How Do We Compare Biological Sequences?

Assembling Genomes

© 2022 Phillip Compeau
Eternity II: The Highest-Stakes Puzzle in History
AN INTRODUCTION TO
GENOME SEQUENCING

© 2022 Phillip Compeau
The Newspaper Problem

stack of NY Times, June 27, 2000

© 2022 Phillip Compeau
The Newspaper Problem

stack of NY Times, June 27, 2000

stack of NY Times, June 27, 2000
on a pile of dynamite

© 2022 Phillip Compeau
The Newspaper Problem

- Stack of NY Times, June 27, 2000
- Stack of NY Times, June 27, 2000 on a pile of dynamite
- This is just hypothetical
The Newspaper Problem

- Stack of NY Times, June 27, 2000
- Stack of NY Times, June 27, 2000 on a pile of dynamite
- This is just hypothetical

© 2022 Phillip Compeau
The Newspaper Problem

stack of NY Times, June 27, 2000

stack of NY Times, June 27, 2000 on a pile of dynamite

this is just hypothetical

BOOM

© 2022 Phillip Compeau
The Newspaper Problem

stack of NY Times, June 27, 2000

stack of NY Times, June 27, 2000 on a pile of dynamite

this is just hypothetical

BOOM

so, what did the June 27, 2000 NY Times say?
The Newspaper Problem

**STOP:** How would you reconstruct the news?

© 2022 Phillip Compeau
The Newspaper Problem

The Newspaper Problem is an overlap puzzle.

stack of NY Times, June 27, 2000

stack of NY Times, June 27, 2000 on a pile of dynamite

this is just hypothetical

BOOM

so, what did the June 27, 2000 NY Times say?

The Newspaper Problem is an overlap puzzle.
The Newspaper Problem

stack of NY Times, June 27, 2000

stack of NY Times, June 27, 2000 on a pile of dynamite

this is just hypothetical

BOOM

so, what did the June 27, 2000 NY Times say?

But what does this have to do with biology?
DNA is a Double Helix of Nucleotide Strands

DNA’s Double Helix (1953)

DNA’s Molecular Structure
Courtesy: Madprime, Wikimedia Commons

© 2022 Phillip Compeau
The Order of Nucleotides Determines Genetics

**Nucleotide:** Half of one “rung” of DNA.

Four choices for the nucleic acid of a nucleotide:

1. Adenine (A)
2. Cytosine (C)
3. Guanine (G)—bonds to C
4. Thymine (T)—bonds to A

Courtesy: Madprime, Wikimedia Commons
**Nucleotide:** Half of one “rung” of DNA.

**Key point:** if we know one strand of DNA, we get the other strand for free because of this “complementarity”.

---

*DNA’s Molecular Structure*

Courtesy: Madprime, Wikimedia Commons

© 2022 Phillip Compeau
Genome “Sequencing” Means “Reading” the Genome

**Genome**: The nucleotide sequence read down one side of an organism’s chromosomal DNA. A human genome has about 3 billion letters.

...CCGTAGTCGCATGGAACAGTATACGAGACAGTACAGATACGATACGATACGATACGATCATTAACCGAGAGTACCAGATTCCAGATCATACGTTACGCTTAGCTACGACGTACGATACCCAGATTACGATCCATATGATATAACCGGTGTGTCTTGCTAATACGTAACGGGGTGCTCGATAGGTCAGAATACCAGATCTCTCGATCTTCTTACAGATACTACGATCCCCAGATACTACCCCTACTGACCCATCGTACGGGTATCTACTACGGATATGATACCGATGTAGAGGGATCCATATATCCCGAGACGTCTCGCGCATAAAGATCATCGTCTAGATACACGTACGTAATAGACTAGCGTATGCTCTTATGATCGTCCCGATCGAGTCGCGTGCTCAGAAAAGCTACGATACGATACCCGATACTAGACCATAG...
Genome “Sequencing” Means “Reading” the Genome

**Genome:** The nucleotide sequence read down one side of an organism’s chromosomal DNA. A human genome has about 3 billion letters.

...CCGTAGTCGCATGAACAGTATACGAGACAGTACAGATACGATACGATACGATACGATATTAACCGAGAGTACCAGATTCCAGATCATACGTTACGCTTAGCTACGGACGTACGATACCCAGATTACGATCCATATAGATATAACCGGTGTCTTGCTAATACGTAACGGGGTGCTCTCGATAGGTCAGAATACCAGATCTCTCGATCTTCTTACAGATACTACGATCCCCAGATACTACCCCTACTGACCCATCGTACGGGTACTACTACGGATATGCCAGATGAGAGGGATCCATATATCCCGAGACGTCTCGCGCATAAAGATCATCGTCTAGATACACGTACGTACTAGACTAGCTAGCTATGCCTCTCTGATCTTCTCTAAGATACTACGATCCCCAGATACTACCCCTACTGACCCCATCGTACGGGTACTACCCGGATATGCCTATCGATCCCTGCCGAGATCGTGCCTCAGAAAAGCTACGATACGATACCCGATACTAGACCATAG...

*Polychaos dubium* (an amoeba) has one of the longest known genomes: 670 billion nucleotides.
Genome “Sequencing” Means “Reading” the Genome

**Genome:** The nucleotide sequence read down one side of an organism’s chromosomal DNA. A human genome has about 3 billion letters.

...CCGTAGTCGCATGGAACAGTATACGAGACAGTACAGATACGATACGATACGATACGATATTAACCAGAGATTCCAGATCATTACG CATCCAGGATACCGATATCCCATATAGGATATACAACCGGTGTGTCTTGCTAATACGTAACGGGGTGCTCTGACTACGATACGGGT CATGGTCTCTCCGTACACGGTCTTCTTACAGATCCTACTACGATCCCCAGATACCTACCTACCTACGACTACTAGCTACTACGGATATGATACCGATGTAGAGGGATCCATATATCCCGAGACGTCTCGCGCATAAAGATCATCGTCTAGATACACGTACGTA CTAGACTAGCGTATGCTTATTATGATCGTCCCGATCGAGTCGCGTGCTCAGAAAAGCTACGATACGATACCGATACTAGACCATAG...

**Key Point:** DNA is submicroscopic! How do we read something that we cannot see?
We Sequence a Species’s Genome to Unlock its Genetic Identity

Darwin’s notebook c. 1837

Hug et al., 2016
Nature Biotechnology, Discovery Magazine
We Sequence an Individual’s Genome to Find What Makes them Unique

2011: First person whose life was saved because of genome sequencing.
Ten years later, genome sequencing saves a life in 13 hours

Source: Owen et al. 2021
Late 1970s: Walter Gilbert and Frederick Sanger develop independent sequencing methods.

Bacterial phage PhiX174 genome (5,386 nucleotides)
History of Genome Sequencing

**Late 1970s:** Walter Gilbert and Frederick Sanger develop independent sequencing methods.

**1980:** They share the Nobel Prize in Chemistry.
History of Genome Sequencing

**Late 1970s**: Walter Gilbert and Frederick Sanger develop independent sequencing methods.

**1980**: They share the Nobel Prize in Chemistry.

However, their approaches cost about $1 per nucleotide.
The Race to Sequence the Human Genome

1990: Human Genome Project given $3 billion to sequence human genome.

James Watson
The Race to Sequence the Human Genome

1990: Human Genome Project given $3 billion to sequence human genome.

1992: James Watson resigns, replaced by Francis Collins.
The Race to Sequence the Human Genome

1990: Human Genome Project given $3 billion to sequence human genome.

1992: James Watson resigns, replaced by Francis Collins.

1997: Craig Venter founds Celera Genomics with same goal.
The Race to Sequence the Human Genome

2000: First draft of human genome published.
Early 2000s: Many more mammalian genomes are sequenced using Sanger’s approach.
From One Mammal Genome to Many

**Problem:** This approach was just too expensive to scale to thousands of species.
Sequencing Cost Has Fallen Faster than Moore’s Law

Cost per Raw Megabase of DNA Sequence

Moore’s Law

NIH National Human Genome Research Institute

genome.gov/sequencingcosts

© 2022 Phillip Compeau
GISAID collects 400k–2 Million SARS-CoV-2 Genomes in One Year Two Years
Scientists aim to sequence 1.5M eukaryotes before 2030
We Now Have Over 2 Million Human Genomes

100,000 Genomes: Sequenced 100,000 UK resident genomes (2012-2018).
Overview of Genome Sequencing

Multiple identical copies of a genome
Overview of Genome Sequencing

Multiple identical copies of a genome

Shatter the genome into reads
Overview of Genome Sequencing

Multiple identical copies of a genome

Shatter the genome into reads

Sequence the reads

(Lab)

AGAATATCA

TGAGAATAT

GAGAATATC

© 2022 Phillip Compeau
Overview of Genome Sequencing

Multiple identical copies of a genome

Shatter the genome into reads

Sequence the reads (Lab)

Assemble the genome using overlapping reads (Computational)
Overview of Genome Sequencing

Multiple identical copies of a genome

Shatter the genome into reads

Sequence the reads

( Lab )

Assemble the genome using overlapping reads

( Computational )

What does genome sequencing remind you of?

AGAATATCA
TGAGAATAT
GAGAATATC

AGAATATCA
GAGAATATC
TGAGAATAT

... TGAGAATATCA ...
Genome Assembly = Overlap Puzzle
Interlude: How Are Reads Sequenced?

https://www.youtube.com/watch?v=fCd6B5HRaZ8
A COMPUTATIONAL PROBLEM FOR GENOME ASSEMBLY
Practical Sequencing Complications

1. DNA may be divided over multiple chromosomes.

2. Reads have imperfect “coverage” of the underlying genome – there may be some regions that are not covered by any reads.

3. Sequencing machines are error-prone.

4. DNA is double-stranded.
Making Some Assumptions is OK!

1. A genome consists of a *single chromosome*.

2. Reads have *perfect “coverage”* of the underlying genome – every possible starting position gets sampled by the sequencer.

3. Sequencing machines are *error-free*.

4. DNA is *single-stranded*. 
Formulating a Computational Problem for Genome Assembly

Genome Assembly Problem
- **Input:** A collection of strings *Reads*.
- **Output:** A string *Genome* reconstructed from *Reads*.
Formulating a Computational Problem for Genome Assembly

**Genome Assembly Problem**
- **Input:** A collection of strings *Reads*.
- **Output:** A string *Genome* reconstructed from *Reads*.

**STOP:** Is this a well-defined problem?
Formulating a Computational Problem for Genome Assembly

**Genome Assembly Problem**
- **Input:** A collection of strings *Reads*.
- **Output:** A string *Genome* reconstructed from *Reads*.

**STOP:** Is this a well-defined problem?

**Answer:** No! We have no sense of what it means to “reconstruct” a genome.
The *k-mer composition* of a string *Text*, denoted \( \text{Composition}_k(\text{Text}) \), is the collection of all *k*-mer substrings of *Text* (including repeats).
We want to solve the reverse problem: given a collection of strings, find a string having this collection as its $k$-mer composition.

**String Reconstruction Problem**

- **Input:** A collection of strings $patterns$ and an integer $k$.
- **Output:** A string $Text$ whose $k$-mer composition is equal to $Patterns$. 
Toward a Computational Problem

STOP: Now is this a well-defined computational problem?

String Reconstruction Problem

• **Input:** A collection of strings *patterns* and an integer *k*.

• **Output:** A string *Text* whose *k*-mer composition is equal to *Patterns*. 
Toward a Computational Problem

**STOP:** Now is this a well-defined computational problem?

**Answer:** Not quite ... what if Patterns = \{AAA, ZZZ\}?
Toward a Computational Problem

STOP: Now is this a well-defined computational problem?

Answer: Not quite ... what if Patterns = {AAA, ZZZ}?

String Reconstruction Problem

• **Input:** A collection of strings patterns and an integer \( k \).
• **Output:** A string Text whose \( k \)-mer composition is equal to Patterns (if such a string exists).
SOLVING THE STRING RECONSTRUCTION PROBLEM?
Exercise: Reconstruct the string corresponding to the following 3-mer composition.

AAT  ATG  GTT  TAA  TGT
Exercise: Reconstruct the string corresponding to the following 3-mer composition.

```
AAT   ATG   GTT   TAA   TGT
TAA
 AAT
  ATG
   TGT
    GTT
     TAATGTT
```
Toward an Algorithm for Genome Assembly

"Greedy" algorithm: for each $k$-mer, look for the $k$-mer of maximum overlap in each direction.

TAA
AAT
ATG
TGT
GTT
TAATGTT
Toward an Algorithm for Genome Assembly

"Greedy" algorithm: for each $k$-mer, look for the $k$-mer of maximum overlap in each direction.

Genome assembly is trivial! We can pack up and go home.
"Greedy" algorithm: for each $k$-mer, look for the $k$-mer of maximum overlap in each direction.

Genome assembly is trivial! We can pack up and go home.

Exercise: Apply this algorithm to the 3-mer composition at right.
Toward an Algorithm for Genome Assembly

AAT
ATG
ATG
ATG
CAT
CCA
GAT
GCC
GGA
GGG
GTT
TAA
TGC
TGG
TGT

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

TAA

AAT
ATG
ATG
ATG
CAT
CCA
GAT
GCC
GGA
GGG
GTT
TAA
TGC
TGG
TGT

© 2022 Phillip Compeau
<table>
<thead>
<tr>
<th>TAA</th>
<th>AAT</th>
<th>AAT</th>
<th>ATG</th>
<th>ATG</th>
<th>ATG</th>
<th>CAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AAT</td>
<td>ATG</td>
<td>ATG</td>
<td>ATG</td>
<td>CCA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GAT</td>
<td>GCC</td>
<td>GGA</td>
<td>GGG</td>
</tr>
<tr>
<td>TAAAT</td>
<td></td>
<td></td>
<td></td>
<td>GGT</td>
<td>TAA</td>
<td>TGC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TGG</td>
<td>TGG</td>
<td>TGT</td>
</tr>
</tbody>
</table>

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

STOP: Which one should we choose?
Toward an Algorithm for Genome Assembly

TAA
AAT
ATG
TGC

TAATGC

AAT
ATG
CAT
CCA
GAT
GCC
GGA
GGG
GTT
TAA
TGC
TGG
TGT

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

TAA
AAT
ATG
TGC
GCC

TAATGCC

AAT
ATG
ATG
ATG
CAT
CCA
GAT
GCC
GGA
GGG
GTT
TAA
TGC
TGG
TGG
TGT

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

TAA
AAT
ATG
TGC
GCC
CCA

AAT
ATG
ATG
ATG
CAT
CCA
GAT
GCC
GGA
GGG
GTT
TAA
TGC
TGG
TGT

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

TAA
AAT
ATG
TGC
GCC
CCA
CAT

TAATGCCAT

AAT
ATG
ATG
ATG
CAT
CCA
GAT
GCC
GGA
GGG
GTT
TAA
TGC
TGG
TGT

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG

AAT
ATG
ATG
ATG
CAT
CCA
GAT
GCC
GGA
GGG
GTT
TAA
TGC
TGG
TGT

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

TAATGCCCATGG

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGA
GGG
TAA
TGC
TGG
TGT

TAATGCCATGGA
Toward an Algorithm for Genome Assembly

TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGA
GAT

TAATGCCCATGGAT

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGA
GAT
ATG

TAATGCCATGGATG

AAT
ATG
ATG
ATG
CAT
CCA
GAT
GCC
GGA
GGG
GTT
TAA
TGC
TGG
TGT

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

TAA
AAT
ATG
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGA
GAT
ATG
TGT
TAATGCCATGGATGT

© 2022 Phillip Compeau
Toward an Algorithm for Genome Assembly

TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGA
GAT
ATG
TGT
GTT
TAATGCCATGGATGTT

© 2022 Phillip Compeau
STOP: Why did our algorithm fail?
Toward an Algorithm for Genome Assembly

Answer: Repeated substrings!
Repeats Make Eternity II Unsolvable ...
... Even a 16-piece “Triazzle” Can Take a Human Hours to Solve...

Courtesy: Dan Gilbert

© 2022 Phillip Compeau
Repeats are very common in genomes; the 300-nucleotide **Alu repeat** occurs over a million times (with minor changes) in every human genome.
Repeats are very common in genomes; the 300-nucleotide **Alu repeat** occurs over a million times (with minor changes) in every human genome.

So what hope do we have of assembling a genome?
GENOME ASSEMBLY AS A HAMILTONIAN PATH PROBLEM
Solution to Previous Exercise

**STOP:** Is this the only solution?
We Can View a Genome as a “Path” in a Graph

**Genome path:** assign each read to a node, connect adjacent reads with edges.
We Can View a Genome as a “Path” in a Graph

**Genome path:** assign each read to a node, connect adjacent reads with edges.

**STOP:** Can you still see the genome?
We Can View a Genome as a “Path” in a Graph

**Genome path:** assign each read to a node, connect adjacent reads with edges.

**STOP:** Can you still see the genome?

**Answer:** No ... we need to know the order of the $k$-mers.
We Can View a Genome as a “Path” in a Graph

**Genome path:** assign each read to a node, connect adjacent reads with edges.

**STOP:** Can you still see the genome?

**STOP:** Could you construct the genome path if you only knew the 3-mer composition?
A Graph Can Represent All Overlapping Strings

- **Prefix:** First $k - 1$ letters in a $k$-mer.
- **Suffix:** Last $k - 1$ letters in a $k$-mer.

© 2022 Phillip Compeau
A Graph Can Represent All Overlapping Strings

- **Prefix**: First $k - 1$ letters in a $k$-mer.
- **Suffix**: Last $k - 1$ letters in a $k$-mer.

**Overlap Graph**: Form a node for each read in *Patterns*, then connect $x$ to $y$ if $\text{Suffix}(x) = \text{Prefix}(y)$.
A Graph Can Represent All Overlapping Strings

- **Prefix**: First $k - 1$ letters in a $k$-mer.
- **Suffix**: Last $k - 1$ letters in a $k$-mer.

**Overlap Graph**: Form a node for each read in *Patterns*, then connect $x$ to $y$ if $\text{Suffix}(x) = \text{Prefix}(y)$. 
A Graph Can Represent All Overlapping Strings

• **Prefix:** First $k - 1$ letters in a $k$-mer.
• **Suffix:** Last $k - 1$ letters in a $k$-mer.

**Overlap Graph:** Form a node for each read in *Patterns*, then connect $x$ to $y$ if $\text{Suffix}(x) = \text{Prefix}(y)$.
A Graph Can Represent All Overlapping Strings

- **Prefix:** First \( k - 1 \) letters in a \( k \)-mer.
- **Suffix:** Last \( k - 1 \) letters in a \( k \)-mer.

**Overlap Graph:** Form a node for each read in *Patterns*, then connect \( x \) to \( y \) if \( \text{Suffix}(x) = \text{Prefix}(y) \).
A Graph Can Represent All Overlapping Strings

- **Prefix**: First $k - 1$ letters in a $k$-mer.
- **Suffix**: Last $k - 1$ letters in a $k$-mer.

**Overlap Graph**: Form a node for each read in *Patterns*, then connect $x$ to $y$ if $\text{Suffix}(x) = \text{Prefix}(y)$. 
A Graph Can Represent All Overlapping Strings

- **Prefix**: First $k - 1$ letters in a $k$-mer.
- **Suffix**: Last $k - 1$ letters in a $k$-mer.

**Overlap Graph**: Form a node for each read in *Patterns*, then connect $x$ to $y$ if $\text{Suffix}(x) = \text{Prefix}(y)$. 
A Graph Can Represent All Overlapping Strings

- **Prefix:** First $k - 1$ letters in a $k$-mer.
- **Suffix:** Last $k - 1$ letters in a $k$-mer.

**Overlap Graph:** Form a node for each read in *Patterns*, then connect $x$ to $y$ if $\text{Suffix}(x) = \text{Prefix}(y)$. 
A Graph Can Represent All Overlapping Strings

- **Prefix**: First $k - 1$ letters in a $k$-mer.
- **Suffix**: Last $k - 1$ letters in a $k$-mer.

**Overlap Graph**: Form a node for each read in *Patterns*, then connect $x$ to $y$ if $\text{Suffix}(x) = \text{Prefix}(y)$. 

© 2022 Phillip Compeau
A Graph Can Represent All Overlapping Strings

- **Prefix**: First $k - 1$ letters in a $k$-mer.
- **Suffix**: Last $k - 1$ letters in a $k$-mer.

**Overlap Graph**: Form a node for each read in Patterns, then connect $x$ to $y$ if $\text{Suffix}(x) = \text{Prefix}(y)$. 

© 2022 Phillip Compeau
A Graph Can Represent All Overlapping Strings

**Note:** we can still see the genome path, but we wouldn’t if we don’t know the order of $k$-mers …

**Overlap Graph:** Form a node for each read in *Patterns*, then connect $x$ to $y$ if $\text{Suffix}(x) = \text{Prefix}(y)$. 
Arranging $k$-mers Lexicographically Makes Genome Vanish

© 2022 Phillip Compeau
Arranging \( k \)-mers Lexicographically Makes Genome Vanish

STOP: If we gave you this graph, what would you look for to find the genome?
We are Looking for a Hamiltonian Path in the Overlap Graph

**Hamiltonian path:** A path through a graph that touches each node exactly once.
STOP: What genome does the highlighted path reconstruct?
And Here’s Another Solution

STOP: How about this highlighted path?
We are Looking for a Hamiltonian Path in the Overlap Graph

**Note:** The graph organizes our reads, but we don’t have an *algorithm* for finding a Hamiltonian path.
We are looking for a Hamiltonian Path in the Overlap Graph

STOP: What does the overlap graph look like if there are many repeats? What if there are none?
A binary string is \textit{k-universal} if it contains every binary \textit{k}-mer once.

**Exercise:** Find a 3-universal string.
A binary string is *\(k\text{-universal}\) if it contains every binary \(k\)-mer once.

**Note:** a \(k\)-universal string corresponds to a Hamiltonian path in the following overlap graph.
1946: Good and de Bruijn independently discover a way to find $k$-universal strings. They cannot imagine that their approach will one day power genome sequencing.
Aside 2: Two Ways to Represent Graphs Computationally

**Adjacency Matrix**

Adjacency matrix: \( A_{i,j} = 1 \) if there is an edge connecting node \( i \) to node \( j \); \( A_{i,j} = 0 \) otherwise.
Aside 2: Two Ways to Represent Graphs Computationally

Adjacency matrix: $A_{i,j} = 1$ if there is an edge connecting node $i$ to node $j$; $A_{i,j} = 0$ otherwise.

Adjacency list: Dictionary; “key” node $i$; “value” is list of nodes that $i$ is connected to.
GENOME ASSEMBLY AS AN EULERIAN PATH PROBLEM
Assigning $k$-mers to *Edges* Instead of *Nodes*

We start again with a “genome path” corresponding to TAATGCCCATGGGGATGTT.
Assigning $k$-mers to *Edges* Instead of *Nodes*

We start again with a “genome path” corresponding to $\text{TAAATGCCATGGGATGTT}$.

**STOP:** How should we label the nodes?
Assigning $k$-mers to *Edges* Instead of *Nodes*

Each node represents the $(k - 1)$-mer corresponding to the overlap between adjacent edges.
Assigning \( k \)-mers to *Edges* Instead of *Nodes*

Each node represents the \((k - 1)\)-mer corresponding to the _overlap_ between adjacent edges.

Unlike with the overlap graph, we will _glue_ together nodes that have the same label.
First: Gluing AT Together
Next: Gluing TG Together
Gluing GG Produces a “Loop”
Gluing GG Produces a “Loop”

This graph is called the **de Bruijn graph** of $Text = TAATGCCCATGGGATGT$ for $k = 3$. 

© 2022 Phillip Compeau
This graph is called the **de Bruijn graph** of $\text{Text} = \text{TAATGCCCATGGGATGTT}$ for $k = 3$.

**Exercise:** Construct the de Bruijn graphs for $k = 4$ and $k = 5$. How do they differ from $k = 3$?
de Bruijn Graph Becomes Less "Tangled" as $k$ Increases (fewer repeats)

$k = 3$

$k = 4$

$k = 5$

© 2022 Phillip Compeau
Gluing GG Produces a “Loop”

This graph is called the de Bruijn graph of $Text = TAATGCCATGGGATGT$ for $k = 3$.

STOP: If we gave you this graph, could you reconstruct $Text$? How?
The genome path is an **Eulerian path** in the de Bruijn graph, or a path that uses every edge exactly once.
The genome path is an **Eulerian path** in the de Bruijn graph, or a path that uses every edge exactly once.

STOP: Can you construct the de Bruijn graph if you don’t already know the text?
Forming de Bruijn Graph from \( k \)-mers

Exercise: Here are the 3-mers from our original dataset represented as isolated edges. By gluing nodes together, what do you obtain?
Forming de Bruijn Graph from $k$-mers
It’s the Same Graph...
Approach for Constructing de Bruijn Graph

1. Form a node for every \((k - 1)\)-mer appearing as a prefix/suffix in \(Patterns\).
2. For every string in \(Patterns\), connect its prefix to its suffix.
1. Form a node for every \((k – 1)\)-mer appearing as a prefix/suffix in \(Patterns\).
2. For every string in \(Patterns\), connect its prefix to its suffix.

STOP: Verify this approach for \(Patterns = \{\text{AAT ATG ATG ATG CAT CCA GAT GCC GGA GGG GTT TAA TGC TGG TGT}\}\).
Which Graph Would You Rather Use?

Overlap Graph – find a Hamiltonian path

de Bruijn Graph – find an Eulerian path
THE ICOSIAN GAME AND THE BRIDGES OF KONIGSBERG
The Origin of “Hamiltonian” Path/Cycle

**Hamiltonian cycle:** A Hamiltonian path that returns to its starting node.

**Exercise:** Can you find a Hamiltonian cycle in this graph? (What algorithm did you use?)
The Origin of “Hamiltonian” Path/Cycle

**Icosian game:** William *Hamilton*, 1857. Objective is to place pegs 1-20 one at a time in adjacent holes.
STOP: Is it possible to walk across each bridge exactly once and return to the starting point?
Define a graph:

- **Nodes = 4 land masses**
- **Edges = 7 bridges**
Define a graph:
• Nodes = 4 land masses
• Edges = 7 bridges
Note: The Bridges of Königsberg question has a solution when this graph has an Eulerian cycle.
STOP: Does this graph help you solve the original question?
Answer: There is no solution because some nodes have an odd degree (number of incident edges).
Even better, Euler would prove how to quickly determine whether a graph has an Eulerian cycle.
Even better, Euler would prove how to quickly determine whether a graph has an Eulerian cycle.

**Key Point:** And yet no one has ever found a polynomial-time algorithm to find a Hamiltonian cycle in a graph!
Similar Problems with Different Fates

Hamiltonian Cycle Problem
\begin{itemize}
\item \textbf{Input:} a network with $n$ nodes.
\item \textbf{Output:} “Yes” if there is a cycle visiting every \textit{node} in the network; “No” otherwise.
\end{itemize}

Eulerian Cycle Problem
\begin{itemize}
\item \textbf{Input:} a network with $n$ nodes.
\item \textbf{Output:} “Yes” if there is a cycle visiting every \textit{edge} in the network; “No” otherwise.
\end{itemize}
FROM EULER’S THEOREM TO AN ALGORITHM FOR GENOME ASSEMBLY
Euler’s Theorem for Directed Graphs

**Indegree:** Number of edges leading into a node.

**Outdegree:** Number of edges leading out of a node.

**Balanced graph:** Every node has indegree equal to outdegree.
Euler’s Theorem for Directed Graphs

**Strongly connected graph:** A graph where it is possible to reach every node from any other node.

---

**FIGURE 3.19** Balanced (left) and unbalanced (right) directed graphs. For the (unbalanced) blue node $v$, $\text{IN}(v) = 1$ and $\text{OUT}(v) = 2$, whereas for the (unbalanced) red node $w$, $\text{IN}(w) = 2$ and $\text{OUT}(w) = 1$.

---

**FIGURE 3.20** A balanced, disconnected graph.

---

**STOP and Think:** We now know that every Eulerian graph is balanced; is every balanced graph Eulerian? The graph in Figure 3.20 is balanced but not Eulerian because it is disconnected, meaning that some nodes cannot be reached from other nodes. In any disconnected graph, it is impossible to find an Eulerian cycle. In contrast, we say that a directed graph is strongly connected if it is possible to reach any node from every other node.

---

**142**

**HOW DO WE ASSEMBLE GENOMES?**

We now know that an Eulerian graph must be both balanced and strongly connected. Euler’s Theorem states that these two conditions are sufficient to guarantee that an arbitrary graph is Eulerian. As a result, it implies that we can determine whether a graph is Eulerian without ever having to draw any cycles.

**Euler’s Theorem:** Every balanced, strongly connected directed graph is Eulerian.

**Proof.** Let $\text{Graph}$ be an arbitrary balanced and strongly connected directed graph. To prove that $\text{Graph}$ has an Eulerian cycle, place Leo at any node $v_0$ of $\text{Graph}$ (the green node in Figure 3.21), and let him randomly walk through the graph under the condition that he cannot traverse the same edge twice.

If Leo were incredibly lucky — or a genius — then he would traverse each edge exactly once and return back to $v_0$. However, odds are that he will “get stuck” somewhere before he can complete an Eulerian cycle, meaning that he reaches a node and finds no unused edges leaving that node.

---

© 2022 Phillip Compeau
Euler’s Theorem for Directed Graphs

**Strongly connected graph:** A graph where it is possible to reach every node from any other node.

**Euler’s Theorem:** Every balanced, strongly connected graph has an Eulerian cycle.
Proof of Euler’s Theorem

Take an arbitrary balanced, strongly connected network, place an ant on any starting node $v_0$, and let it walk randomly.
Proof of Euler’s Theorem

STOP: What must eventually happen when the ant “gets stuck”?

© 2022 Phillip Compeau
Proof of Euler’s Theorem

Answer: Because the graph is balanced, the ant must eventually get stuck at $v_0$!
If this cycle, which we call Cycle_0, is Eulerian, then we stop. Otherwise, move the ant to a node on Cycle_0 that still has unused edges, called ν_1.
Proof of Euler’s Theorem

Make the ant traverse all of Cycle₀ first, then explore unused edges.

STOP and Think: Where is Leo when he gets stuck? Can he get stuck in any node of the graph or only in certain nodes?

It turns out that the only node where Leo can get stuck is the starting node v₀! The reason why is that Graph is balanced: if Leo walks into any node other than v₀ (through an incoming edge), then he will always be able to escape via an unused outgoing edge. The only exception to this rule is the starting node v₀, since Leo used up one of the outgoing edges of v₀ on his first move. Now, because Leo has returned to v₀, the result of his walk was a cycle, which we call Cycle₀ (Figure 3.22(left)).

STOP and Think: Is there a way to give Leo different instructions so that he selects a longer walk through the graph before he gets stuck?

As we mentioned, if Cycle₀ is Eulerian, then we are finished. Otherwise, because Graph is strongly connected, some node on Cycle₀ must have unused edges entering it and leaving it (why?). Naming this node v₁, we ask Leo to start at v₁ instead of v₀ and traverse Cycle₀ (thus returning to v₁), as shown in Figure 3.22(right).

Leo is probably annoyed that we have asked him to travel along the exact same cycle, since as before, he will eventually return to v₁, the node where he started. However, now there are unused edges starting at this node, and so he can continue walking from v₁, using a new edge each time. The same argument as the one that we used before implies that Leo must eventually get stuck at v₁. The result of Leo’s walk is a new cycle, Cycle₁ (Figure 3.23), which is larger than Cycle₀.
The same reasoning implies that the ant will eventually get stuck at $v_1$, creating $Cycle_1$. 
Proof of Euler’s Theorem

We simply iterate this procedure until we are out of unused edges, when we have an Eulerian cycle!
Proof of Euler’s Theorem

We simply iterate this procedure until we are out of unused edges, when we have an Eulerian cycle!
STOP: Why can we be sure that this process will use all the edges?
Proof of Euler’s Theorem

Answer: Because the graph is strongly connected! So note that we have used both conditions in the theorem (balanced and strongly connected).
Exercise: When will an “undirected” graph have an Eulerian cycle?
Euler’s Theorem is “Constructive”

Key Point: This is a “constructive proof”, meaning it implies an algorithm for finding an Eulerian cycle.

EulerianCycle(Graph)

v ← arbitrary node in Graph
Cycle ← randomly walk starting at v (don’t revisit edges) until cycle while there are unexplored edges in Graph
    newStart ← node in Cycle with unexplored edges
    Cycle’ ← cycle formed by traversing Cycle (starting at newStart) and then randomly walking
    Cycle ← Cycle’
return Cycle
From Eulerian Cycles to Paths

STOP: How do we find an Eulerian path in this graph?
From Eulerian Cycles to Paths

**Answer:** Simply draw an edge connecting the two unbalanced nodes to form a balanced graph. Eulerian cycle on right = Eulerian path on left.
From Eulerian Cycles to Paths

STOP: Why will the augmented de Bruijn graph on the right be balanced for any collection of strings Patterns?

© 2022 Phillip Compeau
From Eulerian Cycles to Paths

**Answer:** For every node \( v \) in de Bruijn graph, Indegree(\( v \)) and Outdegree(\( v \)) are both equal to # of patterns containing \( v \) as prefix/suffix, respectively.
String Reconstruction Problem: Reconstruct a string from its k-mer composition.

Input: An integer $k$ and a collection $Patterns$ of $k$-mers.

Output: A string $Text$ with $k$-mer composition equal to $Patterns$ (if such a string exists).

1. Form de Bruijn graph $G$ from $Patterns$. 
**String Reconstruction Problem:** Reconstruct a string from its k-mer composition.

**Input:** An integer \( k \) and a collection \( Patterns \) of \( k \)-mers.

**Output:** A string \( Text \) with \( k \)-mer composition equal to \( Patterns \) (if such a string exists).

1. Form de Bruijn graph \( G \) from \( Patterns \).
2. Add edge to make modified graph \( G' \) balanced.
String Reconstruction Problem: Reconstruct a string from its k-mer composition.

**Input:** An integer $k$ and a collection $Patterns$ of $k$-mers.

**Output:** A string $Text$ with $k$-mer composition equal to $Patterns$ (if such a string exists).

1. Form de Bruijn graph $G$ from $Patterns$.
2. Add edge to make modified graph $G'$ balanced.
3. Find Eulerian cycle in $G'$. 
We Can Assemble a Genome!

String Reconstruction Problem: *Reconstruct a string from its k-mer composition.*

**Input:** An integer $k$ and a collection $Patterns$ of $k$-mers.

**Output:** A string $Text$ with $k$-mer composition equal to $Patterns$ (if such a string exists).

1. Form de Bruijn graph $G$ from $Patterns$.
2. Add edge to make modified graph $G'$ balanced.
3. Find Eulerian cycle in $G'$.
4. Infer Eulerian path in $G$ from this cycle.
String Reconstruction Problem: **Reconstruct a string from its k-mer composition.**

**Input:** An integer \( k \) and a collection \( \text{Patterns} \) of \( k \)-mers.

**Output:** A string \( \text{Text} \) with \( k \)-mer composition equal to \( \text{Patterns} \) (if such a string exists).

1. Form de Bruijn graph \( G \) from \( \text{Patterns} \).
2. Add edge to make modified graph \( G' \) balanced.
3. Find Eulerian cycle in \( G' \).
4. Infer Eulerian path in \( G \) from this cycle.
5. Convert “genome path” into string \( \text{Text} \).
Aside: De Bruijn/Good’s Question

Recall: a binary string is \textit{k-universal} if it contains every binary \textit{k}-mer once.

\textbf{STOP:} How can we find a \textit{k}-universal binary string?
Aside: De Bruijn/Good’s Question

**Answer:** Construct the “de Bruijn graph” for Patterns = all binary $k$-mers; find Eulerian path.
DE BRUIJN GRAPHS FACE HARSH PRACTICAL REALITIES
Practical Sequencing Complications

1. DNA may be divided over multiple chromosomes.

2. Reads have imperfect “coverage” of the underlying genome – there may be some regions that are not covered by any reads.

3. Sequencing machines are error-prone.

4. DNA is double-stranded.
Genomes May Have Multiple Chromosomes

STOP: Any ideas for assembling a genome with multiple chromosomes?
Genomes May Have Multiple Chromosomes

STOP: Any ideas for assembling a genome with multiple chromosomes?

Answer: In theory, we just find an Eulerian path in $n$ different de Bruijn graphs...
Dark Secret: The First *Full* Human Genome Wasn’t Sequenced Until 2020!
Read Coverage is Never Perfect

Draft genome assembly.
Boosting Coverage through Read Breaking

ATGCCGTATGGACACGACT
ATGCCGTATG
GCCGTATGGA
GTATGGACAA
GACAACGACT

Note that these reads don’t overlap perfectly, so building a de Bruijn graph will fail.
Boosting Coverage through Read Breaking

Read breaking: Split each read into all its \( k \)-mer substrings (for a smaller value of \( k \)).
Boosting Coverage through Read Breaking

Read breaking: Split each read into all its $k$-mer substrings (for a smaller value of $k$).
Read breaking: Split each read into all its k-mer substrings (for a smaller value of k).
Boosting Coverage through Read Breaking

**Read breaking**: Split each read into all its k-mer substrings (for a smaller value of $k$).
Boosting Coverage through Read Breaking

Read breaking: Split each read into all its $k$-mer substrings (for a smaller value of $k$).
STOP: What are the trade-offs in choosing a value of $k$?
Answer: The smaller the value of $k$, the higher our coverage will be, but also the more repeats and the more "tangled" our graph.
Assembling Contigs

Even after read breaking, most assemblies have gaps in their coverage, and we will not have a true Eulerian path in the de Bruijn graph.
Assembling Contigs

Even after read breaking, most assemblies have gaps in their coverage, and we will not have a true Eulerian path in the de Bruijn graph.

Real assembly software instead tries to infer (a small number of) **contigs**: contiguous genome segments.
A path in a graph is called **non-branching** if $\text{InDegree}(v) = \text{OutDegree}(v) = 1$ for each “intermediate” node $v$ in the path.
A path in a graph is called **non-branching** if $\text{InDegree}(v) = \text{OutDegree}(v) = 1$ for each “intermediate” node $v$ in the path.

A **maximal non-branching path** is a non-branching path that cannot made longer in either direction.
A path in a graph is called **non-branching** if $\text{InDegree}(v) = \text{OutDegree}(v) = 1$ for each “intermediate” node $v$ in the path.

A **maximal non-branching path** is a non-branching path that cannot be made longer in either direction.

**Note:** In mathematics, “maximum” means “global maximum”; “maximal” means “local maximum.”
Transforming dB Graph into Paths
STOP: Why do you think we are interested in maximal non-branching paths in genome assembly?
STOP: Why do you think we are interested in maximal non-branching paths in genome assembly?

Answer: They represent "subpaths" that must be present in any assembly, and so we can be confident in them.
STOP: Say we sequence both the correct read CGTATGGACA and the incorrect read CGTACGGACA. What will we see in the de Bruijn graph after read breaking for $k = 5$?
STOP: Say we sequence both the correct read CGTATGGACA and the incorrect read CGTACGGACA. What will we see in the de Bruijn graph after read breaking for $k = 5$?

Answer: A “bubble”!
**Bubble**: Two disjoint short path (less than some threshold length) connecting the same pair of nodes in the de Bruijn graph.
**Bubble**: Two disjoint short path (less than some threshold length) connecting the same pair of nodes in the de Bruijn graph.

**STOP**: How might we remove bubbles? What would cause your approach to go wrong?
Inexact repeat: Repeated region in genome with minor variations; the variations look just like sequencing errors!
Inexact repeat: Repeated region in genome with minor variations; the variations look just like sequencing errors!

Lower “multiplicity” paths are likely errors; this is one more benefit of higher coverage in assembly.
dB Graph of *N. meningitidis* (Bacterium) *After Removing Bubbles*

Red edges represent repeats
Pitfalls of Double-Stranded DNA

DNA is double-stranded, and the two strands are reverse complements of each other.

![Diagram of DNA strands](image)

© 2022 Phillip Compeau
Reads may come from *either strand*, so we need to consider each read’s reverse complement.
Pitfalls of Double-Stranded DNA

Note that this example is trivial if we had two de Bruijn graphs (one for the string, one for its reverse complement).
Pitfalls of Double-Stranded DNA

The reality is that we see the amalgamation of both graphs.
The reality is that we see the amalgamation of both graphs.

Even though neither string has a repeat, the graph becomes tangled because ATG and CAT are inverted repeats: the strings are reverse complements of each other.
de Bruijn Assembly in Real Research

An Eulerian path approach to DNA fragment assembly | PNAS

Our main result is the reduction of the fragment assembly to a variation of the classical Eulerian path problem that allows one to generate accurate solutions of large-scale sequencing problems. ... For the last 20 years, fragment assembly in DNA sequencing mainly followed the “overlap–layout–consensus” paradigm (1–6).

by PA Pevzner · 2001 · Cited by 1522 · Related articles

Velvet: Algorithms for de novo short read assembly ... - NCBI

by DR Zerbino · 2008 · Cited by 10047 — We have developed a new set of algorithms, collectively called “Velvet,” to manipulate de Bruijn graphs for genomic sequence assembly.

Abstract · Results · Discussion · Methods

SPAdes: A New Genome Assembly Algorithm and Its ... - NCBI

by A Bankevich · 2012 · Cited by 14402 — SPAdes generates single-cell assemblies, providing information about genomes of uncultivatable bacteria that vastly exceeds what may be obtain...

Abstract · Assembly in SPAdes: An Outline · Results · Universal Genome Assembly