Limits Of Computing

Chapter 17

Complexity of Software

- Commercial software contains errors
  - The problem is *complexity*
  - Software testing can demonstrate the presence of bugs but *cannot demonstrate their absence*
    - As we find problems and fix them, we raise our confidence that the software performs as it should
    - But we can never guarantee that all bugs have been removed

Software Engineering

- **Software requirements** Broad, but precise, statements outlining what is to be provided by the software product
- **Software specifications** A detailed description of the function, inputs, processing, outputs, and special features of a software product

Software Engineering

- A guideline for the number of errors per lines of code that can be expected
  - Standard software: 25 bugs per 1,000 lines of program
  - Good software: 2 errors per 1,000 lines
  - Space Shuttle software: < 1 error per 10,000 lines

Notorious Software Errors

- Mariner 1 Venus Probe

This probe, launched in July of 1962, veered off course almost immediately and had to be destroyed

The problem was traced to the following line of Fortran code:

```
DO 5 K = 1. 3
```

The period should have been a comma.

An $18.5 million space exploration vehicle was lost because of this typographical error

Notorious Software Errors

- Denver baggage handling system - was so complex (involving 300 computers) that the development overrun prevented the airport from opening on time. Fixing the incredibly buggy system required an additional 50% of the original budget - nearly $200m.
- The 2003 North America blackout - was triggered by a local outage that went undetected due to a race condition in General Electric Energy's XA/21 monitoring software.
- FBI in 2005 - $170 million FBI project to update their case management system.

Notorious Software Errors

- Mars Climate Orbiter (1999) - The 125 million dollar Mars Climate Orbiter is assumed lost by officials at NASA. The failure responsible for loss of the orbiter is attributed to a failure of NASA's system engineer process. The process did not specify the system of measurement to be used on the project. As a result, one of the development teams used Imperial measurement while the other used the metric system of measurement. When parameters from one module were passed to another during orbit navigation correct, no conversion was performed, resulting in the loss of the craft. http://mars.jpl.nasa.gov/msp98/orbiter/

Formal Verification

- The verification of program correctness, independent of data testing, is an important area of theoretical computer science research
- Formal methods have been used successfully in verifying the correctness of computer chips
- It is hoped that success with formal verification techniques at the hardware level can lead eventually to success at the software level

Big-O Analysis

- A function of the size of the input to the operation (for instance, the number of elements in the list to be summed)
- We can express an approximation of this function using a mathematical notation called order of magnitude, or **Big-O notation**

Big-O Analysis

$$f(N) = N^4 + 100N^2 + 10N + 50$$

- Then $f(N)$ is of order N^4—or, in Big-O notation, $O(N^4)$.
- For large values of N, N^4 is so much larger than 50, $10N$, or even $100\,N^2$ that we can ignore these other terms

Big-O Analysis

- Common Orders of Magnitude
  - *$O(1)$ is called bounded time*
    - Assigning a value to the ith element in an array of N elements
  - $O(log_2N)$ is called logarithmic time
    - Algorithms that successively cut the amount of data to be processed in half at each step typically fall into this category
    - Finding a value in a list of sorted elements using the binary search algorithm is $O(log_2N)$

Big-O Analysis

- $O(N)$ is called linear is called linear time
  - Printing all the elements in a list of N elements is $O(N)$
- $O(N \log_2 N)$
  - Algorithms of this type typically involve applying a logarithmic algorithm N times
  - The better sorting algorithms, such as Quicksort, Heapsort, and Mergesort, have $N \log_2 N$ complexity

Big-O Analysis

- $O(N^2)$ is called quadratic time
  - Algorithms of this type typically involve applying a linear algorithm N times. Most simple sorting algorithms are $O(N^2)$ algorithms
- $O(2^N)$ is called exponential time

Big-O Analysis

- $O(n!)$ is called factorial time
  - The traveling salesperson graph algorithm is a factorial time algorithm
  - Algorithms whose order of magnitude can be expressed as a polynomial in the size of the problem are called polynomial-time algorithms
  - All polynomial-time algorithms are defined as being in Class P

Big-O Analysis

N	log_2N	$Nlog_2N$	N^2	N^3	2^N
1	0	1	1	1	2
2	1	2	4	8	4
4	2	8	16	64	16
8	3	24	64	512	256
16	4	64	256	4,096	65,536
32	5	160	1,024	32,768	4,294,967,296
64	6	384	4,096	262,144	About 5 years' worth of instructions on a supercomputer
128	7	896	16,384	2,097,152	About 600,000 times greater than the age of the universe in nano-seconds (for a 6-billion-year estimate)
256	8	2,048	65,536	16,777,216	Don't ask!

Table 17.2
Comparison of rates of growth

Big-O Analysis

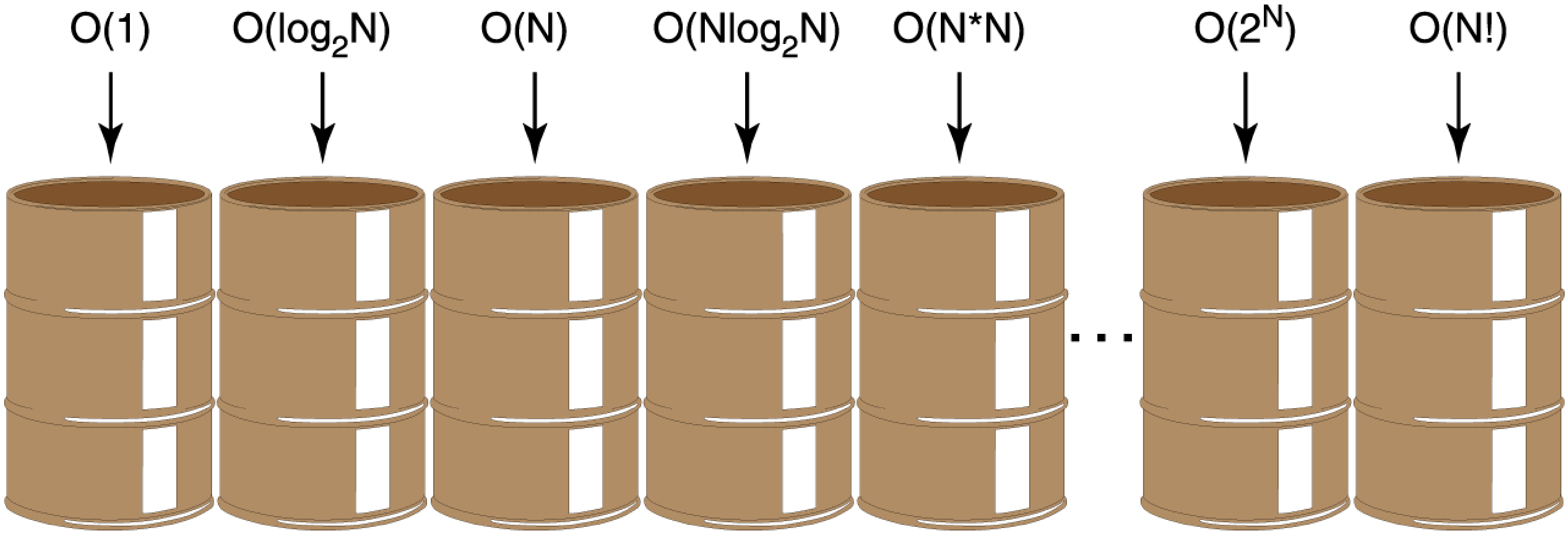


Figure 17.3 **Orders of complexity**

Turing Machines

- Alan Turing developed the concept of a computing machine in the 1930s
- A Turing machine, as his model became known, consists of a control unit with a read/write head that can read and write symbols on an infinite tape

Turing Machines

Figure 17.4 **Turing machine processing**

- Why is such a simple machine (model) of any importance?
  - It is widely accepted that *anything that is intuitively computable can be computed by a Turing machine*
  - If we can find a problem for which a Turing-machine solution can be proven not to exist, then the problem must be unsolvable

Halting Problem

- It is not always obvious that a computation (program) halts
- The **Halting problem:** Given a program and an input to the program, determine if the program will eventually stop with this input
- This problem is unsolvable

Halting Problem

- Assume that there exists a Turing-machine program, called **SolvesHaltingProblem** that determines for any program **Example** and input **SampleData** whether program **Example** halts given input **SampleData**

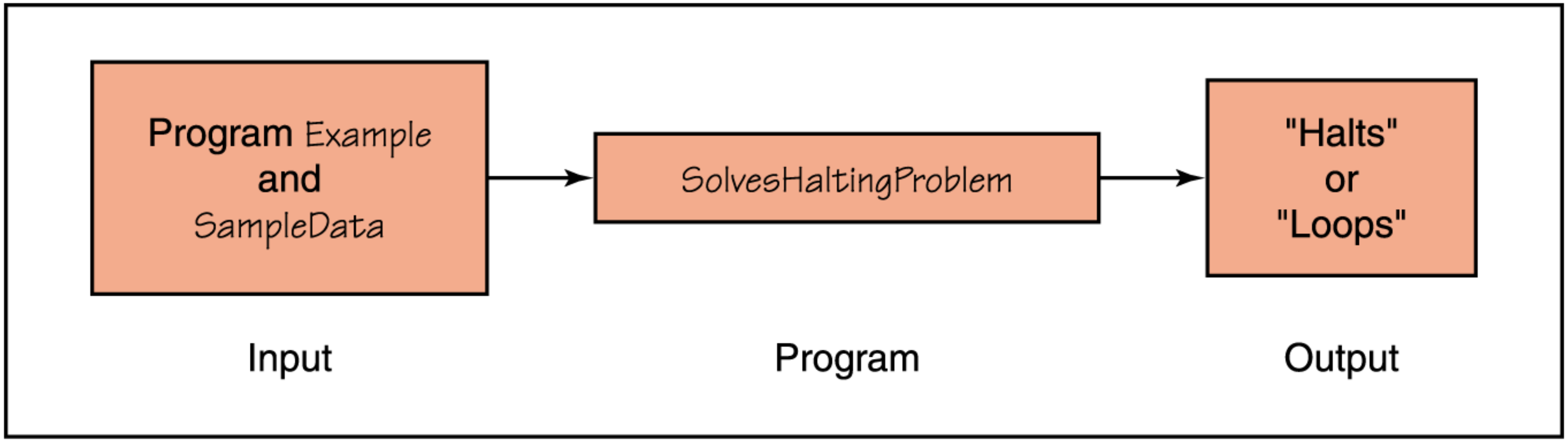


Figure 17.5 **Proposed program for solving the Halting problem**

Halting Problem

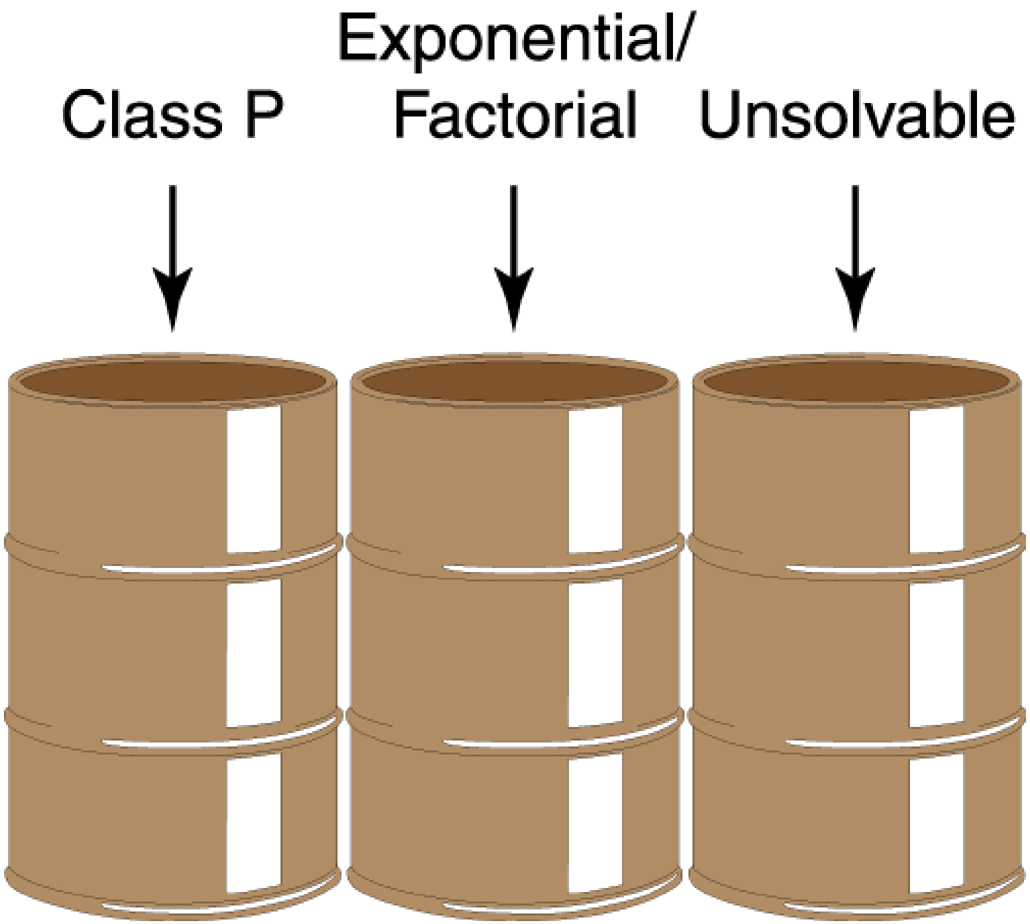


Figure 17.8 A reorganization of algorithm classification

- Let’s reorganize our bins, combining all polynomial algorithms in a bin labeled **Class P**

Halting Problem

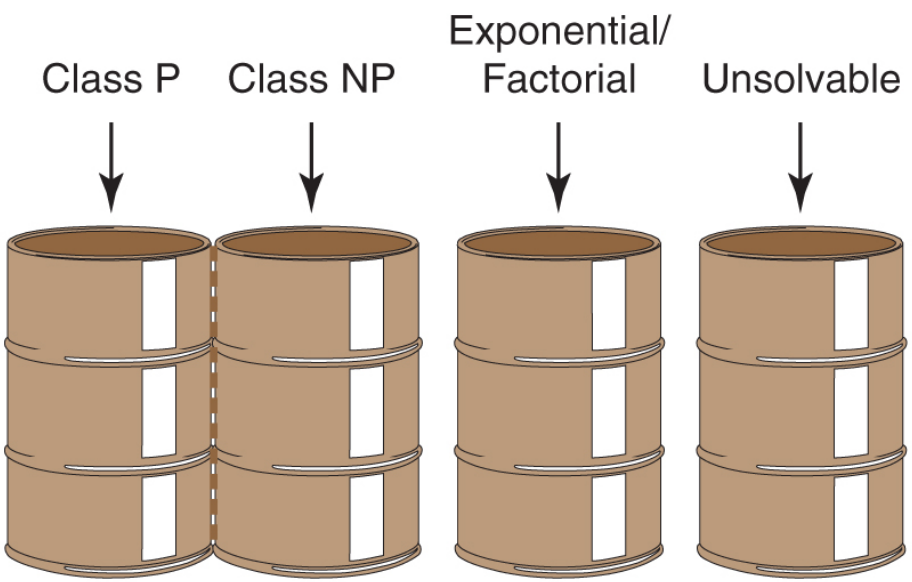


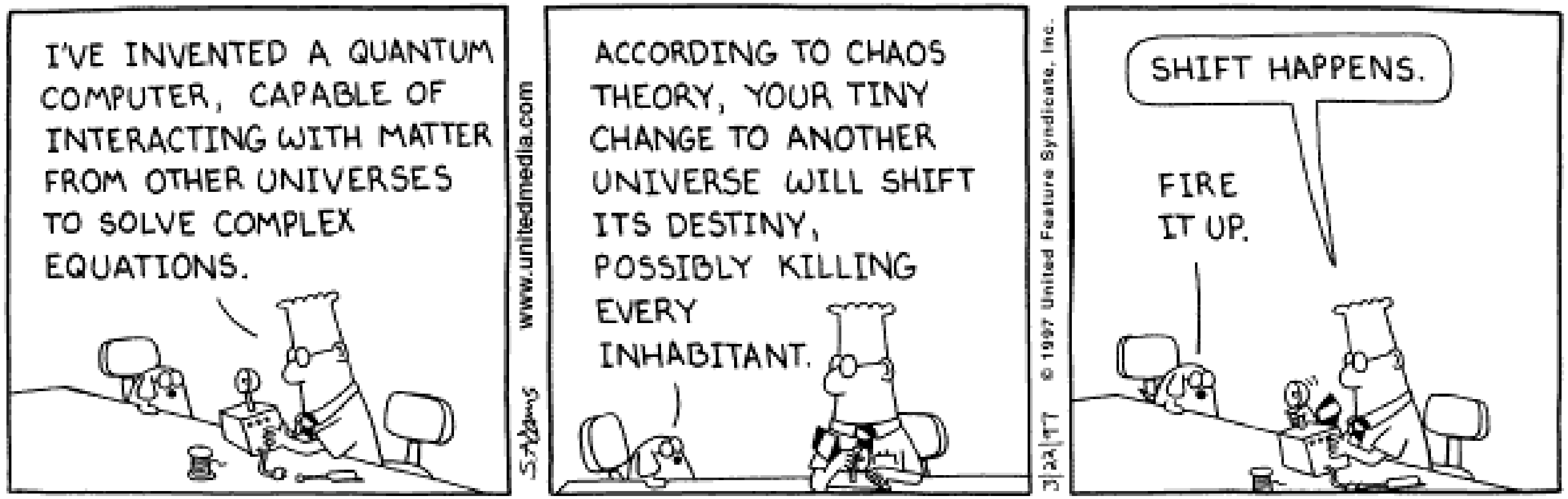
Figure 17.9 **Adding Class NP**

- The algorithms in the middle bin have known solutions, but they are called *intractable* because for data of any size they simply take too long to execute
- A problem is said to be in **Class NP** if it can be solved with a sufficiently large number of processors in polynomial time

The Promise...

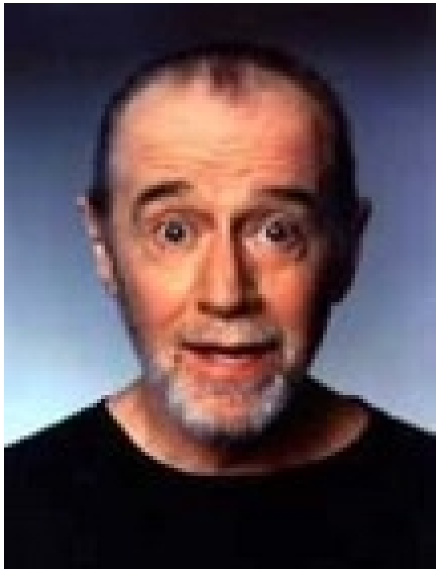
- Quantum Computing – subatomic level; great promise for cryptography
- Photonic Computing – photons replace electrons; no wires!
- Biological Computing - use of living organisms or their components, e.g. DNA strands, to perform computing operations

Dilbert Said It Well…



...But George Carlin Said It Better!!!

A Modern Man



Final Exam

- **Take Home Exam** (on web site)
- Six Questions – Answer All of Them
- **Due December 18 or Earlier!**
- **Absolutely No Lateness**
- All other assignments must be in before the 11th
- **Next Class – Monday, 12/11** – Short Lecture On Limitations of Computing

La commedia e finita' …

…Good Luck…Make A Difference!!!