Algorithms

Definition

- An algorithm is a method for solving a problem
- An algorithm includes a complete description of:
  - the set of inputs, or starting conditions
  - a full specification of the problem to be solved
  - the set of outputs
  - descriptions of valid solutions to the problem
  - a sequence of operations that will eventually produce the output
  - steps must be simple and precise

Attributes of Algorithms

- Even though the study of algorithms is of central importance to computer science, it is hard to specify exactly what an algorithm is
- The steps in an algorithm must be:
  - precise: they must be written in terms understandable by almost anyone
    - but what does "precise" mean? how precise does a step have to be?
  - effective: a step must help the algorithm progress to the final goal
    - but how effective? is there a formal definition of "effective"?
  - practical: a sequence of precise and effective steps may not be useful in practice
    - example (from Knuth): a hypothetical algorithm for playing chess
      - very easy to define a method that considers all possible moves
      - one estimate of the number of possible outcomes of a game with 40 moves: $10^{40}$
      - a "teraflops" computer ($10^{12}$ operations per second) would take $10^{23}$ years
    - age of the universe: $10^{13}$ years

Books on Algorithms

- CIS 315 (required course for CIS majors) uses:
  - Cormen, Leiserson, Rivest
    - Introduction to Algorithms
    - McGraw Hill 1993
      - 1028 pages -- 37 chapters, each on a different type of algorithm
  - A book I used as an undergraduate:
    - Knuth
      - Fundamental Algorithms
      - Addison-Wesley 1968
        - vol. 1 of 3 (was intended as the first of 7)
        - 634 pages in vol 1, alone; vols 2 (numeric) and 3 (sorting and searching) are longer
  - It's a very big and important topic...
Sport / Not a Sport

- From the old *Sports Illustrated* column “Sport or Not a Sport?”
  - **Hot dog eating contests**: “Who can’t eat? Everybody can eat.” — Los Angeles Clippers guard Quentin Richardson (meaning “sport” involves skill).
  - **Bullfighting**: “As long as there’s a chance you could die, it’s a sport.” — Oakland A’s pitcher Steve Sparks (“sport” involves risk).
  - **Ballooning**: “If it’s one guy, not a sport. If it’s a bunch of guys racing, a sport. If they’re racing in a thunderstorm, then it’s a sport televised on Fox.” — U.S. soccer player Clint Mathis (“sport” requires teamwork).
  - **Bass fishing**: “They don’t have jerseys, man. To play a sport, you have to wear a jersey.” — former Denver Broncos running back Terrell Davis.
  - **What’s jai alai?** — Miami Heat forward Chris Gatling

[source: Patrick Hruby, Washington Post]

To illustrate what an algorithm is we’ll play “algorithm / not an algorithm”

Algorithm / Not an Algorithm

Recipe for Enchiladas

- From Chapter 1 of NTO
  - Is a recipe an algorithm?
    - inputs (ingredients)
    - outputs (dinner)
    - steps
  - But are the steps
    - precise?
    - effective?
  - Also: a recipe tells you how to make one dish, but isn’t a specification of how to cook similar dishes...

Driving to Portland

- Problem specification: I’m at work and I need to get to the Portland airport
  - input: my location at UO
  - output: location of PDX
  - steps: milestones on the way

These steps are more precise

Note: an algorithm generated the instructions, but is the set of instructions itself an algorithm?

Sorting a Poker Hand

- Problem: given a set of five cards, arrange them in order from largest to smallest
  - input: a set of 5 cards
  - output: ordered set of cards
  - method:
    - find the largest card, move it to position 1
    - find the next largest card, move it to position 2
    - etc.

If we can be more precise about “find” and “move” then this is a good example of an algorithm

- in the CS literature, it’s known as “bubble sort”
- it’s also a general method that applies to other sets of cards
  - bridge and other games might sort by suit first, then by rank
Algorithm / Not an Algorithm

Greatest Common Divisor
- Problem: find the greatest common divisor of two numbers
  - input: integers $n$ and $m$ (where $m > n$)
  - output: the largest number $x$ such that $n / x$ and $m / x$ are both integers
- Method:
  1. let $r$ be the remainder of $m / n$
  2. if $r = 0$, $n$ is the GCD*
  3. go back to step 1, replacing $m$ with $n$ and $n$ with $r$
- Example: Find the GCD of 26 and 39
  - step | $m$ | $n$ | $r$
  - 1   | 39  | 26 | 13
  - 2   | 26  | 13 | 0
- This is Euclid’s Algorithm
  - may be the first algorithm (300 BCE)

* a number is its own GCD

Integer Factorization
- Problem: find all the prime factors of an integer (e.g. $52 = 2 \times 2 \times 13$)
  - input: an integer $n$
  - output: a list of integers
  - method: check to see if $n$ is divisible by all primes between 2 and $\sqrt{n}$
- Example: Find the GCD of 26 and 39
  - This is clearly an algorithm
    - well-defined input and output
    - each step is precise, effective, practical
    - will work for any choice of $n$

Elements of an Algorithm
- Review: An algorithm is a description of a procedure to solve a problem
  - algorithms have well-defined inputs (starting conditions) and outputs (goals)
  - steps of the procedure must be
    - precise (simple and unambiguous)
    - effective (make progress toward the solution)
    - practical (can be completed)
  - the process must eventually terminate, otherwise the outputs are not defined
- An algorithm is more general than a set of instructions for one problem
  - e.g. find GCD of any pair of integers, sort any set of cards, ...
- Why some of the earlier examples might not be considered algorithms:
  - vague or undefined steps
  - were not methods for solving general problems, but instructions for carrying out a single task

Integer Factorization is a Hard Problem
- Want to earn $200K? Find the prime factors of a 617-digit number
  - http://www.rsasecurity.com/rsalabs
  - The RSA scheme for encrypting messages relies on the fact that finding the prime factors of numbers this large is a very hard problem
    - one of their “smaller” challenges was solved by a network of 80 computers each working 24/7 for 18 months
    - used a much more sophisticated algorithm than the “brute force” method on the previous slide
    - note: The R in RSA = Rivest, one of the authors of the CIS 315 text...
- Methods for analyzing algorithms will be covered in future lectures
  - devise formulas for estimating the number of steps given the size of the input
  - Encryption is the subject of Chapter 37 in NTO
History

The modern word “algorithm” comes from the name of the Persian scholar Muḥammad ibn Mūsā al-Ḵwārizmī (ca. 780 -- ca. 850)

- when his work was published in Latin his name was spelled Algoritmi
- he was the author of several influential works on mathematics and natural science
- his book on the systematic solution of linear equations contained several algorithms
- the title of this book is also the source of our word “algebra”

al-Ḵwārizmī also introduced western philosophers to arabic numerals and the positional number system

People who used this new system of symbol manipulation were known as algoritists

The abacists maintained the old methods

Pseudo-code

The algorithms we’ve seen so far have all been expressed in English

A more common technique is to write “pseudo-code”

- “code” is another term for a program written in a programming language
- “pseudo-code” is a mix of formal notation and English

Specifications in pseudo-code are more precise and (when you get used to it) easier to understand

Another benefit of pseudo-code: easy to implement in many different programming languages

- an algorithm written in pseudo-code is relatively easy to code in Java, Pascal, Perl, Ruby, or many other languages
- an algorithm written in one programming language is much harder to translate to another programming language
- lots of syntax details obscure the underlying meaning of the program

Example: Euclid’s Algorithm

Here is one way to write Euclid’s GCD algorithm in pseudo-code:

```
repeat:
    r ← m / n
    m ← n
    n ← r
until r = 0
print r
```

This form corresponds to the method of building a table row by row

In this description it is clear each step is either

- a very simple mathematical operation, or
- an assignment that moves data around
Euclid’s Algorithm (cont’d)

- Here are two other ways of writing the same algorithm:

```
r ← m / n
while r ≠ 0:
    m ← n
    n ← r
    r ← m / n
print r
```

- These are all examples of loops
  - aka iteration
- Note that indentation is meaningful
- all of these versions specify three steps to be repeated zero or more times

Example: Insertion Sort

- Here is a different way to sort cards, known as “insertion sort”:
  1. start with the card one space over from the left edge (the queen in this example)
  2. use your right hand to pick up the current card
  3. keep looking to the left until you find the first card lower than the one in your right hand, or the end of the poker hand, whichever comes first
  4. insert the card in your right hand back into the poker hand at this location
  5. move one place to the right of the original spot and go back to step 2

- This description of a sorting process is more explicit about the “find” and “move” steps of the previous algorithm
- But it’s pretty verbose, and is still not very precise in places

Example: Insertion Sort

- Here is a pseudo-code version of insertion sort:

```
input: A0, A1, ... An-1
for j = 1 to n-1
    key ← Aj
    i ← j-1
    while i ≥ 0 and Ai > key:
        Ai+1 ← Ai
        i ← i-1
    Ai+1 ← key
```

- This specification is much more concise
- In the next set of slides we’ll see how to represent lists in Ruby and how to write loops like these
- Note this algorithm has two loops, one inside the other
  - these are called nested loops

Insertion Sort (cont’d)

- The following pictures show how insertion sort works
  - The state of a list after the first 3 positions have been sorted:

![Insertion Sort State](Image)

- The first step in the inner loop sets key to the item to be sorted next, and sets 1 to the position to the left of the current position:
Insertion Sort (cont’d)

- The inner loop moves items to the right to make a place for the current item:

  ```
  key ← A_{i}
  i ← j - 1
  while i ≥ 0 and A_{i} > key:
      A_{i+1} ← A_{i}
      i ← i - 1
  A_{i+1} ← key
  ```

Algorithms and Computer Science

- The study of algorithms is a major part of computer science
- Research in computer science involves:
  - developing new and better algorithms
  - exploring relationships between classes of algorithms
  - designing languages that make it easier to express algorithms
- There are several very important problems for which there are no efficient algorithms:
  - factoring integers is very hard
  - it is likely that no efficient algorithm will ever be developed (based on what mathematicians know about the properties of prime numbers)

Artificial Intelligence

- Another branch of computer science research -- **artificial intelligence** -- involves the development of algorithms to simulate human problem solving:
  - natural language: understanding speech, translating text, and other applications
  - described in chapter 41 of NTO
  - managing complex processes
    - example: how to best schedule operations on a factory floor
    - companies lose money when machines are idle waiting for material from other machines
  - making complex decisions
    - does a patient with symptoms X have disease Y?
    - does this image show a cancerous cell or a normal cell?
  - learning: can a computer be trained to carry out repetitive tasks (e.g. recognize and throw away spam e-mail)?
Turing Test

- In one of his last papers, in 1954, Alan Turing examined the question of what it means for a machine to be “intelligent”.
- His criterion is known as the Turing Test: a machine would be considered intelligent if it was indistinguishable from a human.
  - put a computer in one room and a human in another
  - a tester is allowed to pose questions
  - computer and human both respond by typing answers on a terminal
  - if the tester cannot distinguish the computer’s responses from the human’s one would conclude the machine is “intelligent”
- Turing wrote this paper 50 years before the invention of the chat room!
- Turing test today: are there AI programs in your chats?

Review

- This set of slides introduced the concept of an algorithm.
  - An algorithm is a method for solving a problem.
- A type of notation known as “pseudo-code” describes steps in an algorithm.
- Skills:
  - you should be able to understand how a simple algorithm works
  - given an algorithm like GCD and a set of input values you should be able to say what the outputs will be
  - you should (eventually) be able to translate a simple algorithm into Ruby and test your program to see if it is correct.