Algorithm Analysis

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

May 19, 2017

Overview I

1. Algorithm Design
    - Intro
    - Developing Algorithms
    - Algorithm Differences
    - Recursion
    - Tail Recursion

Intro

- An algorithm is a step by step process for solving a problem. It consits of a finite sequence of instructions that when carried out, always terminates.
- Algorithms manipulate data structures.
- They are developed through a process of refinement, from an informal description to a formal description.
- Formal description is written in pseudocode.

Developing Algorithms

Steps to designing algorithm

1. Express in general terms how the algorithm works.
2. Give more detailed, but still informal description of the algorithm, identifying subproblems.
3. May be necessary to treat subproblems in the same way, known as step-wise refinement.
4. Give detailed, unambiguous description in pseudo-code.

Algorithm Differences

- Differeneces between algorithms can impact dramatically the speed of execution.
  - This is measured by run-time efficiency
- Differences can also mean they have different memory requirements
  - They are said to have different space efficiency
- There is often a trade-off between the two.

Recursion

- A recursive definition is something that is defined in terms of itself. (A method calling itself).
- Example is definition of factorial n:

$$\begin{aligned} 1! &= 1 \\ n! &= n \times (n-1)!, \quad \mathit{for} \quad n > 1 \end{aligned} \tag{1}$$

- Rules for a recursive algorithm:
  1. Must have at least one base case and one recursive case.
  2. The recursive case should ensure that the base case is eventually reached.

Tail Recursion

- An algorithm is tail recursive if there is nothing to do after the return (except return its value).
- For example, this return is **NOT** tail recursive because it has to be multiplied by n before return: `return n × factorial(n-1);`
- This, however **IS** tail recursive: `return gcd(y,x%y);`
- Tail recursive algorithms can easily be turne into iterative algorithms.
- Iterative is usually more efficient to use.

The End