# ISyE 3133 Project

The Nucleic Acid Folding Problem is to **predict the secondary structure** of an Nucleic Acid molecule, given only its nucleotide sequence. We let S denote a string of $n$ characters made up of the Nucleic Acid alphabet $\{A, C, U, G\}$. For example, $S = ACGUGCCACGAU$. In this problem, we represent the string as a circle, as in Figure 1.

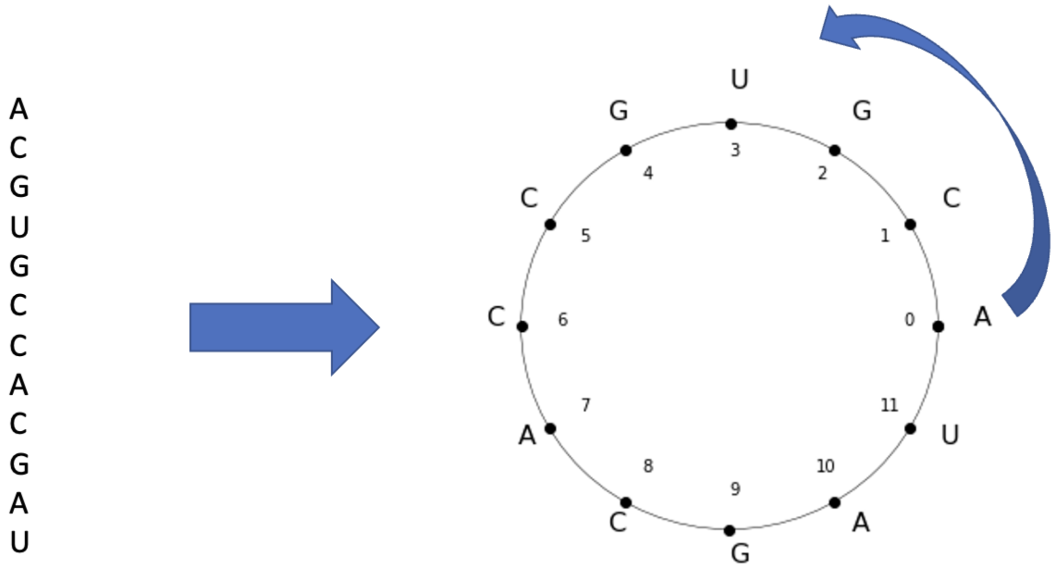

Figure 1: Converting a nucleotide string to a circular string

The secondary structure that we try to predict is the **pairing of nucleotides** in the circle. Every nucleotide can be in at most one pair. An example of a pairing can be seen in Figure 2.

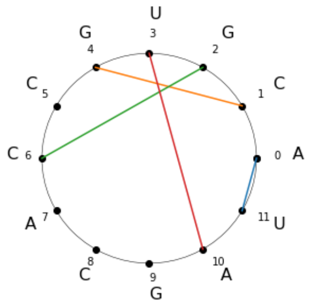

Figure 2: Pairing $\{(0, 11), (1, 4), (3, 10), (2, 6)\}$

We will create a number of integer linear programming models to predict the pairing of a circular string of nucleotides. (Given a particular input string, like $S$, what is the pairing?) Each model will be formulated using different assumptions about the most likely pairing.

Note: this project assumes no prior knowledge of biology, only understanding of integer linear programming formulations and modular/clock arithmetic (see review at end of this document).

## 1 The first crude model

A **nested pairing**, or non-crossing pairing, is a pairing for which each pair is connected by a line inside the circle and none of the lines cross. See Figure 3.

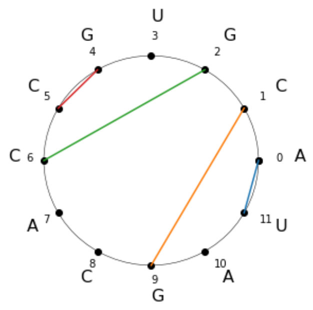

Figure 3: A nested pairing with complementary nucleotides. This pairing is feasible for Model 1.

In the simplest model we consider, we assume:

- Only **nested pairings** are allowed
- Only complementary nucleotides $(A, U)$ and $(G, C)$ can be in a pair
- The most stable pairing is the most likely
- The most stable pairing is the one with most matched pairs

Formulate an integer linear program to find the most likely pairing under these assumptions.

## 2 Simple biological enhancements

In our next model, we assume:

1. Only nested pairings are allowed
2. Any pair of nucleotides must be at least distance 3 away from each other
3. Complementary matched pairs $(A, U)$ and $(G, C)$ are allowed
4. Some non-complementary matched pairs, $(G, U)$ and $(A, C)$ are allowed
5. The most stable pairing is the most likely
6. The stability of a pairing is a weighted function of the number of each type of pairs

The weight of different pairs can be seen in Table 1.

| Pair | Weight |
|---|---|
| $(G, C)$ | 3 |
| $(A, U)$ | 2 |
| $(G, U)$ | .1 |
| $(A, C)$ | .05 |

Table 1: Weights of pair types in the stability function in model 2

Formulate an integer linear program to find the most likely pairing under these assumptions.

## 3 More complex biological enhancements

**Stacked quartets.** A stacked quartet consists of two matched pairs $(i, j)$ and $(i+1, j-1)$. Figure 4 shows two stacked quartets: $\{(9,1),(10,0)\}$ and $\{(8,2),(9,1)\}$.

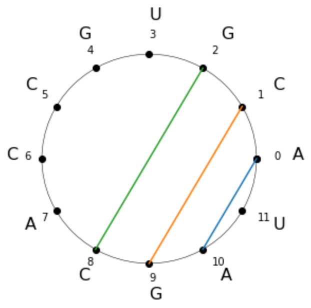

Figure 4: Two stacked quartets, $\{(9,1),(10,0)\}$ and $\{(8,2),(9,1)\}$

**Note.** Be careful! The stacked quartet $\{(1,0),(2,11)\}$ could be seen as having $i = 1$ and $j = 0$ (thus $i+1 = 2$ and $j-1 \mod 12 = 11$) or $i = 11$ and $j = 2$ (thus $i+1 \mod 12 = 0$ and $j-1 = 1$). Make sure not to double count a stacked quartet.

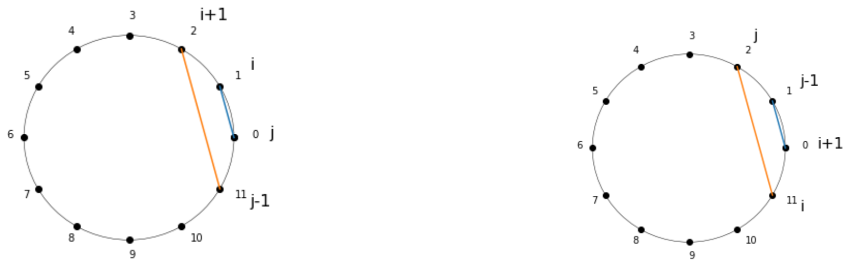

Figure 5: Two ways to label a stacked quartet

In our next model, we assume

1. Any pair of nucleotides must be at least distance 3 away from each other
2. Complementary matched pairs $(A, U)$ and $(G, C)$ are allowed
3. Some non-complementary matched pairs, $(G, U)$ and $(A, C)$ are allowed
4. The most stable pairing is the most likely
5. The stability of a pairing is a weighted function of the number of each type of pairs and the number of **stacked quartets**

The weights of different pairs and the stacked quartets can be seen in Table 1.

| Item | Weight |
|---|---|
| $(G, C)$ pair | 3 |
| $(A, U)$ pair | 2 |
| $(G, U)$ pair | .1 |
| $(A, C)$ pair | .05 |
| Stacked quartet | 1 |

Table 2: Weights in the stability function in model 3

Formulate an integer linear program to find the most likely pairing under these assumptions.

## 4 A model with crossing pairs

In our most complex model, we adjust model 3 to allow up to ten crossing pairs. Figure 2 has three crossing pairs.

## 5 Report on Formulation (Due March 28)

A report should be written on describing your formulation for the above problem.

1. Introduction. Present the optimization problem as if I knew nothing about it beforehand.
2. Model. Describe the way you have chosen to model the problem. What are your variables, objective, and constraints? Do not just write out the formulation - explain each part, as if the reader was only slightly familiar with optimization. What is the size of your formulation as compared to the input size? Thus if the size of the sequence has $n$ symbols, how many constraints and variables does each component of the model need as a function of $n$?

## 6 Feedback on Formulations (Optional; Due March 15)

If you submit your formulations for model 1 and 2 by the Friday before spring break, we will give you feedback that you can incorporate before the report is due. If you submit additional models by this date, we will give you feedback on those as well. You must have formulations for both model 1 and 2 to receive feedback.

## 7 Final Report: Implementation and Analysis (Due April 16)

**Solution and Analysis.** Describe (in words as well as numbers) your optimal solution.

1. Solve each of the models on the strings given in the data file. Plot the running time as the size increases.

Discuss the following questions and possibly other issues that you may have encountered.

1. How does your model scale with the size of the input? How many variables and constraints do you have depending on the size of the string in each of the models.
2. Which of the constraints are the most complicated to solve?
3. Did you update your model while implementing? Why and what updates did you do?

Your first report must be typed and at most 6 pages long. Use the answers to the questions given in the previous section to guide your write-up. Please submit your code along with the report.

## A Modular Arithmetic or *Clock* Arithmetic

The $\bmod$ function is useful to us when working with indices on a circle. Take the indices $\{0, \ldots, 11\}$ on the circle in Figure 6. For any integer $x$, $x \mod 12$ is one of these indices. For instance, $15 \mod 12 = 3$ and $-2 \mod 12 = 10$.

If I want to apply some function to the indices $\{2, 3, 4, 5\}$ in Python, it is pretty straightforward. We will use the print function as an example.

```
  for i in range(2,6):
     print(i)
```

I can use modular arithmetic to extend this idea to indices $\{10, 11, 0, 1\}$.

```
for i in range(4):
   x = (10+i) % 12
   print(x)
```

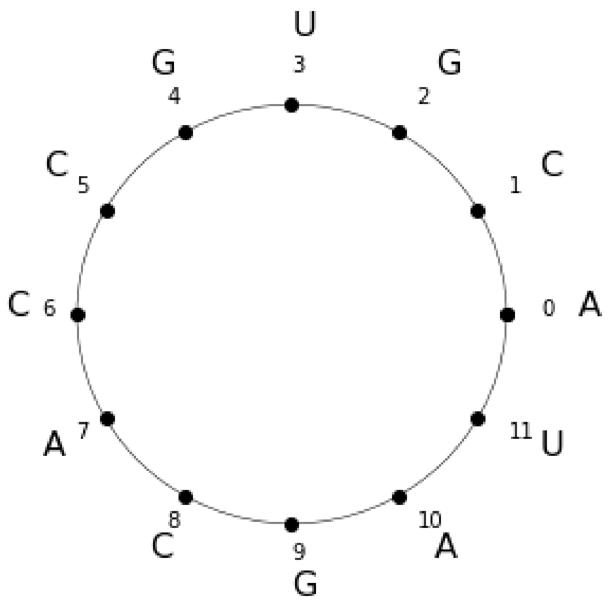

Figure 6: An example for modular arithmetic.