COMP20012 Tutorial 3: Algorithm Design – The questions

If you are having difficulty with any of the concepts in the course, you may like to use this tutorial to discuss them with your tutor, as well as attempting the following exercise in algorithm design.

A Text Editing Problem

The task: Consider an array A of elements indexed from 0 through to N. Let $0 \leq j \leq N$. We consider the task of exchanging the section of array A from 0 through to j inclusive, with the remaining section. Thus for the array of characters (including spaces) "`sat on the mat the cat `" and $j = 14$, the result of the exchange is "`the cat sat on the mat `".

1. Describe an algorithm for the task based upon first reversing the order of the elements in each of the two sections and then performing a further reverse operation to yield the array with the sections exchanged. Explain clearly why your algorithm works i.e. give a *correctness argument*, by showing what happens to an element at a given position. What is the worst-case time complexity of this algorithm?

2. Explain why, if the two sections are of equal length, then exchanging them is straightforward. Use this observation to describe an alternative algorithm for exchanging sections of arbitrary length by exchanging the shorter section with part of the longer section (of equal length) and continuing in this manner until the two sections are fully exchanged.

   Give a recurrence relation for the time complexity of this algorithm in terms of the lengths of the two sections (there are several cases, depending upon the lengths). You need not solve the recurrence relation.

COMP20012 Tutorial 3: Algorithm Design – Tutors notes

Aim: This is an exercise in algorithm design using basic design principles, including divide-and-conquer, which the students see extensively. It is a simple problem whose efficient algorithms can be quite intricate - a recurring theme - that of Part (2) is difficult to analyse fully. Another theme is the range of algorithms available for problems, and this illustrates two quite different algorithms which may have different applications.

1. If the two sections of the array are A_1 and A_2, then the procedure can be concisely expressed as $\text{reverse}(\text{reverse}(A_1).\text{reverse}(A_2))$ where "." is the concatenation of arrays!

   The correctness argument can be put in several forms. Here is one. Let $0 \leq i \leq j$ i.e. i is an index into the first section. When this section is reversed, the element in the i-th position goes to the $(j-i)$-th position. When then the whole array is reversed, this element in the $(j-i)$-th position ends up at $N-(j-i)$. But $N-(j-i) = (N-j)+i$, so the element ends up in the i-th position in the second section as required. Symmetrically for an element initially in the second section.

   As for the time complexity: To reverse an array of length M requires approx $M/2$ interchanges of pairs of elements. Thus to exchange the two sections by this method requires $(j+1)/2 + (N-j)/2 + (N+1)/2 = N+1$ interchanges.

2. The algorithm based upon exchanging equal sections is described in the attached extract from the textbook.