Lecture 2:

Central Dogma of Molecular Biology
& Intro to Programming
Central Dogma of Molecular Biology

- Proteins: workhorse molecules of biological systems
- Proteins are synthesized from the genetic blueprints contained in DNA
- Code is transcribed into RNA, then translated into protein

Proteins

- Biological molecules that do all the work
- Many types:
  - Structural
  - Enzyme
- Made up of amino acids joined together by peptide bonds
Proteins

• Amino acid structure:
  – Central Carbon atom
  – Amino group
  – Carboxyl group
  – Side chain (20 AAs are distinguished from one another by this feature)

source: http://www.genome.gov/Pages/Hyperion/DIR/VIP/Glossary/Illustration
Protein structure

- All amino acids have the same “hooks” that allow them to form their primary structure chain
- The amino (NH$_2$) group of one bonds with the carboxyl group (COOH) of another
- In any amino acid chain, then, there are two ends: the N-terminus and a C-terminus

source: http://www.ichthius.info/Evolution/origin.html
Protein structure

• Primary: sequence
• Secondary: sequences linked by hydrogen bonds
• Tertiary: 3-dimensional structure based on folding
• Quaternary: complex proteins with more than one sequence
DNA

- DNA’s role is memory; stores information necessary to create proteins
- 4 Base types:
  - Cytosine (C)
  - Guanine (G)
    - G complements C
  - Adenine (A)
  - Thymine (T)
    - A complements T
Nucleotides

- DNA
- RNA
- **Structure:**
  - Sugar (ribose or deoxyribose)
  - Phosphate group
  - Base

source: [http://www.genome.gov/Pages/Hyperion/DIR/VIP/Glossary/Illustration](http://www.genome.gov/Pages/Hyperion/DIR/VIP/Glossary/Illustration)
Chromosomes & Genome

- Chromosomes: Tightly packed strands of DNA found in all cells of an organism
- Genome: Complete set of chromosomes in a cell
  - number of chromosomes in genome is characteristic of species; e.g. humans have 23 pairs
  - hundreds of millions of base pairs per chromosome
Reading DNA

• Like any string, there are two ends to a set of braided strands of DNA
• Conventionally, we read a strand from left to right – that is, from the 5’ end to the 3’ end
• Because bases are complementary, the two helices in DNA’s double helix are complementary:

Forward = Reverse Complement (Forward)

AGCTTCTAGTCGACTAGAAGCT

AGCTTCTAGTCGACTAGAAGCT

From Thomas Krahn's presentation.

source: http://berrydna.pbwiki.com/recLOH
Protein synthesis

- DNA archives code that translates into protein
- At the simplest level, a triplet of nucleotides (codon) corresponds to one amino acid
- Any given DNA strand could be read any one of three ways – since there are two strands in the double helix, there are six possible reading frames
ORFs, introns & exons

• An interval of DNA sequence that doesn’t contain a STOP codon is an **open reading frame** (ORF)

• Not all ORFs code for proteins
  – exons: coding regions
  – introns: noncoding regions

source: http://genome.imim.es/courses/Madrid04/exercises/ensembl/index.html
RNA

- RNA is the go-between: fetches information from DNA (transcription), passes it on for protein synthesis (translation)
- Similar to single strand of DNA
  - Sugar is ribose
  - Uracil instead of Thymine
RNA

- Although single-stranded, RNA molecule’s natural inclination is form base pairs, just like the pairs in DNA (A-U instead of A-T)
- Hairpin shapes (like the one shown) are the basic elements of secondary RNA structure

Important synonyms

• Building blocks of nucleic acids may be referred to by any (and all) of the following terms:
  – nucleotide (nt)
  – nucleoside
  – base pair (bp)
  – base

• The length of a DNA molecule is typically given in base pairs – so a sequence with 200 nucleotides total has length 100 bp
“Central Dogma” of computer science

• Source code: program as written by human programmer
• Compiler/interpreter: software that translates source code into form usable by computer
• Executable code: program computer can run
Programs & Programming Languages

- Program: set of instructions for completing a task
  - algorithm
  - encoded in programming language

- Programming Language
  - set of symbols (text, punctuation)
  - rules for their use
Programming Process

- Design
- Implementation
- Testing
- Maintenance
Program Design

• Describe the problem
  – Identify inputs
  – Describe required computational process
  – Identify outputs

• These are components, not steps
Example: greet the user

• Input: user’s name

• Algorithm:
  – Ask for user’s name
  – Read the input
  – Print a greeting message

• Output: greeting
Example: What is my grade?

• Inputs:
  – my scores on assignments, exams, etc.
  – points possible for all of these

• Preliminary algorithm
  – find sum of my scores (myPoints)
  – find sum of points possible (allPoints)
  – find quotient (myPoints/allPoints -> myPercent)
  – compare myPercent with grading scale for class

• Output: my grade
Another example: What’s the complement of this DNA strand?

• Inputs:
  – a string of bases
  – table of complements

• Algorithm:
  – match each base with its complement & produce a new string

• Output: the complementary strand
Algorithm refinement

• Match each base with its complement
  – Examine each character in the string individually
  – if the character is A, replace with T, and vice versa
  – if the character is C, replace with G, and vice versa

• Produce a new string
  – Create a blank string
  – As each replacement character is produced, append it to the end of this new string

• When all bases have been examined and replaced, we are done
Programming & control structures

• A program is a set of instructions

• In the “normal,” or default case, your program proceeds as follows:
  – read & execute 1\textsuperscript{st} instruction
  – read & execute 2\textsuperscript{nd} instruction
  ...
  – read & execute nth instruction
  – done

• This way of doing things is called sequential control – we are running each instruction in sequence
Programming & control structures

• For a very simple algorithm, like the first example, a sequential control pattern works just fine
  – We do each step in order
  – We do each step just once
  – When we have done each step, we quit

• The other two algorithms are more complicated
  – Both require that we repeat some steps
  – Both require us to make decisions
Control structures: loops

• A loop is a set of instructions that is repeated within a program
• We can illustrate the way a loop works using a flow chart like the one on the right
• The shapes represent individual instructions
• The arrows indicate where we should go after each instruction
Loops

- We start by setting up a control variable – this is a value we can check to see if we should continue.
- The next step (diamond in the picture) is the decision point: should we continue?
- If the answer is yes, we proceed into the body of the loop and perform the instructions there.
- When we finish a pass through the loop, we go back to the decision point to find out whether or not we should repeat the process.
Selection structures

- The flow chart on the right illustrates a simple selection structure.
- The diamond shapes again represent decision points.
- Each one is a fork in the road, indicating we have to pick one direction or the other.
Implementation

• At this stage, we are ready to encode our program in a programming language
• For this course, the language in question will be Perl
• We will start working with Perl next time
Testing and Maintenance

• Testing is sometimes considered part of the implementation process: this is where you determine whether or not your program works.

• The maintenance phase is usually the longest in the software life cycle:
  – changes/improvements made over time
  – bug fixes, new versions