CS 331 — Recursion Activity

Mattox Beckman

Introduction

To be a competent programmer, you need a solid grasp of recursion. These questions should help you get there.

Code Samples

```
(defn fun-a [n]
   (if (< n 1) 10
       (* 2 (fun-a (- n 1)))))
```

Figure 1: Program A

```
(defn fun-b [n]
   (if (< n 1) 10
       (* 2 (fun-b (/ n 2)))))
```

Figure 2: Program B

```
(defn fun-c [n]
    (cond (< n 10)   (fun-c (+ n 1))
          (>= n 10)  (fun-c (+ n 2))))
```

Figure 3: Program C

```
(defn fun-d [n]
    (cond (< n 10)   10
          (>= n 10)  20
          :else      (fun-d (- n 2))))
```

Figure 4: Program D

Questions

1. Which of the above functions will go into an infinite loop?

2. Which of the above functions will run in linear ($\mathcal{O}(n)$) time?

3. Which of the above functions will run in logarithmic ($\mathcal{O}(\lg n)$) time?

4. Which of the above functions will never make a recursive call?

You saw in the video an example of the Fibonacci sequence. Here is another way to compute it.

$$\begin{aligned} a_0 &= 1 \\ b_0 &= 1 \\ a_n &= b_{n-1} \\ b_n &= a_{n-1} + b_{n-1} \\ f_n &= a_n \end{aligned}$$

Write a function to compute f_n, given an initial n. For extra niceness, try using the multiple-arity function technique, where one version of the function takes one argument, and the other version takes three arguments.

What time complexity does this function have?