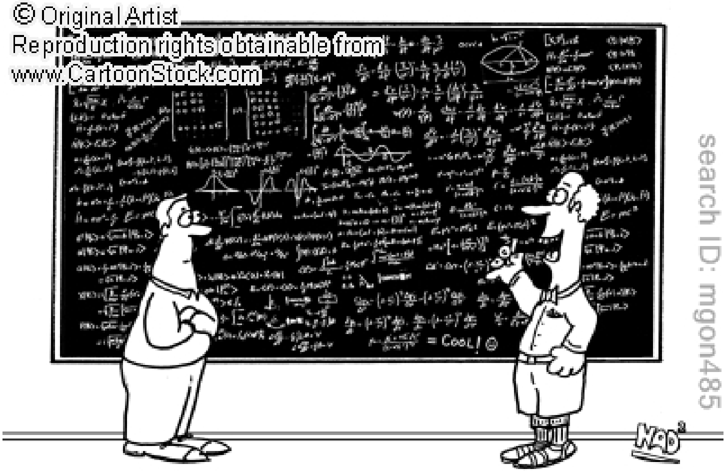
Lecture 8:

Writing: Punctuation, Mathematics, & Algorithms

Faculty of Computing & Informatics
Multimedia University
Cyberjaya, Malaysia

Can you really appreciate the information?



"Eureka! After months of research and formulating algorithms, I've done it... I've discovered the secret to 'being cool'!"

Punctuation

Stop

full-stops or periods

Don’t put a stop at the end of a heading/title

✓ 1.0 Introduction

× 1.0 Introduction.

[note: the stop at the end of **1.0 Introduction**]

Commas

Examples

Commas must appear on both sides of a parenthetical remark

- × The process may be waiting for a signal, or even if processing input, may be delayed by network interrupts.
- ✓ The process may be waiting for a signal, or, even if processing input, may be delayed by network interrupts.

Use final commas in lists

- × I like the subjects research methodology, logic and computer programming.
- ✓ I like the subjects research methodology, logic, and computer programming.

Colons

Colons connect related statements and introduce lists

- ✓ These small additional structures allow a large saving: the worst case is reduced from $O(n)$ to $O(log\ n)$.
- ✓ A colon is usually used for two things: to connect related statements and to introduce lists.
- ✓ There are two advantages of using Bayesian Networks: integrate human expert knowledge, and simple to use.

Semicolons

Semicolons divides a long sentence, or to set off part of a sentence for emphasis.

- ✓ Reading of mathematics is difficult at the best of times; it is unpleasant work if the mathematics is badly presented; and it is pointless if the mathematics does not make sense.
- ✓ in theory the algorithm would be more efficient with an array; but in practice a tree is preferable

Take a look at `http://theoatmeal.com/comics/semicolon`

Hyphenation

Hyphens used in compound adjectives, nouns, and verbs with the numbers and fractions.

Compound Adjectives

- combination of words that, together, act as an adjective. E.g., *long-term*, *low-level*, *high-speed*
- hyphen is needed only if the compound adjective appears **before** the *noun*. E.g., long-term relationship, high-level discussion.
- do not use a hyphen when the adjectives come **after** the noun. E.g., the relationship is long term, the discussion is at high level.

Hyphenation

Compound Nouns

- father-in-law
- brother-in-law

Compound Verbs

- to double-track
- to sun-dry

Tips: the best way to decide whether to use a hyphen is to look the word up in the dictionary. If it is not in the dictionary, hyphenate it, and if it is in the dictionary, spell it as the dictionary requires.

Hyphenation

Adverbs

- ✗ highly-efficient algorithm
- ✓ highly efficient algorithm

To avoid ambiguity

1. Squad helps dog bite victim. (Does the dog really need any help?)
2. Squad helps dog-bite victim. (That was nice of them.)

That vs. Which

"That" and "Which" are NOT interchangable

- × There is one method which is acceptable
- ✓ There is one method that is acceptable
- ✓ There are three options, of which only one is tractable
- × The books which have red covers are new.
- ✓ The books, which have red covers, are new.
- ✓ The books that have red covers are new.

Don't Use Exclamation Marks!

- Especially not two of them!!
- Or even more!!!!

Pluralization

× Machine learning became popular in the 1990's.

✓ Machine learning became popular in the 1990s.

Capitalization

- Either Use All Caps in Your Headings
- Or use initial caps
- Don't mix initial caps and All Caps
- ...Figure 1 shows...
- ...is discussed in Section 1.2.

Punctuation and Quotation Marks

Place a punctuation mark **inside** the quotation mark **only when it was used in the original text**. Example,

- The lecture begins with "Algorithm Design", the easiest lecture in the subject Research Methodology.
- The R&D company claims that "we have the best researchers in the world!"
- Nvidia is leading in the world of "GPU".

For more details:

`http://www.grammar-monster.com/lessons/quotation_(speech)_marks_punctuation_in_or_out.htm`

Mathematics

Theorems

- Number all theorems
- Theorems should stand alone (i.e., not be part of the surround text).Indent or otherwise mark them clearly
- Give a summary of the theorem and proof approach before launching into lemmas and detailed proof
- Omit unimportant details. Leave out arithmetic manipulations

Equations

- Center or indent equations to stand out from the text
- Avoid long sequences of mathematical formulae in the text
- Number equations only if they are needed for later reference.
- Treat displayed equations as part of the sentence in which they are embedded
- Explain your math
  –Avoid unnecessary notation and acronyms

Example of equation:

$$v = x^{z+3} \tag{1}$$

Notation

- Be consistent, standard, and simple!
  –Notation often requires several revisions before you get it right
- Explicitly introduce your notation
  –Don't just start using it
- Try to avoid recursive subscripts or combined subscripts and superscripts
- Avoid "obscure" Greek letters
- Don't reuse symbols for different meanings

Numbers

- Spell out numbers less than 10
  1. except when used mathematically
  2. except for percentages
- Make sure the semantics of percentages and units are clear and unambiguous
  1. There was a 5% increase in performance.
  2. Performance increased by 5%, from 65 to 68 correct answers.
  3. The performance accuracy increased by 5%, from 65% to 70%.

Graphs and Figures

- Figures are great, but only if they actually convey meaning
- Graphs are generally better than tables
- Use clear legends, axis labels, and line type
    1. colored lines, different types of dashes, and different tick marks on lines generally won't reproduce well in black and white
    2. Different line thicknesses are generally good, if there are only two or three types
    3. Be sure you inspect the graphs in their actual size and context

Generating Figures and Graphs

Figures

- xfig
- dia
- latex
- PowerPoint
- Word plus screenshot
- . . . many more.

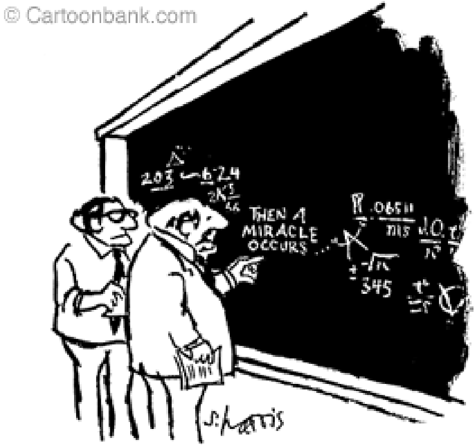
Graph

- Matlab
- Mathematica
- Gnuplot
- Excel
- . . . many more.

Algorithms

Algorithm

Neat and Precise!



"I think you should be more explicit here in step two."

Algorithms

- The value of algorithms is in the problem solved, not in the code.
- Why should the reader bother learning your Algorithm?
- What does "better" mean?
  1. Faster: Theoretically, in practice, or both.
  2. Fewer resources, such as memory, disk, code size.
  3. Less error:
     - Improves the average case.
     - Improves the worst case.
     - Improves a class of cases.
  4. Broader applicability.
  5. Solves previously insoluble problems.
  6. Enables a tradeoff between resources (e.g. time for memory)
  7. Converts static to dynamic, or vice versa.
- What is the cost?

Algorithms

Components of algorithmic paper

- Steps: Detailed steps that make up the algorithm.
- Data: Inputs, Outputs, and internal data structures.
- Scope: How is the algorithm used? Where does it apply?
- Is it globally applicable or limited to a class of problems?
- Limitations: Where does it fail?
- Correctness:
  –How do we know it works? Theoretical proof? Empirical proof?
  –Can you demonstrate correctness?
- Complexity Analysis: Time, Space, Error.
  –At least the worst case.
  –Sometimes the average (expected) case, or distribution data.

Formalism
Algorithm

The main categorizations are:

- **List code:** Numbered steps for each action, with "go to step X" statements.
  Control structure is obscure and ideas are buried.
- **Pseudocode:** Numbered lines of a block-structured language. Detailed structure is clear but statements are terse and overall idea less obvious.
- **Prosecode:** Outline form: Numbered major steps with sub-numbered component steps; combined with explanatory text.
- **Literate code:** Details are introduced gradually, intermingled with discussions of underlying ideas and even asymptotic analysis, proof of correctness, or details of key insights.

DO NOT use flowcharts or other large diagrams.

Details or NOT?

- Algorithms are about ideas. Details are distractions.
- Your Audience: CS Journal readers or conference attendees.
- You are writing for those who are considered experts. They don't need you to
  –Code a loop to add up a list of numbers,
  –Implement a binary search,
  –Write a quicksort,
  –Implement any textbook algorithm.
- Provide the detail needed to implement what you invented,

Mathematical Notation

Use mathematical notation, not programming notation.

- Use positional notation: x_i, not x[i]. use x^n, not x∧n.
- Do not use '*' or 'x' to denote multiplication; use '$\times$' or '$\cdot$'.
- Avoid specific language syntax; '==', 'a=b=c', 'a++', 'a+=c', etc. Do not assume your audience programs in C or C++. Do not use `for(i=0;i<n;i++)` or anything else that requires a knowledge of the syntax of a specific language or programming tool.
- Show *nesting* by <u>indentation</u> or by numbering style (as in an outline), don't use `BeginBlock`, `EndBlock`, $\{\ldots\}$, etc.

Mathematical Notation

Use mathematical notation, not programming notation.

- Use mathematic shortcuts: $\sum$, $\prod$, superscripting and subscripting in place of loops, function calls or braces.
- Variable names of one character leave no room for ambiguity; pq cannot be mistaken for p·q.

Environment

Data Structures, I/O types, OS, Hardware

- provide expected input and output in an algorithm.
- to explain data structure, you should use mathematical notation. E.g., `(id, event, time)`, rather than using a table or pseudocode.
- specify the OS where the experiments were conducted.
- specify hardware constraints: Memory requirements, storage types, network communication information (e.g., speed).
- Provide data types when ambiguous
- Be consistent with notation and descriptions. If “int” and “integer” mean the same thing, pick one. If they do not, define the difference.

Performance of Algorithms

How to evaluate?

- formal proof
- mathematical modelling
- simulation
- experimentation

Performance

Basic of Evaluation

- Functionality (Is your algorithm useful in more situations?)
- Speed (Theoretical or Actual? Measured how, exactly?)
- Asymptotic performance
- Typical performance
- Real data
- Synthetic data
- Compared to ...
  –Benchmark?
  –Standard Library?
  –Specific existing package or canonical algorithm?
- Be realistic. The basis of evaluation should not appear to favour your algorithm.

Performance

Big Numbers

What are you measuring? Is it what people care about?

- CPU time
- Memory requirements
- Disk requirements
- Memory traffic/throughput
- Disk traffic/throughput (fetch time, transfer rate)
- Network traffic/throughput
- Other measures (fewer collisions, fewer errors, more hits, more resilient, tougher to crack, etc)
- Be honest; don't leave out unflattering numbers. If you are twice as fast but use ten times as much memory, this is a trade-off some people might want to make.

Declaration & Acknowledgment