

MC215: MATHEMATICAL REASONING AND DISCRETE STRUCTURES

Monday, 11/17/08

- From last time:
  - More on order of growth
- Today:
  - Recursive functions and algorithms

- READING: 4.4
- EXERCISES: pp. 219-220: 13, 14, 26, 27

Monday, 11/17/08, Slide #1  
Printed 11/17/2008 10:34 AM

---

---

---

---

---

---

---

---

Recursive functions

- A recursive function is a function that is defined in terms of itself.
- Usually  $f$  has domain  $\mathbb{Z}^+$  or  $\mathbb{N}$
- $f$  is defined in two steps:
  - Base case: Define  $f$  directly for  $n = 0$ ;
  - Recursive case: For  $n > 0$ , define  $f(n)$  in terms of  $f(n-1)$
- Factorial function:  $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$
- Recursive definition:
  - $n! = 1$ , if  $n = 1$
  - $n \cdot (n-1)!$ , if  $n > 1$
- Compute  $4!$  using the recursive definition

Monday, 11/17/08, Slide #2  
Printed 11/17/2008 10:34 AM

---

---

---

---

---

---

---

---

More examples

- Give a recursive definition of  $a^n$ , where  $n$  is a nonnegative integer, and  $a$  is a nonzero real number
- The Fibonacci numbers:  $F(n)$  is defined in terms of two preceding values:

$$F(n) = \begin{cases} 1 & \text{if } n = 1 \text{ or } 2 \\ F(n-1) + F(n-2) & \text{if } n > 2 \end{cases}$$

- Compute  $F(0)$ ,  $F(1)$ , ...,  $F(6)$

Monday, 11/17/08, Slide #3  
Printed 11/17/2008 10:34 AM

---

---

---

---

---

---

---

---

## Recursive Algorithms

□ A **recursive algorithm** is one with a definition that has:

- **Base case:** Solves the problem directly for small inputs
- **Recursive case:** Solves the problem for larger input by using solution for smaller inputs

```
Non-recursive linear search:  
LinearSearch(s, n, key) {  
  i := 1;  
  while (i ≤ n and s[i] ≠ key)  
    i := i + 1  
  if (i ≤ n) return (i)  
  else return ("Not Found")  
}
```

Monday, 11/17/08, Slide #4  
Printed 11/17/2008 10:34 AM

---

---

---

---

---

---

---

---

## Recursive Linear Search

□ Idea: If key not found in  $\{s_1, \dots, s_n\}$ , then search the list  $\{s_2, \dots, s_n\}$ , etc.

```
LinearSearch(s, n, key){  
  RLinearSearch(s, 1, n, key)  
  
RLinearSearch(s, first, last, key) {  
  if (first > last) return ("Not Found")  
  else if (s[[first]] = key) return (first)  
  else RLinearSearch(s, first+1, last, key)  
}
```

Monday, 11/17/08, Slide #5  
Printed 11/17/2008 10:34 AM

---

---

---

---

---

---

---

---

## Example: The Towers of Hanoi

□ **Game in which object is to move rings of different sizes**

- from one tower
- to a second tower,
- using a third tower if needed.

□ **Rules:**

- You may only move the **top** ring on a tower
- You are allowed only to place a **smaller** ring on a **bigger** ring.

Monday, 11/17/08, Slide #6  
Printed 11/17/2008 10:34 AM

---

---

---

---

---

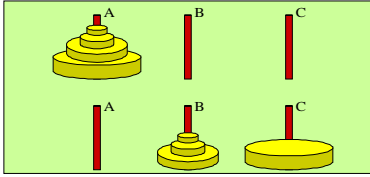
---

---

---

## Two Important Observations

- To move *all* disks from Tower A to Tower C, we must *first* get the *bottom* disk from A to C, which means all the rest must be on Tower B.
- ***But this means we must solve the problem of size  $n-1$  from A to B!!***



Monday, 11/17/08, Slide #7  
Printed 11/17/2008 10:34 AM

---

---

---

---

---

---

---

---