Chapter 17
Limitations of Computing

Chapter Goals
• Describe the limits that the hardware places on the solution to computing problems
• Discuss how the finiteness of the computer impacts the solutions to numeric problems
• Discuss ways to ensure that errors in data transmission are detected
• Describe the limits that the software places on the solutions to computing problems

Chapter Goals
• Discuss ways to build better software
• Describe the limits inherent in computable problems themselves
• Discuss the continuum of problem complexity from problems in Class P to problems that are unsolvable

Limits on Arithmetic

Precision
The maximum number of significant digits that can be represented

With 5 digits precision, the range of the numbers we can represent is -9,999 through +9,999

\[
\begin{array}{c|c}
\text{Largest negative number} & -99999 \\
\text{Zero} & 00000 \\
\text{Largest positive number} & +99999 \\
\end{array}
\]

What happens if we allow one of these digits (let’s say the leftmost one, in red) to represent an exponent?

For example:

\[
\begin{array}{c|c}
\text{represents the number} & +2.345 \times 10^3 \\
\end{array}
\]

Limits on Arithmetic
The range of numbers we can now represent is much larger

\[-9.999 \times 10^9 \text{ to } +9.999 \times 10^9\]

but we can represent only four significant digits

Significant digits
Those digits that begin with the first nonzero digit on the left and end with the last nonzero digit on the right
Limits on Arithmetic

The four leftmost digits are correct, and the balance of the digits are assumed to be zero.

We lose the rightmost, or least significant, digits.

To represent real numbers, we extend our coding scheme to represent negative exponents.

For example

\[4,394 \times 10^{-2} = 0.4394\]

or

\[22 \times 10^{-4} = 0.0022\]

Let current sign be the sign of the exponent and add a sign to the left to be the sign of the number itself.

What is the largest negative number? The smallest positive number? The largest positive number? The smallest negative number?

Representational error or round-off error

An arithmetic error caused by the fact that the precision of the result of an arithmetic operation is greater than the precision of the machine.

Underflow

Results of a calculation are too small to represent in a given machine.

Overflow

Results of a calculation are too large to represent in a given machine.

Cancellation error

A loss of accuracy during addition or subtraction of numbers of widely differing sizes, due to the limits of precision.

Give examples of each of these errors.

There are limitations imposed by the hardware on the representations of both integer numbers and real numbers.

– If the word length is 32 bits, the range of integer numbers that can be represented is 2,147,483,648 to 2,147,483,647.

– There are software solutions, however, that allow programs to overcome these limitations.

– For example, we could represent a very large number as a list of smaller numbers.

Figure 17.1

Representing very large numbers.
Limits on Components

Although most errors are caused by software, hardware components do fail

Have you ever had a hardware failure?

Limits on Communications

Error-detecting codes
Techniques to determine if an error has occurred during the transmission of data and then alert the system

Error-correcting codes
Error-detecting codes that try to determine the correct value

Limits on Communications

Parity bit
An extra bit that is associated with each byte, used to ensure that the number of 1 bits in a 9-bit value (byte plus parity bit) is odd (or even) across all bytes
Parity bits are used to detect that an error has occurred between the storing and retrieving of a byte or the sending and receiving of a byte

Limits on Communications

Odd parity requires the number of 1s in a byte plus the parity bit be odd
For example
- If a byte contains the pattern 11001100, the parity bit would be 1, thus giving an odd number of 1s
- If the pattern were 11110001, the parity bit would be 0, giving an odd number of 1s

Even parity uses the same scheme, but the number of 1 bits must be even

Limits on Communications

Check digits
- A software variation of the same scheme is to sum the individual digits of a number and store the unit’s digit of that sum with the number
- For example, given the number 34376, the sum of the digits is 23, so the number would be stored as 34376–3

Error-correcting codes
- If enough information about a byte or number is kept, it is possible to deduce what an incorrect bit or digit must be

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
Remember the four stages of computer problem solving?
- Write the specifications
- Develop the algorithm
- Implement the algorithm
- Maintain the program

Moving from small, well-defined tasks to large software projects, we need to add two extra layers on top of these: Software requirements and specifications.

Software requirements
A statement of what is to be provided by a computer system or software product

Software specifications
A detailed description of the function, inputs, processing, outputs, and special features of a software product; it provides the information needed to design and implement the software.

Testing techniques have been a running thread throughout this book. They are mentioned here again as part of software engineering.

Can you define walk-throughs and inspections?

Use of SE techniques can reduce errors, but they will occur.

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

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.

AT&T Down for Nine Hours
In January of 1990, AT&T’s long-distance telephone network came to a screeching halt for nine hours, because of a software error in an upgrade to the electronic switching systems.
Notorious Software Errors

Therac-25
- Between June 1985 and January 1987, six known accidents involved massive overdoses by the Therac-25, leading to deaths and serious injuries.
- There was only a single coding error, but tracking down the error exposed that the whole design was seriously flawed.

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:
  
  ```fortran
  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.

Comparing Algorithms

How can we compare algorithms?

Big-O notation

A notation that expresses computing time (complexity) as the term in a function that increases most rapidly relative to the size of a problem.

Function of size factor $N$:

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

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.
Common Orders of Magnitude

– \(O(1)\) is called bounded time
  Assigning a value to the \(i\)th element in an array of \(N\) elements
– \(O(\log_2 N)\) 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 \(O(\log_2 N)\)

– \(O(N)\) 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

– \(O(N^2)\) 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
– \(O(n!)\) is called factorial time
  The traveling salesperson graph algorithm is a factorial time algorithm

Table 17.2

| \(n\) | \(\log_2 n\) | \(\log_2^2 n\) | \(n\) | \(n^2\) | \(2^n\)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>64</td>
<td>256</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>8</td>
<td>64</td>
<td>512</td>
<td>2048</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>16</td>
<td>256</td>
<td>4096</td>
<td>65536</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>32</td>
<td>1024</td>
<td>65536</td>
<td>(4,294,967,296)</td>
</tr>
<tr>
<td>64</td>
<td>6</td>
<td>64</td>
<td>4,096</td>
<td>2,147,483,648</td>
<td>262,144</td>
</tr>
<tr>
<td>128</td>
<td>7</td>
<td>128</td>
<td>8,192</td>
<td>16,777,216</td>
<td>4,294,967,296</td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>256</td>
<td>32,768</td>
<td>68,719,622,352</td>
<td>(9,007,199,254,740,992)</td>
</tr>
</tbody>
</table>

\(n\)'s are beyond our computer's capacity

Don't ask!

Big-O Analysis

Turing Machines

Turing machine

A model Turing developed in the 1930s, that consists of a control unit with a read/write head that can read and write symbols on an infinite tape
Turing Machines

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

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 \texttt{SolvesHaltingProblem} that determines for any program \texttt{Example} and input \texttt{SampleData} whether program \texttt{Example} halts given input \texttt{SampleData}

Now let’s construct a new program, \texttt{NewProgram}, that takes program \texttt{Example} as both program and data and uses the algorithm from \texttt{SolvesHaltingProblem} to write "Halts" if \texttt{Example} halts and "Loops" if it does not halt

Let’s now apply program \texttt{SolvesHaltingProblem} to \texttt{NewProgram}, using \texttt{NewProgram} as data

– If \texttt{SolvesHaltingProblem} prints "Halts", program \texttt{NewProgram} goes into an infinite loop
– If \texttt{SolvesHaltingProblem} prints "Loops", program \texttt{NewProgram} prints "Halts" and stops
– In either case, \texttt{SolvesHaltingProblem} gives the wrong answer
Classification of Algorithms

Polynomial-time algorithms
Algorithms whose order of magnitude can be expressed as a polynomial in the size of the problem are called

Class P
Problems that can be solved with one processor in polynomial time

NP-complete problems
Problems that can be solved in polynomial time with as many processors as desired

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

Ethical Issues

Deep Linking
What is deep linking?
Have you ever experienced it?
How was this issue decided by a court?
What do you think should be done about it?

Who am I?

What machine and what award are named after me?
Why am I on Time’s list of the 100 most influential persons in the 20th century?

Do you know?

Why many lines of software are there for the latest commercial planes?
Why did Dijkstra criticize the use of the term "bugs"?
What is a micro-tagant? How is it used?
What is the traveling salesman problem?