Limitations of Computing

Instructor: Dmitri A. Gusev

Spring 2007

CSC 120.02: Introduction to Computer Science

Lecture 18, May 1, 2007

Limits on Arithmetic

Significant digits begin with the first nonzero digit on the left and end with the last nonzero digit on the right (or a zero digit that is exact). Examples:

100 + 110 = 210
$1*10^2 + 1.1*10^2 = 2*10^2$
123.25 + 46.0 + 86.26 = 255.5

The numbers are assumed to be **measurements** (and therefore probably inexact), so "$1*10^2$" above represents an inexact measurement with only one significant digit!

The maximum number of significant digits (bits) that can be represented is called *precision*. More bits to represent the exponent -> wider range (max – min), but lower precision.

Limits on Arithmetic (cont'd)

Spurious accuracy: If a sprinter is measured to have completed a 100.0 m race in 11.71 seconds, what is his average speed? By dividing the distance by the time using a calculator, we get a speed of 8.53970965 m/s. Obviously, the speed of the sprinter is not known to the nearest 10 nm/s, so it is more sensible to report it to four significant figures (8.540 m/s), because the time is only measured to four significant figures.

A *representational (round-off) error* is the difference between the calculated approximation of a number and its exact mathematical value. It may occur if the precision of the result of an arithmetic operation is greater than the precision of our machine.
30.539*42.736 = 1305.114704 ~ 1305.1

Limits on Arithmetic (cont'd)

Underflow is the condition that occurs when the absolute value of the result of a calculation is too small to represent in a given machine. A non-zero value is then replaced with 0.

Overflow: The result of a calculation is 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 limits of precision. Example of how it looks like: (132+0.000000000000021)-131=1

Limits on Communications

Error-detecting codes determine that an error has occurred during the transmission of data and alert the system. “Please re-transmit this packet.”

Error-correcting codes not only determine that an error has occurred but try to determine what the correct value is.

Odd parity bit ensures that the number of 1s in a unit (say, byte) plus parity bit is odd. If the number is even when the data is retrieved/received, an error has occurred.

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

Check digits (*check sums*): Store the sum of the individual digits of a number with the number.

Reed–Solomon error correction is an error-correcting code that works by oversampling a polynomial constructed from the data. The polynomial is evaluated at several points, and these values are sent or recorded. By sampling the polynomial more often than is necessary, the polynomial is over-determined. As long as "many" of the points are received correctly, the receiver can recover the original polynomial even in the presence of a "few" bad points. Reed–Solomon codes are used in a wide variety of commercial applications, most prominently in CDs and DVDs.

Limitations of Software

A software *bug* is an error, flaw, mistake, failure, or fault in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result). Most bugs arise from mistakes and errors made by people in either a program's source code or its design, and a few are caused by compilers producing incorrect code.

Complexity: Very many lines of code and/or complicated program structure. (Not to be confused with computational complexity!)

Software testing can demonstrate the presence of bugs but **cannot prove their absence**.

Computational Complexity

Big-O notation expresses computing time (~ the number of operations) as the term in a function that increases most rapidly relative to the size of a problem (N). (Can do *average*, *best case*, *worst case* analysis.)

O(1) is called *bounded time*. The amount of work is bounded by a constant.

$O(\log_2 N)$ is called *logarithmic time*. Example: Binary search.

O(N) is called *linear time*. Example: Sequential search.

$O(N \log_2 N)$ is called *N $\log_2 N$ time.* Applying a logarithmic algorithm N times.

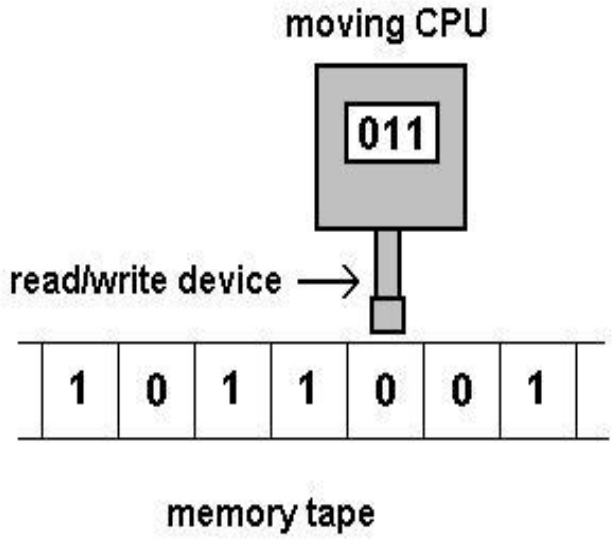
$O(N^2)$ is called *quadratic time*. $O(N^3)$ is called *cubic time*.

Polynomial-time (Class P) algorithms: Algorithms whose complexity can be expressed as a polynomial in the size of the problem.

$O(2^N)$ is called *exponential time*.

O(N!) is called *factorial time*. The traveling salesman problem.

Turing Machines



Each instruction causes:

- A symbol to be read from a cell on the tape
- A symbol to be written into the cell
- The tape to be moved one cell left, to be moved one cell right, or to remain positioned as it was

The Church-Turing thesis: Anything that is intuitively computable can be computed by a Turing machine.

Halting Problem

The *halting problem* is the problem of determining whether any program will eventually stop given a particular input. It is *unsolvable*. ☹

NP and NP-Complete Problems

- Class NP problems: Problems that can be solved in polynomial time with as many processors as desired.
- NP-complete problems: A class of problems within Class NP such that if a polynomial time solution with one processor can be found for any member of the class, such a solution exists for every member of the class.