CIS 170
Spring 2006

Road Map

Theme for the rest of the term: algorithms for strings

Exact Match

- The first topic will be algorithms for finding substrings
  - example: the index method in Ruby's string class
    ```ruby
    >> name.index(" ")
    => 4
    ```

- Simple algorithms compare every character
- More advanced algorithms skip over large parts of the text
  - NTO 61

Indexed Search

- Searches through very large text use an index similar to an index in a book
  - example: suppose you want to learn about the reverse method of the Array class
    (a) start scanning the book from the beginning, looking for "reverse"
    (b) go to the back of the book, look for "reverse" in the index, go to the page numbers stored in the index
  
- A system with a huge index: Google

How to make an index: NTO 43

A system that uses indices: relational database (NTO 63)

Pattern Search

- Sometimes we want to find a string defined by a pattern
  - example: search for any method of the Array class
  
  - pattern is the word “Array” followed by a period followed by a word
  
  - in Ruby and other programming languages: use a regular expression, e.g.
    ```ruby
    text =~ /Array\..\w+/
    ```

- Regular expressions are also used in protein databases
  - e.g. proteins involved in transcription have a short pattern known as a “zinc finger”
  
  To find such a gene you might do a regular expression search in a protein database
Inexact Match

- Another type of search in bioinformatics involves **inexact match**
- The idea is to look through a large text to find substrings that are similar to an input string
  - example: the first 20 characters in the human hemoglobin protein:
    - MVHLTPEEKSAVTALWGKVN
  - a search at NCBI returned the following (differences highlighted in red)
    - chimp: MVHLTPEEKTAVALWGVKN
    - cow: MVHTAEKAAITGVLWGVKN
    - dog: MVHTAEKESLVGLWGVKN
- This type of search is made more difficult by the fact that matches can be shorter or longer than the input string

Search Without a Pattern

- It is even possible to do a search when you don’t know what you are searching for
- A system can “learn” what characterizes items of interest
- Examples:
  - spam filters for e-mail
  - fraudulent use of credit cards
  - potentially meaningful regions of DNA
- Only 2% of the human genome consists of genes
  - small strings somewhere in front of genes are known as “control regions”
  - is there a pattern or some other way of finding these regions?
- General term for this area: **data mining**

Genetic Algorithms

- One way to write a program that learns is to use evolutionary biology as an inspiration
- A **genetic algorithm** is one that tries lots of different tentative partial solutions
  - Promising solutions are “bred” with others
  - Add “mutations” to introduce variability in the “population” of partial solutions
  - Eventually a complete solution emerges
- NTO: 16

The Plan

- For the rest of the term we will explore these ideas related to searching for information in large pieces of text
  - plain text (e.g. books)
  - web pages
  - DNA strings
- Lab projects will use software that manages large databases
  - MySQL relational database
  - BLAST tool for searching biological sequence databases
- Ruby projects will explore ideas behind many of the implementations
- As time permits we will also look at other topics related to strings, e.g. cryptography or text compression