Computing with DNA & Review and Study Suggestions

von Neumann Architecture

- Refers to the existing computer architectures consisting of
  - a processing unit
  - a single separate storage structure to hold both instructions and data
- Implements a Turing Machine
DNA Computing

- Existing genetic engineering tools can slice, create, and rebuild DNA sequences:
  - Custom sequences can be made “to order”.
  - Duplication, replication, and selection operations exist for DNA sequences.
- Many ($\approx 10^{15}$) different DNA sequences can be operated on at the same time in a single test tube.
- DNA acts as memory; computation is carried out by complementary bases.

How can DNA manipulations lead to computation?

In 1993, Leonard Adleman (USC) proposed DNA computing

- Storage
  - DNA and binary strings encode information
- Alphabet
  - DNA has a 4-letter alphabet: A, C, G, T
  - Computers have a 2-letter alphabet: 0, 1
- Operations
  - We can synthesize any strand of DNA
  - We can generate any binary sequence
How can DNA strings carry out computation?

DNA strings attach to each other (twist into a double helix) if they have complementary elements in corresponding positions.
- A and T are complements
- C and G are complements

Adleman devised a DNA manipulation technique to solve the Hamiltonian Path (HP) problem.

• DNA sequences were created
• They allowed the execution of trillions of operations in parallel

From Adleman’s ‘98 Scientific American paper
Adleman’s first DNA computation operated on a 7-node, 14-edge directed graph.

The algorithm is probabilistic.

- With high probability it will generate a Hamiltonian path, if one exists.

It is a hard-wired, not a programmed solution.

- The setup is for one graph.
- It needs to be repeated from scratch for another graph.

Demonstrated feasibility and started a new area: DNA Computing

Overview of the HP DNA algorithm

Given is a directed graph G and nodes s and t

1. Generate DNA sequences to represent nodes and edges.

2. Generate DNA strands representing paths.

3. From among the paths, select a path that:
   - begins at s and ends at t
   - has length n−1
   - is a simple path
Encoding of nodes and edges

For every node $v_i$:

generate a random DNA strand $R_i$ of length 20

For every edge $(v_i, v_j)$:

create the string $S(i,j)$, which consists of the first half of $R_i$ and the second half of $R_j$.

Example:

$v_2$ as $R_2 = ATCGAACGTTTTAACGTAGT$
$v_3$ as $R_3 = TCGAATTACGTAGAACGTTT$

dge $(v_2,v_3)$ as $TTAACGTAGTTCGAATTACG$

Initial conditions

Start node $s$ has strand $R_s$ and end node $t$ has strand $R_t$

For every edge $(s, v_i)$, create the edge strand consisting of $R_s$ and the first half of $R_i$

For every edge $(v_i, t)$, create the edge strand consisting of the second half of $R_i$ followed by $R_t$

Consider a path $< s, v_1, v_2, ... >$

$R_s = TTCAATCGCCCTAAAGGACT$
$R_1 = GTAATACGTTCGGCGTGA$
$R_2 = ATCGAACTATTAACGTTAGT$

...
What goes on in the test tube?

- Complement of the node strands – lots of them
- Edge strands – lots of them

DNA strands attach to form double strands if they have complementary elements in the corresponding positions.

Generate

\[ R_s = \text{TTCATTCGCTTTAAAGGACT} \]
\[ R_1 = \text{GTAAGTTGCTTTTCGGGCTGTA} \]
\[ R_2 = \text{ATCGAAGGTATTAACGTAGT} \]

What goes into the tube

\[ c(R_s) = \text{AAAGTAACGGAATTCCTGA} \]
\[ c(R_1) = \text{CATCATGCAAAGCGCAACA} \]
\[ c(R_2) = \text{GAGCTTGACATAATGCACTCA} \]

edge \((s, v_1)\): \text{TTCATTCGCTTTAAAGGACTGTAGTACGTTTCGGGCTGTA}

edge \((v_1, v_2)\): \text{ATCGAAGGTATTAACGTAGT}

Reaction

\[ \text{TTCATTCGCTTTAAAGGACTGTAGTACGTTTCGGGCTGTAATCGAAGGTATTAACGTAGT} \]
\[ \text{AAGTAACGGAATTCCTGACATCATGCAAAGCGCAACAGAGCTTGACATAATGCACTCA} \]
Putting it all together

1. Generate DNA sequences representing nodes and edges.
2. Mix to generate the paths.
3. From among the paths, select a path that
   - begins at $s$ and ends at $t$
   - has length $n-1$
   - is a simple path

Step 3 involves operations like merge, amplify, test-if-empty, separate, separate-by-length, separate-by-positions.
Progress since 1993

Other problems were solved using Adleman’s approach (formula satisfiability, factoring numbers, combinatorial problems with a large solution space).

It was shown that DNA computation does not provide additional computational power beyond that of a Turing machine.

Between 2002-04, researchers from the Weizmann Institute of Science (Israel) developed a programmable molecular computing machine composed of enzymes (hardware) and DNA molecules (data).

What about quantum computing?

- A quantum computer uses quantum mechanical phenomena - superposition and entanglement - to perform operations on data.
  - Quantum properties are used to represent data and perform operations.
  - Quantum computing has the potential to support a new kind on computation
  - A system with n qubits can perform $2^n$ calculations at one
  - The technology supporting quantum computation has not yet been developed
Quantum Computing

- Quantum computers would be different from traditional computers as well as DNA computers.
- A quantum computer could not solve unsolvable problems (like the Halting Problem)
- But, exponential time algorithms may have feasible solutions.
  - Fast integer factorization would break cryptographic systems
  - Security of existing methods for public key encryption (RSA systems) would no longer be guaranteed

Review and study suggestions

- Exam will have some programming questions, some explain questions
- Sample questions will be posted on Blackboard this week
- Programming questions
  - Write one line statements achieving what is specified
  - Write small programs (like in exam 2)
  - Know how to generate plots/visuals (not Cytoscape)
- Explain Questions
  - Give short answers addressing the point
  - Don’t reproduce material from sources
Review and study suggestions

- Work through lab material and lab problems
  - Understand what the focus of the lab was
  - Read posted solutions
  - Re-solve problems
  - Understand code posted in association with lectures
  - Change it, break it …
  - Make sure to review …
- Recursion
- Graph operations
- Dictionaries
- Be able to read class definitions and methods

---

Review and study suggestions

- **Chapters covered from Zelle**
  - 1, 2, 3, 4
  - 6, 7, 8, 11
  - 13 (except 13.3)
- **Software packages (basic uses)**
  - MatPlotLib, VPython, NetworkX, NumPy
- **Learning Python Book**
  - Mainly for looking up material and seeing different examples
  - There are a number of Python features we did not cover