Genome Reconstruction: A Puzzle with a Billion Pieces
Phillip E. C. Compeau and Pavel A. Pevzner
Outline

I. Problem
II. Two Historical Detours
III. Example
IV. The Mathematics of DNA Sequencing
V. Complications
Problem: Given DNA, how do we find the nucleotide sequence?

• Reduces to two problems:
  1. Read generation (Biological)
  2. Fragment Assembly (Algorithmic/Mathematical)
Introduction to DNA Sequencing

• Four Nucleotides: A, G, C, T
• No known way to read DNA one nucleotide at a time
• Current technology can only 'read' short segments of DNA
  - At most approximately 100 nucleotides in length
  - Short fragments of length k are called k-mers
• Biologists generate these k-mers starting at every nucleotide
• Then use mathematics to attempt to recover the sequence by solving a giant overlap puzzle
Brief Introduction to Read Generation

- First synthesize all possible 3-mers
- Attach these to a grid on which each 1-mer is assigned a unique location
- Take the DNA fragment and fluorescently label it
- Apply this to the DNA array
- Read the complements of fluorescent grids

<table>
<thead>
<tr>
<th>AAA</th>
<th>AGA</th>
<th>CAA</th>
<th>CGA</th>
<th>GAA</th>
<th>GGA</th>
<th>TAA</th>
<th>TGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAC</td>
<td>AGC</td>
<td>CAC</td>
<td>CGC</td>
<td>GAC</td>
<td>GGC</td>
<td>TAC</td>
<td>TGC</td>
</tr>
<tr>
<td>AAG</td>
<td>AGG</td>
<td>CAG</td>
<td>CGG</td>
<td>GAG</td>
<td>GGG</td>
<td>TAG</td>
<td>TGG</td>
</tr>
<tr>
<td>AAT</td>
<td>AGT</td>
<td>CAT</td>
<td>CGT</td>
<td>GAT</td>
<td>GGT</td>
<td>TAT</td>
<td>TGT</td>
</tr>
<tr>
<td>ACA</td>
<td>ATA</td>
<td>CCA</td>
<td>CTA</td>
<td>GCA</td>
<td>GTA</td>
<td>TCA</td>
<td>TTA</td>
</tr>
<tr>
<td>ACC</td>
<td>ATC</td>
<td>CCC</td>
<td>CTC</td>
<td>GCC</td>
<td>GTC</td>
<td>TCC</td>
<td>TTC</td>
</tr>
<tr>
<td>ACG</td>
<td>ATG</td>
<td>CCG</td>
<td>CTG</td>
<td>GCG</td>
<td>GTG</td>
<td>TCG</td>
<td>TTG</td>
</tr>
<tr>
<td>ACT</td>
<td>ATT</td>
<td>CCT</td>
<td>CTT</td>
<td>GCT</td>
<td>GTT</td>
<td>TCT</td>
<td>TTT</td>
</tr>
</tbody>
</table>
a) Map of Konigsberg.  b) The graph formed by compressing each land mass into a vertex and representing each bridge by an edge.

Konigsberg Bridge Problem

• Problem: Is there a walk that traverses each bridge exactly once?

• Euler solved this problem in the 18th century and spawned the branch of mathematics known as Graph Theory.
Hamilton's Game
Simplifying assumptions:
1. The genome we are reconstructing is cyclic.
2. Every read has the same length.
3. All possible substrings of length \( l \) occurring in our genome have been generated as reads.
4. The reads have been generated without any errors.
Example

• Suppose we have the sequence:
  TAATGCCATGGGATGTT

• From this sequence, we yield the 3-mers:
  TAA, AAT, ATG, TGC, GCC, CCA, CAT, ATG, TGG, GGG, GGA, GAT, ATG, TGT, GTT

• We construct a graph from these 3-mers by:
  1. Using the 3-mers as vertices.
  2. Placing a directed edge from vertex 1, \( (v1) \), to vertex 2, \( (v2) \) if the prefix of \( v2 \) is the suffix of \( v1 \).
Example

- Prefix of AAT is AA while suffix of TAA is AA, etc.
In practice, k-mers are given in lexicographic order:
AAT, ATG, ATG, ATG, CAT, CCA, GAT, GCC, GGA, GGG, GTT, TAA, TGC, TGG, TGT

We again use the 3-mers as nodes.

Now we connect two nodes from one to another if the suffix is same as prefix.

For example, we connect AAT to all ATG nodes.

We yield a new graph that looks as follows.

The goal is now to find a path in the graph that passes through every node exactly once. (Hamiltonian Problem)
Example
Building the Path
Sequence then becomes:

TAATGCCCATGGGGATGTT
Example Revisited

We now approach sequence generation in a new way.

• Start again with the sequence:
  TAATGCCCATGGGATGTT

• Generate 3-mers again:
  TAA, AAT, ATG, TGC, GCC, CCA, CAT, ATG, TGG, GGG, GGA, GAT, ATG, TGT, GTT

• The 3-mers now become the *edges* while the prefixes and suffixes become the *nodes*. 
Example Revisited

- **TA** is the prefix of a 3-mer with **AA** as the suffix, so it is connected by an edge labeled by the 3-mer **TAT**, etc.
- The next step is to paste together *nodes* that are the same.
Eulerian Problem

• The goal now is to find a path through the graph that passes through every edge exactly once. (Eulerian Problem)

• When this path is found, concatenate the edges to retrieve the sequence.
When we read the *edges* back, we recover the sequence:

**TAATGCCATGGGATGTT**
The Million Dollar Question

Is the Hamiltonian Problem or the Eulerian Problem easier to solve?
Million Dollar Question

• Turns out that the Hamiltonian Problem is intractable
  - NP-complete
  - You can literally win a million dollars by solving it
  - Hamiltonian strategy still used to sequence the Human Genome and others before 2001

• Eulerian Problem is very easy to solve
  - Proof of Euler's Theorem gives you a very nice algorithm to find the cycle
Euler's Theorem

Theorem: A directed, connected, and finite graph $G$ has an Eulerian cycle if and only if, for every vertex $v$ in $G$, the indegree and the outdegree of $v$ are equal.
Complications

• Eulerian Cycle found might not be unique
  – In our example there is also a cycle that generates the sequence:
    TAATGGGATGCCATGTT

• How does the problem change when the sequence is not cyclic, but rather, a linear DNA sequence?

• How do we adjust for errors in the read generation?
Thank You for Listening. Any Questions?