A *New* Approach To The Teaching of Modern Algorithms

SIGSHWNTEL 2018

Introduction

- Agent-based modelling ☑
- Parallel computation ☑
- Scheduling algorithms ☑

Claim

- Revolutionary breakthrough in diactic technique

Proof

```
Agents = { Counter, Fizzer, Buzzer }
```

(live demo)

Proof (code)

- Fearless concurrency

```
use std::io::stdio::flush;
use std::io::timer::sleep;
use std::thread::Thread;
use std::time::duration::Duration;

fn main() {
    let counter = Thread::spawn(|| -> () {
        let mut i = 0u;
        loop {
            i = i + 1;
            print!("\n{}", i);
            flush();
            sleep(Duration::seconds(1));
        }
    });

    let fizzbuzz = Thread::spawn(|| -> () {
        loop {
            print!("\rfizzbuzz");
            flush();
            sleep(Duration::seconds(15));
        }
    });

    let buzz = Thread::spawn(|| -> () {
        loop {
            print!("\rbuzz");
            flush();
            sleep(Duration::seconds(5));
        }
    });

    let fizz = Thread::spawn(|| -> () {
        loop {
            print!("\rfizz");
            flush();
            sleep(Duration::seconds(3));
        }
    });

    let _ = counter.join();
    let _ = fizzbuzz.join();
    let _ = buzz.join();
    let _ = fizz.join();
}
```

Corollary

- This diactic approach applied to the same class of algorithms

Corollary 1.1 Sleepsort

- O(1) Sleepsort proof

  (live demo)

Q.E.D.

∎