

Review

Chapters 0 to 9

CS105: Great Insights in Computer Science

0: Magic in the Stone

- Computers are everywhere creating new capabilities.
- Computer science is the study of "reduction": making complex out of simple. Relating complex to complex.
- **Skill:** Reading barcodes.

1: Nuts and Bolts

- Bits: 0/1, simple but can represent anything
- Made of charges in silicon, but could be anything.
- **Skill:** Evaluate an expression using and/or/not.

2: Universal Building Blocks

- Universal logic gates can simulate and/or/not.
- Gates can be defined in terms of simpler gates (reduction).
- Logical expressions, truth tables

2: Continued

- Some simple logic gates.
- **Skill:** Recognize some simple functions of logic gates.
- Representing numbers in binary.
- **Skill:** Convert between binary and decimal (base 10).
- **Skill:** Adding and negating numbers in binary.

2: Continued

- Fundamental circuits:
 - CPU interprets machine instructions.
 - Memory holds data and programs and is addressed in binary.
 - Arithmetic circuits perform mathematical calculations.

2: Continued

- Each clock cycle corresponds to a small instruction being executed.
- Machine-language program is made from bytes.
- Everything the computer does is made from these small instructions.
- **Skill:** Converting machine-language instructions to logical formulas.

3: Programming

- *The hierarchy:* application programs → software libraries → high-level languages → machine language → logic blocks → basic logic gates → physical bits (transistors).
- Subroutines can package up repeated operations (like making a square or a song chorus).
- **Skill:** Write a song as a series of subroutines.
- Parameters let you use the same subroutine for related tasks.
- The subroutine stack keeps track of the program's place.

3: Continued

- People write programs in languages that cannot be directly run on the computer.
- Compilers translate the programs into machine language that the computer can run.
- Parsers capture the structure of the program, allow for optimizations and code generation.
- Interpreters don't translate the program, but just simulate it themselves.
- **Skill:** Interpreting expression trees.

4: How Universal Are Turing Machines?

- If an assumption leads to self contradiction, the assumption is broken.
 - Barber paradox, Godel's theorem, Kantor's diagonalization
- Assumption that we can detect whether a program loops forever leads to a program that loops forever **and** doesn't loop forever.
 - The "halting problem" is not solvable.

4: Continued

- Seemingly random numbers can be generated using complex numerical mixing-up functions.
- Random bits useful for sending secret messages.
- If event has probability p , $1/p$ tries before it happens (on average).
- **Skill:** How many attempts before success, if success probability is p ?
- **Skill:** For what values will a given loop halt?

5: Algorithms and Heuristics

- Sock matching.
- Different approaches to a problem (“algorithms”) can be faster than others!
- **Skill:** Count the number of gates in a (multilevel) logic block.

5: Continued

- **Skills:** Decision problems on lists...
 - Is x the median?
 - Sum divisible by 5?
 - Product divisible by 5?
- Fast way and slow way!

5: Continued

- Computer scientists analyze algorithms by their running time as a function of the size of the input.
- Can sing generalizations of songs with n verses.
- Number of syllables is a pain to figure out as a function of n .
- Big-O notation (asymptotic *growth rate*) simplifies the process.

5: Continued

- Major classes of song growth rates:
 - constant-size verses: linear $O(n)$
 - verses grow because each includes a number, which grows: linear-logarithmic ($O(n \lg n)$).
 - verses grow by a constant size: quadratic ($O(n^2)$).
 - verses grow by a constant size and include larger numbers: quadratic-logarithmic ($O(n^2 \lg n)$).
- **Skill:** Recognize growth rate of songs.

5: Continued

- **Skill:** Distinguish proper (growth rate) and improper ("big") uses of the word "exponential". Good = trend (growth, increase), bad = comparison of two points.
- The growth-rate classes are also very useful for analyzing algorithms like sock sorters.
- Some problems seem to only grow exponentially more difficult with size: NP-complete problems.

5: Continued

- Google builds a map of the web by visiting web pages it knows about, then looking on those pages for the addresses of other pages.
- This operation is called "graph search". Used to solve mazes, also.
- Graph defined by nodes, links. Other terms: source, sink, path, cycle, connected components, tour, directed, undirected, tree. Node x is "reachable" from y if there's a path (series of links) from y to x .

5: Continued

- Sorting speeds up problems like the search for information in a list of n .
- Selection Sort: Repeatedly find, remove the smallest. $O(n^2)$.
- Binary search: Ask a question that splits the remaining set of options in half. $O(\lg n)$.
- Quicksort: Split into big/small elements relative to pivot. $O(n \lg n)$.
- Heuristics: Often finds a near best answer (hill climbing).

5: Continued

- **Skill:** Recognize the time needed to solve problems on sorted and unsorted lists.

Example task	Unsorted	Sorted
Find a target element	$O(n)$	$O(\log n)$
Find minimum	$O(n)$	$O(1)$
Find median	$O(n^2)$	$O(1)$
Find mode	$O(n^2)$	$O(n)$
Find mean	$O(n)$	$O(n)$
Element divisible by 5?	$O(n)$	$O(n)$
Element bigger than 5?	$O(n)$	$O(\log n)$
List longer than 5?	$O(1)$	$O(1)$

6: Memory: Information and Secret Codes

- The number of bits that it takes to represent a message varies with the *encoding*.
- Finding a shorter encoding is “compression”.
- Sometimes video compression is “lossy”.
- Huffman encoding uses few bits for common characters.
- **Skill:** Given a string, build a Huffman code for it, encode and decode strings.

7: Speed: Parallel Computers

- Moore's Law: Roughly, computers double in speed every year and a half.
- Some problems can be sped up by letting more than one computer work on them at a time. Some can't! Some others can, but only if you're clever.
- Google uses "map and reduce" to carry out huge calculations quickly.
- Demo: Apply operation to all pixels at once.

8: Computers That Learn And Adapt

- Classifiers map feature vectors to yes/no. Killers? Speech recognition?
- Classifiers can be created automatically (learned) by analyzing a training set of examples.
- Decision trees can be constructed from examples and applied to new instances.

8. Continued

- “forever” or “while True” loops useful when constructing programs that must continue to do the right thing.
- Reinforcement learning defines task by specifying the reward function / goal and letting the program change its behavior to make things work.
- Robots are just computers.

9: Beyond Engineering

- Classical engineering methods can fail badly.
- Genetic algorithm is a heuristic that uses a population to find good solutions.
- Uses populations of individuals (often sequences of bits!) that mate if their fitness is sufficiently high to produce new generations of improved individuals.
- Nice summary of earlier concepts!

With sincere apologies to Simon and Garfunkel...



Charles Babbage's machines