Lesson 20: Intro to Algorithms (W06D2)

Balboa High School

Michael Ferraro

September 22, 2015

Do Now

1. On a sheet of paper, divide 16289 by 7 using *long division*.
2. You may be called upon to guide your teacher — who is a math teacher! — through the same task. To prepare, think of the *reasons* why you did everything you did when evaluating $16289 \div 7$.

Sign-off opportunities for PS #3's paper form:

- later this period
- Homeroom tomorrow
- immediately after 6th Period tomorrow

Aim

Students will be introduced to algorithms, pseudocode, flowcharts, and basic iteration using `while()`.

Do Now: Teach Me Long Division!

To the board. . .

Algorithm

- A step-by-step process for doing something
- We write programs that *implement* algorithms, putting them to use

Algorithms You've Used

Examples of algorithms you've benefited from

- Getting directions via Google Maps — *shortest path* algorithms from *Graph Theory*

Algorithms You've Used

Examples of algorithms you've benefited from

- Getting directions via Google Maps — *shortest path* algorithms from *Graph Theory*
- Visiting any website — algorithms running on IP routers decided how and when to move your data packets over the Internet requesting content from remote servers

Algorithms You've Used

Examples of algorithms you've benefited from

- Getting directions via Google Maps — *shortest path* algorithms from *Graph Theory*
- Visiting any website — algorithms running on IP routers decided how and when to move your data packets over the Internet requesting content from remote servers
- Typing on a smartphone — an algorithm predicts what word(s) you might be typing based on the letters typed thus far

Describing Algorithms

Two ways of describing algorithms:

- **Pseudocode:** not quite a high-level language (simpler syntax and style!); useful for describing a set of instructions that can then be implemented in your language of choice (e.g., Java!)
- **Flowcharts:** a visual means of showing an algorithm's flow

An Algorithm You'd Better Not Know

- How does a procrastinator work?
- What is his (or her) M.O.[1]?

[1] M.O. is short for *Modus Operandi*, which is Latin for *mode of operation*

An Algorithm You'd Better Not Know

Pseudocode:

An Algorithm You'd Better Not Know

Pseudocode:

$\mathbf{t_{min}} \leftarrow$ `est. minimum time to complete PS#4`

An Algorithm You'd Better Not Know

Pseudocode:

$\mathbf{t_{min}}$ ← `est. minimum time to complete PS#4`
$\mathbf{t_{remaining}}$ ← `(time due) - (current time)`

An Algorithm You'd Better Not Know

Pseudocode:

$\mathbf{t_{min}}$ ← `est. minimum time to complete PS#4`
$\mathbf{t_{remaining}}$ ← `(time due) - (current time)`

`while (` $\mathbf{t_{remaining}}$ `>` $\mathbf{t_{min}}$ `):`

An Algorithm You'd Better Not Know

Pseudocode:

```
t_min ← est.  minimum time to complete PS#4
t_remaining ← (time due) - (current time)

while ( t_remaining > t_min ):
    d ← choose distraction
```

An Algorithm You'd Better Not Know

Pseudocode:

```
t_min ← est.  minimum time to complete PS#4
t_remaining ← (time due) - (current time)

while ( t_remaining > t_min ):
    d ← choose distraction
    engage in d for 1h...
```

An Algorithm You'd Better Not Know

Pseudocode:

$\mathbf{t_{min}}$ ← est. minimum time to complete PS#4
$\mathbf{t_{remaining}}$ ← (time due) - (current time)

while ($\mathbf{t_{remaining}}$ > $\mathbf{t_{min}}$):
    d ← choose distraction
    engage in **d** for 1h...
    recalculate $\mathbf{t_{remaining}}$

An Algorithm You'd Better Not Know

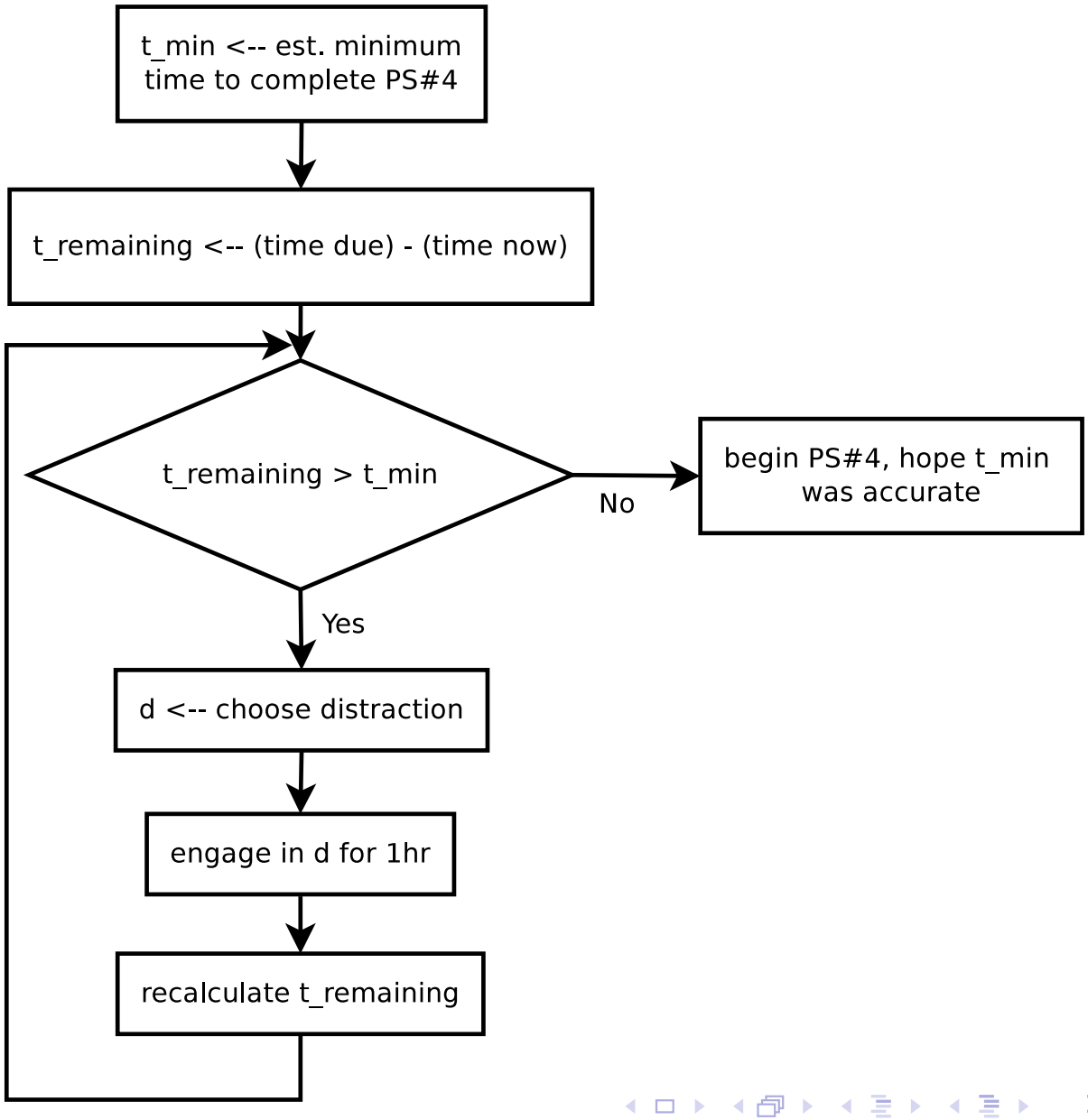
Pseudocode:

```
t_min ← est.  minimum time to complete PS#4
t_remaining ← (time due) - (current time)

while ( t_remaining > t_min ):
    d ← choose distraction
    engage in d for 1h...
    recalculate t_remaining

begin problem set, hope t_min was accurate
```

An Algorithm You'd Better Not Know



while()

- your first flow control statement!

while()

- your first flow control statement!
- finally, non-sequential programs

while()

- your first flow control statement!
- finally, non-sequential programs
- syntax:

```
while ( <condition is true?> ) {
    run the code in this block...
}
```

while() Example

What does this code snippet do?

```
int a = 0;

while ( a < 10 ) {
    System.out.println(a);
}
```

`while()` Example

What does this code snippet do?

```
int a = 0;

while ( a < 10 ) {
    System.out.println(a);
}
```

→ See what the code does:

- Create a new project called `Lesson20` in `workspace0`
- Create class `WhileOne` in the new project
- Insert the `while()` loop into that class' `main()` and run it

while() Example

```
int a = 0;

while ( a < 10 ) {
    System.out.println(a);
}
```

- The **stopping condition** is what causes the `while()` loop to terminate (finish).

`while()` Example

```
int a = 0;

while ( a < 10 ) {
    System.out.println(a);
}
```

- The **stopping condition** is what causes the `while()` loop to terminate (finish).
- In this case, it's when it is no longer true that a < 10. In other words, if a ≥ 10, the program may end.

`while()` Example

```
int a = 0;

while ( a < 10 ) {
    System.out.println(a);
}
```

- The **stopping condition** is what causes the `while()` loop to terminate (finish).
- In this case, it's when it is no longer true that a < 10. In other words, if a ≥ 10, the program may end.
- We need a **loop variant** here — something that changes each time the loop iterates — so that we get closer to the stopping condition over time.

while() Example: Fixed!

```
int a = 0;

while ( a < 10 ) {
    System.out.println(a);
    a = a + 1;          // loop variant:
                        // 'a' changes over time
}
```

Playing with `while()`

- Replace the

  ```
  a = a + 1;
  ```

  line with

  ```
  a++;
  ```

  [2]

- change the `while()` condition from

  ```
  a < 10
  ```

  to

  ```
  a <= 10
  ```

[2] "++" is an **incrementer** operator, increasing the value of the var by 1

Programming Exercises

1. Write a program to display all positive integers up to 100.
2. Write a program to display all even positive integers up to 84.

HW

Work on PS #4a, §§1-3.4, inclusive.[3]

[3] That means §1 through §3.4, *including* §1 and §3.4!