Complexity

Extended Summary from Alasdair Urquhart's Paper,
Chapter 2 in The Blackwell Guide to the Philosophy of Computing and Information

What is Computational Complexity?

- **The concern**
  The theory of computational complexity is concerned with estimating the resources a computer needs in order to solve a problem.
- **The important resources**
  Time : The number of steps in a computation.
  Space : Amount of memory used for computation.
- **What does the theory signify?**
  Not all problems solvable by a form of Turing machine are solvable in practice.

  Another important finding is that problems fall into strict hierarchies in accordance with their space and time requirements.

The Model for Computation – *Turing Machine*

- **The time and space complexity of a problem depends on the chosen model for computation!**
  The model for the computation for our discussion will be the Turing Machine (from now on).

- **Time**
  Time is defined as the number of steps taken before the (Turing) machine halts.

- **Space**
  Space is defined as the number of cells of tape visited by reading head during the computation.

Time Complexity – *An Example*

- A common standard for machine speed for today's computers is the number of floating-point operations per second (flops).
- Suppose we have a machine of 1Mflops (1 million operations per second).
- Suppose we have various algorithms whose running time are n^2, n^3, and 2^n (where n is the size of the input).
- n^2 algorithm would require quarter of a second for an input size of 500.
- n^3 algorithm would require two minutes and five seconds for an input size of 500.
- 2^n algorithm would require 35 years for an input size of 50 (fifty, not 500, no mistake ☺).
- **The problems which seem to require algorithms of the last type is not rare, nor uninteresting, as one might hope.**

Solvable Problems (Decidability)

- **Alphabet:** A set of symbols. Usually denoted with Σ.
  The set of all finite strings that can be constructed from the symbols in alphabet Σ is denoted by Σ^*.
- **Language (Problem)**
  Let Σ be a finite alphabet. A subset of Σ^* is called a language, or a problem. A string in a problem is called an "instance" (usually just "string" for the language case).
- **Characteristic function**
  Let L be a language. The characteristic function f is defined as:
  $$f(s) = \begin{cases} 0 & s \notin L \\ 1 & s \in L \end{cases}$$
- **Decidable language (Solvable problem)**
  A language is decidable if there is a Turing machine that computes its characteristic function.

Time and Space Bounds of Algorithms

- **The O notation**
  For $f, g : \Re^2$, $f(x)$ is said to be $O(g(x))$ if there exists a constant c such that for sufficiently large x, $f(x) < c \cdot g(x)$
- If f is a computable function, whose Turing machine takes $O(T(n))$ steps to halt with the output for an input of n symbols, then the function f is said to be computable in time $T(n)$.
- The same applies to space: if f is a computable function, whose Turing machine visits $O(T(n))$ different places before halting with the output for an input of n symbols, then the function f is said to be computable in time $T(n)$.
- **The definitions are introduced here in a not-so-rigorous manner.**

Complexity Classes

- **The class P**
  The class P (which stands for "polynomial") includes those problems whose running time is $O(p(n))$ for some polynomial $p(n)$.
- **The class PSPACE**
  The class PSPACE includes those problems whose space requirement is $O(p(n))$ for some polynomial $p(n)$.
- **The classes EXPSPACE and EXPTIME**
  The classes EXPSPACE and EXPTIME contains those problems whose space or time requirement is $O(2^{n^k})$ where k is a constant.
- **The class NP**
  The class NP contains problems whose solution can be tested in a polynomial time, whereas the number of candidate solutions is exponential.
- **The core problem in CS: Is P=NP?**

The Hierarchy Theorems

- **Space constructible functions**
  A function $S(n)$ is said to be space constructible if there is a Turing machine M that is $S(n)$ space bounded, and for each n there is an input of length n on which M actually uses $S(n)$ tape cells.
- **Space hierarchy theorem (Hartmanis, Lewis, and Stearns, 1965)**
  If $S_1(n)$ and $S_2(n)$ are space constructible functions, and S_2 grows faster than S_1 asymptotically, so that
  $$\liminf_{n\to\infty} \frac{S_1(n)}{S_2(n)} = 0$$
  then there exists a problem solvable in space $S_2(n)$, but not in space $S_1(n)$.
- **Time hierarchy theorem**
  There exist a similar theorem for the time hierarcy.

Effective Reduction

- **For what purpose is reduction used?**
  Reduction of a language (problem) L_1 to L_2 is used to show that L_2 is at least as hard as L_1.
- **Formalization of effective reduction**
  Let L_1 and L_2 be problems expressed in alphabets Σ_1 and Σ_2. L_1 is said to be polynomial-time reducible (or just reducible) to L_2 if there is a polynomial-time computable function f from Σ_1^* to Σ_2^* such that for any string s in Σ_1^*, s is in L_1 if and only if $f(s)$ is in L_2.
- **Completeness**
  If C is a complexity class, and L is a problem in C so that any problem in C is reducible to L, then L is said to be C-complete.

Cook's Theorem and NP Completeness

- **Many interesting problems are in NP.**
- **The satisfiability problem**
  In mathematics, a formula of propositional logic is said to be satisfiable if truth values can be assigned to its free variables in a way that makes the formula true. The satisfiability problem is deciding whether a given formula is satisfiable or not.
- **Cook's theorem:**
  The satisfiability problem is NP-complete.
- The satisfiability problem was the first proven NP-complete problem. Using reduction, many others have been shown to be NP-complete.
- Note that NP and EXPTIME are not the same classes. See http://www.csee.umbc.edu/help/theory/classes.shtml

Complexity and Philosophy – *The Imitation Game*

- Philosophical treatments of the concept of computation often ignores issues related to complexity.

- **Turing's imitation game**
  There are two rooms that are completely isolated from one another except for a teletype connection between them. In one room there is a human with a terminal, and a computer. In the other, there is an interrogator whose purpose is to carry out a conversation with both parties for five minutes to understand which is computer and which is the human.

  "I believe that in about fifty years' time it will be possible to program computers, with a storage capacity of about 10^9, to make them play the imitation game so well that an average interrogator will not have more than 70 percent chance of making the right identification after five minutes of questioning." (Turing 1950)

Complexity and Philosophy – *The Imitation Game*

- **A machine that would pass the Turing test**
  Suppose there is a limit on the number of symbols exchanged instead of the original "five minutes" limit. Consider all sequences of symbols representing a possible sequence of questions and answers in the imitation game. Some of these sequences are good in the sense that the operator have no more chance than random guess. Then a Turing machine (in mathematical sense) can be constructed from the table of such good responses, which would, by definition, pass the Turing's test.
- **What exactly is the problem?**
  Ignorance of the complexity. The complexity of the problem is so high that we cannot solve it in the brute-force way described above, with any known means.
  Machines in mathematical abstract sense are not always realizable (at least with current state-of-technology).

Complexity and Philosophy

- **Imaginary experiments**
  Frequently, philosophical discussions on the foundations of cognitive science revolves around implausible thought experiments like Searle's "Chinese room" argument.

  Arguments based on such imaginary experiments appear quite powerful, until one considers also the computational resources needed.

- Thus it is extremely important to distinguish between objects that exist in the purely mathematical sense (like the Turing machine that succeeds in the imitation game), and the physically plausible resource bounded machines.

References

- The second chapter and all its references ☺
- http://www.fact-index.com/, search for "complexity theory for computation". Many good definitions. Don't get lost ;)

Complexity – *Too complex to ever be fully finished…*

ANY QUESTIONS?