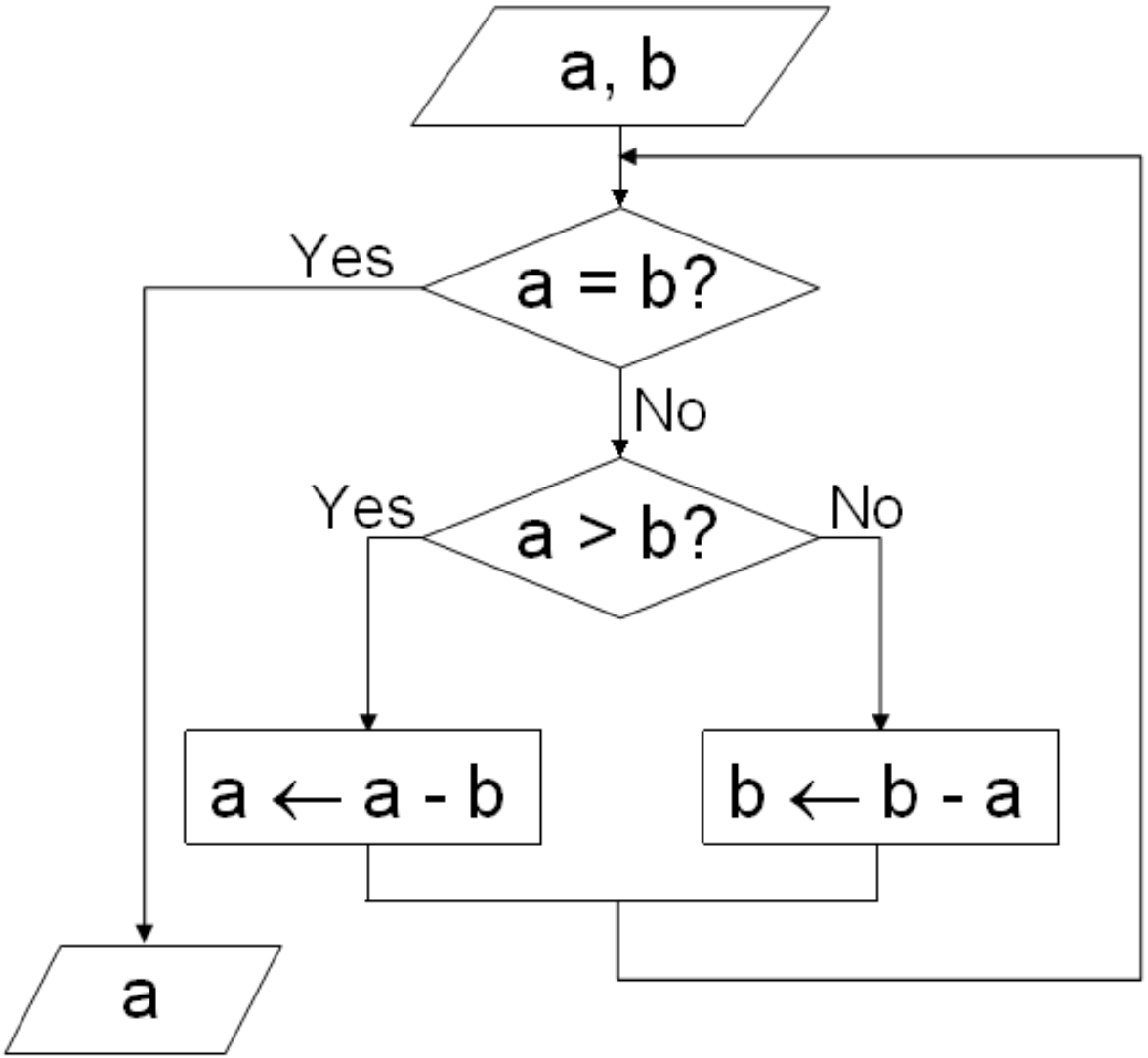


Practice Using the Algorithm

Find the GCF of the following pairs, using a spreadsheet to track the number of iterations:

- $\{63, 14\}$
- $\{16, 128\}$
- $\{55, 7\}$

Implementing the GCF Algorithm



Implementing the GCF Algorithm

Let's look at a Scratch implementation of Euclid's GCF algorithm:

- Download `euclid-GCF.sb`
- Upload the Scratch file to cloud-based Scratch

Implementing the GCF Algorithm

- "$a = b$?" decision leads to a *loop*; can be thought of as "`while` $a \neq b$, `do` `...`"

Implementing the GCF Algorithm

- "$a = b$?" decision leads to a *loop*; can be thought of as "`while` $a \neq b$, `do ...`"
- In Java, $\neq$ is expressed as `!=`

```
public static int gcf(int a, int b) {
    while ( a != b ) {
        ...
    }
    return a;
}
```

Implementing the GCF Algorithm

- "$a = b$?" decision leads to a *loop*; can be thought of as "`while` $a \neq b$, `do ...`"
- In Java, $\neq$ is expressed as `!=`

```
public static int gcf(int a, int b) {
    while ( a != b ) {
        ...
    }
    return a;
}
```

$\rightarrow$ what logic needs to be implemented in the `while()` loop?

Implementing the GCF Algorithm

The "$a > b$?" decision point is missing.

- Think of as:

  "`if` $a > b$, `then ...; otherwise, ...`"

Implementing the GCF Algorithm

The "$a > b$?" decision point is missing.

- Think of as:

  "`if` $a > b$, `then ...; otherwise, ...`"

- Enter the `if()` statement!

Implementing the GCF Algorithm

```
public static int gcf(int a, int b) {
    while ( a != b ) {

        if ( a > b ) {
            a = a - b;
        }



    }
    return a;
}
```

Implementing the GCF Algorithm

```
public static int gcf(int a, int b) {
    while ( a != b ) {

        if ( a > b ) {
            a = a - b;
        } else {
            b = b - a;
        }

    }
    return a;
}
```

Implementing the GCF Algorithm

```
public static int gcf(int a, int b) {
    while ( a != b ) {

        if ( a > b ) {
            a -= b;
        } else {
            b -= a;
        }

    }
    return a;
}
```

For rest of period. . .

- Write a program to find the GCF of the pairs you worked with earlier:
  - $\{63, 14\}$
  - $\{16, 128\}$
  - $\{55, 7\}$
- Work on PS #4a, §4

HW

Finish §4 of PS #4a.