COMP523 Continuous Assessment 1

John Sylvester

Due at 17:00 on 20/03/2023

A. In the RAM model we assume that multiplication of two n-bit numbers takes constant time however this is not actually the case.

One approach to multiplying two large numbers is to break them up by powers of the base, for example (in base 10)

$$\begin{aligned} 913 \cdot 302 &= (9 \cdot 10^2 + 1 \cdot 10^1 + 3 \cdot 10^0) \cdot (3 \cdot 10^2 + 2 \cdot 10^0) \\ &= 27 \cdot 10^4 + 3 \cdot 10^3 + 9 \cdot 10^2 + 18 \cdot 10^2 + 2 \cdot 10^1 + 6 \cdot 10^0 \\ &= 275726. \end{aligned}$$

This works nicely as multiplying by a power of the base (i.e. 10^4 above or 2^3 if we are working in binary) is quick, since you just shift the digits. This suggests a way of breaking the problem down into smaller sub-problems. Generally, given two n-bit numbers n_1 and n_2 we express $n_1 = a_1 \cdot 2^{\lceil n/2 \rceil} + b_1$ and $n_2 = a_2 \cdot 2^{\lceil n/2 \rceil} + b_2$ and then (ignoring shifting by powers of 2 and some additions) we have broken the problem of multiplying two n-bit numbers into the 4 multiplications of $\lceil n/2 \rceil$-bit numbers. This approach leads to an $O(n^2)$ algorithm.

However, there is a smarter trick. Given $n_1 = a_1 \cdot 2^{\lceil n/2 \rceil} + b_1$ and $n_2 = a_2 \cdot 2^{\lceil n/2 \rceil} + b_2$ we let

$$c = a_1 \cdot a_2, \qquad d = b_1 \cdot b_2 \quad \text{and} \quad e = (a_1 + b_1) \cdot (a_2 + b_2) - c - d, \tag{1}$$

Then we have the expression

$$n_1 \cdot n_2 = c \cdot 2^n + e \cdot 2^{\lceil n/2 \rceil} + d. \tag{2}$$

Notice here that we used only 3 multiplications of $\lceil n/2 \rceil$-bit numbers, we used more additions but these are 'cheap' - we can assume that adding two n-bit numbers can be done in time $O(n)$. Likewise assume multiplication by any power 2^k takes time $O(k)$.

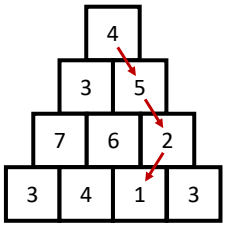
(i) Develop this idea (the equations given by (1) and (2)) into an algorithm for multiplying two n-bit numbers and provide pseudocode, explaining clearly which algorithmic principles you are using. [8 marks]

(ii) Show correctness of your algorithm. This requires verifying the formula given by (2) holds and arguing that your algorithm computes this correctly. [8 marks]

(iii) Analyse your pseudocode to derive a recurrence relation for $T(n)$, the running time of your algorithm on an input of two numbers both with n bits. [7 marks]

(iv) Solve this relation to give the asymptotic running time of your algorithm. You can either prove this running time by induction or use the Master theorem, however in the case of the Master theorem you need to justify why your application is valid. [7 marks]

B. You are given a pyramid of boxes each of which contains a number. You start at one at the top and at each step you move to the level below in one of the adjacent boxes. Your goal is to find the path with minimum sum of entries. For example, in the image below the smallest sum is achieved by the path shown:



(i) Find a counterexample for each of the following greedy algorithms:

- Start from the root and always follow the smallest value. [5 marks]
- Start from the bottom and always follow the smallest value. [5 marks]
- Start from the root and always choose the next step that would give the optimal path when looking at most two steps ahead. [5 marks]

(ii) Design an algorithm to efficiently solve this problem and give pseudocode for your algorithm, explaining clearly which algorithmic principles you are using. [15 marks]

(iii) Argue about correctness of your algorithm and determine the time and space complexity. [10 marks]

C. Consider a directed graph with edge capacities, representing a rail network. There are three types of vertex: supplies, demands, and ordinary interconnection points. There is a single type of cargo we wish to carry. Each demand vertex v requires $d_v > 0$ units. Each supply vertex v has a maximum amount it can produce $s_v > 0$ units. We wish to determine if we can route exactly d_v units to each demand vertex v from the supply vertices.

(i) Explain how to express this problem as a max flow problem. Justify correctness of your construction. [10 marks]

We now add a further constraint: each interconnection point v has a capacity $i_v \geq 0$, and no more than i_v units of flow can pass through v. More formally if an interconnection point v has edges $e_1, \ldots, e_k$ then the flow though these edges can be at most i_v.

(ii) Adapt your model for the original problem above to take account of this new constraint. Justify correctness of your construction. [10 marks]

We now allow each demand vertex to receive any number of units. The question is now: can we route all available units from supply vertices to some demand vertices?

(iii) Design a polynomial time algorithm to solve this problem. Justify correctness of your algorithm. [10 marks]