CS 476/590 Bioinformatics Project III:

DNA Re-Assembly, a.k.a. Solving Shortest Superstring

Description: You are to implement the graph-theoretic Shortest Superstring problem presented in class and in Chapter 8 of your text.

Specifications: Your input is a set of DNA « fragments » (motivated by snippets that may result from Next Generation Sequencing technologies). Your goal is to reconstruct the larger contiguous DNA strand that the fragments were sequenced from. There is a good degree of expected **overlaps** between fragments that guides the problem formulation. In fact, you want to find the **shortest contiguous DNA sequence that contains all input fragments.** This is equivalent to finding the **DNA sequence containing all input fragments that maximizes the TOTAL OVERLAP between consecutive fragments.**

However, be careful to first DISCARD any fragment that is a proper SUBSTRING of any other fragment.

After doing so and computing the OVERLAP matrix, you may view each remaining fragment as a NODE i in a directed graph G. And G has all possible edges (i,j) weighted such that the SCORE of edge (i,j) is the value Overlap(i,j). A Hamiltonian Path in such a graph is a path that visits all vertices exactly once. Shortest superstring reduces to finding a Hamiltonian path with greatest total Overlap sum. And, following the path, you may reconstruct the Contig (the re-assembled superstring DNA strand).

Although finding Hamiltonian paths in general graphs is a hard problem (and such paths need not exist in general graphs), the types of graphs based on possible overlaps (including allowance for overlaps of zero) are completely connected and permit many possible Hamiltonian paths. The more difficult optimization is the Ham-path of maximal total overlap. However, we are contending with a **heuristic** approach here, based on a **greedy approximation** : Sort edges in descending order of their overlap scores, and proceeding in the order from maximal to minimal overlap, select the current edge as long as it is feasible to do so. All this means is that selecting the edge would NOT induce any of the following :

- A « fork » situation where there are two out-edges from a single node
- An inverse « fork » situation where there are two in-edges to a single node
- A cycle, meaning that there is already a path from the « to edge » back to the « from edge »

Remembering that a Hamiltonian path has |V|-1 edges, you know when to stop the loop.

I have provided input files based on examples covered in slides and lectures : **fragex1.txt** and the text example is **fragtext162.pl**.

I will demo a Perl implementation of this project – at least demo it…