Dynamically Rearranging Synteny Blocks in Comparative Genomes

With this project, I built a program that takes in two strings, and finds the least number of flipped sections necessary within the first string to get as close as possible to the second.

This project was inspired by some of the subjects tackled near the end of the year in learning about the existence of synteny blocks and the surprisingly few flips of DNA sections necessary to get from one species’s genes to another. The example of the mice and human’s *entire chromosome X* only needing 7 total flips was astounding, and this project was an attempt to begin approaching a method that could produce the best possible flips with any two DNA sequences - whether they be a dozen bp or as many bp as a chromosome.

For data to be analyzed, unfortunately I never ended up using real genes. The primary concern of this project was runtime, which will be attended to later, and real genes are big - really, really big for the scale of this program’s mechanics. While I constructed test sequences of various sizes that would include all the sorts of variation that would exist between two sequences, I have yet to use it in a real world setting. It would require converting a DNA sequence’s export data into a string, but that would be all.

With these test sequences, however, I was still able to construct what I feel with sufficient processing power/time would be an effective program. The program starts with two strings, one that is to ultimately be transformed into the other. From the first string, every possible string that could be made from flipping pieces of the first string is produced. Each of these strings is then compared with the ultimate string through GlobalAlignment, and the strings with the best score are retained. Then, the same process occurs for these strings through a recursion loop. This continues until either the # of loops reaches a cap that is set ahead of time, or an exact match with the ultimate string is found. Once either occurs, a sequence of strings, and the flips needed to produce them, is created starting with the first string, and then ending at the ultimate string.
The sequence would then be uploaded onto GraphSpace in a format similar to that drawn below:

Fig. 1: Mock-up of GraphSpace representation

However, I found a lot of difficulty in discerning the GraphSpace API, and it was very late into the process of producing this section of the program that I did not end up producing such graphical representations of the sequence process. However, the lists that would be necessary to do so are easily accessible at the 'end' of the program's functioning if time is made to do so.

The results of the program itself were surprisingly effective. In fact, even the test strings that I had been using at the start I thought needed a minimum of three flips to transform them, but the program found a way to do it in two! (Though that may speak more to my inadequacy than the program's efficacy.) However, it did continue to work for whatever strings I threw at it after some bug-fixing - I just have yet to approach the largest strings as a result of runtime. Since it uses an alignment score for choosing optimal arrangement, it is able to work even with strings of different length or very low same-nucleotide conservancy. The most significant difficulty that I ran into as a result of this program was absolutely runtime. Since it's so reliant on a recursive function that takes advantage of a greedy algorithm, runtime expounds exponentially in huge amounts with every extension of length to the string inputs.

There is still plenty to do with this program - primarily in the realms of optimization and figuring out the intricacies of GraphSpace or other graphing tools that might be effective for this. If anything, the command line output is readable
enough that one could hand-produce a graph from the data provided. The program was a blast to produce, and I am proud of the results that I’ve made so far. Hopefully I’ll have a chance and the motivation to fully finish it out in future research! Thank you for teaching this course - I learned a lot and have a whole lot more confidence in my skills as a biologist, as well as a computer scientist. Have a great day, and I’ll see you around!