CIS529: Bioinformatics

Denovo Genome Assembly

Presented by

Dr. Fayyaz-ul-Amir Afsar Minhas

http://faculty.pieas.edu.pk/fayyaz

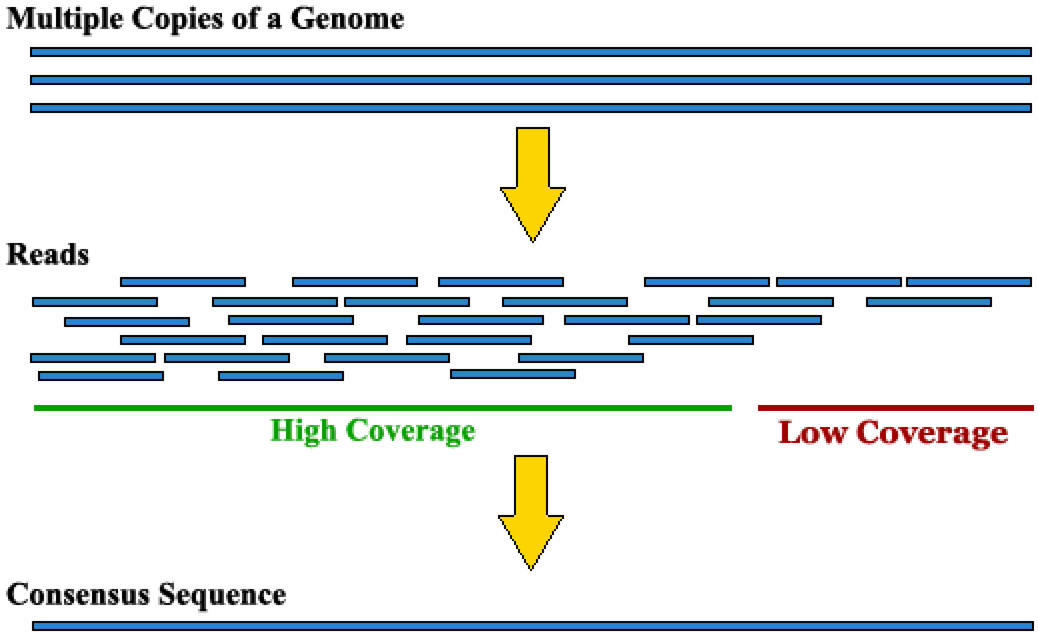
Department of Computer & Information Sciences
Pakistan Institute of Engineering & Applied Sciences
PO Nilore, Islamabad 45650
Pakistan

Review

- Exploding newspapers
- De Bruijn Graphs

The Assembly Problem

- Input: A collection of string Reads
- Output: A string Genome constructed from Reads



Assembly as a String Reconstruction Problem

- Input: A collection of k-mers
- Output: A genome such that
  - Composition_k (Genome) = collection of k-mers

- String Composition
  - Let s = TAATGCCATGGGATGTT
  - Composition_3 (s) is:
    - TAA AAT ATG TGC GCC CCA CAT ATG TGG GGG GGA GAT ATG TGT GTT
    - In reality, we don’t know the order so we can write it in lexicographic order (as in a dictionary)
    - AAT ATG ATG ATG CAT CCA GAT GCC GGA GGG GTT TAA TGC TGG TGT

Naïve Solution

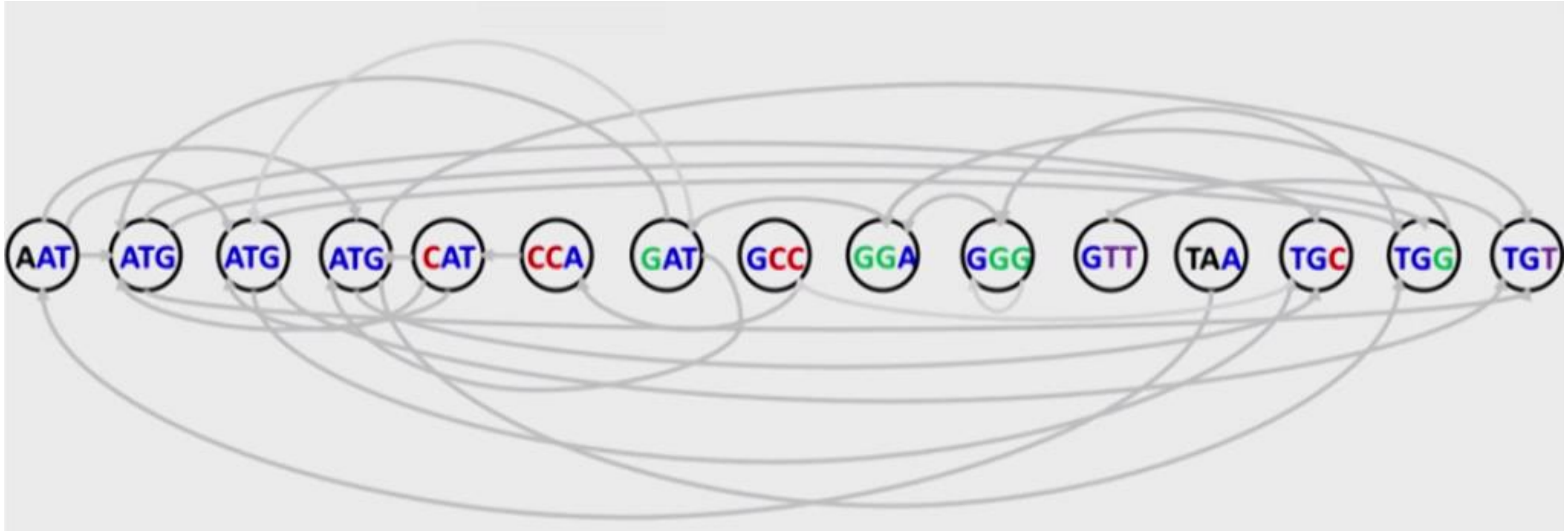
- AAT ATG ATG ATG CAT CCA GAT GCC GGA GGG GTT TAA TGC TGG TGT
- Pick a random Start and take the next k-mer such that
  - suffix(s_i) = prefix (s_{i+1})

```
TAA
 AAT
  ATG
   TGT
    GTT
```

Stuck!!

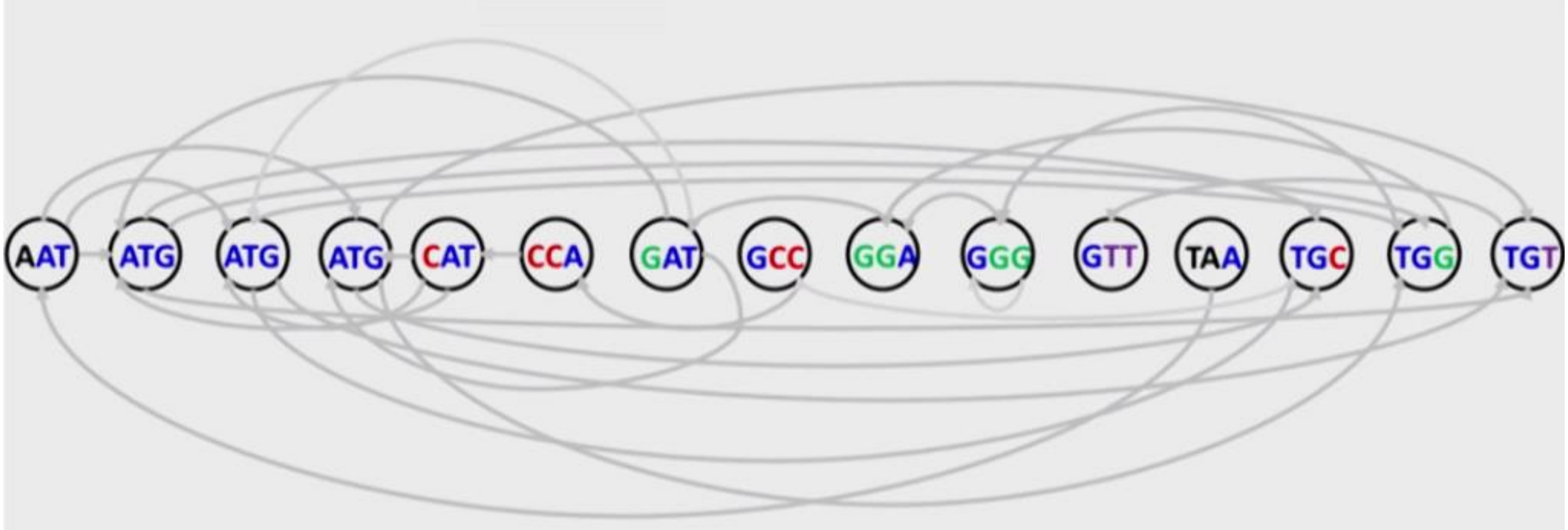
As a Hamiltonian Path Problem

- AAT ATG ATG ATG CAT CCA GAT GCC GGA GGG GTT TAA TGC TGG TGT
- Represent each k-mer as a node
- Join two nodes if suffix(s_i) = prefix (s_{i+1})
- Find the Hamiltonian Path

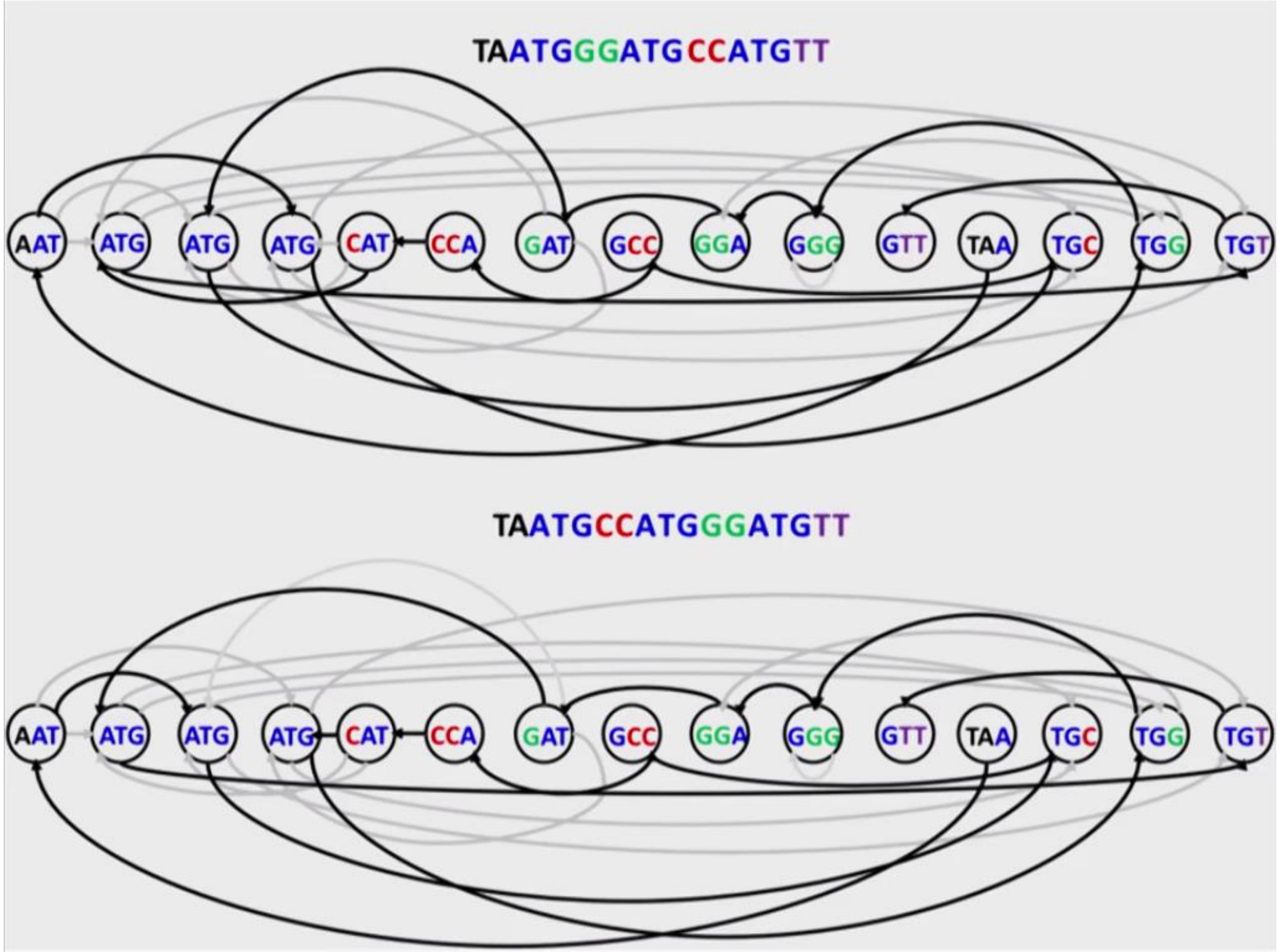


As a Hamiltonian Path Problem

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

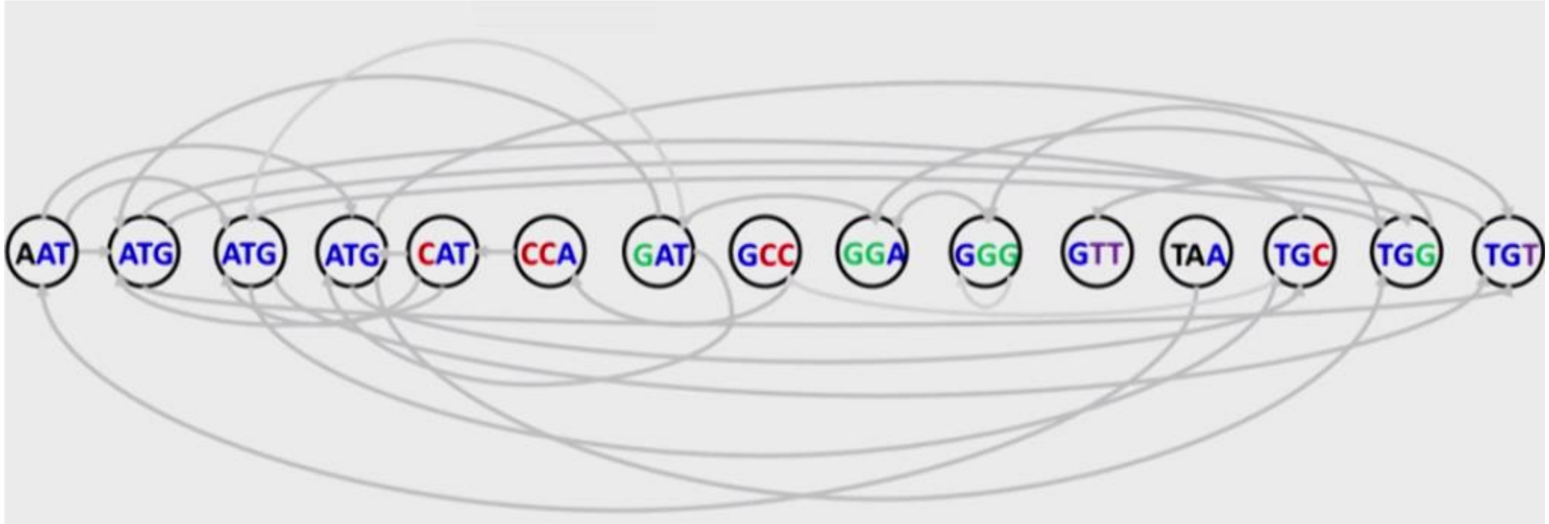


Multiple Hamiltonian paths



From De Bruijn Graphs

- Is this graph a De Bruijn Graph?



- The task here is to find a 3-universal string!

Problems with Hamiltonian Paths

- The problem with Hamiltonian Paths is that
  - There isn't an efficient algorithm known for them

- We know an efficient algorithm for finding the Eulerian cycles which can be adapted for finding Eulerian paths

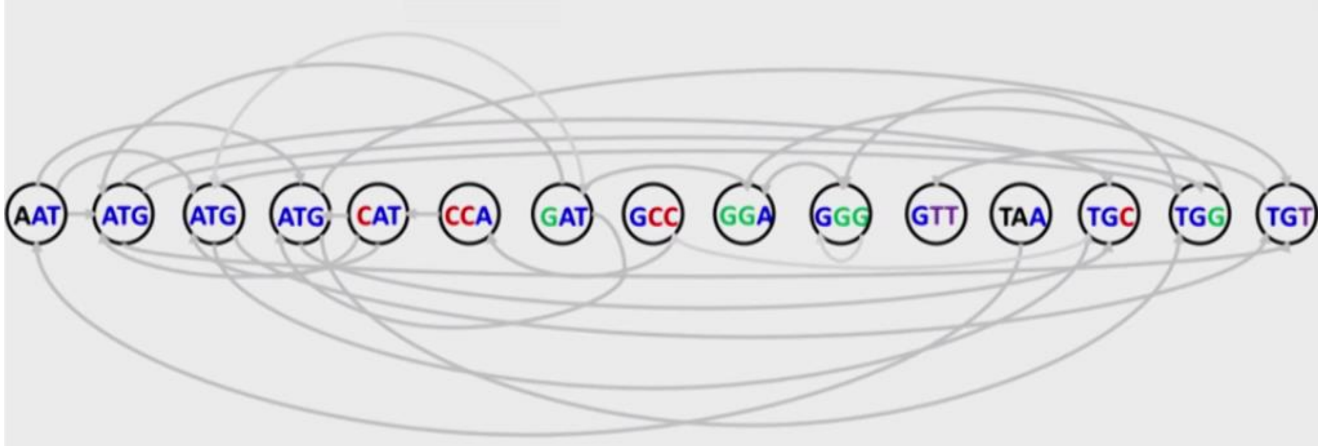
Finding Eulerian Paths

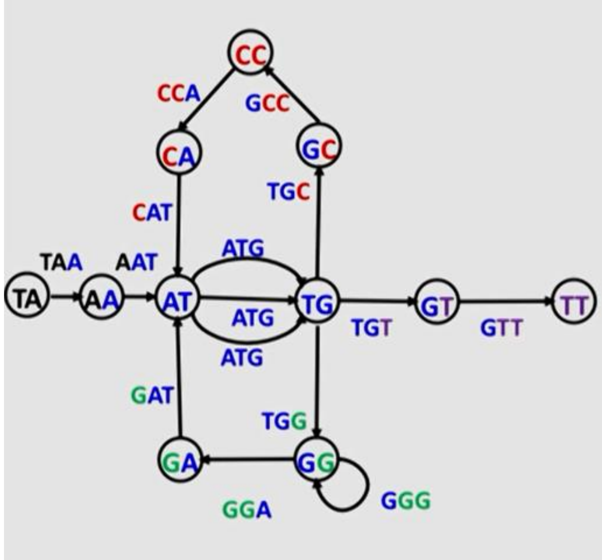
- We know that
  - H(G(m,k)) = k universal string
  - E(G(m,k)) = k+1 universal string
  - Thus to find the k universal string we are interested in, we need to find
    - E(G(m,k-1))
  - That means we need to construct G(m,k-1)
    - We know that L(G(m,k)) = G(m,k+1)
    - Thus: G(m,k-1) = L^{-1}((G(m,k))
    - How do we apply the inverse Line Operation on the graph?
      - Nodes in G(m,k) are edges of G(m,k-1)
      - If we know an edge, we can construct two nodes such that prefix of one is the suffix of the other

Example

- Let’s say we have a node ‘AAT’ in G(m,k)
- This will become an edge in G(m,k-1)
- What should be the nodes
  - AA
  - AT

G(m,k-1) from G(m,k)

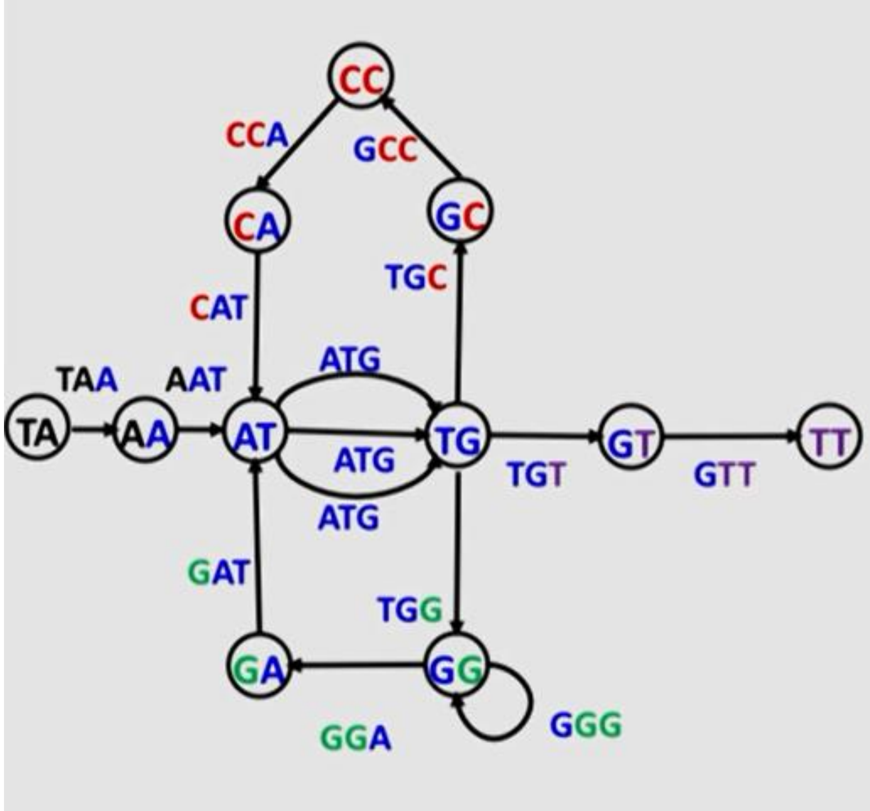




Now let’s find the Eulerian path

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

Genome: TAATGCCATGGGATGTT



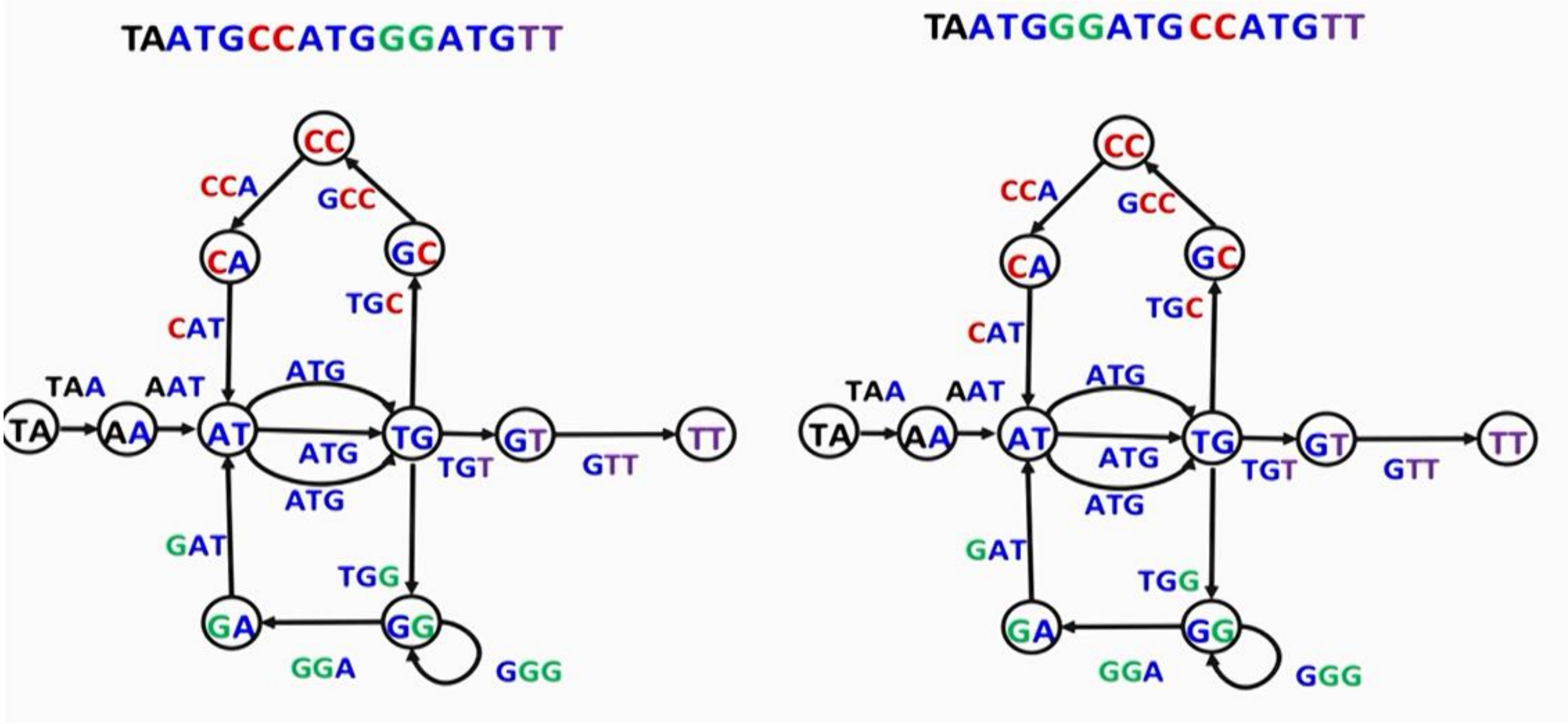
Assembly

- Find the reads (of length k)
- Construct the De Bruijn Graph G(m,k-1)
- Find an Eulerian Path
  - O(number of edges)

- DONE!

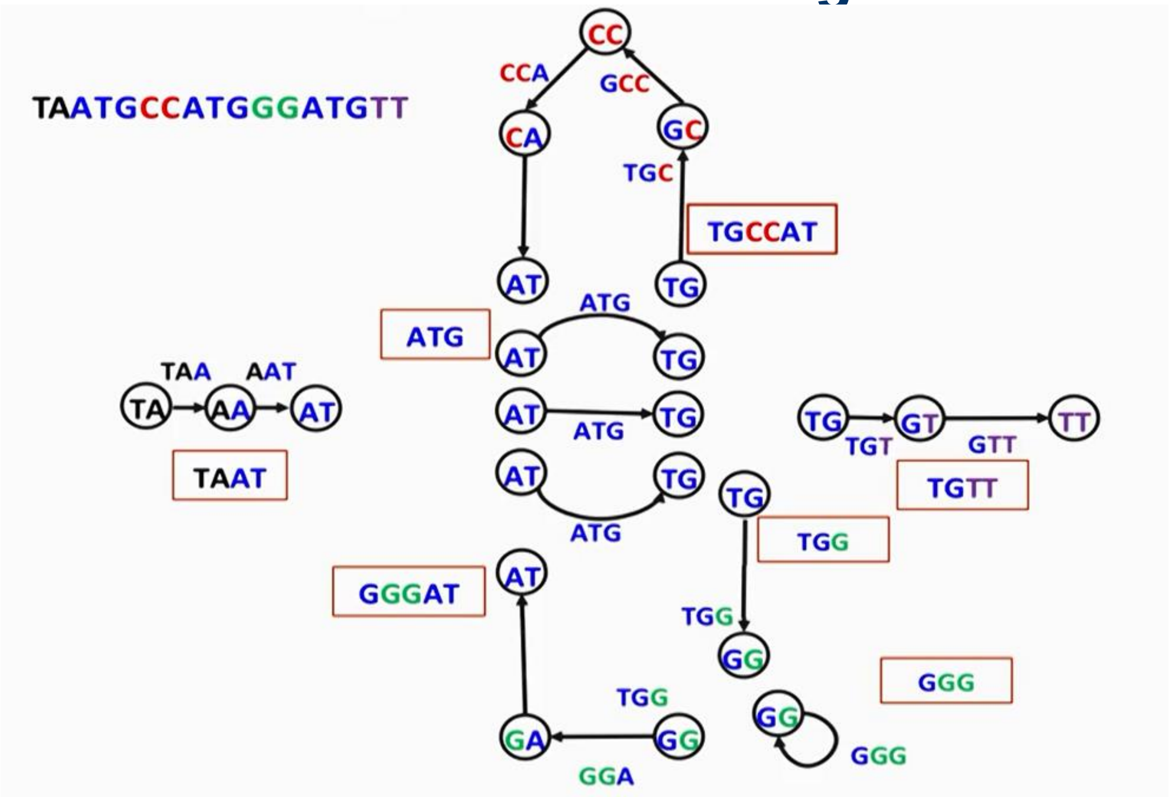
Practical issues

- Multiple Eulerian Paths
  - Which one to use?

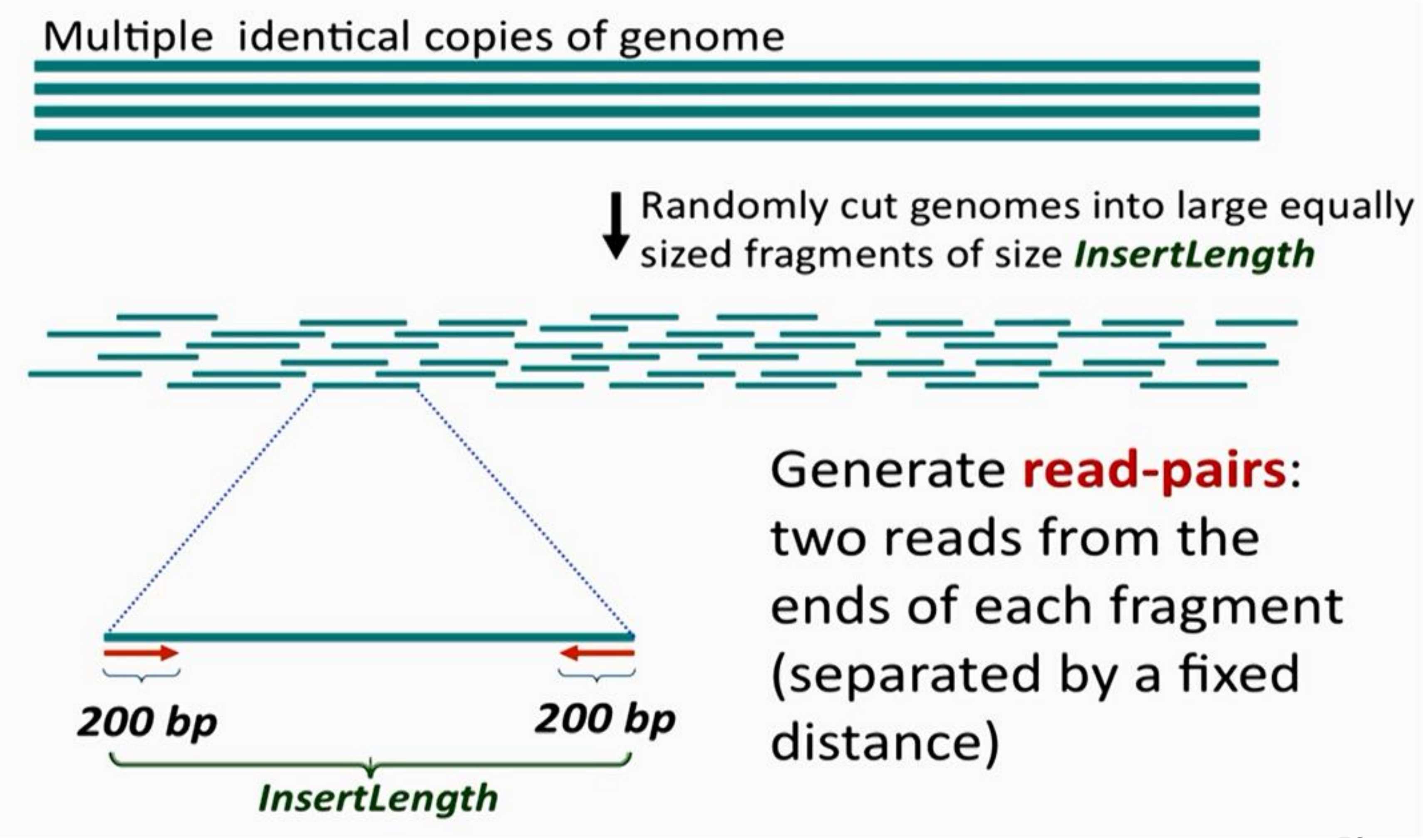


Contigs

- Dismember the De Bruijn graph to get ‘contigs’
  - Ideally we want just one contig
  - Reduce the number of contigs!



Solution: Paired end reads

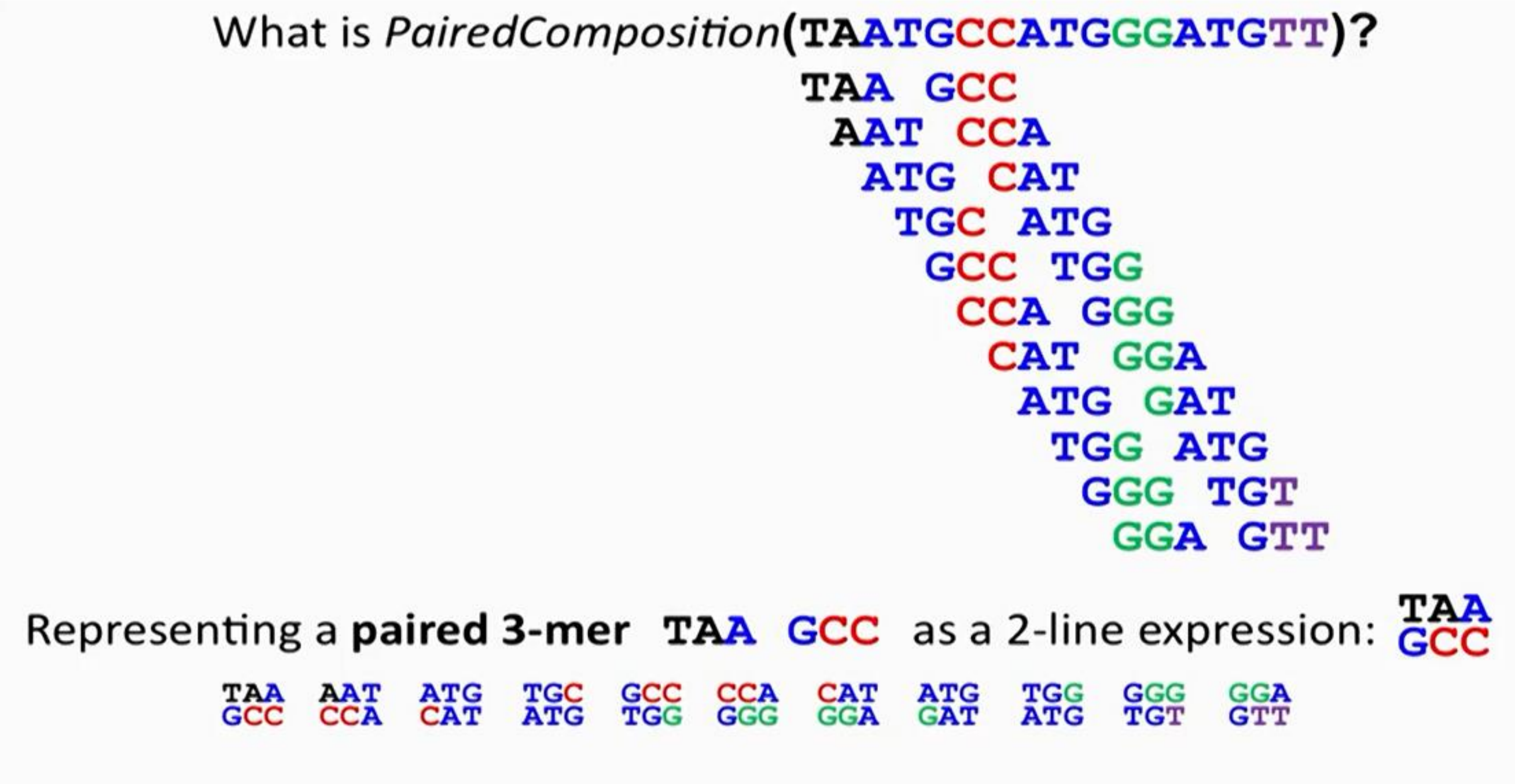


Multiple identical copies of genome

Randomly cut genomes into large equally sized fragments of size *InsertLength*

Generate **read-pairs**: two reads from the ends of each fragment (separated by a fixed distance)

Paired Composition



What is *PairedComposition*(TAATGCCATGGATGTT)?

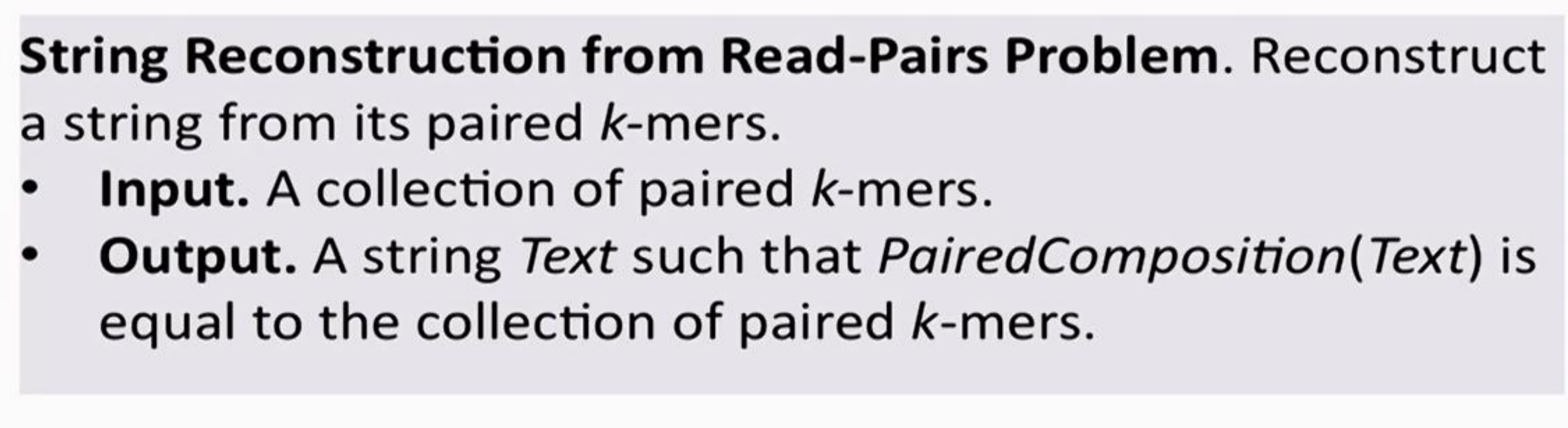
```
TAA GCC
 AAT CCA
  ATG CAT
   TGC ATG
    GCC TGG
     CCA GGG
      CAT GGA
       ATG GAT
        TGG ATG
         GGG TGT
          GGA GTT
```

Representing a **paired 3-mer** TAA GCC as a 2-line expression:

```
TAA
GCC
```

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

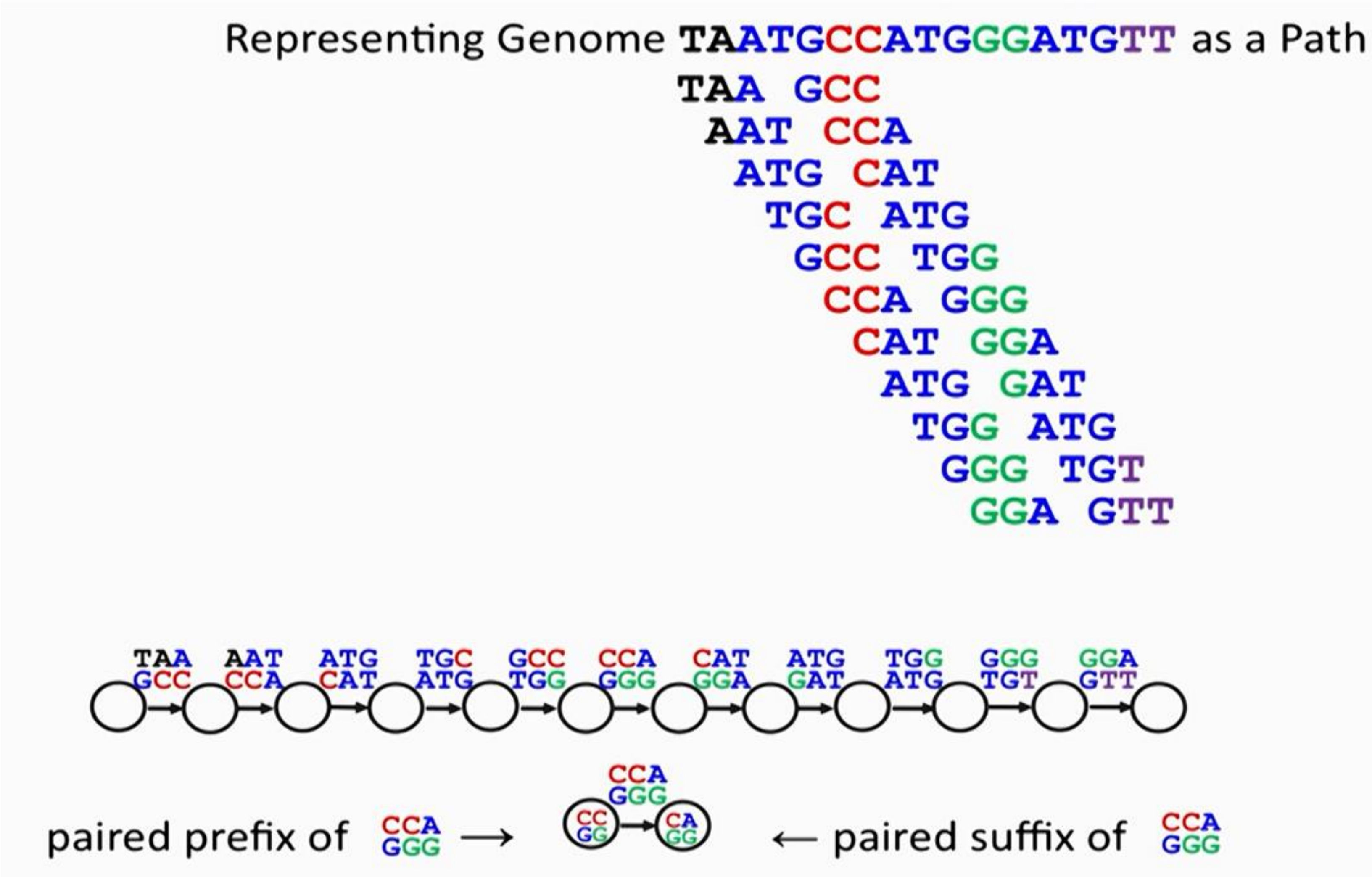
Paired String Reconstruction Problem



String Reconstruction from Read-Pairs Problem. Reconstruct a string from its paired *k*-mers.

- **Input.** A collection of paired *k*-mers.
- **Output.** A string *Text* such that *PairedComposition*(*Text*) is equal to the collection of paired *k*-mers.

Solution: Paired De Bruijn Graphs



Representing Genome TAATGCCATGGGATGTT as a Path

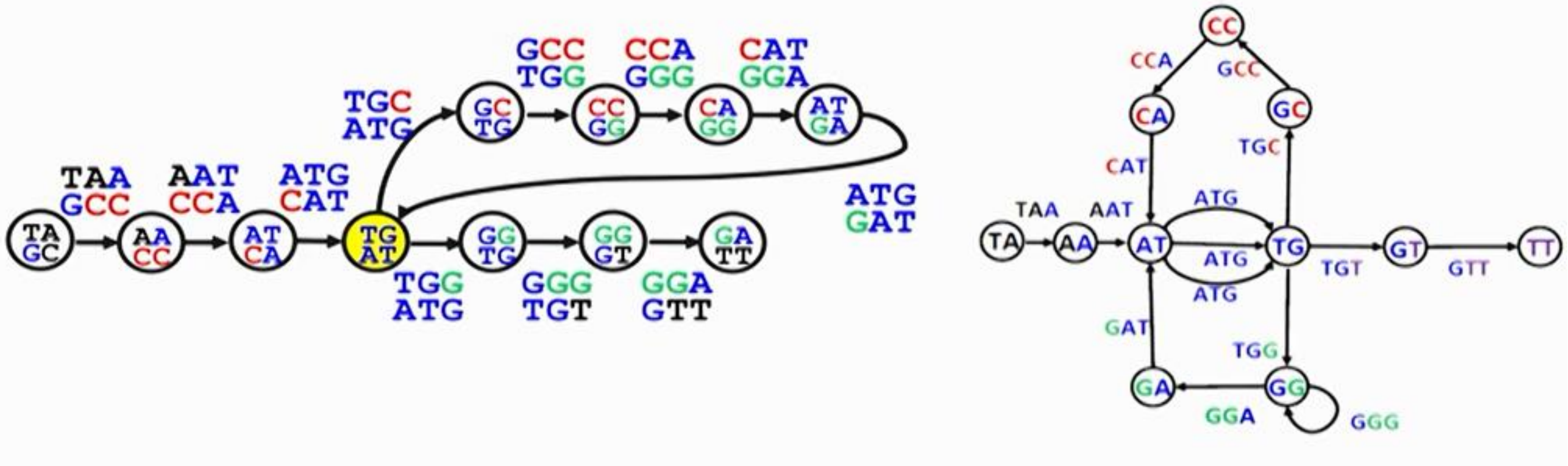
TAA GCC
AAT CCA
ATG CAT
TGC ATG
GCC TGG
CCA GGG
CAT GGA
ATG GAT
TGG ATG
GGG TGT
GGA GTT

paired prefix of CCA GGG →

← paired suffix of CCA GGG

Solution: Paired De Bruijn Graphs

- Lesser number of contigs in paired
  - One Eulerian Path



Paired de Bruijn Graph

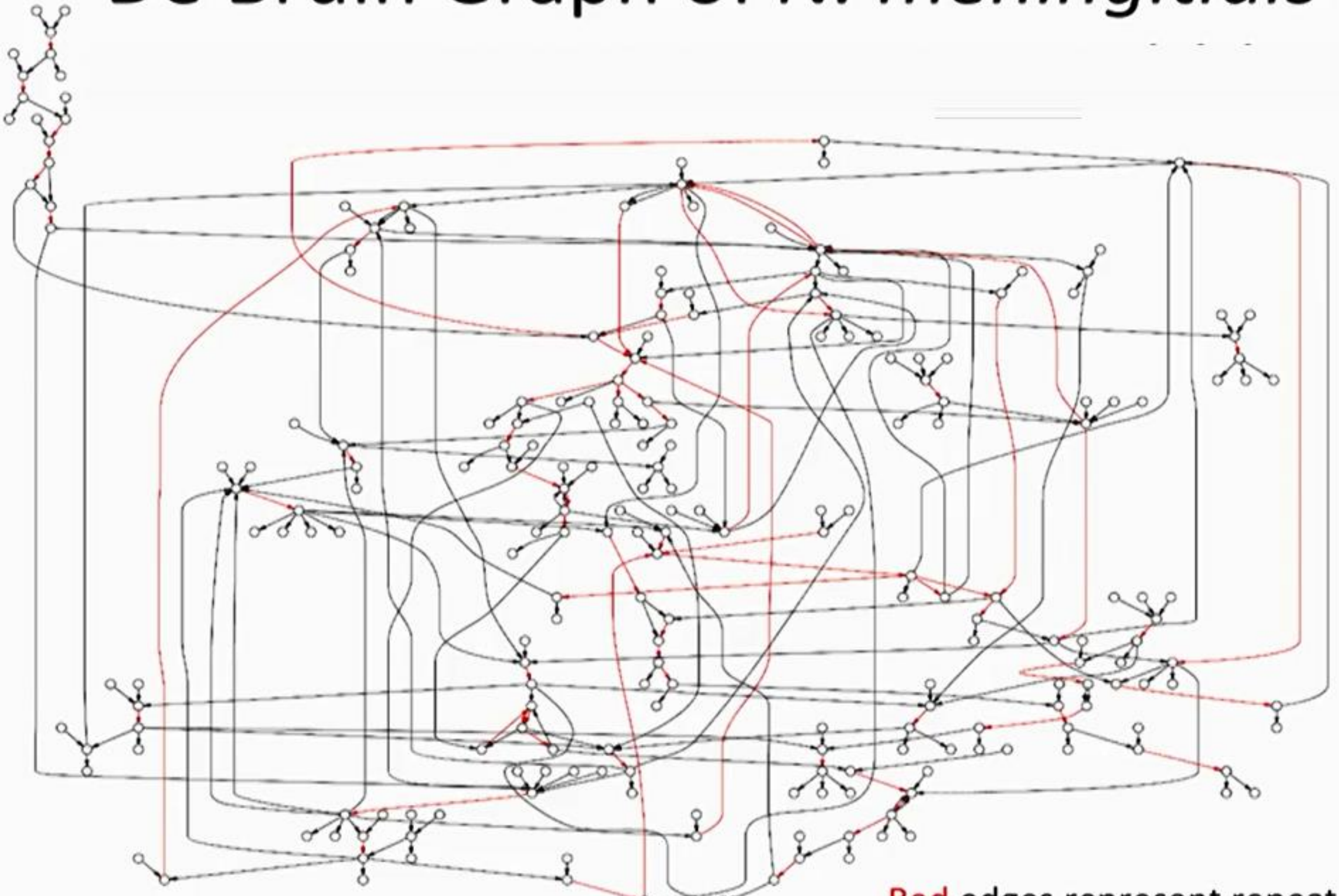
De Bruijn Graph

Unrealistic Assumptions

- Perfect coverage of genome by reads
    - Every k-mer from the genome is represented by a read
- Reads are error free
- Multiplicities of k-mers are known
- Distances between reads within read-pairs are exact

- A lot of effort by a lot of people to make it feasible!

De Bruin Graph of *N. meningitidis*

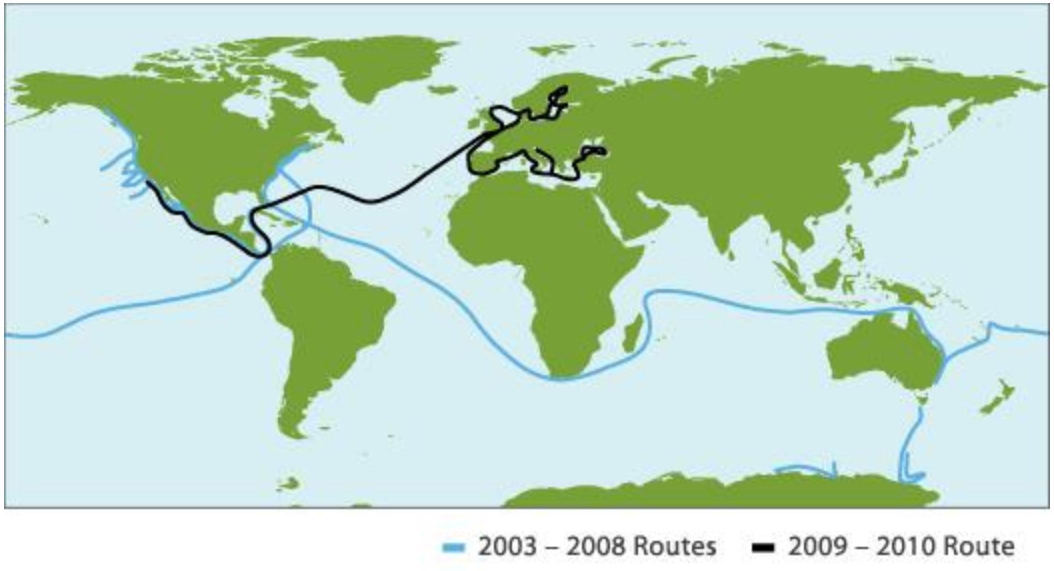


Red edges represent repeats

Sequencing

- A lot of sequencing going on
- Metagenomics
  - Sequencing samples
- Genomes in ocean water
  - http://www.jcvi.org/cms/research/projects/gos/overview/

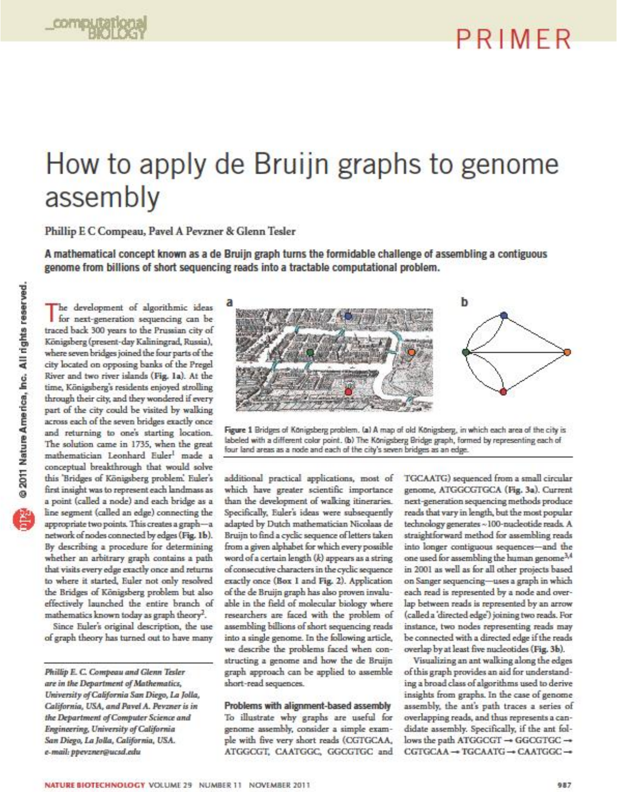
Sorcerer II



Required Reading

- http://www.nature.com/nbt/journal/v29/n11/full/nbt.2023.html

Compeau, Phillip E. C., Pavel A. Pevzner, and Glenn Tesler. 2011. "How to Apply de Bruijn Graphs to Genome Assembly." *Nature Biotechnology* 29 (11): 987–91. doi:10.1038/nbt.2023.



_computational BIOLOGY PRIMER

How to apply de Bruijn graphs to genome assembly

Phillip E C Compeau, Pavel A Pevzner & Glenn Tesler

A mathematical concept known as a de Bruijn graph turns the formidable challenge of assembling a contiguous genome from billions of short sequencing reads into a tractable computational problem.

© 2011 Nature America, Inc. All rights reserved.

The development of algorithmic ideas for next-generation sequencing can be traced back 300 years to the Prussian city of Königsberg (present-day Kaliningrad, Russia), where seven bridges joined the four parts of the city located on opposing banks of the Pregel River and two river islands (**Fig. 1a**). At the time, Königsberg's residents enjoyed strolling through their city, and they wondered if every part of the city could be visited by walking across each of the seven bridges exactly once and returning to one's starting location. The solution came in 1735, when the great mathematician Leonhard Euler[1] made a conceptual breakthrough that would solve this 'Bridges of Königsberg problem.' Euler's first insight was to represent each landmass as a point (called a node) and each bridge as a line segment (called an edge) connecting the appropriate two points. This creates a graph—a network of nodes connected by edges (**Fig. 1b**). By describing a procedure for determining whether an arbitrary graph contains a path that visits every edge exactly once and returns to where it started, Euler not only resolved the Bridges of Königsberg problem but also effectively launched the entire branch of mathematics known today as graph theory[2].

Since Euler's original description, the use of graph theory has turned out to have many additional practical applications, most of which have greater scientific importance than the development of walking itineraries. Specifically, Euler's ideas were subsequently adapted by Dutch mathematician Nicolaas de Bruijn to find a cyclic sequence of letters taken from a given alphabet for which every possible word of a certain length (*k*) appears as a string of consecutive characters in the cyclic sequence exactly once (Box 1 and **Fig. 2**). Application of the de Bruijn graph has also proven invaluable in the field of molecular biology where researchers are faced with the problem of assembling billions of short sequencing reads into a single genome. In the following article, we describe the problems faced when constructing a genome and how the de Bruijn graph approach can be applied to assemble short-read sequences.

Figure 1 Bridges of Königsberg problem. (**a**) A map of old Königsberg, in which each area of the city is labeled with a different color point. (**b**) The Königsberg Bridge graph, formed by representing each of four land areas as a node and each of the city's seven bridges as an edge.

Problems with alignment-based assembly

To illustrate why graphs are useful for genome assembly, consider a simple example with five very short reads (CGTGCAA, ATGGCGT, CAATGGC, GGCGTGC and TGCAATG) sequenced from a small circular genome, ATGGCGTGCA (**Fig. 3a**). Current next-generation sequencing methods produce reads that vary in length, but the most popular technology generates ~100-nucleotide reads. A straightforward method for assembling reads into longer contiguous sequences—and the one used for assembling the human genome[3,4] in 2001 as well as for all other projects based on Sanger sequencing—uses a graph in which each read is represented by a node and overlap between reads is represented by an arrow (called a 'directed edge') joining two reads. For instance, two nodes representing reads may be connected with a directed edge if the reads overlap by at least five nucleotides (**Fig. 3b**).

Visualizing an ant walking along the edges of this graph provides an aid for understanding a broad class of algorithms used to derive insights from graphs. In the case of genome assembly, the ant's path traces a series of overlapping reads, and thus represents a candidate assembly. Specifically, if the ant follows the path ATGGCGT → GGCGTGC → CGTGCAA → TGCAATG → CAATGGC →

Phillip E. C. Compeau and Glenn Tesler are in the Department of Mathematics, University of California San Diego, La Jolla, California, USA, and Pavel A. Pevzner is in the Department of Computer Science and Engineering, University of California San Diego, La Jolla, California, USA. e-mail: ppevzner@ucsd.edu

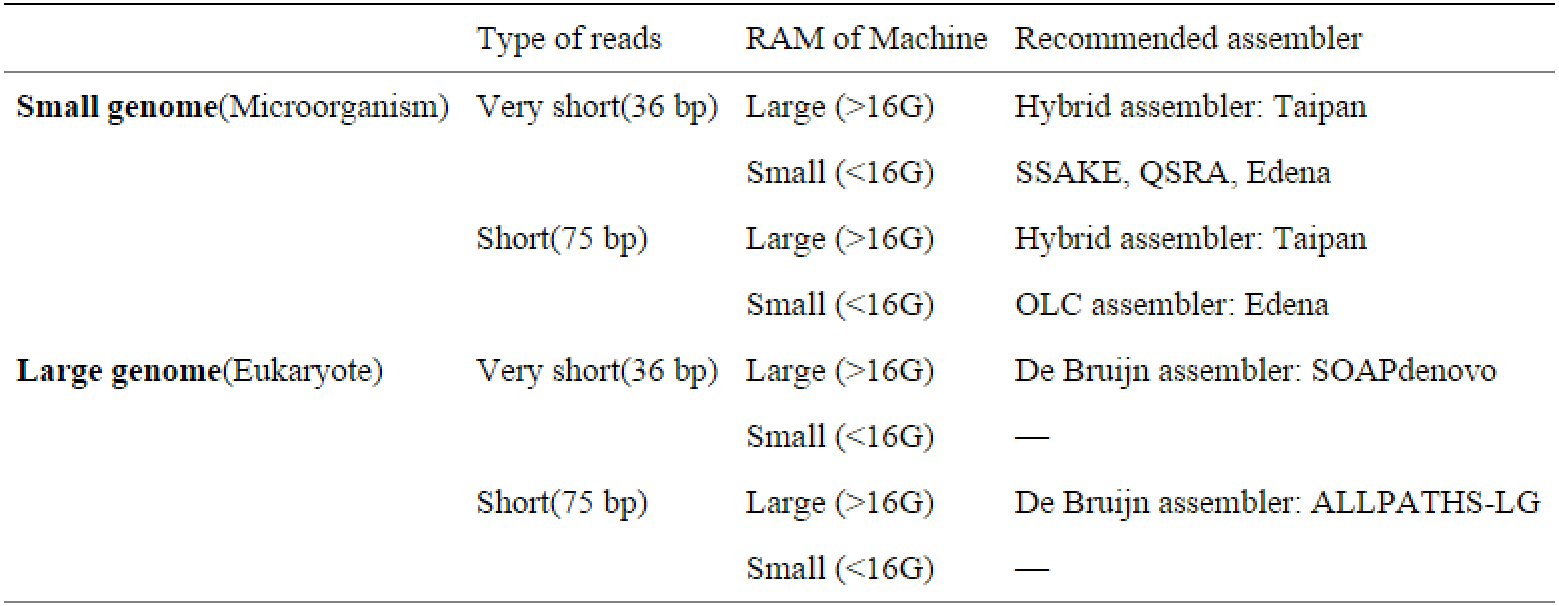
NATURE BIOTECHNOLOGY VOLUME 29 NUMBER 11 NOVEMBER 2011 987

Tools

- UGENE
  - SPADES
- VELVET
  - BIOLINUX
    - http://environmentalomics.org/bio-linux-software-list/

Comparison of de novo assemblers

- Zhang, Wenyu, Jiajia Chen, Yang Yang, Yifei Tang, Jing Shang, and Bairong Shen. “A Practical Comparison of De Novo Genome Assembly Software Tools for Next-Generation Sequencing Technologies.” *PLoS ONE* 6, no. 3 (March 14, 2011). doi:10.1371/journal.pone.0017915.



	Type of reads	RAM of Machine	Recommended assembler
Small genome(Microorganism)	Very short(36 bp)	Large (>16G)	Hybrid assembler: Taipan
		Small (<16G)	SSAKE, QSRA, Edena
	Short(75 bp)	Large (>16G)	Hybrid assembler: Taipan
		Small (<16G)	OLC assembler: Edena
Large genome(Eukaryote)	Very short(36 bp)	Large (>16G)	De Bruijn assembler: SOAPdenovo
		Small (<16G)	—
	Short(75 bp)	Large (>16G)	De Bruijn assembler: ALLPATHS-LG
		Small (<16G)	—

General Applications of NGS

- The applications of NGS seem almost endless
  - Resequencing of the human genome to identify genes and regulatory elements involved in disease
  - Whole-genome sequencing of different species for comparative biology analyses
  - Sequencing of microbial species to identify novel virulence factors involved in pathogenesis and spread of disease
  - Gene expression studies using RNA-seq allow researchers and clinicians to visualize expression in sequence form
- As NGS continues to grow in popularity, it is inevitable that novel applications will continue to appear

Grada, Ayman, and Kate Weinbrecht. “Next-Generation Sequencing: Methodology and Application.” *Journal of Investigative Dermatology* 133, no. 8 (August 2013): e11. doi:10.1038/jid.2013.248.

Clinical Applications of NGS

- Whole-exome sequencing
  - The exome consists of only the protein-coding regions of the genome (a little over 1% of the genome)
  - Sequencing of the exome is used in gene discovery research
  - Exome sequencing can facilitate the discovery of disease-causing mutations
- Targeted sequencing
  - Sequencing that specifically targets regions of the genome that are of interest to researchers or clinicians
  - Targeted sequencing is more affordable and yields much higher coverage of genomic regions of interest
  - Sequencing panels can be developed to target specific genomic regions or disease-causing mutation hotspots

Input

- FASTQ Files: Reads stored
  - Also have quality information

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

  - Phred quality scoring

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

  - Different machines use different formats on quality

Input: Reference Alignments

- FASTA files

Output of De Novo Alignment

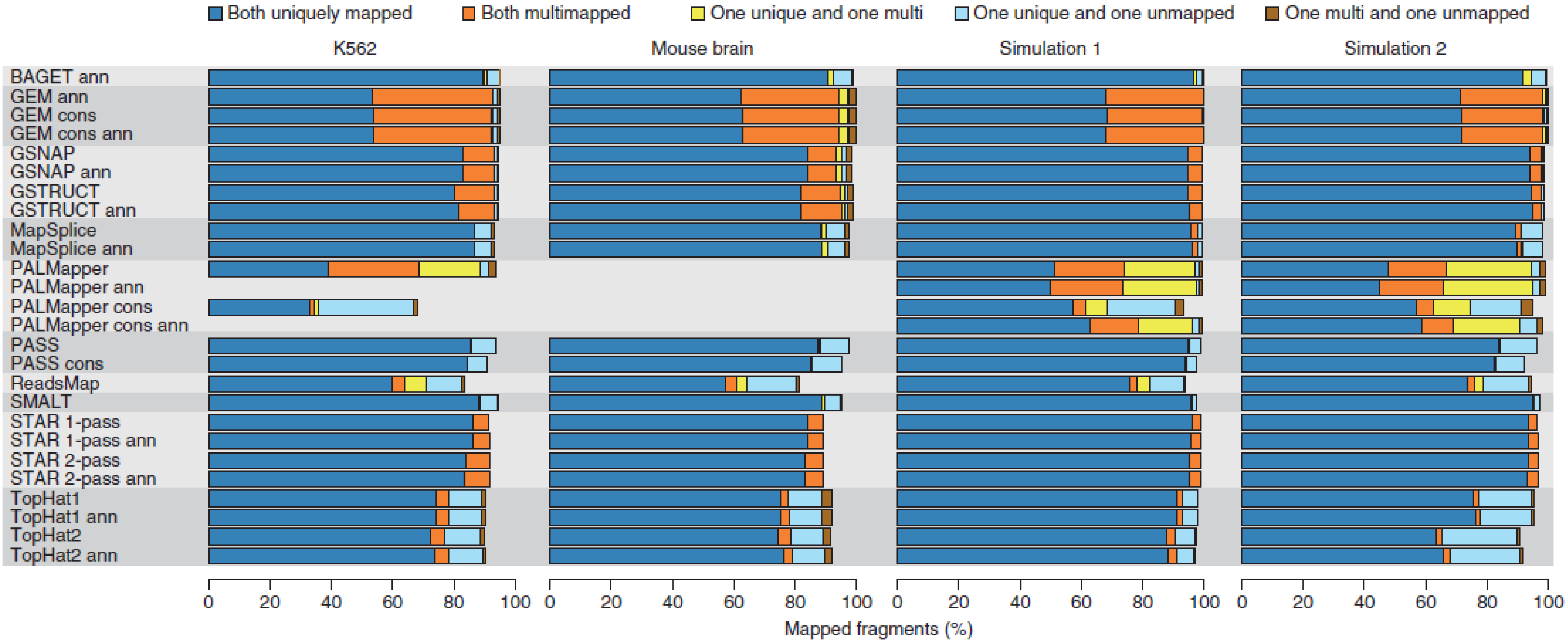
- Contigs (FASTA):
    - A group of overlapping clones representing regions of the genome; the contiguous sequence of DNA created by assembling these overlapping chromosome fragments.
- Genome

Output of Reference Alignment

- SAM or BAM
- SAM: Text format for storing sequence data
- BAM: Binary

- Tell where each read is aligned

Read mapping tools



- Engström, Pär G., Tamara Steijger, Botond Sipos, Gregory R. Grant, André Kahles, The RGASP Consortium, Gunnar Rätsch, et al. “Systematic Evaluation of Spliced Alignment Programs for RNA-Seq Data.” *Nature Methods* advance online publication (November 3, 2013). doi:10.1038/nmeth.2722.

Tools

- **Short read mapping (RNA-Seq, ChIP Seq, mirRNA-Seq)**
  - **TopHat**
  - **BWA**
- **Prediction of Alternative Splicing**
  - **SpliceGrapher**
- **Differential Expression**
  - DESeq
  - EdgeR
- **https://usegalaxy.org/**
- **UGENE**