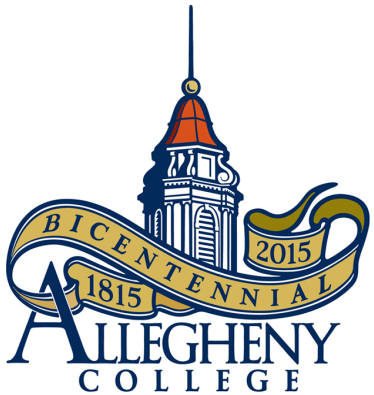
CMPSC 112

Lecture 9: Recursion

Dr. Aravind Mohan

Allegheny College

September 28, 2017



Last Time

- How to declare Arrays
- How to generate values in the elements of an Array
- How to access the elements in an Array

Reminder: Mastery Quiz.

Reminder: Midterm exam 01 date is changed to 10/05/2017 during the lab session. (Reason for the change is additional time needed to take the exam).

Reminder: No lecture class on 10/05/2017, in order to give some extra preparation time.

is

What is Recursion?

- Sometimes, the best way to solve a problem is by solving a smaller version of the exact same problem first.
- Recursion is a technique that solves a problem by solving a smaller problem of the same type.

Recursion Vs Iteration

- Iteration can be used in place of recursion.
  - An iterative algorithm uses a looping construct.
  - A recursive algorithm uses a branching structure.
- Recursive solutions are often less efficient, in terms of both time and space, than iterative solutions.
- Recursion can simplify the solution of a problem, often resulting in shorter, more easily understood source code.

How do I write a recursive function?

- Determine the size factor
- Determine the base case(s)

  (the one for which you know the answer)
- Determine the general case(s)

  (the one where the problem is expressed as a smaller version of itself)
- Verify the algorithm

  use the (”Three-Question-Method”)

Three-Question Verification Method

1. The Base-Case Question:

   Is there a nonrecursive way out of the function, and does the routine work correctly for this base case?

2. The Smaller-Caller Question:

   Does each recursive call to the function involve a smaller case of the original problem, leading inescapably to the base case?

3. The General-Case Question:

   Assuming that the recursive call(s) work correctly, does the whole function work correctly?

Example 1: Factorial Calculation

- Question: What is ”12!”?
  - 12! = 12*11*10*9*8*7*6*5*4*3*2*1
  - 12! = 479,001,600
- Iterative calculation: Put it in a for-loop.
- Recursive calculation: Use fact(n-1) to calculate fact(n).
- Do each of these provide identical answers?
- Do each of these run at (roughly) the same speed?

Example 2: Fibonacci Calculation

- Question: What are the Fibonacci numbers?
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233
  - Each number is the sum of the two numbers preceding it.
- Iterative calculation: Put it in a for-loop.
- Recursive calculation: Use fib(n-1) and fib(n-2) to calculate fib(n).
- Do each of these provide identical answers?
- Do each of these run at (roughly) the same speed?

Example 3: n Choose k (combinations)

- Question: Given n things, how many different sets of size k can be chosen?

$$ {}_nC_r = \frac{n!}{r!(n-r)!} $$

Any Questions

- Reminder 01: REVIEW FORM