COSC 2011 Section N

Tuesday, March 20 2001

Overview

- Mathematical Induction
  - Definition, Examples
- Introduction to Recursion
  - Definition, Examples
- Assignment 1 Notes/Questions?
  - For loop time analysis

Loop Invariants: (1)

- An assertion that remains true each time the statements of a loop are executed
  - Tells us something about the values of the loop variables while executing a loop.
  - Should be true at the beginning of each iteration, including the first!

$\{p\}$ **while** C **do** $S \rightarrow \{p \wedge \neg C\}$

Bubble Sort Algorithm (1)

Algorithm Bubblesort(*sequence*):

Input*:* sequence of integers *sequence*

Postcondition*: sequence* is sorted & contains the same integers as the original sequence

length = length of *sequence*

for *i* = 0 **to** *length* **-** 1 **do**
  for *j* = 0 **to** *length* **-** *i* **-** 2 **do**
    if *j*th element of *sequence* > (*j*+1)th element of *sequence*
    then
      swap *j*th and (*j*+1)th element of *sequence*

Bubble Sort Algorithm (2)

- Loop Invariant – Outer Loop:
  - Last i elements of *sequence* are sorted and are all greater or equal to the other elements of the sequence.
- Loop Invariant – Inner Loop:
  - Same as outer loop and the jth element of *sequence* is greater or equal to the first j elements of s*equence*.

Bubble Sort Algorithm (3)

- Running Time Analysis:
  - Assume access to and swap of elements takes O(1) time.
  - Running time of ith pass:

    O(sum[n-i+1])
  - Can re-write it as:

  O(n + (n+1) + … + 2 + 1)

  O(sum(i))

  By proposition 3.4:

  sum(i) = [n(n+1)] / 2

Mathematical Induction: (1)

- Can be used only to prove results obtained some some other way:
  - Not a tool for discovering formulas or theorems!
- Many theorems state P(n) is true for all positive integers n
  - Mathematical induction is used to prove assertions (propositions) of this kind.

Mathematical Induction: (2)

- Used top prove statements of the form $\forall n P(n)$, for all positive integers.
- A proof by mathematical induction that P(n) is true for all positive integers consists of two steps
  1. Basis Step: show P(1) (or n = some other finite value) is true.
  2. Inductive Step: Show $P(n) \rightarrow P(n+1)$ is true for every positive integer n.

Mathematical Induction: (3)

- P(n) is called the ***inductive hypothesis.*** When both steps are done, then we have shown $\forall n P(n)$.

$$[P(1) \land \forall n(P(n) \rightarrow P(n+1)] \rightarrow \forall n P(n)$$

  - To prove the inductive step for every n, we need to show P(n) cannot be false when P(n) is true.
    - Assume P(n) is true & show that under this assumption, P(n+1) must be true.

Mathematical Induction: (4)

- **Remark**: It is not assumed P(n) is true for all positive integers! Only shown that if it is assumed P(n) is true then P(n+1) is also true.

- When using induction, we show that P(1) is true. Then since P(1) implies P(2), P(2) must be true. Then P(3) is true because P(2) implies P(3). Continuing along these lines, P(k) is true for any positive integer k.

COSC 2011 Section 9

Mathematical Induction: (5)

- **Useful Illustration**:
  - Consider a line of people, person 1, person 2 etc. A secret is told to the first person and each person tells the secret to the next person in line.
  - Let P(n) be the statement that person n knows the secret.
    - P(1) is true since it was told to first person.
    - P(2) is true since person 1 tells person 2 and so on…

COSC 2011 Section 10

Mathematical Induction: (6)

- **Another Illustration:**
  - Infinite row of dominos, labeled 1,2,3,….,n & each domino is standing up.
  - Let P(n) be the statement that domino n is knocked over.
  - If the first domino is knocked over, P(1) is true.
  - If whenever first domino is knocked over – P(1) is true, it knocks the (n+1)th domino over – P(n) $\rightarrow$ P(n+1) is true, then all dominos are knocked over!

COSC 2011 Section 11

Mathematical Induction: (7)

- Sometimes we need to show P(n) is true for n=k, k+1, k+2… where k is an integer other than 1.
  - Can still use induction as long as we change the *Basis Step*.
  - Show P(k) is true and then show P(n) P(n+1) is true for n = k, k+1, k+2…
  - K can be negative, positive or zero.

COSC 2011 Section 12

Recursive Methods (1):

- Sometimes we can reduce the solution to a problem with a particular input to the solution of the same problem with smaller input.
  - Solution to the original problem can be found with a sequence of ***reductions*** until problem is reduced to some initial case where solution is known.

COSC 2011 Section 13

Recursion (Math): (2):

- Defintion (Mathematical):
  - When an object is defined in terms of itself.
- Can be Used to Define:
  - Sequences, functions sets.
- Example:
  - Sequence of powers of 2 is given by $a_n = 2^n$
  - Can also be defined as:
    - Give the first term of the sequence: $a_0 = 1$

COSC 2011 Section 14

Recursion (Math): (3):

- Rule for finding a term of the sequence from the previous one.

- Recursively Defined Functions:

1. Specify the value of the function at 0.
2. Give rule for finding its value as an integer from its values at smaller integers.

COSC 2011 Section 15

Recursion (Math): (4):

- Example:
  - f is defined recursively as:

$f(0) = 3$

$f(n + 1) = 2f(n) + 3$

$f(1) = 2f(0) + 3 = (2 \times 3) + 3 = 9$

$f(2) = 2f(1) + 3 = (2 \times 9) + 3 = 21$

$f(3) = 2f(2) + 3 = (2 \times 21) + 3 = 45$

- Question:
  - Give an inductive defintion of the factorial function: $f(n) = n!$

COSC 2011 Section 16

Recursion (Math): (5):

- Solution:
  - Initial value:

    $f(0) = 1$

  - Rule for finding $f(n+1)$:
    - (n+1)! is computed by multiplying n! by (n+1)

    $f(n+1) = f(n) \times (n+1)$

- Determining the Value of the Factorial Function:
  - Use the rule that shows how to express $f(n+1)$

COSC 2011 Section 17

Recursion (Math): (6):

  - In terms of $f(n)$ several times:

- Example of $f(4) = 4!$

$f(4) = 4f(3) = 4 \times 3f(2) = 4 \times 3 \times 2f(1) = 4 \times 3 \times 2 \times 1 \times f(0) = 4 \times 3 \times 2 \times 1 \times 1 = 24$

  - When $f(0)$ is the only function that occurs, no more reductions are necessary.
    - Only thing to do is insert $f(0)$ into formula.

COSC 2011 Section 18

Recursion (Math): (7):

- Example: Fibonacci Numbers f_0, f_1, f_2, … f_n are defined as follows:

  $f_0 = 0$,

  $f_1 = 1$ and

  $f_n = f_{n-1} + f_{n-2} \quad n = 2,3,4\ldots$

  - What are f_2, f_3, f_4, f_5?

COSC 2011 Section 19

Recursion (Math): (8):

  - Solution to Fibonacci Numbers for n = 2,3,4,5,6

    $f_2 = f_1 + f_0 = 1 = 0 = 1$

    $f_3 = f_2 + f_1 = 1 + 1 = 2$

    $f_4 = f_3 + f_2 = 2 + 1 = 3$

    $f_5 = f_4 + f_3 = 3 + 2 = 5$

    $f_6 = f_5 + f_4 = 5 + 3 = 8$

COSC 2011 Section 20

Recursive Methods (1):

- A method is called *recursive* if it solves a problem by reducing it to an instance of the same problem with smaller input.
  - A Recursive Method calls itself as a subroutine with.
  - Each time it calls itself, the problem is "reduced" until we reach a point where the problem is small enough to be easily solved.

Recursive Methods (2):

- Candidate Problems for Recursion Have the Following Characteristics:
  - One or more simple cases of the problem have simple non-recursive solution (*base cases*).
  - For other cases there is a process for substituting one or more reduced cases of the problem that are closer to the base case.

Recursive Methods (3):

- Eventually the problem can be reduced to base cases only, all of which are easy to solve!
- Recursive algorithms we will encounter will generally be of the form:

If *base case* reached **then**
solve it
else
reduce problem using recursion

Recursive Methods (4):

- Recursive Factorial:

```
public static long factorial(long n){
  if (n <= 1)
    return 1;
  else
    return n * factorial(n – 1);
}
```

- Method calls itself recursively to compute factorial of n-1.
- When recursive call terminates, returns (n-1)!

Recursive Methods (5):

- (n-1)! Is then multiplied by n to compute n!.
- In turn, recursive invocation calls itself to compute the factorial of n-2, etc…
- To compute n!, multiply n by factorial of n-1. But how do we calculate factorial(n-1)?
  - We call the factorial method with n-1 as an argument…

COSC 2011 Section 25

Recursive Methods (6):

- Important Properties Every Recursive Methods Should Possess:
  - Method must terminate!
    - Base case
    - Even infinite recursive method will terminate!
  - Always perform the recursive call on a smaller input value.
    - Reduce the problem at every recursive call.

COSC 2011 Section 26

Recursive Methods (7):

- Example: Fibonacci Numbers

```
public static long fibonacci(long n){
  if (n = = 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return fibonacci(n – 1) +
     fibonacci(n-2);
}
```

COSC 2011 Section 27

Recursion & Induction:

- Notice the Similarity Between Recursion and Induction!
  - Induction can be used to prove the correctness of many recursive formulas & functions!
- Problem with Recursion:
  - Usually require more computation and space over an iterative approach

COSC 2011 Section 28