CPSC 500
Lecture 1: Sept. 4, 2013

Scribe: Kevin Woo

1 Overview

This first lecture covered briefly the course expectations and recapped computational models used as a basis to compare and analyze the complexity of algorithms to be studied later in the course. This is part of the introductory lecture set to provide prospective students a feel for the pacing and expectations for the course, as well as recap some basic algorithm analysis concepts for motivation.

2 Course expectations

Fundamental algorithms will be covered in more depth than undergraduate courses. While the course may have a warm-up cost for students unfamiliar with the prerequisite material, the required readings will be posted so students can read up on them beforehand. Students are expected to keep up with these basic material on their own. Lecture slides will be posted either before or shortly after each class.

This is a broad theory course with many topics covered quickly with depth dependent on time and interest. While the course will mainly follow the textbook chapters (PDF copy available online), it can evolve with student feedback to facilitate “a living and breathing course”. A questionnaire will be handed out in October for voting preferred topics to be covered in addition to the mandatory core material, and which topics to investigate further.

3 Computational models

What is meant by $O(n^3)$ time, $O(n \log n)$ space? Computational (abstract machine) models provide a description of a computer hardware and software system defined over its input, output, and operations [1] usable to analyze the complexity of algorithms running on it. Designs of these abstract models are kept reasonable and not overpowered as to prevent allowing it to compute incomputable things, such as a solution to halting problem.

The measure of complexity varies for each application. For instance, algorithms involving matrix computation may be measured with FLoating-point Operations Per Second (FLOPS), per-row operation time, and space occupied by matrices stored or used in memory. Graph algorithms may be measured by a function of its input number of nodes.

This lecture reviewed two sequential models: the Turing machine and Random-access Machine (RAM).

3.1 Turning machine

Alan Turing’s model describes a machine that reads and manipulates symbols on an input tape via its tape head according to a predefined rule table [2]. The tape contains partitioned memory cells whose values can be changed individually, and thus acting as the machine's storage medium. The internal state of the machine is determined wholly and deterministically by the state machine rules executed sequentially with these read and manipulation operations acting on one adjacent tape section at a time. A more powerful variant, the Universal Turing Machine, takes as additional input a description of another Turing machine that it can execute as a simulation within itself.[1] This separate storage of Turing machine "programs" for later execution is used by the later von Neumann architecture [3]. As modern computers are based on this architecture, a Turing-complete implementation of the model (I/O issues aside), all computation can be executed on a Turing machine and can still be studied with the Turing model. However, due to the unrealistic inefficiency of the mechanism, it is not as popular as the RAM model.

3.2 Random-access machine

The RAM is a register machine with an internal capacity-unbounded integer register storage with the capability for non-sequential to access any location in memory, which is a more realistic and efficient method compared to the sequential memory access of Turing machines that have no built-in ability to skip forward on the tape and no sense of locality (addressing). The RAM model uses these advantages to perform indirect addressing: the ability to refer to data at addresses specified by pointers [4]. The pointer values can be computed and stored for use within its registers, and thus allowing for complex operations such as ADD, STORE, and RECALL much like a modern computer. An advanced variant of the RAM model combines the ability of the Universal Turing Machine to arbitrarily execute stored programs. This is the Random-access Stored-program Machine (RASP) [5], and is itself an actual example of the von Neumann architecture.

3.3 Example: Palindrome

Palindromes are character strings that are unchanged with the characters reversed from left to right. The strings are preprocessed to remove punctuation, symbols, and spaces to include only letters. For example, "Madam, I'm Adam" has the palindrome "MADAMIMADAM" and "Able was I ere I saw Elba" has the palindrome "ABLEWASIEREISAWELBA".

Verifying a palindrome takes $O(n^2)$ time on a Turing machine. This is a result from the necessary traversal of the tape memory containing the input string between the current leftmost to rightmost characters for each comparison. The tape head bounces back and forth until converging to the center of the string. A video demonstrating a Turing machine solving a palindrome is available online [6].

In contrast, a RAM machine can do this in linear $O(n)$ time since it is able to compute the address of each character in memory and access them directly in one step each, thus having only to access each memory cell once. This corresponds to the modern character array access approach when programming in C code.

4 Summary

Following a brief course introduction, two basic computational models were introduced along with a simple example algorithm that can be analyzed with these models. Although small, this should provide an overall motivation for this course. While more algorithms and analysis techniques will be covered in greater detail later on, the take-home message here is that complexity theory serves an important purpose for rigidly evaluating and comparing the performance of algorithms so we can make more informed choices when designing or simply using them in whichever field we study as computer scientists.

[1] I heard you like Turing Machines, so I input a Turing Machine into your Turing Machine so you can simulate a Turing Machine on your Turning Machine

References

[1] Wikipedia, "Abstract machine," [Online]. Available: http://en.wikipedia.org/wiki/Abstract_machine.

[2] Wikipedia, "Turing machine," [Online]. Available: http://en.wikipedia.org/wiki/Turing_machine.

[3] Wikipedia, "Universal Turing machine," [Online]. Available: http://en.wikipedia.org/wiki/Universal_Turing_machine.

[4] Wikipedia, "Random-access machine," [Online]. Available: http://en.wikipedia.org/wiki/Random-access_machine.

[5] Wikipedia, "Random-access stored-program machine," [Online]. Available: http://en.wikipedia.org/wiki/Random-access_stored-program_machine.

[6] YouTube, "Turing Machine Video Tutorial/Demo/Example/Demonstration: Palindrome Detection," [Online]. Available: http://www.youtube.com/watch?v=LqCho9eAbsc.