ALGORITHMS, CORRECTNESS AND EFFICIENCY
G52ACE

January 10, 2017

University of Nottingham
Department of Computer Science

Contents

INTRODUCTION

Algorithms, Correctness and Efficiency is a 20 credit module. There are two exams. The first is on the 23rd January 2017 and is worth 25%. The second is in the summer and is worth 50%.

LECTURE 1: ANALYSING ALGORITHMS

What is an algorithm

An algorithm is a step-by-step solution for solving a problem in a **finite** amount of time. Most algorithms convert input data to ouput data. The running time typically grows with the input size. The average running time of an algorithm is often very hard to calculate so we focus on the worst case. We often refer to the worst case as $T(n)$.

We can guess the running time through experimentation however this may not be accurate. It may also be time consuming. It is not possible if you want to know the running time ahead of computation. You will also have to implement the algorithm which won't be ideal if you want to evaluate algorithms before programming them in code.

Theoretical analysis of algorithms characterizes running time as a function of the runnign time **n**.

It is important to write the algorithm in pseudo code in order to analyse it. We assume that the algorithm runs on a RAM (Random Access Machine different to the memory in your computer) which is a theoretical computer with a CPU and an unlimited supply of memory. The RAM model is not representative of a real computer and has certain differences i.e. unlimited memory, each memory cell can hold an arbitrary number etc.

Algorithm *arrayMax*(A, n)	# operations
currentMax $\leftarrow A[0]$	2
for $i \leftarrow 1$ **to** $n-1$ **do**	1
if $A[i] >$ *currentMax* **then**	$2\,(n-1)$
currentMax $\leftarrow A[i]$	$2\,(n-1)$ (worst case)
{ increment counter: i++ }	$2\,(n-1)$ ("hidden")
{ test counter: $i \leq$ (n-1) }	$2\,(n-1)$ ("hidden")
return *currentMax*	1
Total	$8n-4$

Figure 1: Counting primitive operations of an algorithm calculating the largest number in an array

Unfortunately counting is quite vague. For example, one person might come up with 2 and another 4. Either could be correct but 2n will not be correct. Actual running time depends on many things including CPU architecture, RAM speed, compiler speed etc. It is impossible to take all these things into account so don't worry about finer details.