CS341 Advanced Topics in Algorithms Notes

Based on lectures by Dr. Alex Tiskin
Notes by Alex J. Best

May 6, 2014

1 Introduction

These are some rough notes put together for CS341 in 2014 to make it a little easier to revise. The headings correspond roughly to the contents of the module that is on the module webpage so hopefully these are fairly complete, however they are not guaranteed to be.

2 Computation by circuits

A computational model is an abstract computing device used to reason about computations and algorithms. For example, Turing machines and quantum computers.

An algorithm in a specific model is a description of some input, processing steps and output. The encoding for the input and output along with the steps in a specific model should be given.

The complexity of an algorithm can depend on the model used, for example sorting can be easier if we can do more than simply compare two objects and factoring large numbers can be done faster on a quantum computer.

A basic special-purpose computational model we will use is a *circuit*, this is a directed acyclic graph where nodes represent operations and edges represent the flow of inputs and outputs. The computation is *oblivious* the order of operations is independent of the input. A circuit will only allow for input and output of a fixed size, so to compute with a varying number of inputs we must give an infinite family of circuits, this may or may not admit a finite description! Some common operations performed by a node in a circuit are arithmetic, boolean logic and input comparison, we assume these operations are constant time. The *size* of a circuit is the number of nodes and the *depth* is the maximum path length from the input to the output.

A *comparison network* is a circuit of comparator nodes that move the largest input to the rightmost output. A *merging network* is a comparison network that takes as input two sorted sequences of fixed lengths and merges them into one sorted sequence. A *sorting network* is a comparison network that takes an arbitrary input sequence and gives a sorted output sequence. A finite description of a sorting (or merging) networks is equivalent to an oblivious sorting (or merging) algorithm. The size/depth of these networks determine the sequential/parallel complexity of the algorithms respectively. For example bubble sort on n inputs has size $n(n-1)/2$ and depth $2n-1$, in fact insertion sort gives exactly the same circuit.

Claim 1 (Zero-one principle)**.** A comparison network is sorting if and only if it sorts all input sequences of 0s and 1s.

Proof. Only if is clear, for the if direction we argue by contradiction. Assume there is some input $x = (x_1, \ldots, x_n)$ that the network does not sort and instead outputs a sequence $(y_1, \ldots, y_n)$ which

is not in the correct order, let k and l be the indices of the output whose elements are the wrong way round. Then let

$$X_i = \begin{cases} 0 & \text{if } x_i < y_k, \\ 1 & \text{if } x_i \geq y_k, \end{cases}$$

and run the network on input $(X_1, \ldots, X_n)$. Each X_i will follow exactly the same path through the network as the x_i did, but in the position k of the output there will be a 1 and position l will contain a 0, contradicting the fact that the network sorts all 0,1 sequences. □

This principle applies to sorting merging and other comparison based problems, such as selection. It massively reduces the amount of work required to check a sorting network automatically. A merging network for sequences of length m and n can now be checked in only $(m+1)(n+1)$ pairs of input sequences.

General merging can be done non-obliviously with only $O(n)$ comparisons, how fast can we do this obliviously?

The odd-even merging network is one way of merging obliviously, the definition is recursive. First we merge the elements of each sequence that lie in odd positions and then those that lie in even positions. Then we complete the merge by comparing pairwise the second output of the odd merge with the first of the even, then the third odd with the second even etc.

To see that this is a merging network we apply the zero-one principle inductively. The base case is trivial, for the inductive step

3 Parallel computation models

4 Basic parallel algorithms

5 Further parallel algorithms

6 Parallel matrix algorithms