Algorithms

Definition

- More than anything else computer science is the study of algorithms
- 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

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...

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 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 winning a chess tournament: "for each game, consider all possible moves, choose the best one"
    - very easy to define a method that considers all possible moves
    - one estimate of the number of possible boards in a game with 40 moves: $10^{43}$
      - a "teraflops" computer ($10^{12}$ operations per second) would take $10^{23}$ years
    - age of the universe: $10^{13}$ years
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 (i.e. “sport” involves risk).
  - **Balooning**: “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 Sports Network.” — U.S. soccer player Clint Mathis (i.e. “sport” requires teamwork).
  - **Bass fishing**: “They don’t have jerseys. To play a sport, you have to wear a jersey.” — former Denver Broncos running back Terrell Davis.

To try to understand what an algorithm is we’ll play a round of “Algorithm or Not an Algorithm”

Algorithm / Not an Algorithm

**Recipe for Enchiladas**

- Is a recipe an algorithm?
  - inputs (ingredients)
  - outputs (dinner)
  - steps
- Are the steps
  - precise?
  - effective?

Algorithm or not an algorithm?

**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

Algorithm or not an algorithm?

**Sorting a Hand in a Card Game**

- Problem: given a set of cards, arrange them in order from largest to smallest
  - input: a set of cards
  - output: ordered set of cards
  - method:
    - compare leftmost two cards, put smallest on left
    - move the third card to its correct spot before/between/after the left two
    - move the fourth card to its correct spot before/in the middle of/after first three
    - continue until entire hand is sorted
- If we can be precise about “compare”, “move” and “correct spot” then this is a good example of an algorithm
- in the CS literature, it’s known as “insertion sort” and it’s a general method that applies to other sets of cards or other types of data

Algorithm or not an algorithm?
Algorithm / Not an Algorithm

The Sieve of Eratosthenes

- Problem: make a list of prime numbers less than some number \( n \)
  - input: upper limit \( n \)
  - output: a list of numbers \( p_0, p_1, \ldots \) such that each \( p_i \) is prime, and for any number \( 2 \leq m \leq n \) if \( m \) is prime it is in the list

- Method:
  - make a list \( a \) containing all integers between 2 and \( n \)
  - make an empty list \( p \) (to hold the output)
  - repeat until \( a \) is empty:
    - set \( m \) to the first element from \( a \); remove this number from \( a \) and append it to \( p \)
    - remove every multiple of \( m \) from \( a \)
  - (examples on the next slide)

Algorithm / Not an Algorithm

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 \( p \) such that \( p_0 \times p_1 \ldots = n \)
  - method: check to see if \( n \) is divisible by all primes between 2 and \( \sqrt{n} \)

- This is clearly meets the requirements of an algorithm
  - well-defined input and output
  - each step is precise, effective, practical
  - will work for any choice of \( n \)

Algorithm / Not an Algorithm

The Sieve of Eratosthenes (cont’d)

- See also: “sieve of eratosthenes” at Wikipedia (has an animation)

Algorithm / Not an Algorithm

Integer Factorization is a Hard Problem

- RSA Labs (a company that sells encryption software) ran a contest
  - offered cash prizes for the first person or team able to factor large numbers
  - e.g. \$200K to find the prime factors of a 616-digit (decimal) integer

- A group using a much more sophisticated algorithm than the “brute force” method on the previous slide factored a 640-digit number
  - required a network of 80 computers working 24/7 for 5 months (“30 CPU-years”)
  - for more information look for “cryptographic challenges” at http://rsa.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
  - note: The R in RSA = Rivest, one of the authors of the CIS 315 text...
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 an algorithm 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 a method for solving a class of problems
  - e.g. find primes less than any number \( n \), sort any set of items (cards, strings, ...)
- 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

History

- The earliest known algorithms were defined by Greek mathematicians
  - e.g. Euclid’s method for the greatest common divisor of two integers, ca. 300 BC
- 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”

History (cont’d)

- 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 algorists
- The abacists maintained the old methods
  - woodcut illustration from a 1504 encyclopedia
  - see the CIS 315 web site

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 than plain English
- Another benefit of pseudo-code: easy to implement in many different programming languages
  - an algorithm written in pseudo-code is relatively easy to re-code in Java, Pascal, Perl, Ruby, or many other languages
Example: Sieve of Eratosthenes

- Here is what the sieve algorithm might look like in pseudo-code:

  ```
  a ← [2..n]
  p ← []
  while a ≠ []
    m ← a₀
    remove a₀ from a
    append m to p
    for i = 0 to |a|-1
      delete aᵢ if aᵢ mod m = 0
  ```

- Some of these operations (e.g. assignment) are part of (almost) every programming language
- Others (e.g. remove, append) may be part of a language (e.g. a method of the Array class), or if not, are simple enough to write in a few steps

Example: Insertion Sort

Here is a more precise verbal statement of the insertion sort algorithm:

1. the initial key is one space over from the left edge (the Q in this example)
2. use your right hand to pick up the key
3. scan left until you find the first item lower than the one in your right hand, or the front of the list, whichever comes first
4. insert the key back into the list at this location
5. set the new key to the item to the right of the location of the previous key and go back to step 2

- This description of a sorting process is more explicit about the “find” and “move” steps of the original description
- 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:

```pseudocode
input: a list a₀, a₁, ... an-1
for j = 1 to n-1
  key ← a_j
  i ← j-1
  while i ≥ 0 and aᵢ > key:
    aᵢ+1 ← aᵢ
    i ← i-1
  aᵢ+1 ← key
```
Insertion Sort (cont’d)

- The following pictures show an example of how insertion sort works (using a list of numbers instead of cards)
- Suppose the first 3 positions (0 through 2) have been sorted:

![Diagram showing sorted positions]

- The first step in the inner loop sets key to the item to be sorted next, and sets i to the position to the left of the key’s position:

![Diagram showing key assignment]

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

![Diagram showing item movement]

```
for j = 1 to n-1
    key ← a_j
    i ← j-1
    while i ≥ 0 and a_i > key:
        a_i+1 ← a_i
        i ← i-1
    a_i+1 ← key
```

Insertion Sort (cont’d)

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)

Algorithms and Computer Science (cont’d)

- A conjecture in one branch of theoretical computer science is that for a certain class of problems no efficient algorithm exists
  - these problems are intractable
  - an important group of intractable problems are the “NP-complete” problems
- Another very interesting question: what are the limits to what can be solved via algorithms?
  - some problems are known to be unsolvable
  - one can prove mathematically that it is impossible to design an algorithm to solve such problems
- Both of these topics are covered in Ch 12 of the textbook
  - we’ll talk about them later in the term...
Artificial Intelligence

- Another branch of computer science known as **artificial intelligence** involves the development of algorithms to simulate human problem solving
  - natural language: understanding speech, translating text, and other applications
    - natural language = English, Spanish, Chinese, ...
    - computer language = Java, Ruby, ...
  - 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)?

Historical Note: 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
  - computer and human both respond to questions by typing answers on a terminal
  - if a 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: 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 the Sieve of Eratosthenes or insertion sort and a set of input values you should be able to say what the outputs will be

- You will not be required to translate pseudo-code into Ruby
  - you might be asked to modify a Ruby program