CS472 Module 10 Part A - What is computational complexity?

Athens State University

March 28, 2016

Algorithm Analysis vs. Computational Complexity

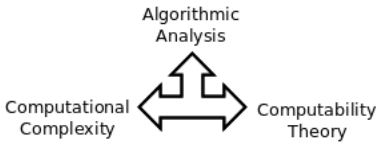
Computational complexity branch of computer science that focuses on classifying computational problems according to their inherent difficulty, and relating those classes to each other

- Study of all possible algorithms that solve a given problem
- Determine a lower bound on efficiency of all algorithms for a given problem
- Problem analysis as opposed to algorithm analysis

How to measure complexity?

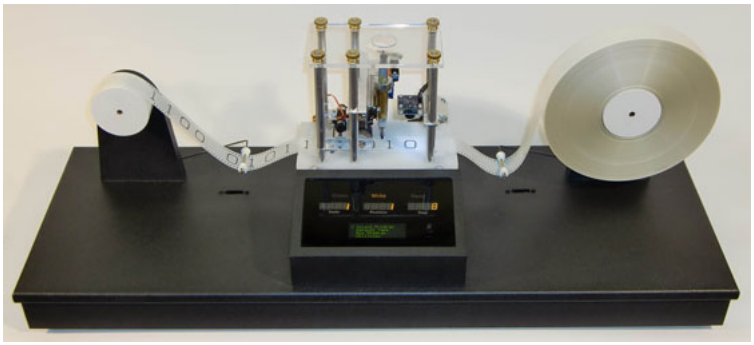
- A problem is difficult if its solution requires significant resources whatever the algorithm used

Algorithm Analysis vs. Computational Complexity



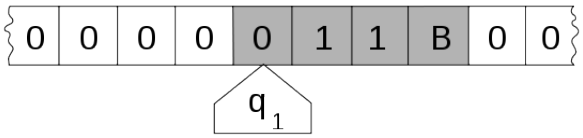
- **Algorithm analysis**: determine amount of resources required by a particular algorithm to solve problem
- **Complexity theory**: What algorithms for solving a problem can be found for a problem given a restricted set of resources?
- **Computability theory**: What can of problems can, in principle be solved using algorithms?

Some results from computability theory: Turing Machines



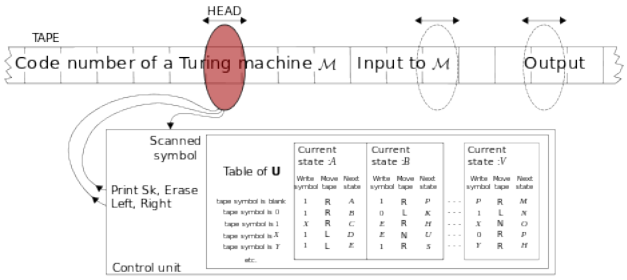
A **Turing Machine** is hypothetical device that manipulates symbols on strip of table according to a table of rules. A Turing machine can be adapted to simulate the logic of any computer algorithm.

Computability theory: Operation of a Turing Machine



There is a mathematical notation that can be used to describe the structure and behavior of a Turing machine. The details of this notation may be of interest but is not required for this discussion.

Computability theory: A Universal Turing Machine



A **Universal Turing Machine** (UTM) is a Turing machine that can simulate an arbitrary Turing machine running on any arbitrary input.

There are results in the theory of computer science that says a UTM can calculate any recursive function, decide any recursive language, and accept any recursively enumerable language. In particular, the Church-Turing thesis says the problems solvable by a UTM are exactly those problems solvable by an algorithm for any reasonable definition of the term.

Computability theory and Computational Complexity

- There will be some results from the theory of computation that we will take as assumptions in this discussion:
  - The *Church-Turing thesis* states that if a problem can be solved by an algorithm, then there exists a *Turing machine* that can solve the problem.
    - It can be shown that modern digital computers are, in fact, a form of a Turing machine.
  - Every model of computation identified to date can be converted to a Turing machine (or vice versa) without providing any extra computational power.

Some terms

Computational problem A (possibly infinite) collection of *instances* together with a *solution* for each instance

Problem instance A string composed of items from an alphabet. Typically, we work with *bitstrings* taken from the binary alphabet $0, 1^*$

- Objects other bitstrings must be suitably encoded; for example, integers can be represented in binary notation, graphs by encoding their adjacency matrix or adjacency list

Decision problem vs. function problem

Decision problem Any arbitrary yes-or-no question on an infinite set of inputs

- Example: "Given two numbers x and y, does x evenly divide y?"
- A decision problem that can be solved by an algorithm is called a *decidable* problem

Function problem A computational problem where a single output is expected for every input, but the output is more complex than that of a decision problem

- Example: The Traveling Salesman Problem

Function problems are just decision problems

- It is possible, perhaps with a representation change, to recast a function problem into an integer
- For example, the multiplication of two integers can be expressed as a set of triples (a, b, c) such that $a * b = c$. Deciding if a given triple is a member of this set corresponds to solving the problem of multiplying two integers

Measuring the size of an instance

- To reason about what it means to solve a problem using a given amount of time and space, we need:
  - A mathematical model of computation such as the deterministic Turing machine
  - Some measure of the number of steps such a machine will take in order for the machine to halt and output the answer to decision problem of interest.
- Other measures of complexity exist but in general we can operate with the asymptotic behavior of input size

Measuring the size of an instance

- Algorithm analysis uses the rate-of-growth functions to reason about the performance of algorithms
- For algorithm analysis, we look at upper bound on time complexity of algorithms
- For computational complexity, we look towards the lower bound on the amount of time an algorithm requires to solve an instance