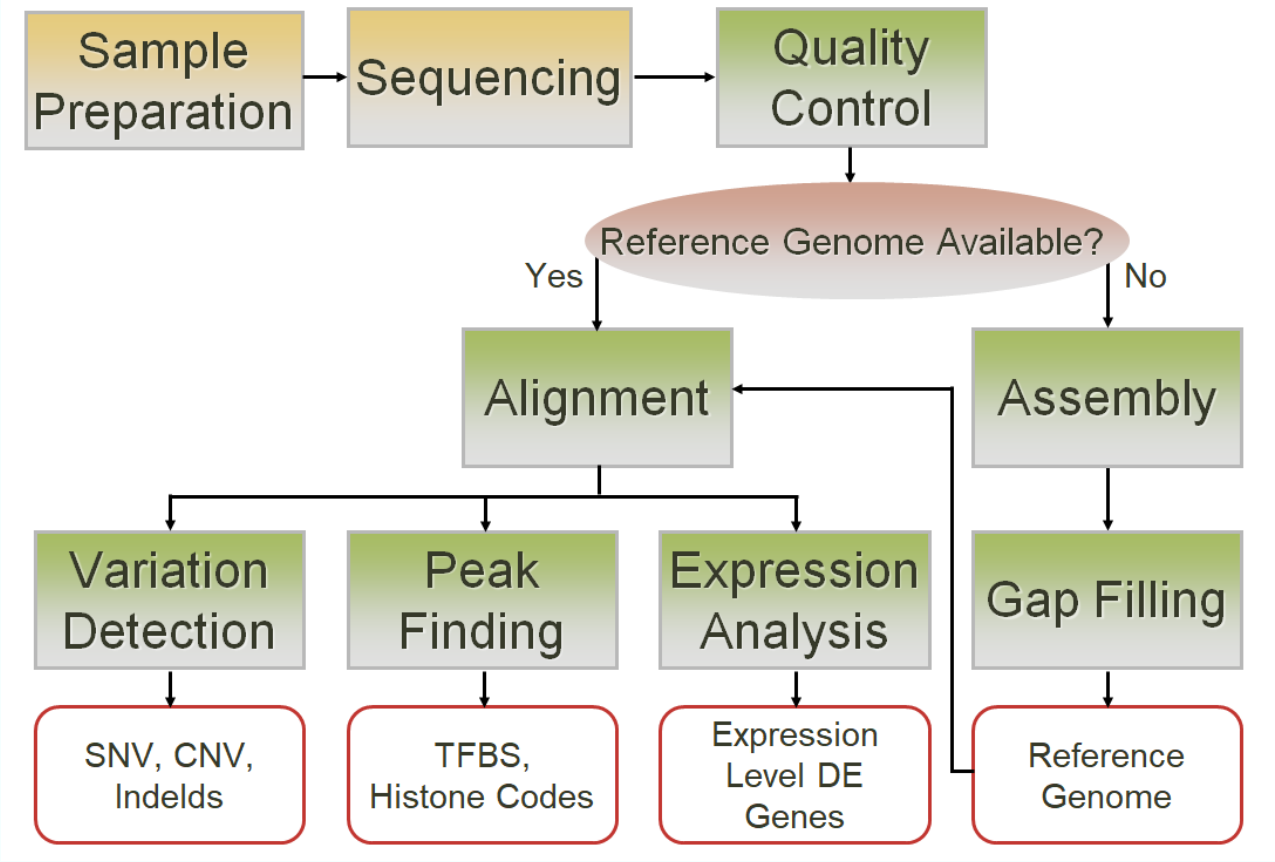
Genome Assembly

Introduction to Bioinformatics Course

Niloufar Razani
Department of Computer Engineering
Sharif University of Technology

Fall 2023

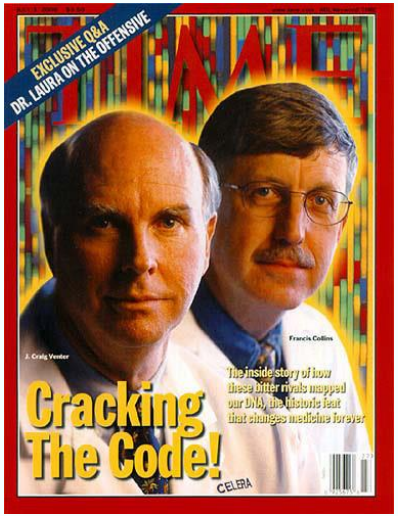
Analysis Pipeline



The Race to Sequence the Human Genome

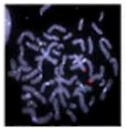
- 1990: The public Human Genome Project, headed by Francis Collins, aim to sequence the human genome by 2005.

- 1997: Graig Venter founds Celera Genomics, a private firm, with the same goal.



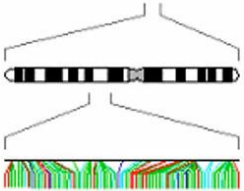
STRATEGIES FOR SEQUENCING THE HUMAN GENOME

Human Genome Project
(Hierarchical Shotgun Sequencing)
(Clone-by-clone sequencing)

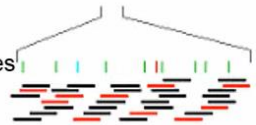


Celera Genomics Inc.
(Whole Genome Shotgun Sequencing)

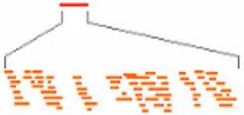
1. **Construction of maps**
   - Cut into 150Mb
   - Clone into BAC
   - Transform into *E. Coli*
   - Construct landmark map



2. **Golden Tiling Path**
   - Identify overlapping clones
   - Select tile path (red)
   - Subclone into plasmid



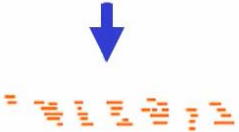
3. **Shotgun sequencing of subclones**



4. **Assemble into contigs**

1. **Randomly sheer genomic DNA**

2. **Clone into plasmid**



3. **Sequencing of insert clones**

4. **Align sequences**

5. **Assemble into contigs**

Clone by Clone Sequencing

- – Took ~11 years (1990-2001) to sequence 1 human and publish draft genome (complete genome: 2003).
- – $3B Budget, labs in 20 countries.
- – 2 Separate processes.
- – Clone libraries unstable, maps hard to complete.
- – Sequencing libraries must be made for every clone.
- + Assembly problem ‘easy’ and well understood.

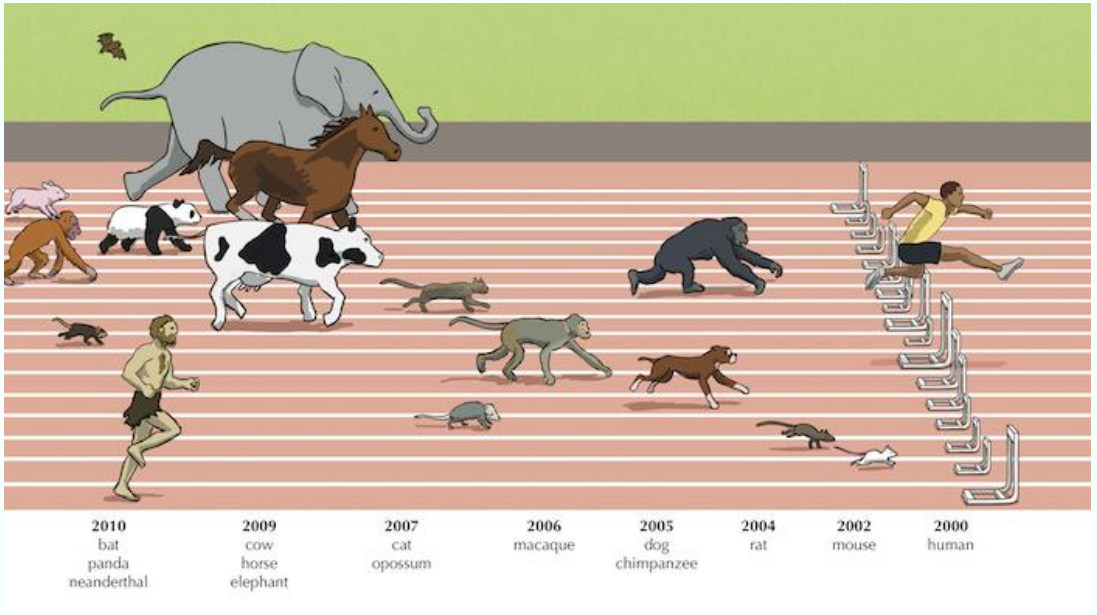
Whole Genome Shotgun Sequencing

- \+ Took ~3 years to sequence 1 human.
- $300M budget, labs by one company!
- – Assembly problem ‘challenging’.

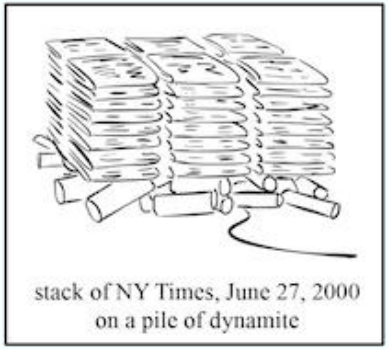
From Human to Mouse to Rat to …

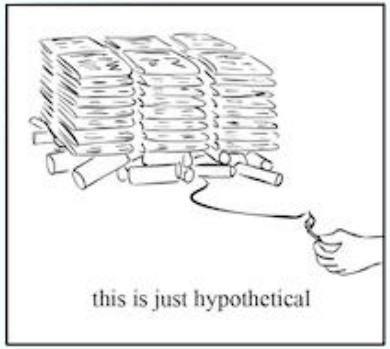
- The first sequenced genome, belonging to a φX174 bacterial phage (i.e., a virus that preys on bacteria), had only 5,386 nucleotides and was completed in 1977 by Frederick Sanger.

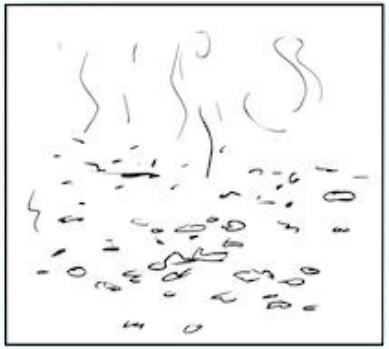
- genome sequencing has raced to the forefront of bioinformatics research, as the cost of sequencing plummeted. Because of the decreasing cost of sequencing, we now have thousands of sequenced genomes, including those of many mammals.

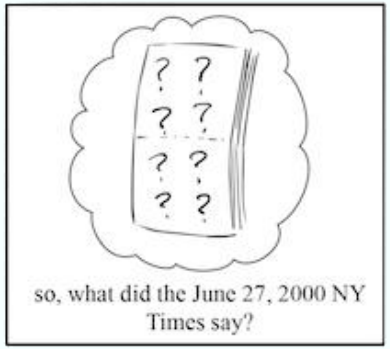


Newspaper Problem

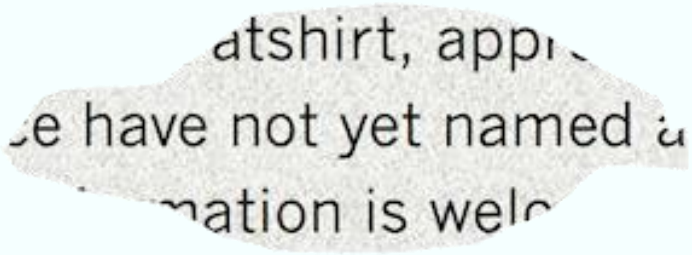


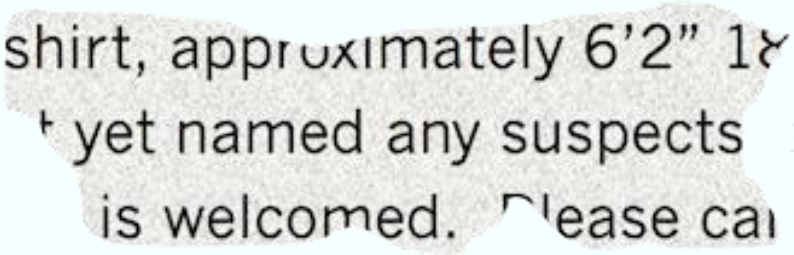






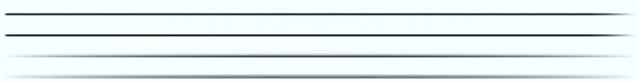
Newspaper Problem





Genome Assembly Problem

Multiple identical copies of a genome



Shatter the genome into reads

Sequence the reads

Assemble the genome using overlapping reads

AGAATATCA
GAGAATATC
TGAGAATAT
...TGAGAATATCA...

Greedy Algorithm

Find two sequences with the largest overlap and merge them

```
Let T = set of all fragments

do {

    For the pair (s,t) in T with maximum overlap do

     merge(s,t)

} while there are at least two strings in T

Output the only string in T
```

- It ignores the biological problems of: orientation, read errors, insertions and deletions.
- It is used in TIGR Assembler, Phrap and CAP3 and is the base of some other assemblers.

The String Reconstruction Problem

Genome assembly is more difficult than you think!

1. DNA is double-stranded, and we have no way of knowing a priori **which strand a given read derives from**.
2. Modern sequencing machines are not perfect, and the reads that they generate often contain **errors.**
3. Some regions of the genome may not be covered by any reads, making it impossible to reconstruct the entire genome.
4. **Repeats** and **Bias** in Reads.

Assumptions

- Reads are all **k-mers for some value of k.**
- All reads come from the **same strand, have no errors, and exhibit perfect coverage**, so that every k-mer substring of the genome is generated as a read.

The String Reconstruction Problem

- Given a string Text, its **k-mer composition** $\text{COMPOSITION}_k(\text{Text})$ is the collection of all k-mer substrings of Text (including repeated k-mers).

COMPOSITION_3 (TATGGGGTGC) = {ATG, GGG, GGG, GGT, GTG, TAT, TGC, TGG}.

String Composition Problem: Generate the k-mer composition of a string.

Input: An integer k and a string Text.

Output: $\text{Composition}_k(\text{Text})$, where the k-mers are arranged in lexicographic order.

In order to model genome assembly, we need to solve inverse problem.

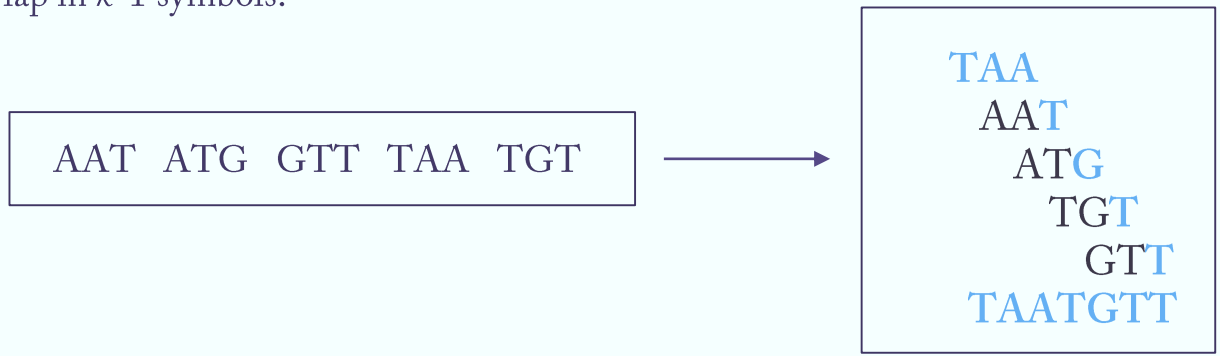
The String Reconstruction Problem

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

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

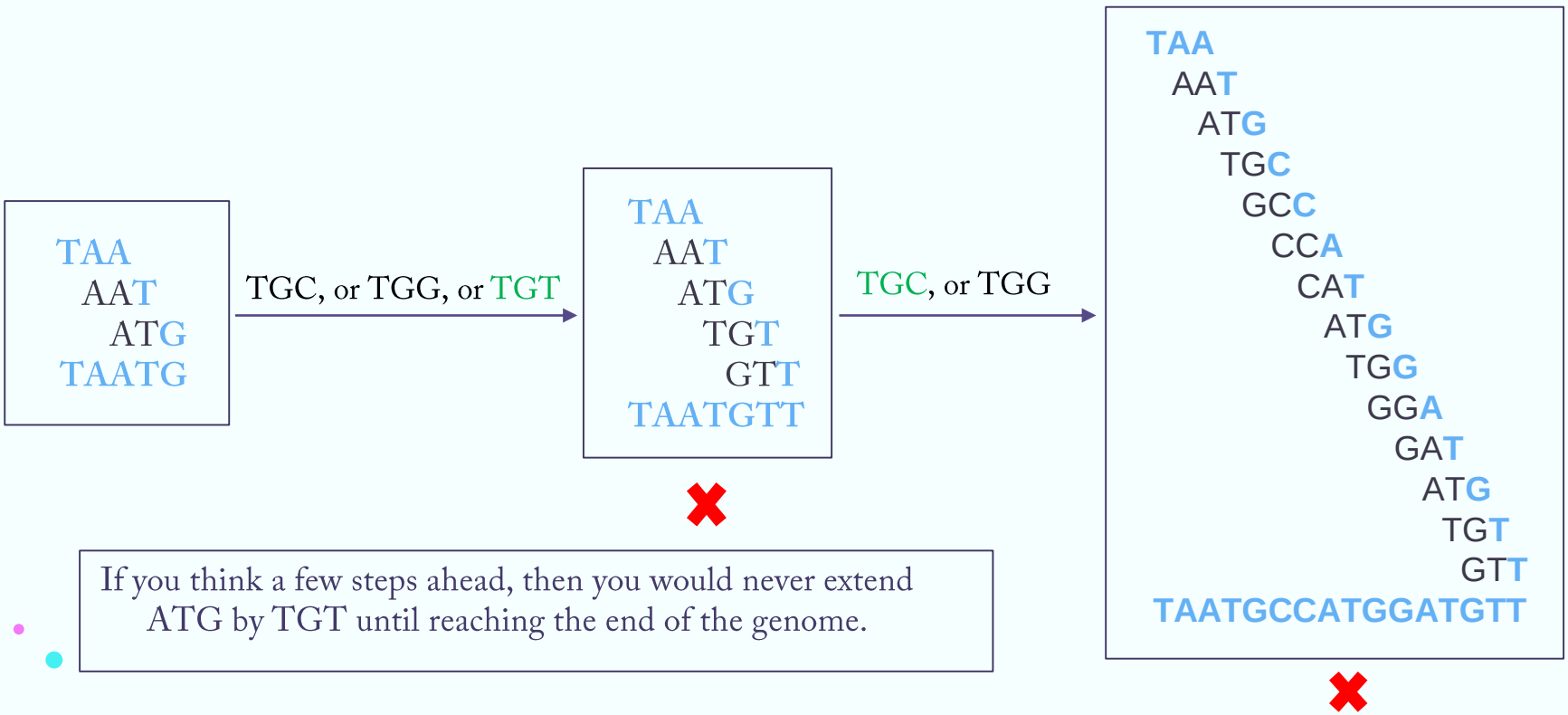
Output: A string Text with k-mer composition equal to Patterns (if such a string exists).

- The most natural way to solve the String Reconstruction Problem is to "connect" a pair of *k*-mers if they overlap in *k*-1 symbols.



Repeats complicate genome assembly

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



If you think a few steps ahead, then you would never extend ATG by TGT until reaching the end of the genome.

Repeats complicate genome assembly

- With millions of reads, repeats make it much more difficult to “look ahead” and construct the correct assembly.

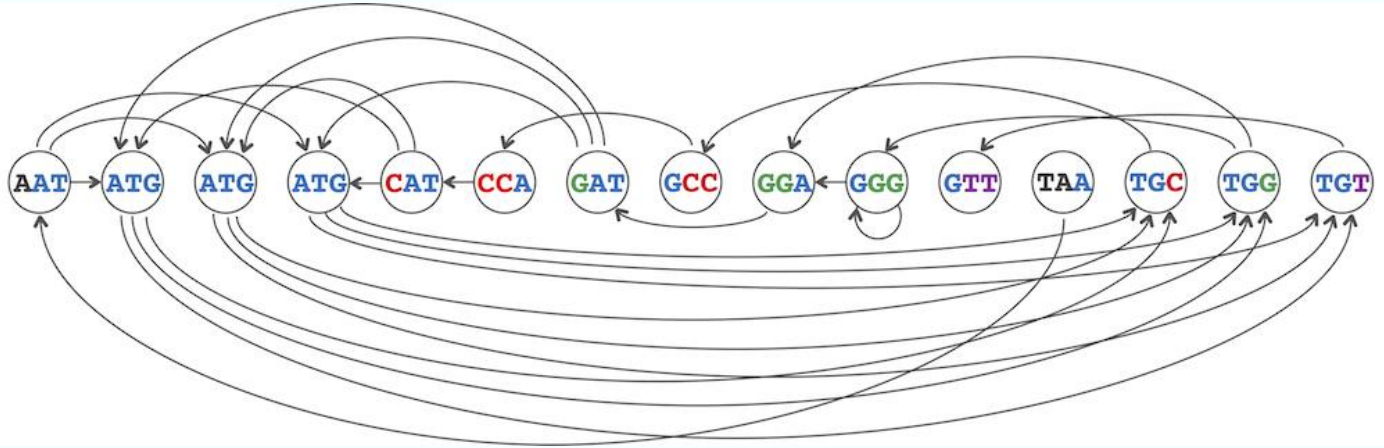
- approximately 50% of the human genome is made up of repeats, e.g., the approximately 300 nucleotide-long **Alu sequence** (The most common transposon in humans) is repeated over a million times, with only a few nucleotides inserted/ deleted/ substituted each time.

String Reconstruction as a Walk in the Overlap Graph

- Repeats in a genome necessitate some way of looking ahead to see the correct assembly in advance.

- We form a node for each k-mer in Patterns and connect k-mers Pattern and Pattern' by a directed edge if SUFFIX(Pattern) = PREFIX(Pattern'). The resulting graph is called the **overlap graph** on these k-mers.

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

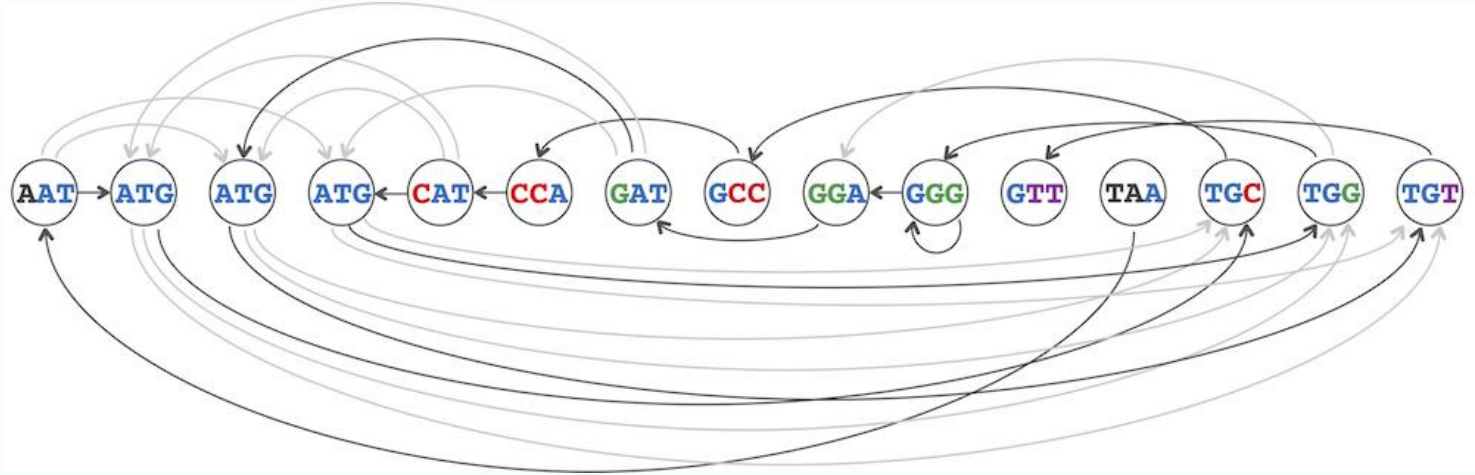


overlap graph

String Reconstruction as a Walk in the Overlap Graph

- We now know that to solve the String Reconstruction Problem, we are looking for a path in the overlap graph that visits every node exactly once. A path in a graph visiting every node once is called a **Hamiltonian path.**

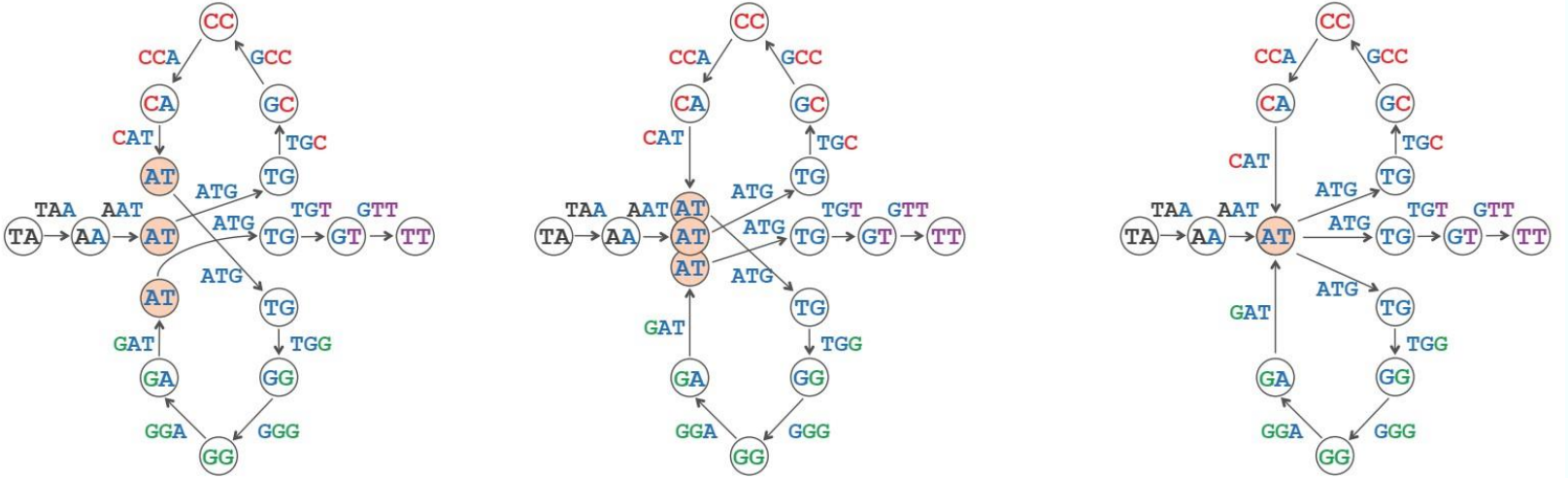
AAT ATG ATG ATG CAT CCA GAT GCC GGA GGG GTT TAA TGC TGG TGT



The **genome path** spelling out TAATGCCATGGGATGTT

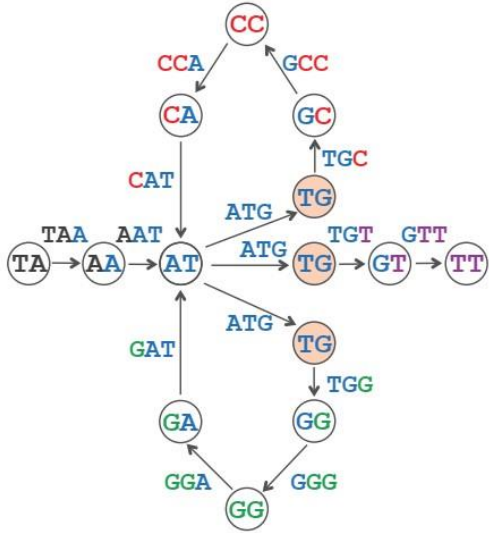
Another Graph for String Reconstruction

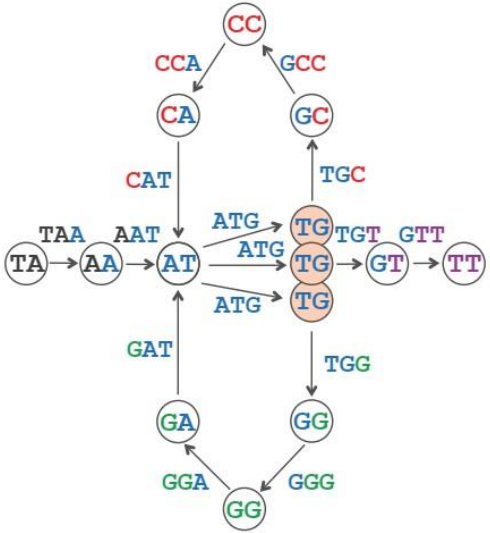
- This time, instead of assigning these 3-mers to nodes, we will assign them to edges.
- we will label each **node** of this graph with **a 2-mer representing the overlapping nucleotides** shared by the edges on either side of the node.
- Nothing seems new here until we start **gluing** identically labeled nodes.

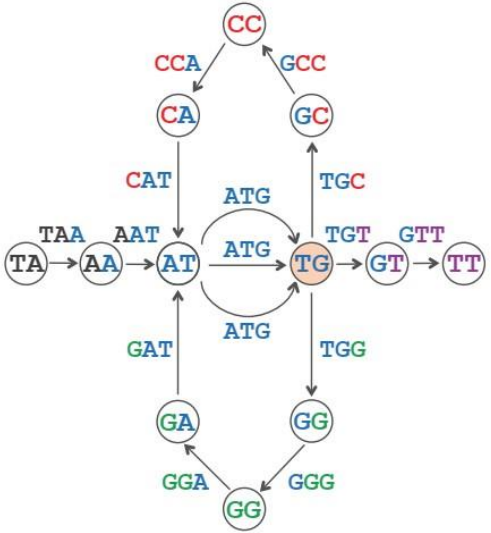


TAATGCCATGGGATGTT

Another Graph for String Reconstruction





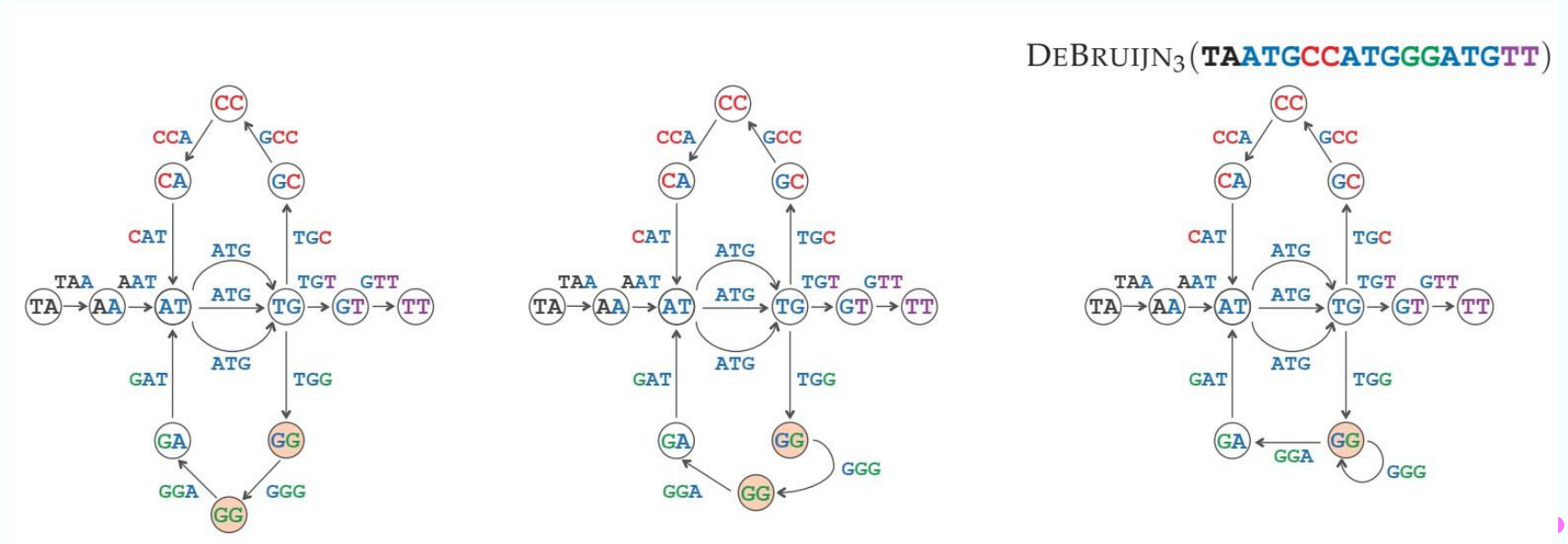


TAATGCCATGGGATGTT

Another Graph for String Reconstruction

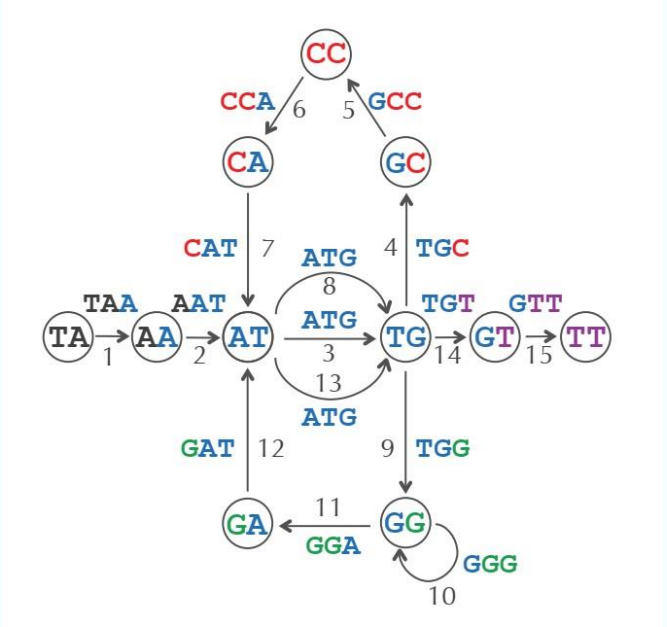
- The number of nodes in the resulting graph has reduced from 16 to 11, while the number of edges stayed the same.

- This graph is called the **de Bruijn graph** of TAATGCCATGGGATGTT, denoted $DeBruijn_3$(TAATGCCATGGGATGTT).



Walking in the de Bruijn Graph

- Even though we have glued together nodes to form the de Bruijn graph, we have not changed its edges, and so the path from TA to TT reconstructing the genome is still hiding in $DeBruijn_3$(TAATGCCATGGGATGTT).
- Therefore, solving the String Reconstruction Problem reduces to finding a path in the de Bruijn graph that visits every edge exactly once.
- Such a path is called an **Eulerian Path.**

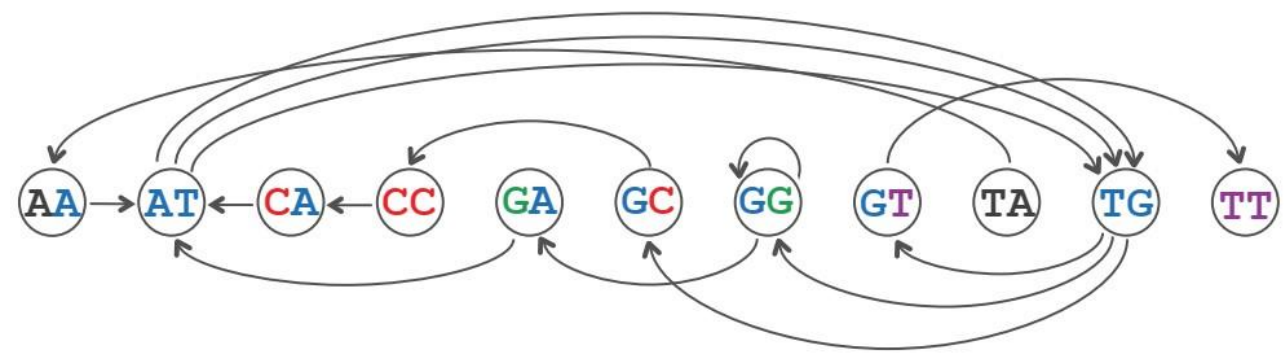


STOP and Think: Can you construct $DeBruijn_k$(Text) if you don't know Text but you do know its k-mer composition?

Constructing de Bruijn graphs from k-mer composition

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

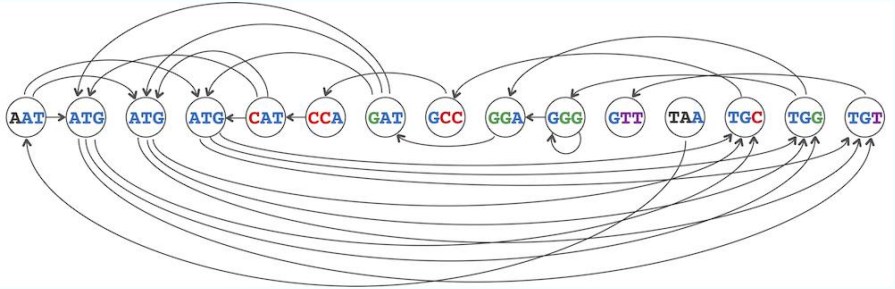
AA AT CA CC GA GC GG GT TA TG TT



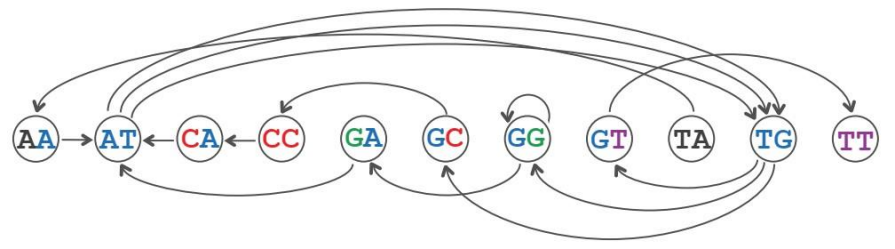
De Bruijn graphs versus overlap graphs

- We now have two ways of solving the String Reconstruction Problem.
- We can either find a **Hamiltonian path** in the overlap graph or find an **Eulerian path** in the de Bruijn graph.

STOP and Think: Which graph would you rather work with, the overlap graph or the de Bruijn graph?



overlap graph



De Bruijn graph

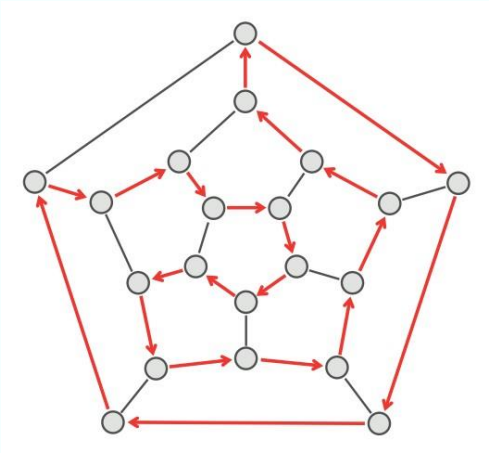
Hamiltonian paths/Cycles

Hamiltonian Path Problem: Construct a Hamiltonian path in a graph.

Input: A directed graph.

Output: A path visiting every node in the graph exactly once (if such a path exists).

Determining whether such paths and cycles exist in graphs (the Hamiltonian path problem and Hamiltonian cycle problem) are **NP-complete.**

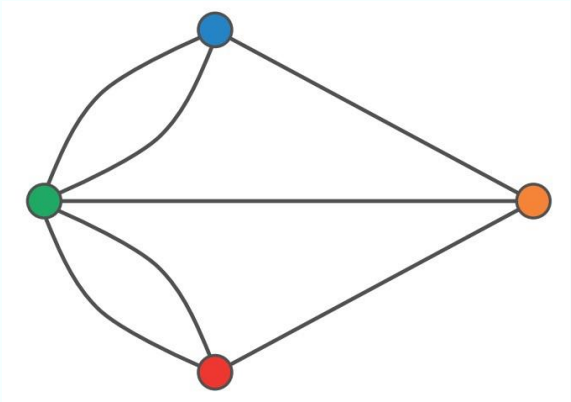


Eulerian paths/Cycles

Eulerian Path Problem: Construct an Eulerian path in a graph.
Input: A directed graph.
Output: A path visiting every edge in the graph exactly once (if it exists).

Euler proved a theorem dictating when a graph will have an **Eulerian cycle.**

Good News!
Euler's algorithm is polynomial.

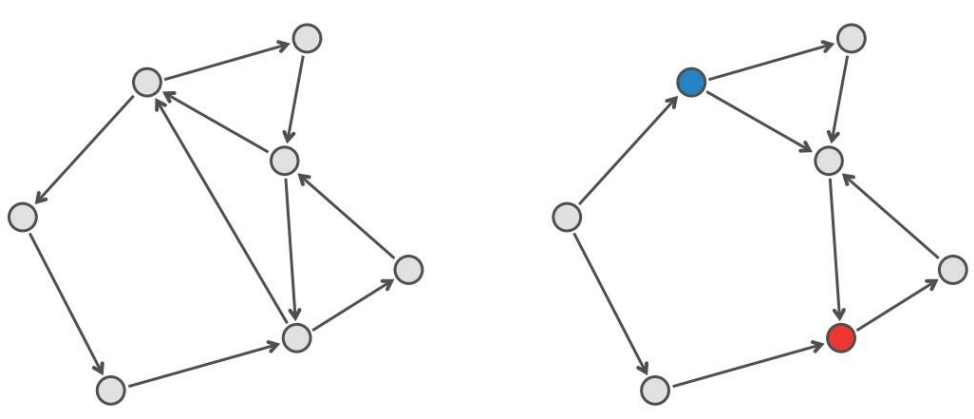


Euler's Theorem

- Euler's Theorem states that two conditions are sufficient to guarantee that an arbitrary graph is Eulerian(has Eulerian cycle):

1. The graph must be **Balanced**. (A node v is **balanced** if IN(v)=OUT(v), and a graph is **balanced** if all its nodes are balanced)

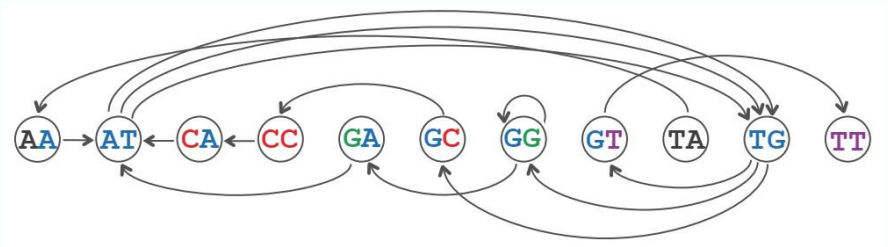
2. The graph must be **Strongly Connected.** (it is possible to reach any node from every other node)

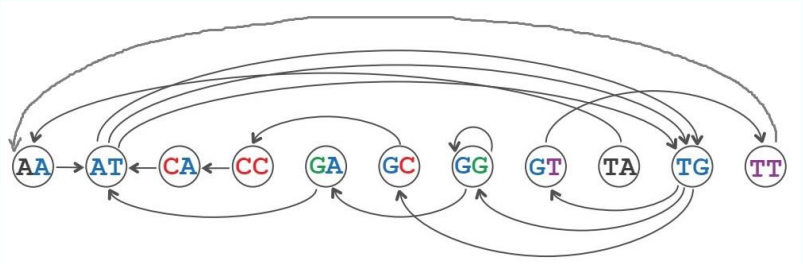


Balanced (left) and unbalanced (right) directed graphs.

From Eulerian cycles to Eulerian paths

- We can now check if a directed graph has an Eulerian cycle, but what about an Eulerian path?
- How many unbalanced nodes does a graph with an Eulerian path have?
- If an Eulerian path in a graph connects a node v to a different node w, then the graph is **nearly balanced**, meaning that all its nodes except v and w are balanced.
- **The De Bruijn graph representation of k-mers satisfies the prerequisites of Euler's theorem. This allows the original string to be reconstructed in linear time complexity.**



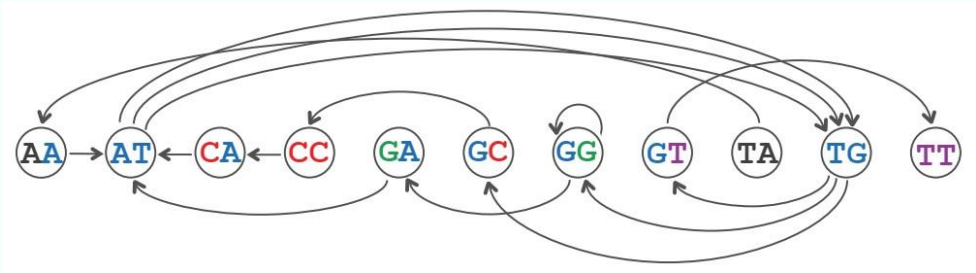


From reads to read-pairs

- Despite many attempts, biologists have not yet figured out how to generate *long* and *accurate* reads. The most accurate sequencing technologies available today generate reads that are only about 300 nucleotides long, which is too short to span most repeats, even in short bacterial genomes.

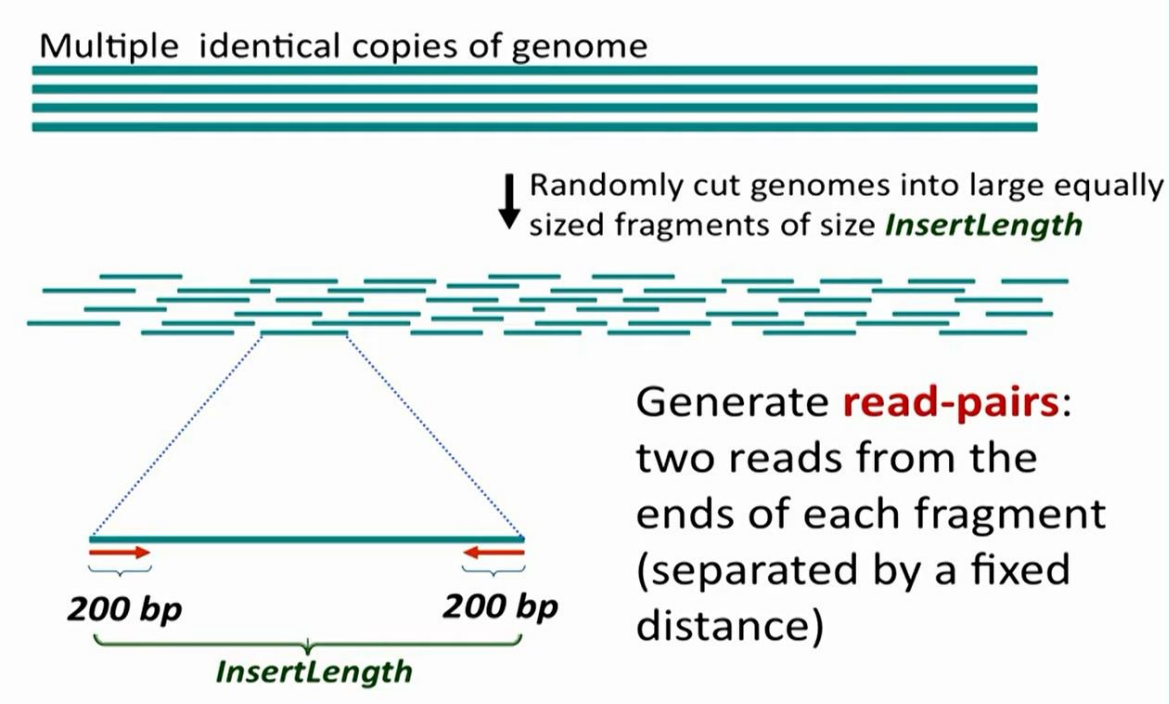
- The string TAATGCCATGGGATGTT cannot be **uniquely** reconstructed from its 3-mer composition since another string (TAATGGGATGCCATGTT) has the same 3-mer composition, There is two Eulerian path in De Bruijn graph. Which string are you going to give to biologist!

STOP and Think: What additional experimental information would allow you to uniquely reconstruct the string TAATGCCATGGGATGTT?



DNA sequencing with Read-pairs

Biologists have suggested an indirect way of increasing read length by generating **read-pairs.**

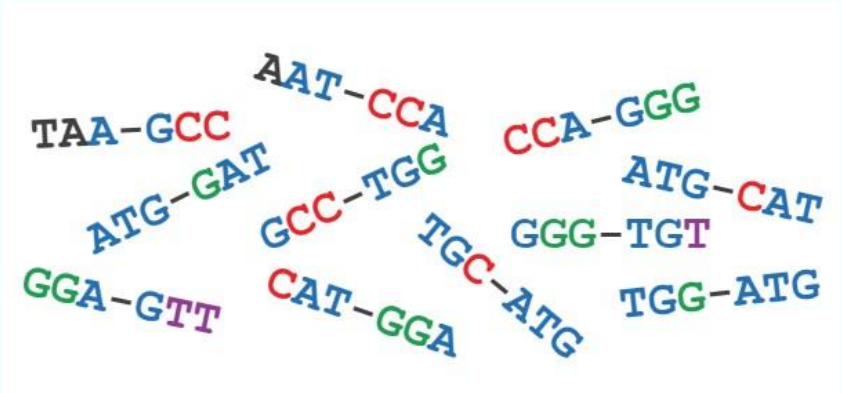


From reads to read-pairs

- **Read-pairs** are pairs of reads separated by a fixed distance *d* in the genome. A long "**gapped**" read of length *k* + *d* + *k* whose first and last *k*-mers are known but whose middle segment of length *d* is unknown.

- Read-pairs contain more information than *k*-mers alone, and so we should be able to use them to improve our assemblies.

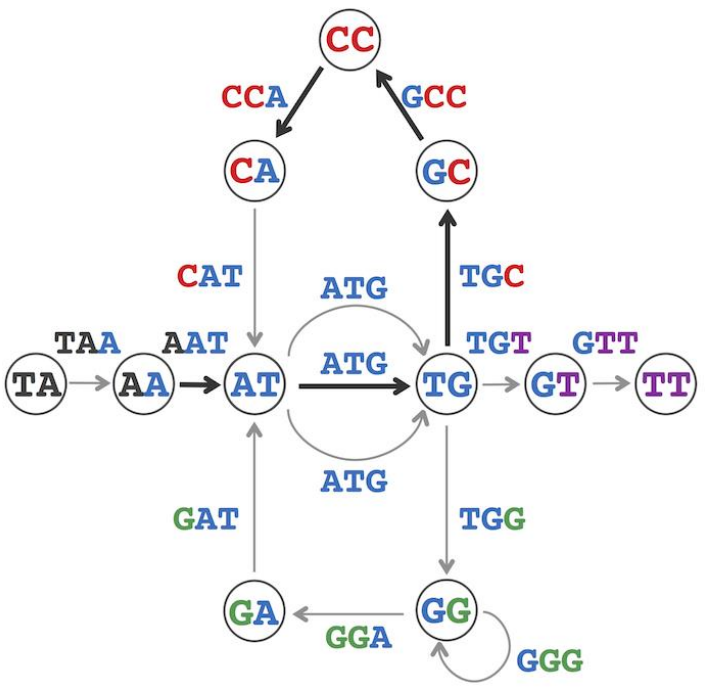
- If only you could infer the nucleotides in the middle segment of a read-pair, you would immediately increase the read length from *k* to 2 · *k* + *d*.



Read-pairs(d=1) sampled from TAATGCCATGGGATGTT

Transforming read-pairs into long virtual reads

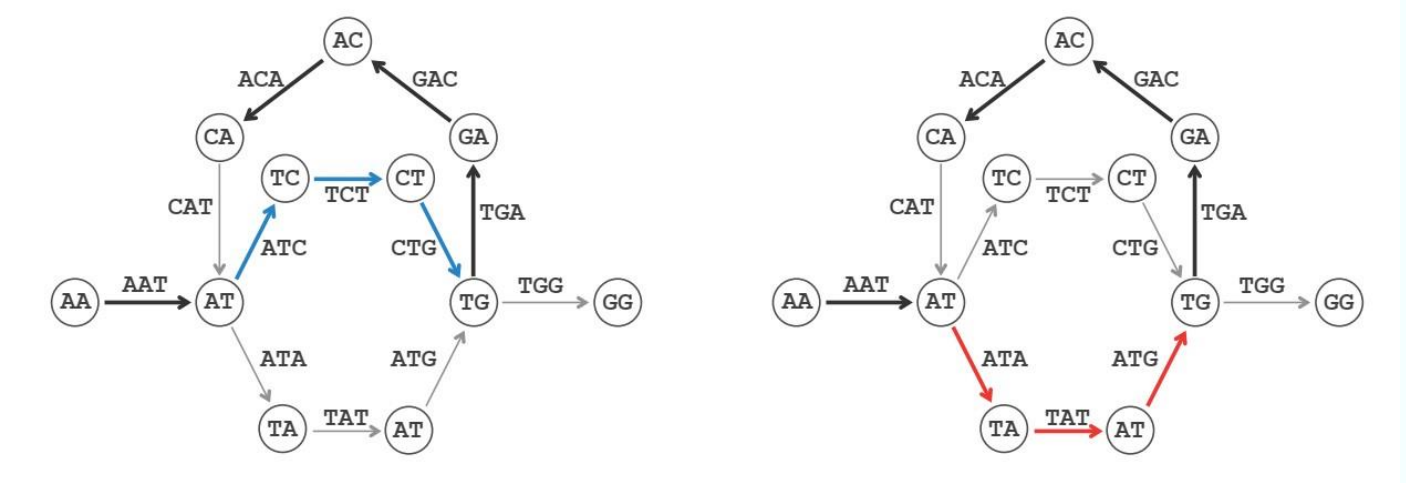
- Since these reads are separated by distance d in the genome, there must be a path of length $k + d$ in $DeBruijn_k(Reads)$ connecting the node at the beginning of the edge corresponding to $Read1$ with the node at the beginning of the edge corresponding to $Read_2$.
- If there is **only one path of length $k + d + 1$ connecting these nodes, or if all such paths spell out the same string**, then we can transform a read-pair formed by reads $Read_1$ and $Read_2$ into a virtual read of length $2 \cdot k + d$.
- After preprocessing the de Bruijn graph to produce long virtual reads, we can simply construct the de Bruijn graph from these long reads and use it for genome assembly.



Each path of length 3 + 1 + 1 = 5 (connecting node AAT to node CCA) spells out AATGCCA.

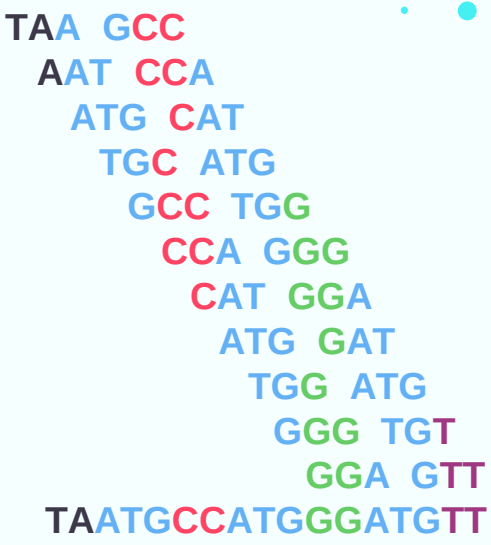
Transforming read-pairs into long virtual reads

- The previous assumption limits the application of the long virtual read approach to assembling read-pairs because highly repetitive genomic regions often contain multiple paths of the same length between two edges spelling different strings.



From composition to paired composition

- Given a string Text, a **(k,d)-mer** is a pair of k-mers in Text separated by distance d. We use the notation (Pattern$_1$|Pattern$_2$) to refer to a (k,d)-mer whose k-mers are Pattern$_1$ and Pattern$_2$.
- The (k,d)-mer composition of Text, denoted **PairedComposition$_{k,d}$(Text)**, is the collection of all (k,d)- mers in Text (including repeated (k,d)-mers).



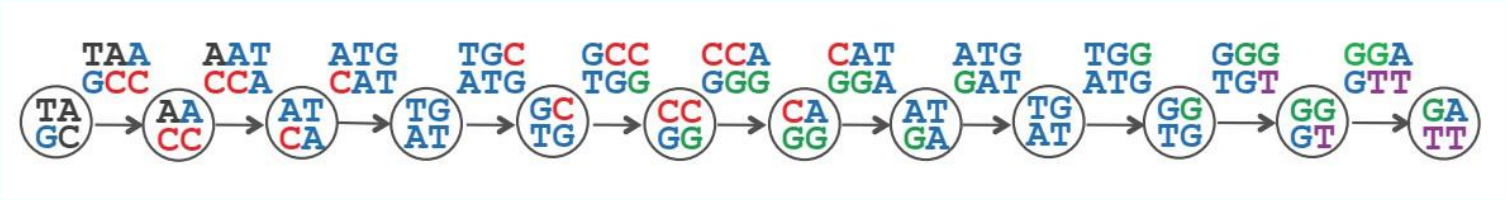
- Although TAATGCCATGGGATGTT and TAATGGGATGCCATGTT have the same 3-mer composition, they have different (3,1)-mer compositions.
- Thus, if we can generate the (3,1)-mer composition of these strings, then we will be able to distinguish between them.
- **But how can we reconstruct a string from its (k,d)-mer composition? And can we adapt the de Bruijn graph approach for this purpose?**

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

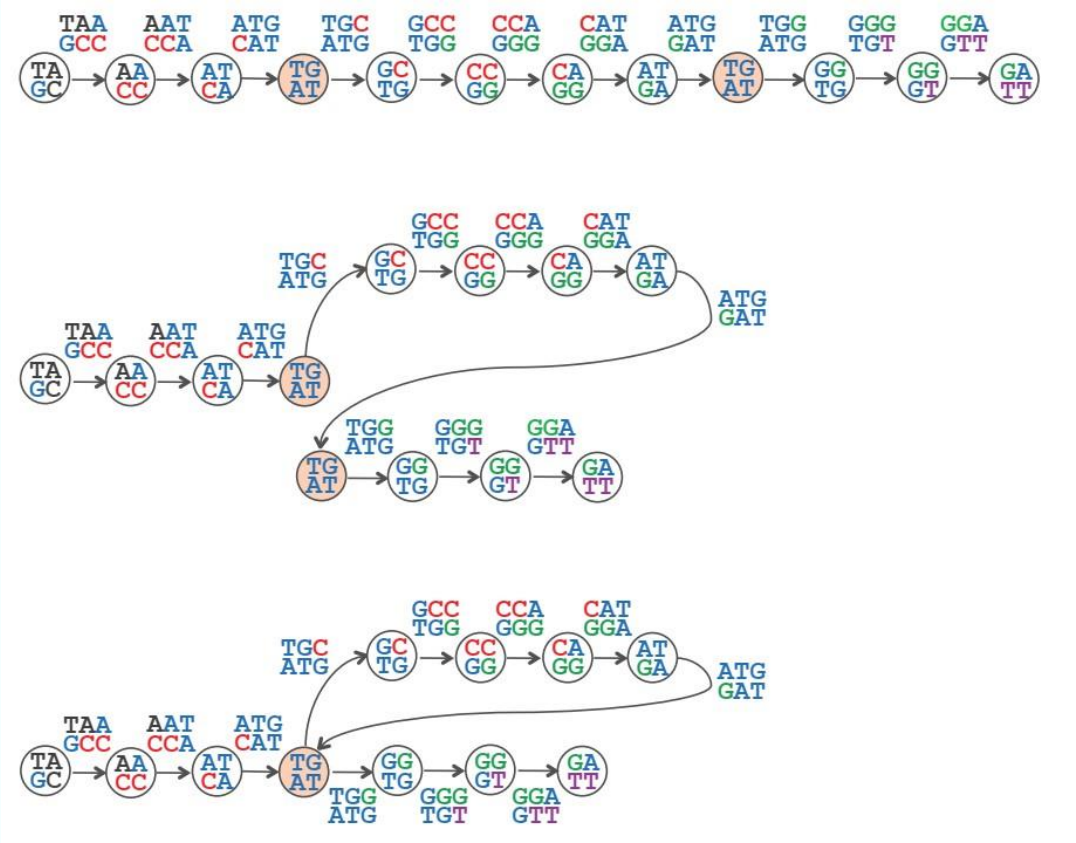
Paired de Bruijn graphs

- Given a (k, d)-mer $(a_1 \ldots a_k \mid b_1 \ldots b_k)$, we define its **prefix** as the (k – 1, d + 1)-mer $(a_1 \ldots a_{k-1} \mid b_1 \ldots b_{k-1})$, and its **suffix** as the (k - 1,d + 1)-mer $(a_2 \ldots a_k \mid b_2 \ldots b_k)$. For example, **Prefix((GAC|TCA)) = (GA|TC) and Suffix((GAC|TCA)) = (AC|CA).**

- Given a string Text, we construct a graph $\text{PathGraph}_{k,d}$(Text) that represents a path formed by |Text| - (k + d + k) + 1 edges corresponding to all (k, d)-mers in Text. We label edges in this path by (k, d)-mers and label the starting and ending nodes of an edge by its prefix and suffix, respectively. The figure below illustrates $\text{PathGraph}_{3,1}$(TAATGCCATGGGATGTT).

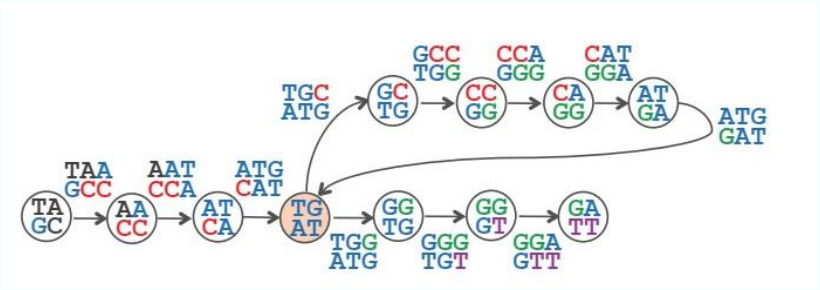


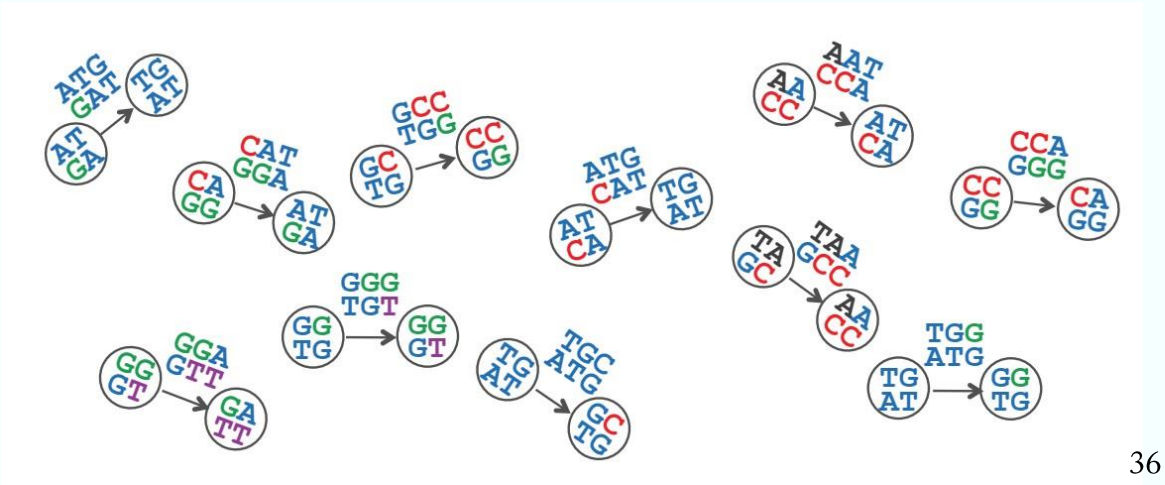
Paired de Bruijn graphs



How construct the paired de Bruijn graph from the (*k*,*d*)-mer composition?

- We represent each paired k-mer by an edge from the paired prefix to paired suffix. Then glue nodes with identical labels together.
- The result is the same graph as the one we constructed in the previous slide!

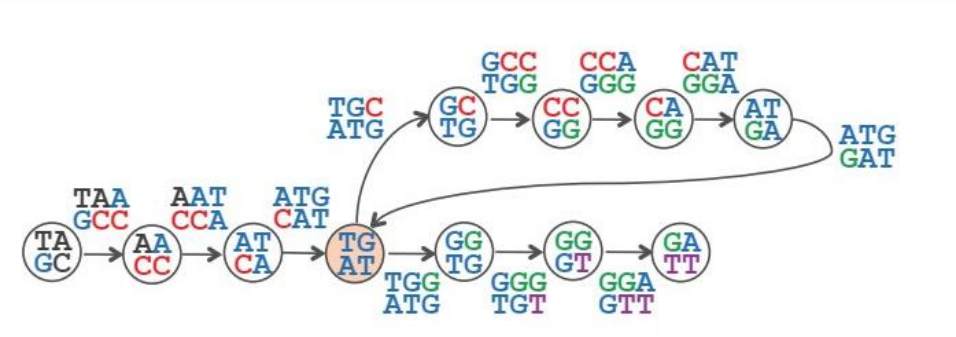




Which Graph Represent a Better Assembly?

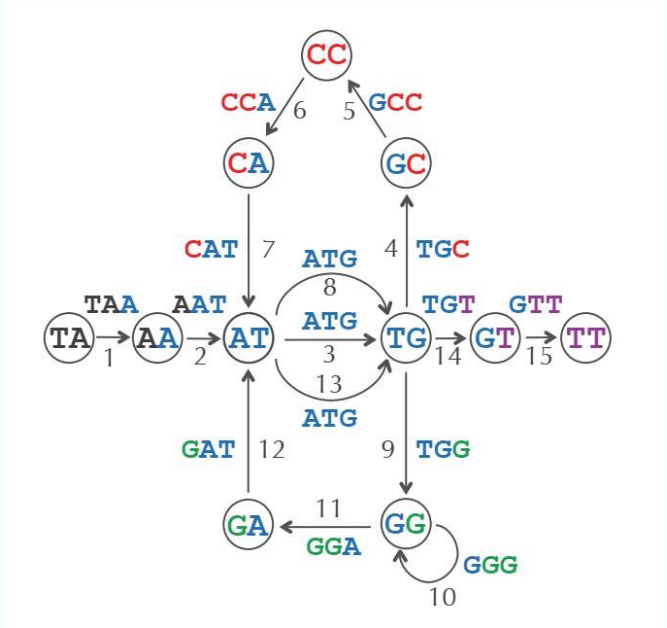
The paired De Bruijn graph is simpler than the De Bruijn Graph and there is only one Eulerian path it paired De Bruijn graph

Paired De Bruijn Graph



TAATGCCATGGGATGTT

De Bruijn Graph



TAATGCCATGGGATGTT
TAATGGGATGCCATGTT

Some 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!

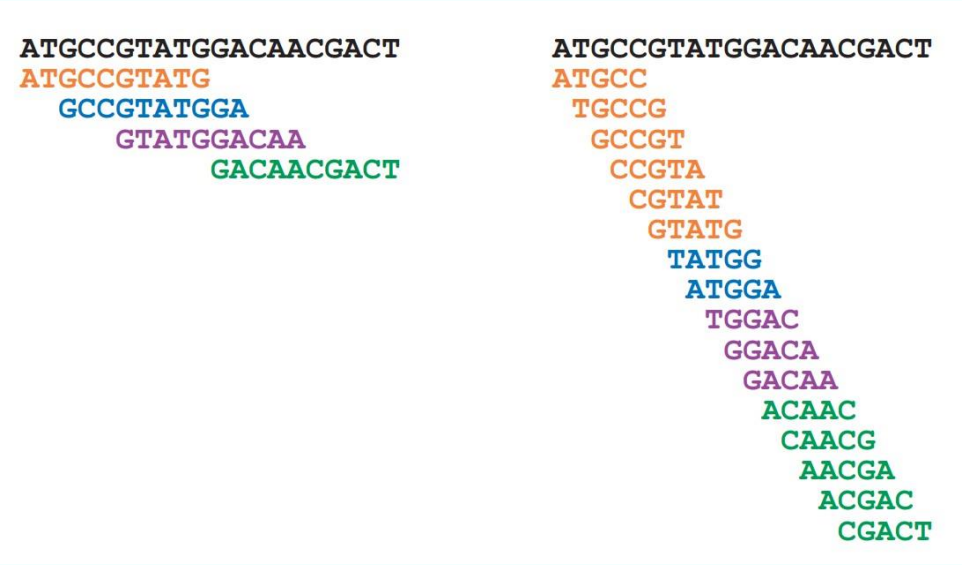
Some Unrealistic Assumptions!

- **Imperfect** coverage of genome by reads (every k-mer from the genome is represented by a read).
- Reads are **error-prone**.
- Multiplicities of k-mers are **unknown**.
- Distances between reads within read-pairs are **inexact**.
- **Etc., etc., etc.**

First Unrealistic Assumption: Perfect Coverage

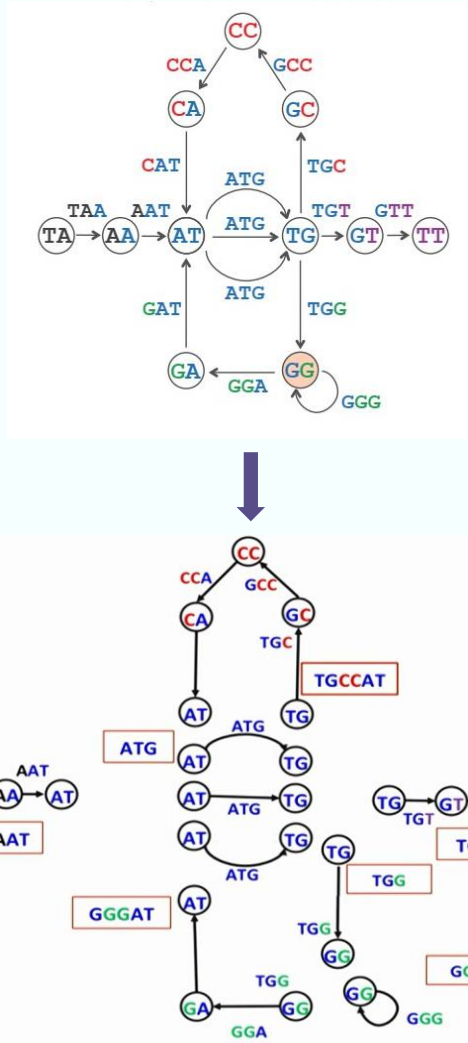
- We have taken for granted that a sequencing machine can generate all *k*-mers present in the genome, but this assumption of "perfect coverage" does not hold in practice.
- For example, the popular Illumina sequencing technology generates reads that are approximately 300 nucleotides long, but this technology still misses many 300-mers present in the genome, and nearly all the reads that it does generate have sequencing errors.

- **Read Breaking** approach, in which we break reads into shorter *k*-mers, is used by many modern assemblers.
- Read breaking must deal with a practical trade-off.



Splitting the genome into contigs

- Even after read breaking, most assemblies still have gaps in *k*-mer coverage, causing the de Bruijn graph to have missing edges, and so the search for an Eulerian path fails.
- Biologists often settle on assembling **contigs** (long, contiguous segments of the genome) rather than entire chromosomes.
- Fortunately, we can derive contigs from the de Bruijn graph, by breaking the de Bruijn graph into maximal non-branching paths
- In our example there is nine maximal non-branching paths representing contigs TAAT, TGTT, TGCCAT,ATG, ATG, ATG, TGG, GGG, and GGAT.

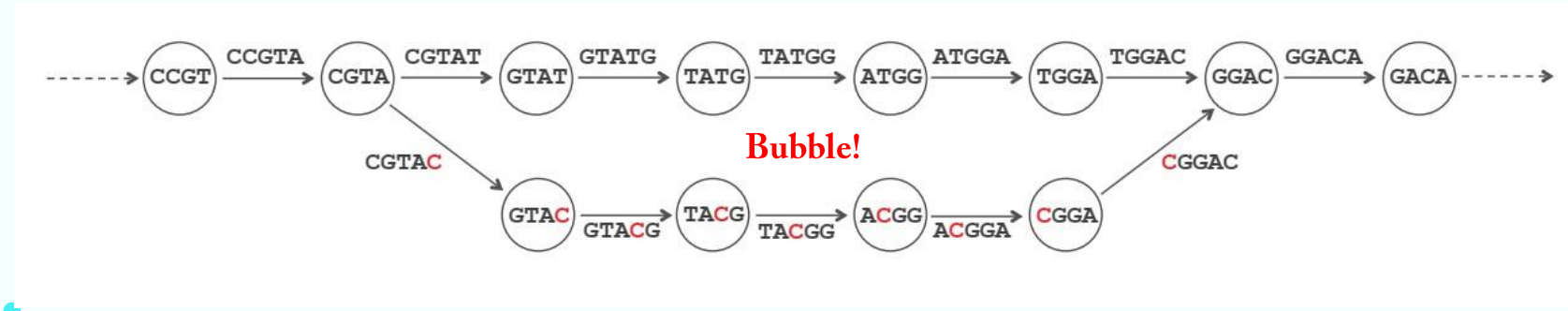


Second Unrealistic Assumption: Error-free Reads

- Adding the single erroneous read CGTACGGACA (with a single error that misreads T as **C**) to the set of "broken" 5-mer reads results in erroneous 5-mers CGTAC, GTACG, TACGG, ACGGA, and CGGAC after read breaking.

- These 5-mers result in an erroneous path from node CGTA to node GGAC in the de Bruijn graph below, meaning that if the correct read CGTATGGACA is generated as well, then we will have two paths connecting CGTA to GGAC in the de Bruijn graph.

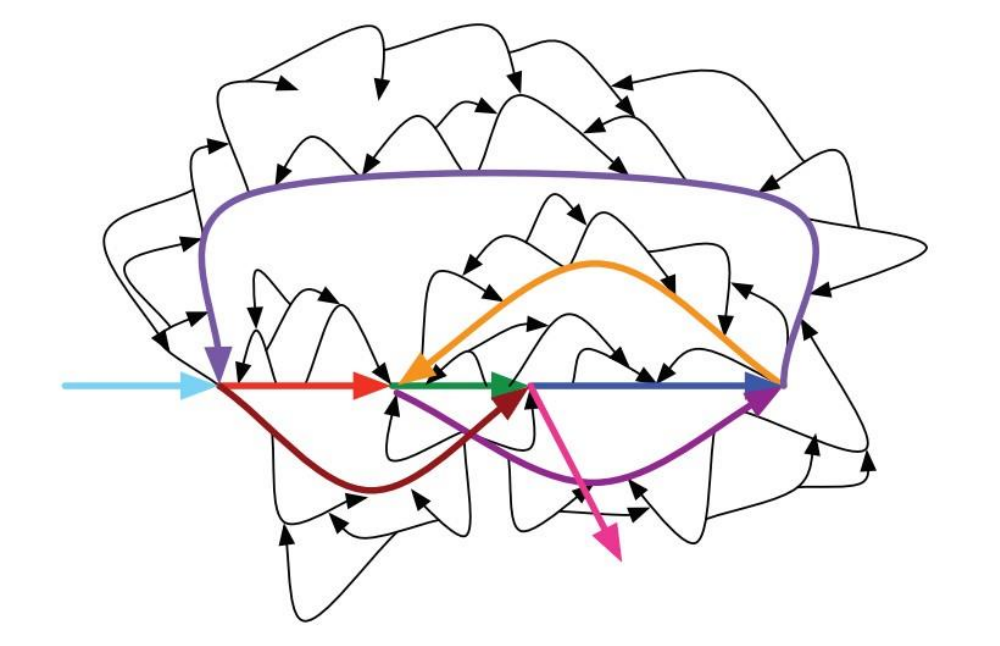
- This structure is called a **bubble**, which we define as two short disjoint paths (e.g., shorter than some threshold length) connecting the same pair of nodes in the de Bruijn graph.



Bubble Explosion

Where are the Correct Edges of the de Bruijn Graph?

- Existing assemblers remove bubbles from de Bruijn graphs. The practical challenge is that, since nearly all reads have errors, de Bruijn graphs have millions of bubbles



RESOURCES

- Compeau, Phillip, and Pave Pevzner. "Bioinformatics algorithms: an active learning approach, Chapter 3: How do we assemble the genome?