Defining Information Technology & Data Organization

Data Types and Computer Basics
Administrative notes

• Labs will be posted each Friday the week before you will do it in lab.
• Please note that the readings are from a mix of the textbook and modules posted on the website.
• Speaking of the textbook, about the 5th edition…
• Connect woes
• Piazza discussion board
  • Only use e-mail for personal questions – you’ll get faster answers on Piazza!
  • For either, use meaningful comments
  • The secret password is…
Learning Goals for the Course

• **understand** how data structures, interfaces, and processes on computers are designed;
• **design** your own digital artifacts using computer applications and programs, by applying your understanding of data, interfaces, and processes using other resources available to you;
• **connect** your computing knowledge with your knowledge and interest in other disciplines
Meta point: what is a learning goal?

- Meta-meta point: what is a meta point?
  - Meta: Pertaining to a level above or beyond. For example, metadata is data that describes data, metalanguage is language that describes language, etc. (wiktionary)

- A learning goal is designed to tell you what information you should learn about a topic

- At the beginning of the topic, the learning goal may not even make any sense

- By the end of a topic, it should help you know what to study
Learning goals

- Understand what an algorithm is and why they matter.
- Give an example of cases when one algorithm might perform better or worse than another algorithm.
- Recognize examples of **data types** and illustrate how properties associated with familiar data types can influence the behaviour of computer applications which act on these data types.
- Recognize examples of **data structures** and classify data structures as networked, hierarchical, and/or tabular when applicable.
- Convert between small instances of graphical and text based representations of hierarchical structure.
Hardware vs. Software

Last time we discussed what a computer is. One thing we didn’t differentiate is hardware vs. software.

**Hardware** is the machinery.

**Software** is the programs that run on top of the hardware to implement applications.
Hardware
RQ

How does a computer fit a large amount of data into its very small components?
How has this changed?

Eniac:  ~19,000 pieces of data  

My laptop:  ~4,000,000,000,000,000 pieces of data  

http://www.computerhistory.org
“..will we reach a limit to how small our computer chips can become?”
Moore’s Law

Computer speed and memory on a chip will double every 18 months to 2 years

http://en.wikipedia.org/wiki/Moore's_law
Software
An operating system is a special kind of software that allows the other software to run. Probably most of you use Windows or Mac OS. In this course (especially in lab next week), we’ll also be teaching you how to use Unix, mostly the command line – you'll need this for your project!
“Why does the process of booting up sometimes (or most of the time) fix problems on a computer?”
How does it all work?

We’ll cover how the hardware works later in the course.

For now, let’s concentrate on the software. Software works by programmers writing down algorithms in a special languages that the computer can understand.

An algorithm is a precise, systematic method for producing a specified result.
RQ

If programs are algorithms that have been specialized to a specific set of conditions and assumptions, how do programs avoid an error or mistake as a result of a situation where the assumption does not indefinitely apply? Will such a situation occur? More specifically, what is an example of an assumption in a program?
“An algorithm has to be finite. Why is that the case? I’m sure that you can create one that goes on for infinity, looping continuously.”

http://xkcd.com/1411/
So that’s an algorithm. Now what?

Well, next we need to be able to turn it into something the computer will understand. Which means we need to be able to store our data.
Data structures

A data structure describes relationships among data. There are many including:

- Network or graph data structures
- Hierarchical data structures
- Tabular data structures
Data Structures

network structures show relationships between pairs of data items
Data Structures

Hierarchical structures can be represented visually in many distinct ways.
Data Structures

in **hierarchical structures**, data is organized at branch points and/or leaves of a “tree”
Now that we’ve defined a hierarchy, let’s think about how the computer would store one
Matching Parentheses

- parentheses provide a text-based way to describe hierarchical structure
- what are pros and cons of text-based and graphical representations?

(A+(B+(C+D)))
**Step 1** (match the parentheses):

working from left to right,

- when you reach a “)”, match it with the last unmatched “(“ to its left
- use an arc to connect the pair of matched ( )’s so you remember which ones are matched

repeat until you get to the end

\[( A + ( B + ( C + D ) ) ) \]
Convert: Text → Graphical

- result at end of Step 1
Matching Parentheses

**Step 2 (build the tree):**
- put a node in the tree for each arc, and add links from each to the arcs and letters directly nested within it
Matching Parentheses

- result at end of Step 2
Matching Parentheses

( A + ( B + ( C + D ) ) )
So what does the computer do with these data structures?

- Different applications will deal with different data in different ways
- Sometimes they’ll treat the same data structure in a very different fashion…
Learning goals revisited

- Understand what an algorithm is and why they matter
- Give an example of cases when one algorithm might perform better or worse than another algorithm
- Recognize examples of **data types** and illustrate how properties associated with familiar data types can influence the behaviour of computer applications which act on these data types
- Recognize examples of **data structures** and classify data structures as networked, hierarchical, and/or tabular when applicable
- Convert between small instances of graphical and text based representations of hierarchical structure