LC101 JavaScript Track: Class 3.2

Error-Correcting Codes (1)

PART 1: THEORY & SIMULATION

The genetic code is the set of rules by which information encoded within genetic material is translated into creating proteins. DNA is molecule that carries the instructions used in the growth, development, and reproduction of organisms.

DNA consists of basic building blocks called *nucleotides* which can be composed of one of the following four bases:

1.) Cytosine (C)
2.) Guanine (G)
3.) Adenine (A)
4.) Thymine (T)

Furthermore, a sequence of nucleotide triplets such as CGT forms a *codon* (kind of like how letters spell out words). Each codon specifies which amino acid will be added next according to the table below:

Amino acids biochemical properties | nonpolar | polar | basic | acidic

Termination: stop codon

Standard genetic code

1st base	2nd base T		2nd base C		2nd base A		2nd base G		3rd base
T	TTT	(Phe/F) Phenylalanine	TCT	(Ser/S) Serine	TAT	(Tyr/Y) Tyrosine	TGT	(Cys/C) Cysteine	T
	TTC		TCC		TAC		TGC		C
	TTA	(Leu/L) Leucine	TCA		TAA[B]	Stop (*Ochre*)	TGA[B]	Stop (*Opal*)	A
	TTG		TCG		TAG[B]	Stop (*Amber*)	TGG	(Trp/W) Tryptophan	G
C	CTT		CCT	(Pro/P) Proline	CAT	(His/H) Histidine	CGT	(Arg/R) Arginine	T
	CTC		CCC		CAC		CGC		C
	CTA		CCA		CAA	(Gln/Q) Glutamine	CGA		A
	CTG		CCG		CAG		CGG		G
A	ATT	(Ile/I) Isoleucine	ACT	(Thr/T) Threonine	AAT	(Asn/N) Asparagine	AGT	(Ser/S) Serine	T
	ATC		ACC		AAC		AGC		C
	ATA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	A
	ATG[A]	(Met/M) Methionine	ACG		AAG		AGG		G
G	GTT	(Val/V) Valine	GCT	(Ala/A) Alanine	GAT	(Asp/D) Aspartic acid	GGT	(Gly/G) Glycine	T
	GTC		GCC		GAC		GGC		C
	GTA		GCA		GAA	(Glu/E) Glutamic acid	GGA		A
	GTG		GCG		GAG		GGG		G

In summary, nucleotides spell out codons, and codons reference some biologically-important compound to be created. The entire sequence is read, triplet-by-triplet until a *stop codon* is reached. There are three stop codons: TAA, TAG, and TGA (think of this as a stop sign that tells the reader we've reached the end).

TODO: write a function `generate()` that randomly generates a string of nucleotides, with a space in between each codon. Keep going until you randomly encounter a stop codon. Then, return the string.

EXAMPLE: ATT ACC CCT CAT GAG GAT **TGA**

Occasionally, random mutations will occur due to errors in the DNA replication process. Although these usually have little effect, they may cause cancer or other genetic disorders if uncorrected.

TODO: Write a function `mutate(strand)` that takes in a generated DNA strand and randomly reassigns a nucleotide about 10% of the time. In reality, there are lots of pre-existing processes to correct these errors, and the error rate is really small (roughly 10^{-9}; the human genome has about 3×10^{9} nucleotides). However, we're not about to print out a trillion numbers.

EXAMPLE (from above): ATT ACC CCA CAT GTG GAT TGA
In this case, 'C' was randomly changed to 'C' again, and 'T' was randomly changed to 'A'.

Next, DNA is transcribed into RNA. In this process, we "pair up" each of the nucleotides in the DNA according to the following rules:

1.) G → C
2.) C → G
3.) T → A
4.) A → U

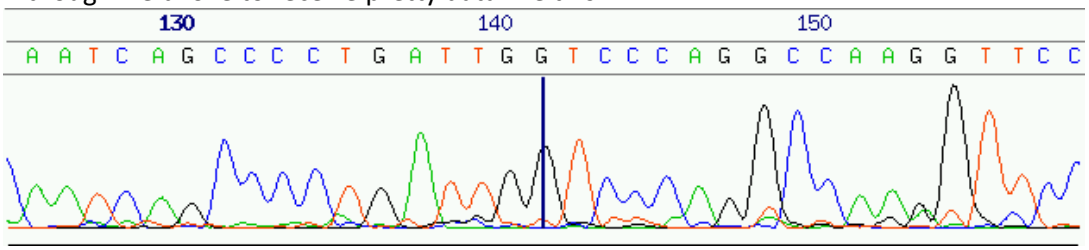
Where 'U' is a nucleotide found in RNA called Uracil.

TODO: Write a function `transcribe(strand)` that takes in a DNA strand and returns the complementary RNA strand.

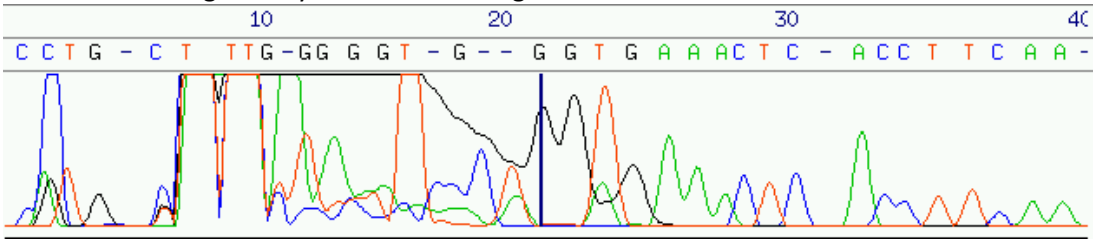
EXAMPLE (from our mutated strand above): UAA UGG GGU GUA CAC CUA ACU

PART 2: THE LAB & REALITY You're now a lab technician working on sequencing the human genome. Unfortunately, the data we're receiving is muddled with noise due to a variety of reasons such as flaws in the translation operations of the electrophoresis signal, stochastic operations in chromatography, and other fancy-sounding processes that sound like nonsense.

Although we'd love to receive pretty data like this…



…we sometimes get noisy data with missing or incorrect values.



As a result, we may need to perform these measurements several times. Let's say we do this twice.

Attempt 1: ATC CTG GTA AGT AGG

Attempt 2: ATG GTG ACG GGG ACG

To keep it simple, let's assume we'll always measure *something*, even though the two attempts might not match. The Hamming distance between two strings is simply the number of positions where the corresponding symbols are different. For example, AT**C** and AT**G** have a Hamming distance of 1. Meanwhile, **A**G**T** and **G**G**G** have a Hamming distance of 2.

TODO: Write a function `hammingDistance(str1, str2)` that takes in two strings and returns their Hamming distance.

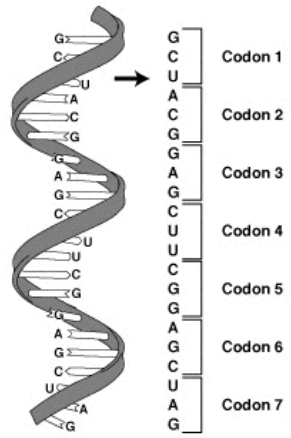
Lastly, let's write a simple program to correct these errors after transcription. Here's a table (provided to you as a JSON file) of how frequently a particular codon shows up in the human genome (per thousand):

```
UUU 17.6  UCU 15.2  UAU 12.2  UGU 10.6
UUC 20.3  UCC 17.7  UAC 15.3  UGC 12.6
UUA  7.7  UCA 12.2  UAA  1.0  UGA  1.6
UUG 12.9  UCG  4.4  UAG  0.8  UGG 13.2

CUU 13.2  CCU 17.5  CAU 10.9  CGU  4.5
CUC 19.6  CCC 19.8  CAC 15.1  CGC 10.4
CUA  7.2  CCA 16.9  CAA 12.3  CGA  6.2
CUG 39.6  CCG  6.9  CAG 34.2  CGG 11.4

AUU 16.0  ACU 13.1  AAU 17.0  AGU 12.1
AUC 20.8  ACC 18.9  AAC 19.1  AGC 19.5
AUA  7.5  ACA 15.1  AAA 24.4  AGA 12.2
AUG 22.0  ACG  6.1  AAG 31.9  AGG 12.0

GUU 11.0  GCU 18.4  GAU 21.8  GGU 10.8
GUC 14.5  GCC 27.7  GAC 25.1  GGC 22.2
GUA  7.1  GCA 15.8  GAA 29.0  GGA 16.5
GUG 28.1  GCG  7.4  GAG 39.6  GGG 16.5
```



If there's a distance of 1 between codons, such as UAC and UGC, compare them in the table and select the one with the highest chance of happening. In this case, we'd choose UAC instead of UGC because 15.3 > 12.6. If the distance is greater than 1, then simply ignore the error.

TODO: Write a function `merge(strand1, strand2)` that takes in 2 sampled RNA strands, makes any necessary corrections codon-by-codon, and then returns a single corrected strand.

EXAMPLE: AUC ACG UAA
AUU GCA UAA → AUC GCA UAA

TODO: Create a HTML page that simulates the whole process. First, generate two DNA sequences and mutate one of them (this simulates taking two different measurements). Display both of these strands, highlighting the errors. Include a "Transcribe" button, which displays the corresponding RNA strands when clicked. Afterwards, remove the "Transcribe" button and show a "Correct" button, which displays the final corrected strand. A pair of DNA sequences should be generated each time the page is refreshed.