Algorithms Worksheet 1

This worksheet contains mostly pen and paper calculations. The solutions should be submitted as plain text, files in other formats, like .doc or .pdf will not be accepted. You can write x^n and a_n for x^n and a_n. For each part of a question write the answer and a short description of how the answer was obtained, so, for example, each part of questions 1 and 2 should have a one sentence description. Each question is worth a fifth of the marks, question 4 and 5 are the extension questions.

1. This question is about estimating the algorithmic complexity of simple operations. Take multiply and addition of a single digit as roughly one operation and assume that powers are calculated by doing the corresponding number of multiplications.

   a) What is the big-oh complexity of adding two n-digit numbers?

   b) What is the big-oh complexity of multiplying two n-digit numbers using the standard long multiplication method.

   c) What is the big-oh complexity of calculating x^n of a single digit number?

2. This question is about estimating the algorithmic complexity of evaluating a polynomial or multiplying matrices. Here, consider fixed sized variables, so multiplication and addition take roughly one step, irrespective of how many digits the number has. Once again, powers are calculated by multiplication.

   a) What is the big-oh complexity of evaluating, that is finding the value of $p(x)$, of an order n polynomial

   $$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

   using straight-forward substitution?

   b) Horner's method is a quicker method for evaluating a polynomial. If x_o is the value that the polynomial needs to be evaluated on, let $b_n = a_n$ and then

   $$b_{n-1} = a_{n-1} + x_o b_n$$

   and

   $$b_{n-2} = a_{n-2} + x_0 b_{n-1}$$

   right down to

   $$b_0 = a_0 + x_0 b_1$$

   and $b_0 = p(x_0)$ is the answer. What is the big-oh complexity?

   c) A matrix is an algebraic object with two indices, in a square matrix the indices both have the same range, say one to n. We write $A = [a_{ij}]$ where A is the matrix and a_{ij} are the components. The multiple of two matrices is given by

   $$C = AB$$

   where the components c_{ij} of C are given by

   $$c_{ij} = \sum\nolimits_{k=1}^{n} a_{ik} b_{kj}$$

   with $A = [a_{ij}]$ and $B = [b_{ij}]$. What is the big-oh complexity of multiplying two square matrices?

3. This question is intended to help you practice counting things. In the Arthur C. Clarke short story *The Nine Billion Names of God* a ‘Mark V Automatic Sequence Computer’ is purchased by a religious community to list all the possible names of their deity, a task whose importance is central to their belief system. We are told that this name is nine or fewer characters long and that no characters can occur more than three times in sequence, so AABBAABBA is a possible name, as is AAABBAABB, but AAAABBAAB is not (see note below). We are not told in the story how many characters the alphabet contains but for definiteness lets assume there are 18 letters in the alphabet. Show how you could calculated how many names there are with n; in other words, describe a method for calculating the number of names of up to length n, this method could be used to calculate the number of names of length nine, but that isn’t required, just the method.

4. Consider another arithmetic operation, or a different algorithm for one of the operations above, describe it and estimate the big-oh complexity. Examples might be long division or the power of an m digit number or the peasant algorithm for multiplication.

5. Consider some other non-sorting algorithm, describe it and estimate the big-oh complexity. Examples might be the Tower of Hanoi or the Traveling Salesman problem.

Note from the Arthur C Clarke story in Q3

The restriction permits letters to repeat three times but not four may seem surprising, this is noted in the story

> ‘[W]e use a special alphabet of our own. Modifying the electromatic typewriters to deal with this is, of course, trivial. A rather more interesting problem is that of devising suitable circuits to eliminate ridiculous combinations. For example, no letter must occur more than three times in succession.’
>
> ‘Three? Surely you mean two.’
>
> ‘Three is correct: I am afraid it would take too long to explain why, even if you understood our language.’