CS 420-02: Undergraduate Simulation, Modeling and Analysis

Rahul Simha

Department of Computer Science
College of William & Mary
Williamsburg, VA

Chapter 1

Introduction

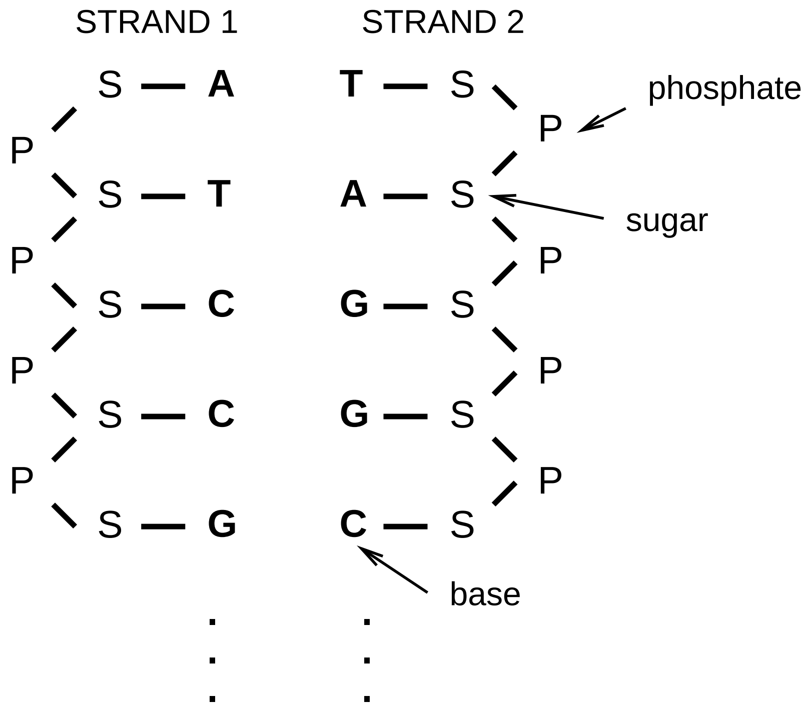
CS 420-02: Undergraduate Simulation, Modeling and Analysis

1.1 Introduction

- Introduction in three parts:
  1. *A problem in computational biology*:
     - Background: DNA and sequencing.
     - The alignment problem: formulation.
     - The alignment problem: solution using graph theory.
  2. *Cellular automata*:
     - Von Neumann's question.
     - Cellular automata: the Game of Life.
  3. *Genetic algorithms*:
     - Evolution as a metaphor for algorithm design.
     - The basic genetic algorithm.
     - Example: function optimization.
- Overview of course.

1.2 Background: Molecular Biology

- Two key chemicals in living things:
  1. *Proteins*:
     - Proteins occur in many forms/functions:
       * Enzymes (catalysts).
       * Structural proteins (cell walls, skin, hair).
       * Transporters of substances (e.g., hemoglobin, carrier of oxygen).
       * Transporters of information (receptors, hormones).
     - Fact: human body has about 100,000 distinct proteins.
     - A protein is a chain of *amino acids*.
     - There are 20 different amino acids.
     - $3 \leq$ #amino acids $\leq$ thousands.
       $\Rightarrow$ amino acids can repeat along the chain.
     - If we use the symbols $\Omega = \{A_1, A_2, \ldots, A_{20}\}$ for the 20 amino acids, a protein is a "word" over the alphabet Ω.
  2. *DNA*:
     - DNA is a molecular structure that resides inside a cell's nucleus.
     - DNA is the "information carrier" of a cell.
     - DNA is a (really long) "word" over a 4-letter alphabet: $\{A, T, C, G\}$. (Bases: A=Adenine, T=Thymine, C=Cytosine, G=Guanine).
     - DNA occurs as two *complementary* strands: e.g.,



– The 3D structure is a double helix:

Insert Fig 1.5, p 9

- The phosphates and sugars carry no information. Only the the bases do.
- The strands are complementary: given one you can construct the other.
  $\Rightarrow$ the information in DNA is really one long string over the letters $\{A, T, C, G\}$.
- Substrings of DNA encode useful things (proteins)
  $\Rightarrow$ substrings $\approx$ genes.
- The entire string is called a cell's genome.
- Human genome: 3 billion letters long.

- DNA's dual purpose:
  1. To encode proteins.
  2. To replicate itself.

- Encoding proteins:
  - Recall:
    * A protein is a string of amino acids.
    * There are 20 amino acids (20 letters in the Amino Alphabet).
  - Groups of bases (from the DNA Alphabet) encode amino acids.
  - How many bases are required to encode 20 amino acids?
    * How many amino acids can one base position encode?
      Answer: 4.
    * How many amino acids can two bases encode?
      Answer:
    * How many bases are needed for 20 amino acids?
      Answer:

* Thus, 3 bases are enough.
* 3 bases can encode $4^3 = 64$ things. What about the excess combinations?

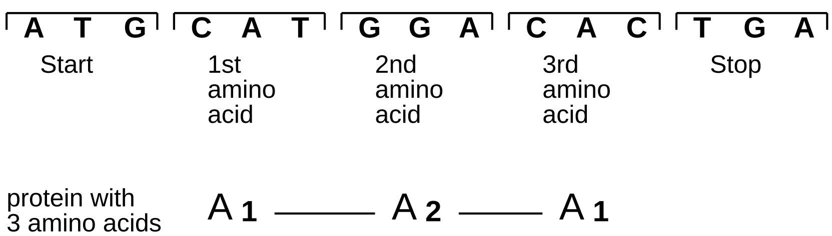
  ⇒ Many duplicates; some combinations for punctuation (start, stop).

– Thus, bases are read three-at-a-time to encode amino acids (or punctuation)

  ⇒ a long sequence of DNA can encode a long sequence of amino acids

  ⇒ a protein!

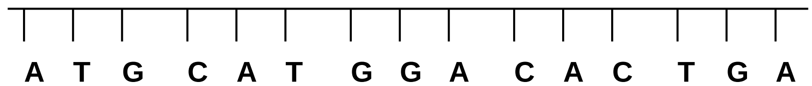
– Example:



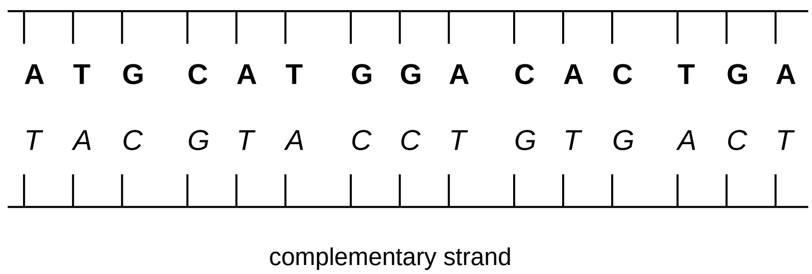
– The genetic code (the fixed translation scheme):

Page 11

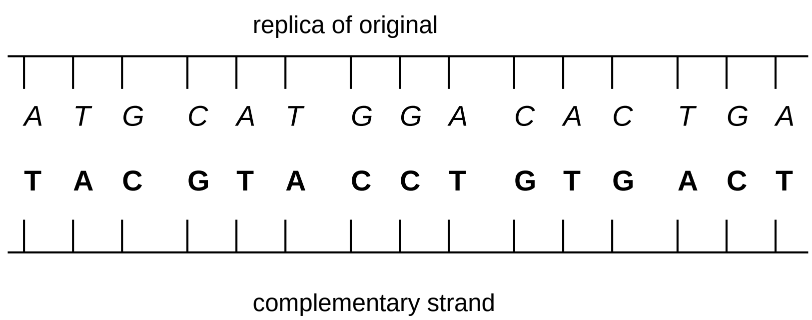
- DNA replication:
  - DNA replicates by a simple 'photocopying' process: (negative of a negative)
    - A strand separates out:



    - The strand creates its complement, drawing on available bases:



    - Then, the complement creates its complement:



- Related facts:
  - *Human Genome Project*: to obtain the entire sequence for a (any one) human.
    - Human genome: about 3 billion bases.
    - Takes days to get a 1000-letter sequence by tedious experimentation.
    - Expected completion date: 2005.
  - 98% of human DNA is thought to be useless (only 2% carries information).

– Most traits are determined by multiple genes.

– A gene has 50-500 bases
  ⇒ 100,000 genes in human DNA.

– *Mutations*: "Mistakes" during replication often result in small changes to DNA.

– *Some key players*:

  ∗ *Gregor Mendel* (1860's): principles of heredity, experiments with peas.

  ∗ *Thomas Hunt Morgan* (1910-20): confirmation of Mendel's principles by experimentation with the fruit fly.

  ∗ *Oswald Avery* (1944): identified DNA as the "information carrier."

  ∗ *James Watson* and *Francis Crick* (1953): identified the stucture of DNA (double helix).

  ∗ *Fred Sanger* (Cambridge), *Walter Gilbert* (MIT): devised DNA sequencing technology in the period 1955-75.
  (Sanger: two Nobel prizes).

1.3 A Problem in Computational Biology

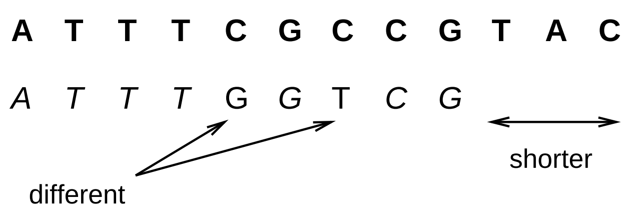
- Molecular biologists often compare DNA sequences or protein sequences from different animals to see if they are "similar."

  Why? Suppose, in animal X the DNA sequence ATTTCGCCGTAC has the function "sharp claws".

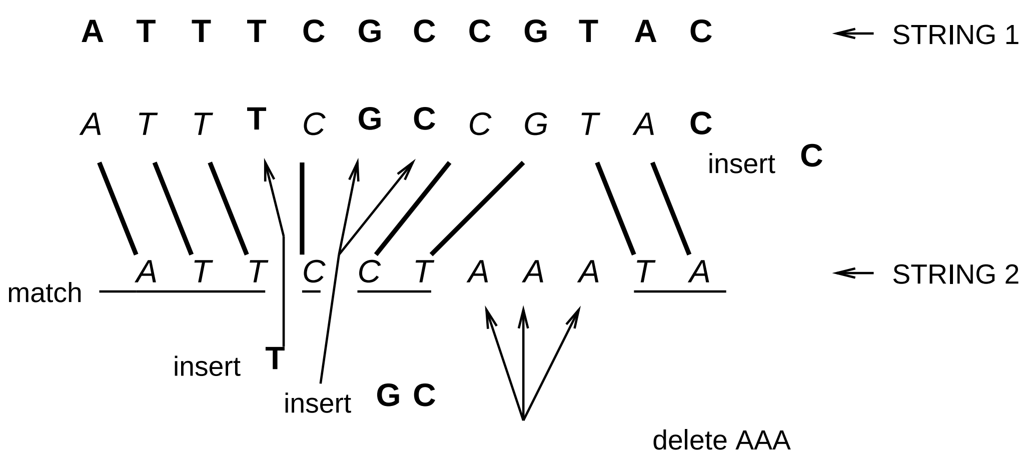
  By looking for a similar sequence in animal Y's genome, you can narrow the search for its "claw sharpness" gene.

- It would be easy if "claw sharpness" genes were identical across all animals.
  $\Rightarrow$ Unfortunately, that is not so.

- Often, they are similar, e.g.,



In fact, the similarity can be more complex:



General idea:
by some insertions, some deletions and some mismatches, two sequences can be aligned.

- We will define a "discrepancy score" for every alignment:
  - Large values for insertions and deletions.
  - Moderate values for mismatches.
  - Low or zero values for exact matches.
- Problem: given two sequences (strings), compute the minimal-discrepancy alignment.
- General problem formulation: Given
  1. An alphabet $\Omega = \{A, B, \ldots\}$;
  2. A mismatch cost function
  $$\delta(x, y) = \text{ cost when } x \text{ is mismatched with } y;$$
  3. An insertion cost function;
  4. A deletion cost function;
  5. Two sequences (strings) over the alphabet;

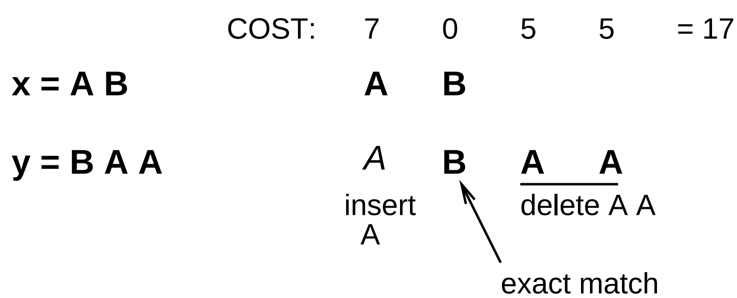
  Compute the minimal alignment.

  Note: For DNA, $|\Omega| = 4$, for proteins, $|\Omega| = 20$.
- Example:
  - $\Omega = \{A, B\}$, $x = AB$, $y = BAA$.
  - Cost table:

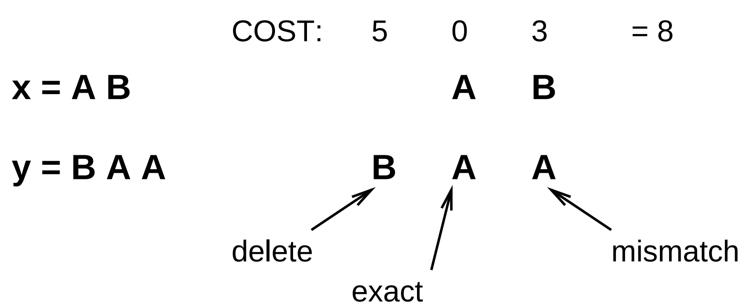
x \ y	A	B	deletion
A	0	4	5
B	3	0	5
insertion	7	8	

cost of aligning y's A with x's B

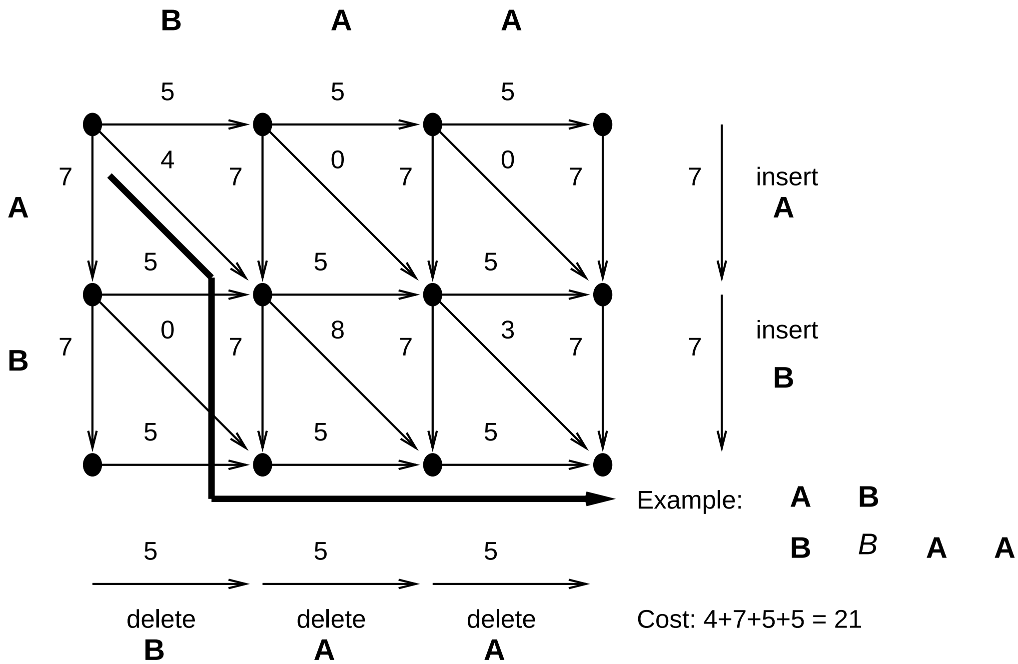
– Sample alignment:



– Another possible alignment:



• Solution method: create a cost-graph, e.g.,



Solution: find shortest path from top-left corner to bottom-right corner.

- Key ideas in algorithm:
  - For each vertex v define:

$$\begin{aligned} f(v) &= \text{shortest cost from top-left to } v \\ N(v) &= \text{vertex North of } v \\ W(v) &= \text{vertex West of } v \\ NW(v) &= \text{vertex Northwest of } v \\ \delta(u,v) &= \text{cost on edge } (u,v) \end{aligned}$$

  - Scan vertices row by row.
  - For each vertex encountered in scan:

$$\begin{aligned} f(v) = \min(\; & f(N(v)) + \delta(N(v),v), \\ & f(W(v)) + \delta(W(v),v), \\ & f(NW(v)) + \delta(NW(v),v)) \end{aligned}$$

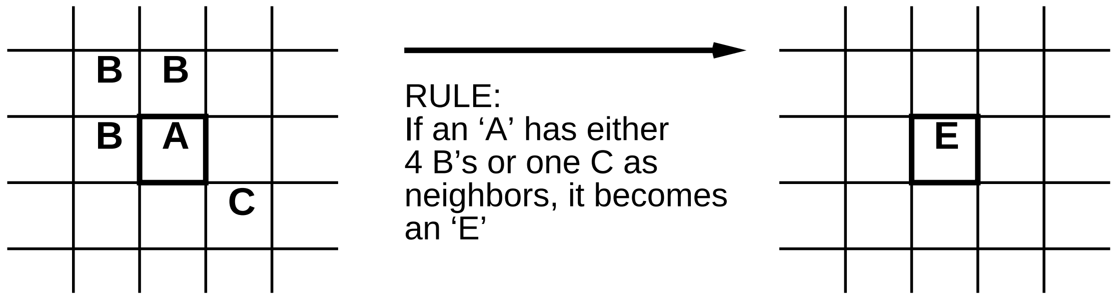
- Minimal cost: f(bottom-right).
- If strings are of length m and n, what is the complexity of the algorithm?

1.4 Summary I

- DNA information structure: string of alphabets.
- Facts about DNA:
  - Substrings encode function (protein).
  - DNA replicates by photocopying.
  - Mutations are occasional changes in DNA during replication.
- DNA sequencing poses many computational problems.
- Other computational biology problems:
  - Statistical heredity theory (mapping approximate locations of genes).
  - Search and database problems in genome databases.
  - Macro-geometry of DNA and other molecules.
  - The protein folding problem.

1.5 Von Neumann's Theory of Self-Reproduction

- John Von Neumann (1903-1957): a giant of 20-th century science:
  - 20's - 30's: Pure and applied mathematics, quantum physics.
  - 30's - 40's: Game theory and economics.
  - 40's - 50's: Computing (stored program machine).
  - 50's: Automata, self-reproduction theory.
- Von Neumann's goal: to prove mathematically that self-reproduction (or any kind of reproduction) is possible.
- First attempt: the kinematic model.
- The blueprint problem: can a blueprint contain itself?
- Second attempt: the cellular model (cellular automata).
- What is a cellular automaton?
  - An infinite 2D grid.
  - Each cell in the grid can be in one of a finite number of *states*. Set of states $\Omega = \{A, B, \ldots\}$.
  - The system evolves in discrete time steps.
  - At each step the next state of a cell depends on:
    1. its current state;
    2. the current state of each of several surrounding cells (usually, the eight neighbors), e.g.,



- The Game of Life:
  - A particular kind of cellular automaton.
  - Devised in 1970 by John Horton Conway, a Cambridge mathematician.
  - Two states: $\Omega = \{0, 1\}$ (off, on).
  - Rules for cell state-change (only consider 8 neighbors):
    1. [**Birth**] If exactly 3 neighbors are *on*, next state is *on*.
    2. [**Status-quo**] If exactly 2 neighbors are *on*, next state is current state.
    3. [**Death**] In all other cases (0,1,4,5,6,7 or 8 neighbors *on*), next state is *off*.
- Interesting configurations in the Game of Life:
  - 3-dot blinker.
  - 4 dots: block, T-tetronimo.
  - 5 dots: glider.
  - The eater.
  - The R-pentonimo.

1.6 Von Neumann's Quandary

- The problem:
  - Von Neumann wanted to show that self-reproduction is possible in the cellular world, but not in a trivial way.
  - Trivial self-reproduction: the 3-dot blinker in Life.
  - He defined non-trivial reproduction as a cellular automaton that:
    1. Embedded a Turing machine (universal Turing machine).
    2. Embedded a Universal Constructor.
       (UC: reads building instructions (blueprint) on "tape" and builds cells accordingly).
    3. Reproduced itself entirely, including blueprint.
- Von Neumann never constructed a complete self-reproducing automaton: he died before he could.

  However, most experts agree he showed that it was possible.

  Von Neumann's automaton: 29 states/cell and 200,000 cells in initial configuration:

  - He showed how a Turing machine could be constructed.
  - He showed how a Universal Constructor (UC) would work.
  - He showed the UC worked by reading instruction from a "tape" (the blueprint).
  - He showed that the blueprint problem was solvable: the blueprint was "photocopied" into the new offspring.

1.7 Summary II

- A simple mathematical object like a cellular automation is capable of complex behavior.
- It is not obvious that a given rule system will produce "interesting" behavior.
- Cellular automata have been used to demonstrate self-reproduction mathematically.
- Cellular automata are essentially a particular type of discrete-event simulation.

1.8 Genetic Algorithms

- Genetic algorithms: evolutionary biology as a metaphor for algorithm design.
- Genetic algorithms are used in optimization.
- Key ideas:
  - Start with a large number of potential solutions (initial population).
  - At each step, generate a new population:
    * Weak (high-cost) solutions "die."
    * Strong (low-cost) solutions "survive."
  - By evolution, the hope is that the optimal solution will dominate the population eventually.
- Example: we will look at a really simple problem.
  - Let $S = \{0, 1, 2, \ldots, 31\}$, the set of potential solutions.
  - Define, for $x \in S$, $f(x) = x^2$, the cost function.
  - Problem: find $x^* \in S$ that maximizes f.

- Steps in the genetic algorithm (example):
  - Step 1: Express potential solutions as bit-strings:

    $$S = \{00000, 00001, 00010, \ldots, 11111\}.$$

    Treat each bit-string as a collection of 1-bit genes
    $\Rightarrow$ each solution is a genome.
  - Step 2: Start with a large random initial population, e.g.,

Hundred each of 01101, 11000, 01000, 10011.

Thus, initial population is 400.

– Step 3: Repeat the following for a long time:

1. Compute the "fitness" (f-value) of each unique genome:

ID	Genome	Decimal	f(genome)	fraction of total (PDF)	CDF
1	01101	13	169	0.144	0.144
2	11000	24	574	0.492	0.691
3	01000	8	64	0.055	0.691
4	10011	19	361	0.309	1.000
			1170		

2. Use a genome's fraction as its survival probability for next generation.
   $\Rightarrow$ e.g., P[survival for 11000] = 0.492.
3. Create 400 new survivors based on random generation from the PDF.

   Note: at this point it seems we will never create any new genome.
4. Apply crossover rules:
   * Pick two genomes at random, e.g., 01101 and 11000.
   * Pick a random crossover point, e.g., 4-th bit.
   * Crossover to get two new genomes: 01100, 11001
     $\Rightarrow$ also called "mating."
5. Apply mutation rules:
   * For each genome and each bit, flip the bit with some small probability.

1.9 Summary III

- A biological phenomenon (evolution) was used as a metaphor for algorithm design.
- By simulating evolution, the genetic algorithm is able to solve optimization problems.
- Genetic algorithms are applied widely to discrete optimization problems.

1.10 Overview of course

- Modeling of systems with interacting components:
  - Reliability.
  - Structure of materials.
- Simulation as a research tool.
  - Artificial life systems.
- Physical and man-made systems as a metaphor for algorithm design:
  - Simulated annealing (metaphor from metallurgy).
  - Price-directed optimization (metaphor from economics).