Algorithms and Imperative Programming

COMP26120

Todd Davies

December 14, 2014

Introduction

This is a two-semester practical introduction to algorithms and data structures, concentrating on devising and using algorithms, including algorithm design and performance issues as well as 'algorithmic literacy' - knowing what algorithms are available and how and when to use them.

To reflect the emphasis on practical issues, there are two practical (laboratory) hours to each lectured hour. Lectures serve to motivate the subject, orient students, reflect on practical exercises and impart some basic information. A range of practical applications of algorithms will also be presented in the lectures. Other information resources will be important, including a set textbook, which will provide essential support.

The course-unit starts with a 5-week primer on the C programming language, enabling students to become competent programmers in this language as well as in Java (and, possibly, in other languages). This teaching is supported by an on- line C course and extensive laboratory exercises.

There is a follow-up course unit on Advanced Algorithms in the Third Year. This presents the foundational areas of the subject, including (1) measures of algorithmic performance and the classification of computational tasks by the performance of algorithms, (2) formulating and presenting correctness arguments, as well as (3) a range of advanced algorithms, their structure and applications.

Aims

- To make best use of available learning time by encouraging active learning and by transmitting information in the most effective ways.
- To give students a genuine experience of C.
- To make students aware of the importance of algorithmic concerns in real-life Computer Science situations.
- To emphasise practical concerns, rather than mathematical analysis.
- To become confident with a range of data structures and algorithms and able to apply them in realistic tasks.

Additional reading

Algorithm design: foundations, analysis and internet examples - Goodrich, Michael T. and Roberto Tamassia

Contents

1 Algorithmic complexity and performance

Complexity	Growth rate	
$O(1)$	None	
$O(logn)$	Logarithmic	
$O(n^k)$	Polynomial	
$O(n)$	Linear	All of these are special cases of polynomials, n^1, n^2 and n^3 respectively
$O(n^2)$	Quadratic	
$O(n^3)$	Cubic	
$O(k^n)$	Exponential	
$O(n!)$	Factorial	

Table 1: A number of common complexities and their equivalent growth rates

Growth rates that are either exponential or factorial in nature (or are perhaps even worse than this) are said to be intractable , while algorithms with other computational complexities are said to be tractable.

Tractable (*Adjective*)
Easy to deal with.

In order to simplify the big-oh complexity of an algorithm you just isolate the fastest growing term in the equation (i.e. whatever term comes furthest down in Table **??**). You then remove all the constants from the equation.

2 Algorithmic correctness

3 Data structures

4 Basic algorithms

4.1 Sorting

4.2 Searching

4.3 Tree and graph traversal