1. Set up

We’re back to Go for this assignment.

1. Inside of your src directory, create a directory called fold.

2. Copy the canvas.go file from HW4 and put it into the fold directory. Also download the code.google.com directory from Piazza if it isn’t already in your src directory.

3. Set your GOPATH environment variable to the location of your go directory that you made above. On a Mac,

   ```
   export GOPATH=/Users/pcompeau/Desktop/go
   ```
   
   where you replace the directory name after the = with the location of the go directory you just made.

   On Windows, use

   ```
   set GOPATH=C:\Users\pcompeau\Desktop\go
   ```

2. Assignment

2.1 Protein folding

Proteins are linear molecules that primarily consist of a chain of amino acids strung together. There are 20 commonly occurring amino acids in most organisms, so we can think of a protein as a string formed from a 20-symbol alphabet. This is called the primary structure of the protein. Yet the function of a protein also depends on its three-dimensional structure after it folds in on itself (called its secondary structure). A fundamental (and difficult!) challenge in modern biology is therefore to predict a protein’s secondary structure from its primary structure.

For example, here are two example protein folds and their amino acid sequences:
To begin modeling this biological problem computationally, we assume that a protein folds into its lowest-energy conformation. So if we assume that we have a function $\text{energy}(S)$ that takes a structure and evaluates its energy, we are looking for a structure $S$ that minimizes $\text{energy}(S)$. While a lot of progress has been made on this problem over the years (including by the winners of the 2013 Nobel Prize in Chemistry), the problem remains difficult; attempting to formulate a decision problem for whether a given amino acid sequence can fold into a protein of energy beneath some threshold leads to NP-Complete problems everywhere we look.

### 2.2 The HP model

In order to make progress on several aspects of the protein folding process, a simplified model — introduced by Dill\(^1\) — has been studied. This model is called the “HP model” because we simplify the 20 amino acid alphabet down to just two types of amino acids: “H”, which are hydrophobic (like to be away from water), and “P” which are polar (don’t mind being near water).

The second simplification in our case is that proteins are modeled in 2-D instead of 3-D, and that they only fold on a square lattice. The following are examples of such an HP protein fold, where the circles represent amino acids and the solid nodes are the “H” amino acids (the leftmost HP protein has the sequence $\text{HHPHPHHPPHH}$):

Notice that each folded protein is represented by a walk through the lattice, in which case we can

---

represent such a walk by a sequence of three relative directives: right \((r)\), left \((l)\), forward \((f)\). The walk is described as though you were giving driving directions. For example, a walk that looks like the number “7” would be represented as \(\langle r, r, f \rangle\) (we assume that initially, the “walker” is facing up). Almost all walks can be described in this way; the only exception to this is the first point: in this scheme, we cannot represent the walk that looks like a “u” because there is no way to specify “go backwards” at the first point. Fortunately, this does not matter since we don’t care about the orientation of the protein. For example, the leftmost structure in the row of 4 above, is specified by \(\langle l, l, r, f, r, f, r, f, r, r, f, l \rangle\). This sequence of instructions is called the fold or the structure.

The last simplification we make relates to the energy function. When dealing with real proteins, the energy function usually is a combination of many physical energies. For the HP model, the energy function measures how “buried” the “H” atoms are. Let the amino acid sequence be given by \(\langle p_1, p_2, \ldots, p_n \rangle\), where \(p_i = 1\) if the \(i\)th amino acid is an “H” and \(p_i = 0\) otherwise. Then the energy of a folded protein is

\[
\text{energy}(S, \vec{p}) = 10x - \sum_{i=1}^{n} p_is_i,
\]

where \(s_i\) equals the number of amino acids at neighboring lattice points, including diagonals, that are not adjacent to the \(i\)th point in the walk, and \(x\) is the number of times the walk crosses itself. In this example,

\[
\begin{array}{ccccccc}
& & & & & & \\
& 
\cdot & & & & & \\
& 
\cdot & & \cdot & & \cdot & \\
\cdot & & \cdot & & \cdot & & \\
& & \cdot & & \cdot & & \\
& & & & \cdot & & \\
\end{array}
\]

\(s_i = 4\) because of the 8 nearby points, 4 contain amino acids and are not adjacent to \(i\) in the structure.

Our goal in this assignment will be to write a program that is given an HP protein sequence \(\vec{p}\) and finds a structure \(S\) (walk along a square lattice) that minimizes \(\text{energy}(S, \vec{p})\). An alternative problem formulation is to find a non-crossing structure \(S\) that minimizes \(\text{energy}(S, \vec{p})\).

### 2.3 Simulated annealing

To find a low energy structure, we will use simulated annealing, which is a widely useful optimization framework. Our simulated annealing framework depends on several parameters and works as follows:

1. Set \(i = 1\); Choose a random structure \(S_i\) (i.e. a random sequence of \(l, r, f\)).
2. Compute \(E_i = \text{energy}(S_i, \vec{p})\).
3. Change a random letter of the walk \(S_i\) to a random different letter to obtain a slightly different structure \(S'_i\).
4. Compute \(E'_i = \text{energy}(S'_i, \vec{p})\).
5. If $E_i' < E_i$, set $S_{i+1} = S_i'$. Otherwise, if $E_i' > E_i$, let $q = e^{-(E_i' - E_i)/kT}$; with probability $q$ set $S_{i+1} = S_i'$, and with probability $(1 - q)$ set $S_{i+1} = S_i$.

6. Let $i = i + 1$, and if $i \mod 100 = 0$ let $T = 0.999T$.

7. If the structure hasn’t changed for $m$ iterations, stop; otherwise go to step 2.

8. Return the $S_j$ with the lowest energy.

In other words, we keep making random small changes to the structure. If the change improves the structure (lower energy), we keep the change. If the change doesn’t improve the energy, we keep the change with probability $e^{-(\Delta)/kT}$

where $\Delta$ is the amount by which the energy went up. The idea here is that we always accept improvements, and we accept “bad” changes with probability inversely proportional to how bad they are. The constant $k$ is a normalizing factor to scale the energy differences.

Variable $T$ is the interesting one: when $T$ is large, we are more likely to accept changes that increase the energy, and when $T$ is very small, we nearly never do. $T$ can be viewed as a temperature of the system: when the system is “hot”, we bounce around a lot, and as it “cools” we head towards low-energy solutions.

Reasonable choices for the parameters for a protein of length $n$ are $m = 10n$, $k = 6n$, and the initial $T = 10n$.

2.4 What you should do

You should write a Go program that can be run with the following command line:

```
./fold HP
```

where HP is a string of H and Ps representing the HP protein sequence.

Your program should output two lines of text:

```
Energy: ENERGY
Structure: lrf
```

where ENERGY is the energy (integer) of the final structure, and lrf is a sequence of $l, r, f$ that specifies the lowest energy structure you found.

You should also write a file called fold.png that draws the structure. You can use any nice format for the drawing that makes the Hs dark colored and the Ps light colored. For example using edges of length 10 pixels, and drawing the H-residues as solid black squares and P-residues as solid white squares with a black boarder. Drawing the final fold is extra credit for 02-201 students.

Note: You can use any algorithm you want to find a low-energy structure. Of course, simulated annealing is a good choice, but you can modify simulated annealing if you need to in order to obtain a better structure. You can also modify aspects of the basic simulated annealing algorithm above. For example, you can change how often $T$ is changed (maybe even sometimes increasing it), or you could run several independent simulated annealing runs from different starting points, and return the best solution found, etc.
2.5  Tips on how to start

First write the main function to parse the arguments from the command line.

Then write a function `randomFold` that generates a random fold as a sequence of `l`, `r`, and `f` commands.

Then, write a function `drawFold` that “draws” the fold onto a sufficiently large 2-D matrix.

Next, write and test a function `energy(S, p)` that computes the energy of a fold `S` and sequence `p`. This function probably calls `drawFold`.

Next, write a `randomFoldChange` function that takes a fold and randomly changes one of its commands.

Next, write `optimizeFold(p)` that returns the lowest-energy fold you can find for `p` (likely by running the simulated annealing algorithm). Play around with the parameters and the algorithm to see if you can get better and better folds.

Finally, write `paintFold` that draws the fold onto a canvas and saves the canvas to `fold.png`.

2.5.1  Tip for getting your program to run faster:

In the simulated annealing code, if you find a new structure with the same energy, don’t switch to it.

2.5.2  Tips for making the code easier to write:

- For the drawing, you can use any format you want for the drawing so long as it shows the structure in a reasonable way.

- The data structure the solution uses for laying out the fold is `[][]string`. Then `M[x][y]` lists all the residues that are in that position (if crossings are allowed).

- If you’d prefer, you can simply reject structures that have crossings (i.e. they have infinite energy).

- Don’t try to track which amino acids are adjacent to one another. Rather, lay them out in a 2D array, compute the score ignoring which residues are adjacent in the walk, and then subtract 2 for every `H` in the middle of the sequence, and 1 for every `H` at the ends of the sequence.

3.  Learning outcomes

After completing this homework, you should have

- implemented a simulated annealing optimizer (or other approach for optimizing);
- learned about the HP protein folding model;
- gained additional practice writing Go programs.