Lecture 1: Introduction

Phylogeny

Language Phylogeny

Tree of Life

Menu
- Motivating Problem
- Course Structure, Expectations, Goals
- Analyzing Algorithms

From http://www.shef.ac.uk/language/quantling/

From http://tolweb.org/
Finding a Phylogeny
- Speculate on history based on current evidence
  - Not guaranteed to be correct
- Find the “most likely” history
  - Parsimony: find the evolutionary tree that explains the observations with the fewest possible changes

Measuring Changes
- Natural Languages
  - Grammatical Rules
  - Lexicon
  - Hard to quantify how similar two languages are
- Species
  - Genomes (only recently)
  - Easy to quantify: genome differences are measurable

How Species Evolve
- Point Mutations (Substitution): one base is replaced with another
  - UV Ray
  - ...CAT...
  - ...CTT...
  - With only point mutations, easy to tell how close two genomes are, just count the different bases

How Species Evolve 2
- Insertions: one or more bases are inserted
  - ...GCATG...
  - ...GACATG...
- Deletions: one or more bases are removed
  - ...GATCATG...
  - ...CATG...
  - Caused by copying errors (enzymes slipping, etc.)

Measuring Genome Similarity
- Insertions and Deletions this hard
  - ACATCATCATCAT
  - CATCATCATCAT
  - are more “similar” than
  - ACATCATCAT
  - TCGTTGCCGAAAA

Sequence Alignment
- Align sequences by inserting gaps:
  - ACATCATCATCAT
  - CATCATCATCAT
  - Find best alignment inserting gaps given:
    - value of matching bases (point mutations) = c
    - cost of a gap (insertion/deletion) = g
  - We use c = 10, g = 2: goodness = 12 * c – g = 118
Brute Force Alignment

To find the best alignment of sequences $U$ and $V$ with correct value $c$ and gap penalty $g$:

- if $U$ or $V$ is empty
  - $U$, $V$ is the best alignment
- otherwise,
  - [ next slide ]

Brute Force Alignment: Otherwise...

Try three possibilities:

1. First elements of $U$ and $V$ are aligned:
   [ best alignments of $U[1:]$ and $V[1:]$ + $c$ if $U[0] == V[0]$ ]

2. First element of $U$ is aligned with a gap in $V$
   [ score of best alignments of $U[1:]$ and $V + g$ ]

3. First element of $V$ is aligned with a gap in $U$
   [ score of best alignments of $U$ and $V[1:] + g$ ]

Pick the choice with the highest score

Course Structure, Expectations, Goals

Meetings

- Lectures: 2 per week
  - Will include material not in the book
  - Most lectures will use slides and notes
- Section meetings: 1 per week
  - You must sign up for one of the sections
  - Classroom work, group exercises, review, quizzes, ...
- Staffed time in Small Hall
  - Take advantage of help from the ACs and your classmates

Problem Sets

- 8 total, 1-2½ weeks each
- Work on them when and where you want (but take advantage of staffed lab time in Small Hall)
- Usually will work with partners
- Mix of programming and analysis
- Main way most will learn
- Turn in on paper at beginning of class (first is due Wednesday)
My Teaching Philosophy:
Drinking from a Firehose

Don’t be overwhelmed!
You will do fine.
It may hurt a little bit, and a lot of water will
go by you, but you won’t go away thirsty!

Expectations:
Programming Background
• You understand basic programming:
  – Can write a program longer than a screenful
  – Can understand multi-file programs
  – Familiar with common control structures, procedures, recursive definitions
• You don’t freak out when you are expected to learn a new language on your own

Expectations:
Math and Logic Background
• You remember some things from CS202 (or will learn/re-learn them when you need them):
  – Arithmetic, logarithms, sets, graphs
  – Symbolic logic, implication
  – Proof techniques (induction, contradiction)
• The textbook is quite mathematical – you may need to read things more than once

Course Goals

Course Goal 1
Learn to write delightful programs.
correct, readable, elegant, economical, efficient, scalable, maintainable, secure, dependable

Course Goal 2
Be able to predict how decisions about data representation will impact properties of an implementation.
running time, memory use, ease of implementation, scalability, ...
Course Goal 3
Understand how a program executes at levels of abstraction ranging from a high-level programming language to machine memory.

We will talk about what this means in Monday’s class.

Algorithm Properties
- Implementable – can be readily expressed as a program
- Termination – always finishes
- Correctness – always gives the correct answer
- Efficient – uses resources wisely

Note: Chapter 2 of text has a similar list but separates “Implementable” into Effectiveness and Program Complexity

Is it Implementable?
def bestAlignment (U, V, c, g):
    if len(U) == 0 or len(V) == 0: return U, V
    else:
        (U0, V0) = bestAlignment (U[1:], V[1:], c, g)
        scoreNoGap = goodnessScore (U0, V0, c, g)
        if U[0] == V[0]: scoreNoGap += c
        # try inserting a gap in U (no match for V[0])
        (U1, V1) = bestAlignment (U, V[1:], c, g)
        scoreGapU = goodnessScore (U1, V1, c, g) - g
        # try inserting a gap in V (no match for U[0])
        (U2, V2) = bestAlignment (U[1:], V, c, g)
        scoreGapV = goodnessScore (U2, V2, c, g) - g
        if scoreNoGap >= scoreGapU and scoreNoGap >= scoreGapV:
            return U[0] + U0, V[0] + V0
        elif scoreGapU >= scoreGapV:
            return GAP + U1, V[0] + V1
        else:
            return U[0] + U2, GAP + V2

Is this a “good” solution?
if U or V is empty
    U, V is the best alignment otherwise,
    Try three possibilities:
    1. First elements of U and V are aligned:
       score of best alignments of U[1:] and V[1:] + c if U[0] == V[0]
    2. First element of U is aligned with a gap in V
       score of best alignments of U[1:] and V + g
    3. First element of V is aligned with a gap in U
       score of best alignments of U and V[1:] + g
    Pick the choice with the highest score

Algorithm Properties
✓ Implementable – can be readily expressed as a program
✓ Termination – always finishes
✓ Correctness – always gives the correct answer
✓ Efficient – uses resources wisely
Termination?

if $U$ or $V$ is empty
$U$, $V$ is the best alignment otherwise,
Try three possibilities:
1. First elements of $U$ and $V$ are aligned:
   score of best alignments of $U[1:]$ and $V[1:]$ + $c$
   if $U[0] == V[0]$
2. First element of $U$ is aligned with a gap in $V$
   score of best alignments of $U[1:]$ and $V[1:]$ + $g$
3. First element of $V$ is aligned with a gap in $U$
   score of best alignments of $U$ and $V[1:]$ + $g$
Pick the choice with the highest score

if $U$ or $V$ is empty
$U$, $V$ is the best alignment otherwise,
Try three possibilities:
1. First elements of $U$ and $V$ are aligned:
   score of best alignments of $U[1:]$ and $V[1:]$ + $c$
   if $U[0] == V[0]$
2. First element of $U$ is aligned with a gap in $V$
   score of best alignments of $U[1:]$ and $V[1:]$ + $g$
3. First element of $V$ is aligned with a gap in $U$
   score of best alignments of $U$ and $V[1:]$ + $g$
Pick the choice with the highest score

Every attempt, at least one element is removed (and none added). Initial length is finite, so must terminate.

Algorithm Properties

✓ Implementable – can be readily expressed as a program
✓ Termination – always finishes
• Correctness – always gives the correct answer
  – Very informally: it tries all possibilities and picks the best one
• Efficient – uses resources wisely

Efficiency?

• What resources do we care about?
  – Programmer Time
  – Running Time
  – Space Use

Measuring Resource Use

• Space
  – Fundamental unit: bit
• Running Time
  – No fundamental unit
    • Number of steps?
    • How much can you do in one step?
    • How long does a step take?
• How does it scale with the size of the input

Answering for this algorithm is a PS1 question

Charge

• Registration Survey
  – Linked from course web site
  – Submit by Friday 5pm
• Text: Read chapters 1-3
• PS1: Out now, due in 1 week
  – Start now – the section time is not for doing PSs
• Monday: Levels of Abstraction, Order Notation