

Spring 2008 22C:021:A01/A03

Computer Science II : Data Structures

Instructor: Wenli He (wenli-he@uiowa.edu)

Supervisor: Jim Cremer (cremer@cs.uiowa.edu)

Department of Computer Science

University of Iowa

01/31/2008

Outline

- Page 4 : compute series $\sum_{i=1}^{\infty} i/2^i$
- Page 10 : Recursion and Induction (Theorem 1.4)
- Quiz (20 minutes)
 - NOTE: open textbook, but **closed** notes

Compute series $\sum_{i=1}^{\infty} i/2^i$

$$S = \sum_{i=1}^{\infty} i/2^i$$

$$2S = 2 \cdot \sum_{i=1}^{\infty} i/2^i = \sum_{i=1}^{\infty} i/2^{i-1} = 1 + \sum_{i=2}^{\infty} i/2^{i-1}$$

$$= 1 + \sum_{i=1}^{\infty} (i+1)/2^i = 1 + \sum_{i=1}^{\infty} 1/2^i + \sum_{i=1}^{\infty} i/2^i$$

$$= \sum_{i=0}^{\infty} (1/2)^i + S, \text{ because } 1 = (1/2)^i, \text{ where } i = 0.$$

$$= \frac{1}{1-\frac{1}{2}} + S, \text{ because } \sum_{i=0}^{\infty} A^i = \frac{1}{1-A}, 0 < A < 1$$

Thus, $2S = 2 + S$, that is, $S = 2$.

Prove correctness of recursive routine using induction

Figure 1.4 Recursive routine to print an integer

```
/* print nonnegative n */
public static void printOut(int n)
{
    if( n >= 10 )
        printOut( n / 10 );
    printDigit( n % 10 );
}
```

Base case: `printDigit(n)` if $0 \leq n < 10$.

Theorem 1.4

Theorem 1.4.

The recursive number-printing algorithm is correct for $n \geq 0$.

Proof (by induction on the number of digits in n).

Base case: n has one digit.

Proof: If n has one digit, the program is trivially correct.

Inductive hypothesis:

If `printOut` works for all numbers of k or fewer digits, then `printOut` works for all numbers of $(k + 1)$ digits.

Proof: (To be continued)

Theorem 1.4

Proof: (Continued)

$n = d_k d_{k-1} \dots d_1 d_0 = d_k d_{k-1} \dots d_1$ followed by d_0 .

$d_k d_{k-1} \dots d_1 = \lfloor \frac{n}{10} \rfloor$, line 4 prints first k digits.

$d_0 = n \% 10$, line 5 prints last digit.

Thus, any number of $(k + 1)$ digits is printed correctly.

Theorem is proved by induction.

Four Fundamental Rules of Recursion

1. *base cases*
2. *making progress*
3. *design rule*
4. *Compound interest rule.* Never duplicate work by solving the same instance of a problem in separate recursive calls.

(NOTE: This is the reason that it's generally a "bad" idea to use recursion to evaluate simple mathematical functions, e.g., Fibonacci numbers.

"bad" - doesn't mean recursion itself is bad, but poor use of it. In particular, DONOT "redo" work already done before. See example of Fibonacci numbers.)

Quiz

Quiz

(20 minutes)

Open textbook, closed notes!