

## 1 Sequencing by Hybridization

For each sequence we obtain the list of all the  $l$ -mers that occur by seeing what they hybridize to on a DNA array. In practice there are issues with an  $l$ -mer occurring more than once and with false-positives, but we will assume these do not occur. Assembly using these  $l$ -mers is somewhat similar to the assembly with very short fragments as described by the guest lecturer.

Formally, for sequencing by hybridization (SBH) we want to obtain a string where all the  $l$ -mers that are substrings of that string is exactly the set of  $l$ -mers we obtain. There are two approaches to solving this problem that I will discuss. More details on the background and algorithms (such as proofs) can be found in chapter 8 of the Jones and Pevzner book.

## 2 Hamiltonian path approach

The Hamiltonian path of a graph is a sequence of vertices connected by edges that visits each vertex exactly once. Determining (much less obtaining) the Hamiltonian path of a graph is NP-complete, meaning it is very computationally difficult (we can't really do too much better than trying all possible paths, unless  $P = NP$ ).

We can pose the SBH problem as the solution to a Hamiltonian path for a graph where each node is an  $l$ -mer that occurs and there is an edge from  $x_1$  to  $x_2$  if the last  $l - 1$  characters of  $x_1$  equals the first  $l - 1$  characters of  $x_2$ . We simply overlap the  $l$ -mers along the Hamiltonian path to obtain the desired string.

## 3 Euler path approach

The Euler path of a graph is a path that visits each edge exactly once. It is not too difficult to prove that a connected graph has a Euler path if and only if the absolute difference of the in-degree and the out-degree for every vertex is 1 for at most 2 vertices and 0 for the rest. Thus, Euler path is certainly not NP-complete (we can check whether or not a graph has an Euler path by just looking at each of the vertices). Actually computing the Euler path is fairly simple, as we will see.

To find the Euler path, start with an arbitrary node. Perform a traversal until it is not possible to move any further. Then start at another node and continue. By merging these paths and cycles as described in the book, we will obtain a Euler path.

The SBH problem can be transformed into an Euler path problem by having the  $l$ -mers represent the edges between vertices that represent  $l - 1$ -mers. We have an edge from  $x_1$  to  $x_2$  if there is an  $l$ -mer whose first  $l - 1$  is  $x_1$  and whose last  $l - 1$  characters is  $x_2$ . Once we have the Euler path we simply read off the edges and add a new character for each one (except the first, where we take the whole edge).

## 4 Examples

Examples of both of these algorithms being applied to SBH can be found in the book.