Thinking Algorithmically

David K. Elson

CS4HS @ Columbia

July 7, 2011

Overview

- Resources for Integrating CS Lessons
  - CS Unplugged
  - TIP @ Columbia
    - Sample Lab: Programming PB&J Robots
  - NACLO
    - North American Computational Linguistics Olympiad

Computer Science, Hold The Computer

- **Best way to introduce computer science is without computers!**
  - True even for CS majors in college
    - All final exams are computer-less
  - CS is, in a sense, applied math and problem solving
  - Theory before practice
  - Avoids "let's grab the shiny thing" syndrome
  - Lower costs, less paperwork than bringing in hardware

But Gadgets Play A Role

- Familiar tech still useful to inspire students and is intrinsically interesting to them
  - "How does an iPod hold your songs?"
  - "What happens when you touch an iPhone screen?"
  - "What does Google do? Why is it free?"
  - "If you wanted to run a Facebook-like service inside the classroom with pencil and paper, how would you do it? "
    - "How would you **scale** up to 700 million users?"
  - "How are 3D graphics drawn in real time?"
  - "How does the Kinect know how you are standing?"

CS Unplugged

- http://csunplugged.org/
- Sponsored by Google, CMU, University of Canterbury
- Free learning activities that teach CS through engaging games and puzzles
  - No computers necessary
  - Relate to math and science curricula

Age Appropriateness

- CS Unplugged has exercises meant for all grade levels.
- Seemingly "younger" ones can be adapted to high school level with the addition of more complex exercises and discussions

Columbia TIP

- GK-12: NSF-funded K-12 outreach program
- TIP: Technology Integration Partnerships
- Free labs on integrating STEM topics including computer science
  - E.g., Programmable LEGO robots
- http://goo.gl/X04CC

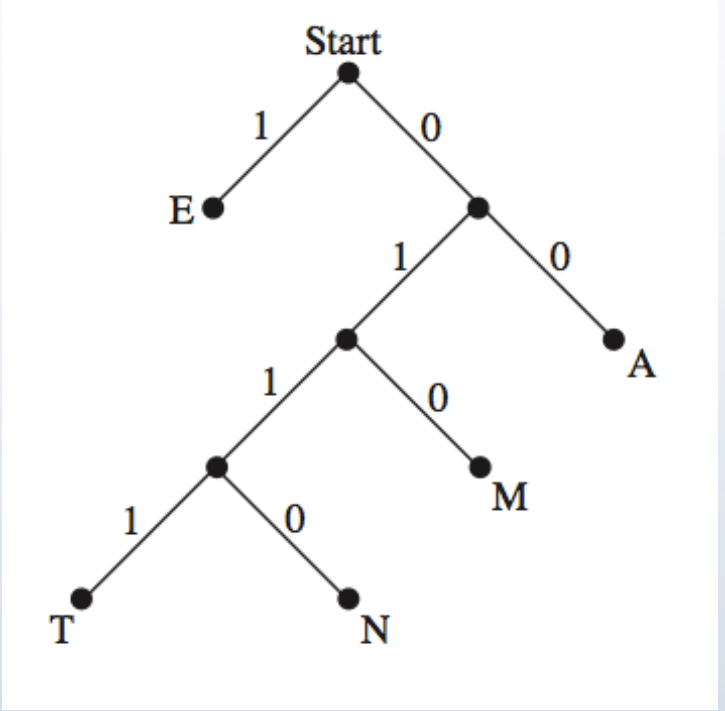
Areas Covered

- Binary Numbers
  - Understanding bases other than Base 10
  - Practicing arithmetic with binary (addition, subtraction, multiplication)
  - Converting text to binary code and back

$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	Decimal Equivalent
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Areas Covered

- Text Processing & Compression
  - Making Information Smaller: Understanding how we can **efficiently send information** by identifying patterns
    - "How can we send Netflix movies over 3G?"
  - Apply to texting, Morse code, etc.
    - Howud u bild a mor efixint languij?
  - Text prediction in user interfaces
    - "How does the Android phone know what you are trying to _____?"



Areas Covered

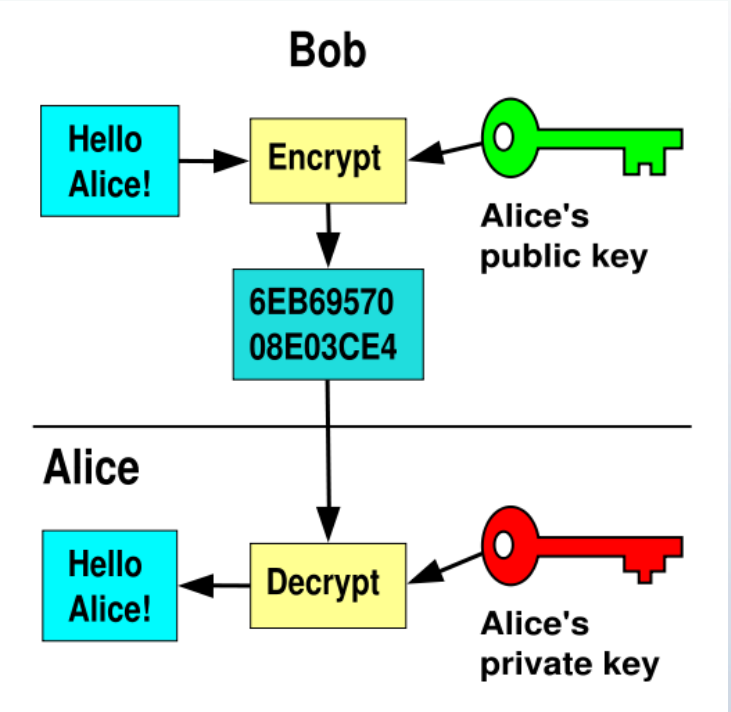
- Sorting and Searching
  - What would be the most efficient way to sort a list of random numbers?
  - Brain teasers:
    - If you have 8 identically shaped balls and a scale, and one ball is heavier, how few weightings do you need to determine the heavy ball?

2!

- Weigh {1,2,3} against {4,5,6}
- If {1,2,3} is heavier…
  - Weigh 1 against 2
  - If 1 and 2 are the same…
    - 3 is heavier
- If {1,2,3} weighs the same as {4,5,6}
  - Weigh 7 against 8
- Etc

Areas Covered

- Encryption
  - How do you keep information secret when it is transmitted?
  - Historical perspective: Wax seals, locked boxes
  - Breakable schemes:
    - ROT-13 and similar (A=N, B=O)
  - Unbreakable but inconvenient:
    - One-Time-Pad (random scramble)
  - New Hotness:
    - Public key encryption (like a mail slot)



Algorithms

- Define a set of possible **instructions**
- Define a **start state** and **end goal**
- How do you get from the start state to the end goal using the instructions?
  - Key insight: A computer will do what you tell it to do, not what you want it to do!
  - GPS directions: Small instruction set (turn left, turn right), but challenging to find the **best** route between two places
  - Various labs can have students think algorithmically

Example: PB&J Lab

- As on Columbia TIP page (http://goo.gl/X04CC)
- A set of commands for a 2-handed robot to make PB&J sandwiches with PB&J knives and unlimited bread:
  - Pick_up(OBJECT)
  - Drop()
  - Spread_Jelly(SLICE)
    - Requires JELLY_KNIFE picked up; 2 time steps
  - Spread_PB(SLICE)
    - Requires PB_KNIFE picked up; 2 time steps
  - Assemble(SLICE, SLICE)
    - Requires first SLICE picked up; drops it onto second SLICE
- How can we make a sandwich?
  - Not done until it's on the table

- Pick_up(OBJECT)
- Drop()

- Spread_Jelly(SLICE)
- Spread_PB(SLICE)
- Assemble(SLICE, SLICE)

Make 2 Sandwiches

ONE HAND	LEFT HAND	RIGHT HAND
Pick_up(PB_KINFE)		
Spread_PB(slice1)		
….		
Drop()		
Pick_up(JELLY_KNIFE)		
Spread_Jelly(slice2)		
….		
Drop()		
Pick_up(slice1)		
Assemble(slice1, slice2)		

- Pick_up(OBJECT)
- Drop()

- Spread_Jelly(SLICE)
- Spread_PB(SLICE)
- Assemble(SLICE, SLICE)

Make 2 Sandwiches

ONE HAND	LEFT HAND	RIGHT HAND
Pick_up(PB_KINFE)	Pick_up(PB_KNIFE)	Pick_up(JELLY_KN)
Spread_PB(slice1)	Spread_PB(slice1)	Spread_Jelly(slice3)
….	…	….
Drop()	Spread_PB(slice4)	Spread_Jelly(slice2)
Pick_up(JELLY_KNIFE)	…	….
Spread_Jelly(slice2)	Drop()	Drop()
….	Pick_up(slice1)	Pick_up(slice3)
Drop()	Assemble(slice1, slice2)	Assemble(slice3, slice4)
Pick_up(slice1)		
Assemble(slice1, slice2)		

- Pick_up(OBJECT)
- Drop()

- Spread_Jelly(SLICE)
- Spread_PB(SLICE)
- Assemble(SLICE, SLICE)

Make 4 Sandwiches

LEFT HAND	MIDDLE HAND	RIGHT HAND

- Pick_up(OBJECT)
- Drop()

- Spread_Jelly(SLICE)
- Spread_PB(SLICE)
- Assemble(SLICE, SLICE)

Make 4 Sandwiches

LEFT HAND	MIDDLE HAND	RIGHT HAND
	Pick_up(PB_KNIFE)	Pick_up(JELLY_KNIFE)
	Spread_PB(slice1)	Spread_Jelly(slice2)
	…	….
Pick_up(slice1)	Spread_PB(slice3)	Spread_Jelly(slice4)
Assemble(slice1, slice2)	…	….
Pick_up(slice3)	Spread_PB(slice5)	Spread_Jelly(slice6)
Assemble(slice3, slice4)	…	…
Pick_up(slice5)	Spread_PB(slice7)	Spead_Jelly(slice8)
Assemble(slice5, slice6)	…	…
Pick_up(slice7)		
Assemble(slice7, slice8)		
	FAST….	

- Pick_up(OBJECT)
- Drop()
- Spread_Jelly(SLICE)
- Spread_PB(SLICE)
- Assemble(SLICE, SLICE)

Make 4 Sandwiches

LEFT HAND	MIDDLE HAND	RIGHT HAND
Pick_up(PB_KNIFE)	Pick_up(PB_KNIFE)	Pick_up(JELLY_KNIFE)
Spread_PB(slice7)	Spread_PB(slice1)	Spread_Jelly(slice2)
…	…	….
Drop()	Spread_PB(slice3)	Spread_Jelly(slice4)
Pick_up(slice1)	…	….
Assemble(slice1, slice2)	Spread_PB(slice5)	Spread_Jelly(slice6)
Pick_up(slice3)	…	…
Assemble(slice3, slice4)	Drop()	Spead_Jelly(slice8)
Pick_up(slice5)	Pick_up(slice7)	…
Assemble(slice5, slice6)	Assemble(slice7, slice8)	drop(JELLY_KNIFE)
	FASTER!	

PB&J: Lessons Learned

- Optimization (reducing unnecessary steps)
- Parallel processing
- Memory stores (hand's capacity to hold)
- Pipelining (Factory Production)
- Performance analysis
  - Can we prove that an algorithm is the **fastest possible?**
  - Can we optimize for **space** rather than **time**?
    - Use the least amount of "table space," fewest knives, etc.
    - Many situations where space is more constrained than time

NACLO

- North American Computational Linguistics Olympiad
  - Coach your students to think algorithmically about linguistics and complete in the annual contest (6 years and counting!)
  - Winners go on to compete in the International Olympiad in Linguistics (IOL) and bring honor to their school and nation
- http://www.naclo.cs.cmu.edu/
- No prior knowledge of linguistics is necessary

NACLO 2011

- 1039 students participating
- 100 university and high school locations
- 125 students qualified for international round

- NACLO 2012: Expected registration through the NACLO web site in the fall

What Are NACLO Problems?

- Use logic to understand languages
  - Translation
  - Decryption
  - Encoding
  - Other cutting edge issues

- Get a taste of the CS area of *Natural Language Processing*
  - Machine translation, Web search, intelligent systems, etc.
  - Uses a lot of machine learning

What Are NACLO Problems?

- Use logic to understand languages
  - **Translation**: Understand foreign syntax & grammar
  - **Writing systems**: Figure out how a foreign language is written (e.g., without vowels)
  - **Calendar systems**: Deduce how a foreign culture measured time by its use of symbols and language
  - **Formal problems**: Design an algorithm to process language in some way (e.g., make a verb passive voice)
  - **Phonological problems**: Understand the connection between written and spoken forms of a language

Sample Problem: Contextual Understanding

- Suppose we are learning a new language and see words we've never seen before. Can we determine from context what **kind** of object they represent:
  - I: Individual, discrete food items
  - L: Liquids, undifferentiated masses, or masses of uncountably small things
  - C: Containers or measurements

Sample Problem: Contextual Understanding

- A hidden gem in Lower Uptown! Get the **färsel-försel** with **gorse-weebel** and you'll have a happy stomach for a week. And top it off with a **flebba** of **sweet-bolger** while you're at it!
- The food was pretty good… But I would have liked more **gorse-weebel** and fewer **göngerplose**. You really feel like the chef is skimping on the good stuff.
- I went to Wolserl last year for a holiday, and this is the real thing. If you order the **gelbelgarg**, though, make sure you also get at least one **rolse** of **sweet-bolger** – it's how the locals like it!

Sample Problem: Structured Understanding

- Determine both **vocabulary** and **grammatical structure** from the following sample sentences:

Mwamni sileng.	He drinks water.
Nutsu mwatbo mwamni sileng.	The child keeps drinking water.
Nutsu mwegau.	The child grows.
Nutsu mwatbo mwegalgal.	The child keeps crawling.
Mworob mwabma.	He runs here.
Mwerava Mabontare mwisib.	He pulls Mabontare down.
Mabontare mwisib.	Mabontare goes down.
Mweselkani tela mwesak.	He carries the axe up.
Mwelebte sileng mwabma.	He brings water.
Mabontare mworob mwesak.	Mabontare runs up.
Sileng mworob.	The water runs.

Mwamni sileng.	He drinks water.
Nutsu mwatbo mwamni sileng.	The child keeps drinking water.
Nutsu mwegau.	The child grows.
Nutsu mwatbo mwegalgal.	The child keeps crawling.
Mworob mwabma.	He runs here.
Mwerava Mabontare mwisib.	He pulls Mabontare down.
Mabontare mwisib.	Mabontare goes down.
Mweselkani tela mwesak.	He carries the axe up.
Mwelebte sileng mwabma.	He brings water.
Mabontare mworob mwesak.	Mabontare runs up.
Sileng mworob.	The water runs.

- Some new words:

sesesrakan	teacher
mwegani	eat
bwet	taro (a kind of sweet potato)
muhural	walk
butsukul	palm-tree

- Translate:
  - **"The teacher carries the water down."**
  - **"The child keeps eating."**
  - **"The palm-tree keeps growing upwards."**

mwamni	drink	*mwisib*	(go) down
mwatbo	keep (doing)	*mwesak*	(go) up
mwerava	pull	*mweselkani*	carry (of axe)
mworob	run	*mwelebte*	carry (of water)
mwegau	grow	*sileng*	water
mwegalgal	crawl	*nutsu*	child
mwabma	(go) here, approach	*tela*	axe

sesesrakan	teacher
mwegani	eat
bwet	taro (a kind of sweet potato)
muhural	walk
butsukul	palm-tree

- "The teacher carries the water down."
  - ***Sesesrakan mweselkani sileng mwisib.***
- "The child keeps eating."
  - ***Nutsu mwatbo mwegani.***
- "The palm-tree keeps growing upwards."
  - ***Butsukul mwatbo mwegau mwesak.***

NACLO Recap

- Lots of sample problems from past Olympiads and information about the annual NACLO contest:
  - http://www.naclo.cs.cmu.edu/
- NACLO committee (including a chapter here at Columbia) can help with organizing training sessions and Olympiad testing centers, recruiting students, etc.
  - Great preparation for a college and professional career in CS or linguistics

Summary

- Algorithmic Thinking: Plenty to learn about Computer Science without using computers
  - Inexpensive and fun
- CS Unplugged and Columbia TIP web sites offer links to lesson plans & resources
- NACLO offers a structured, social environment for learning skills in computer science and linguistics