Algorithms

A precise sequence of instructions for problem solving

Alwin Tareen

The Definition of an Algorithm

- Algorithms are precise sequences of instructions for processes that can be executed by a computer, and are implemented using programming languages.

Some commonly known algorithms are:

- Luhn's Algorithm: Performs credit card number validation.
- Deflate: Executes lossless data compression.
- PageRank: Google's method of measuring a web page's reputation.

Sequencing, Selection and Iteration

Every algorithm in existence can be constructed using some combination of the following 3 concepts:

- **Sequencing:** application of each step of an algorithm in some particular order.
- **Selection:** Using boolean conditions to perform decision making capabilities.
- **Iteration:** Repeating a particular section of code a certain number of times, or until some condition is met.

Properties of Algorithms

- Algorithms can be combined to make new algorithms.
- The language used to express an algorithm should be general enough such that it can be implemented in any high-level programming language.
- Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure that the new algorithm is correct.

Properties of Algorithms

- Knowledge of standard algorithms can assist with constructing new algorithms.
- Different algorithms can be developed to solve the same problem.
- Developing a new algorithm to solve a problem can yield significant insight into the problem itself.

Expressing Algorithms

Algorithms can be expressed using:

- The English language
- Pseudocode
- A flowchart
- Generally, any method that conveys information.

Algorithms are conceptual definitions of how to accomplish a task, usually written in pseudocode.

Analysis of Algorithms

- When we analyze an algorithm, we count the number of steps that are executed.
- **Running time** or **Runtime**: This refers to the total number of steps or operations that are executed by the algorithm. Note that this is not the same concept as the clock duration.
- Generally, the **worst case** input configuration is used to analyze an algorithm. This gives us an upper bound on the expected runtime.

Analysis of Algorithms

- When we analyze algorithms in terms of their order of growth, we generally consider the dominant terms, and disregard the lower order terms.
- For example, $10n^2 + 4 \log n + n$ would be considered an n^2 algorithm.
- An algorithm is considered to be correct if, for every input, it calculates the correct output, and doesn't run forever, or cause an error.

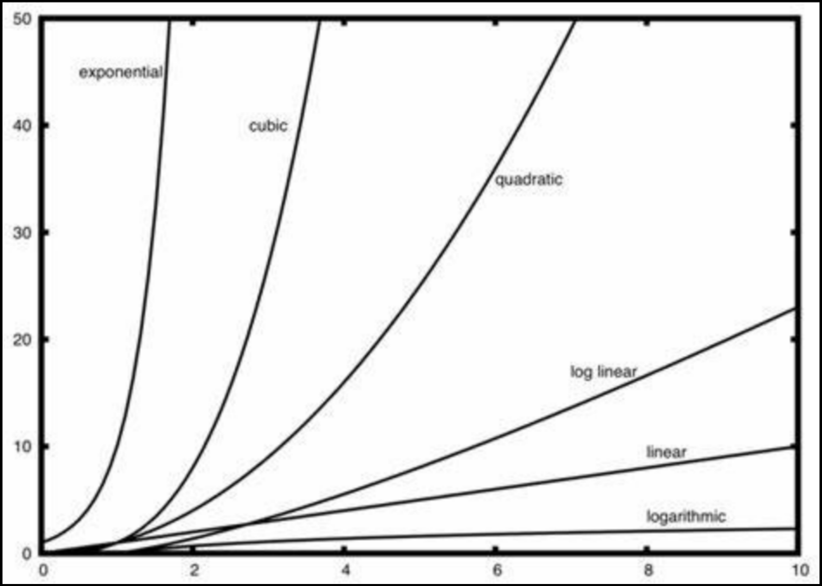
Analysis of Algorithms

We can rank the runtime of an algorithm by how quickly it grows with respect to the input n.

- Constant runtime: $O(1)$
- Logarithmic runtime: $O(\log n)$
- Linear runtime: $O(n)$
- Polynomial runtime: $O(n^c)$
- Exponential runtime: $O(c^n)$

Note that the symbol $O(n)$ is known as **big-oh** notation. It is not a zero, it is actually the Greek letter omnicron. You can think of it as meaning, "On the order of."

Runtimes of Algorithms



Algorithmic Complexity

Problems can be categorized in the following manner:

- Some problems are **tractable**: they can be solved with efficient solutions, in a reasonable amount of time.
- Some problems are **intractable**: they cannot be solved in a reasonable amount of time, regardless of the solution methodology.
- Some problems can be solved **approximately**, but not precisely, in a reasonable amount of time.
- Some problems have no known efficient solution.
- Some problems are not solvable.

Tractable Problems

Efficient solutions with a reasonable solution time

- The order of growth is polynomial with respect to the size of the problem.

Examples in this class of problems include:

- Searching for an element in a list.
- Sorting a list.

Typical runtimes include:

- $O(1)$, $O(\log n)$, $O(n)$, $O(n^c)$

Intractable Problems

No reasonable solution time

- Solutions do exist for this class of problems, but they cannot be solved in a reasonable amount of time.

Examples in this class of problems include:

- Factoring a number into its constituent primes.
- Boolean satisfiability.

Typical runtimes include:

- $O(c^n)$

Problems with Approximate Solutions

Calculating estimates

- Generally, you cannot find the exact answer in this class of problems, but you can calculate an estimate in a reasonable amount of time.
- The solution may involve the use of a **heuristic**: a non-conventional insight, or a clever simplification that provides a solution method.

Examples in this class of problems include:

- The travelling salesman problem.
- The knapsack problem.

Problems with No Known Efficient Solution

Extremely difficult problems

- Some problems are so difficult, that there are currently no known methods to solve them.

Examples in this class of problems include:

- The subset sum problem: Given the following set of numbers, are there a handful of these numbers(at least 1) that add together to get 0?

-2	-3	15
14	7	-10

Decidable and Undecidable Problems

Decidable problems

- A problem is **decidable** if it can produce a yes or no answer for any combination of inputs.

Undecidable problems

- A problem is **undecidable** if it cannot produce a yes or no answer for all combination of inputs.

Algorithms: End of Notes