Sequence Assembly

Fall 2016

BMI/CS 576

www.biostat.wisc.edu/bmi576/

Irene Ong

Irene.ong@wisc.edu

Outline

- What Is Genome Sequencing?
- Exploding Newspapers
- The String Reconstruction Problem
- String Reconstruction as a Hamiltonian Path Problem
- String Reconstruction as an Eulerian Path Problem
- Similar Problems with Different Fates
- De Bruijn Graphs
- Euler's Theorem
- Assembling Read-Pairs
- De Bruijn Graphs Face Harsh Realities of Assembly

Representing a Genome as a Path

Composition_3(TAATGCCATGGGATGTT)=

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

Representing a Genome as a Path

Composition_3(TAATGCCATGGGATGTT)=

Representing a Genome as a Path

Composition_3(TAATGCCATGGGATGTT)=

Can we construct this genome path without knowing the genome TAATGCCATGGGATGTT, only from its composition?

Representing a Genome as a Path

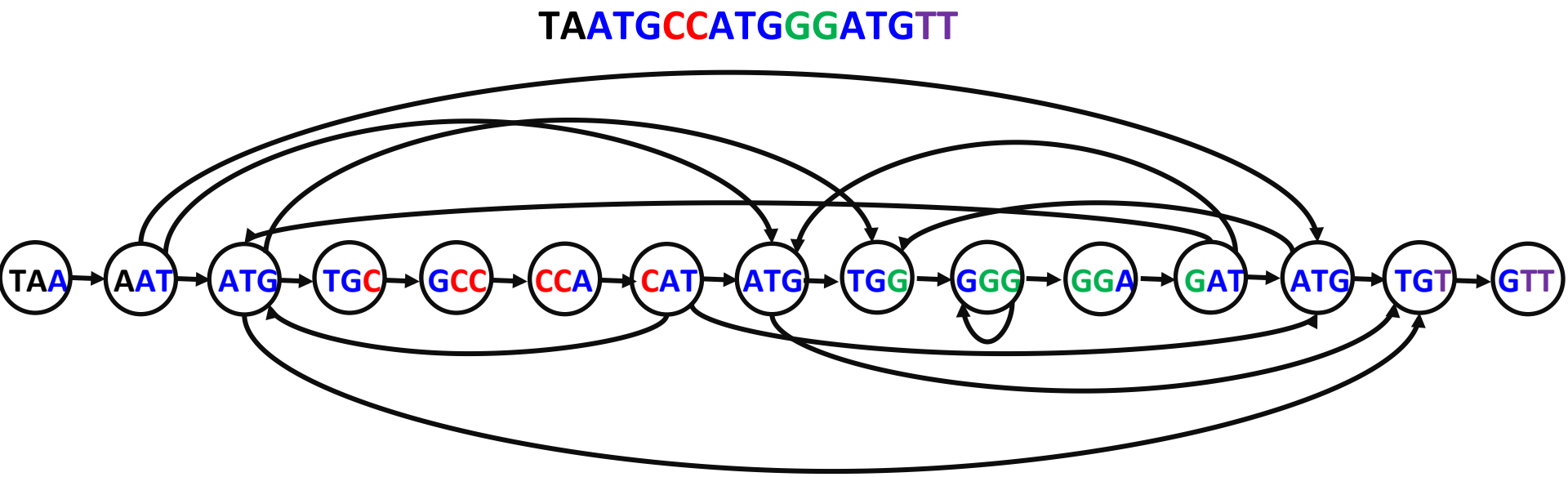
$\text{Composition}_3(\text{TAATGCCATGGGATGTT}) =$

Can we construct this genome path without knowing the genome TAATGCCATGGGATGTT, only from its composition?

Yes. We simply need to connect k-mer_1 with k-mer_2 if suffix(k-mer_1)=prefix(k-mer_2).

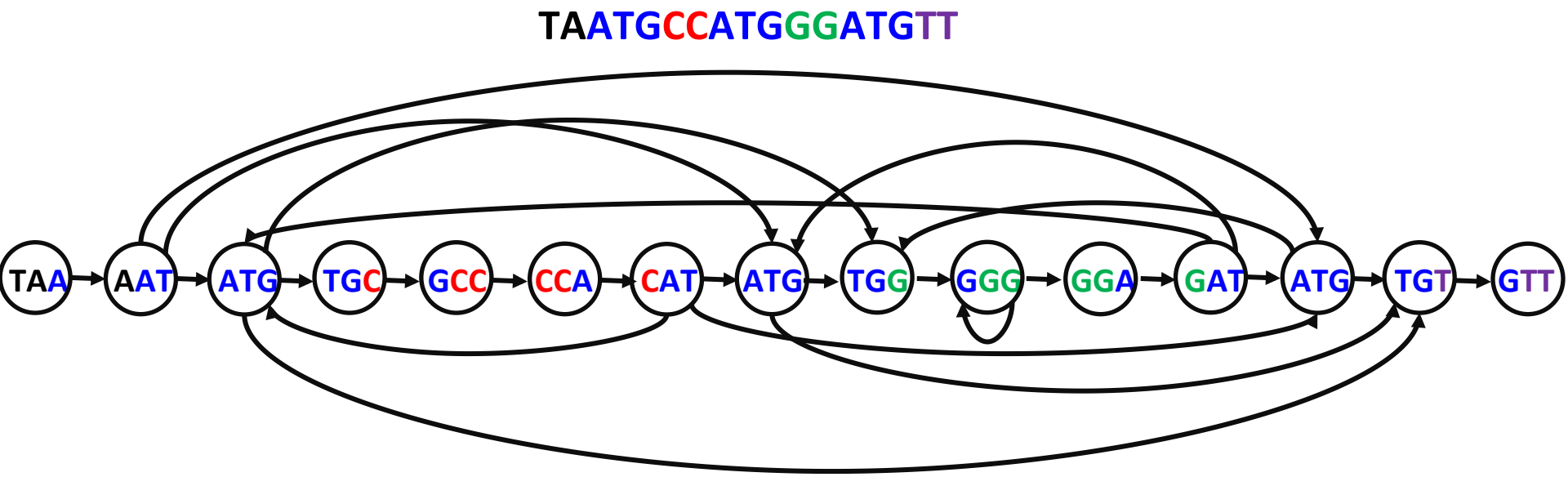
E.g. TAA → AAT

A Path Turns into a Graph



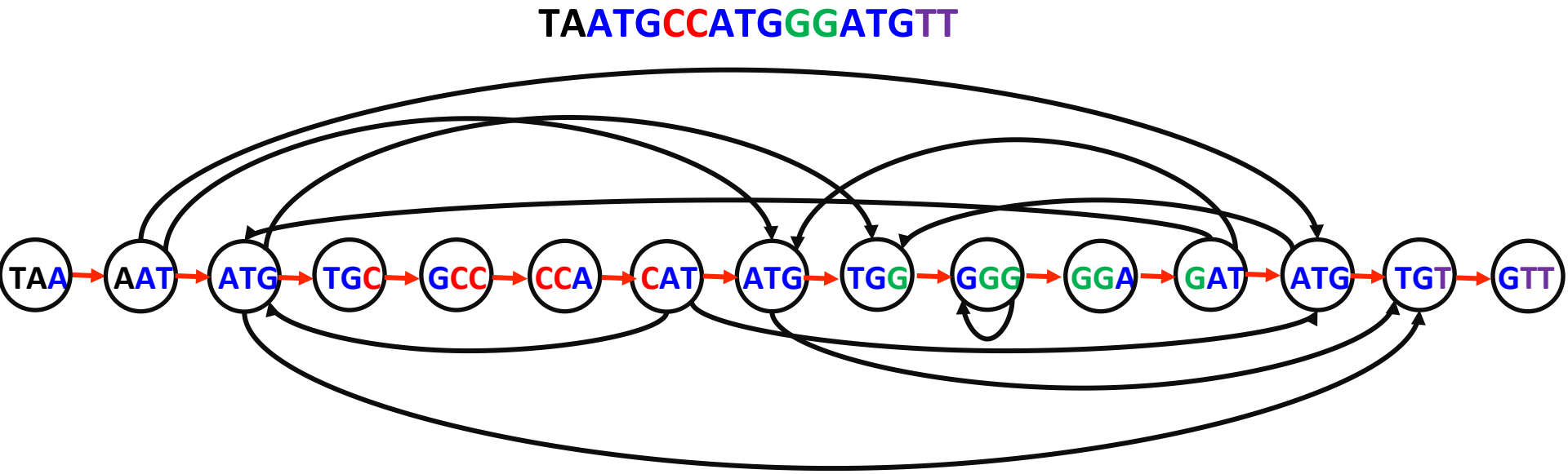
Yes. We simply need to connect k-mer$_1$ with k-mer$_2$ if suffix(k-mer$_1$)=prefix(k-mer$_2$).
E.g. TAA → AAT

A Path Turns into a Graph



Can we still find the genome path in this graph?

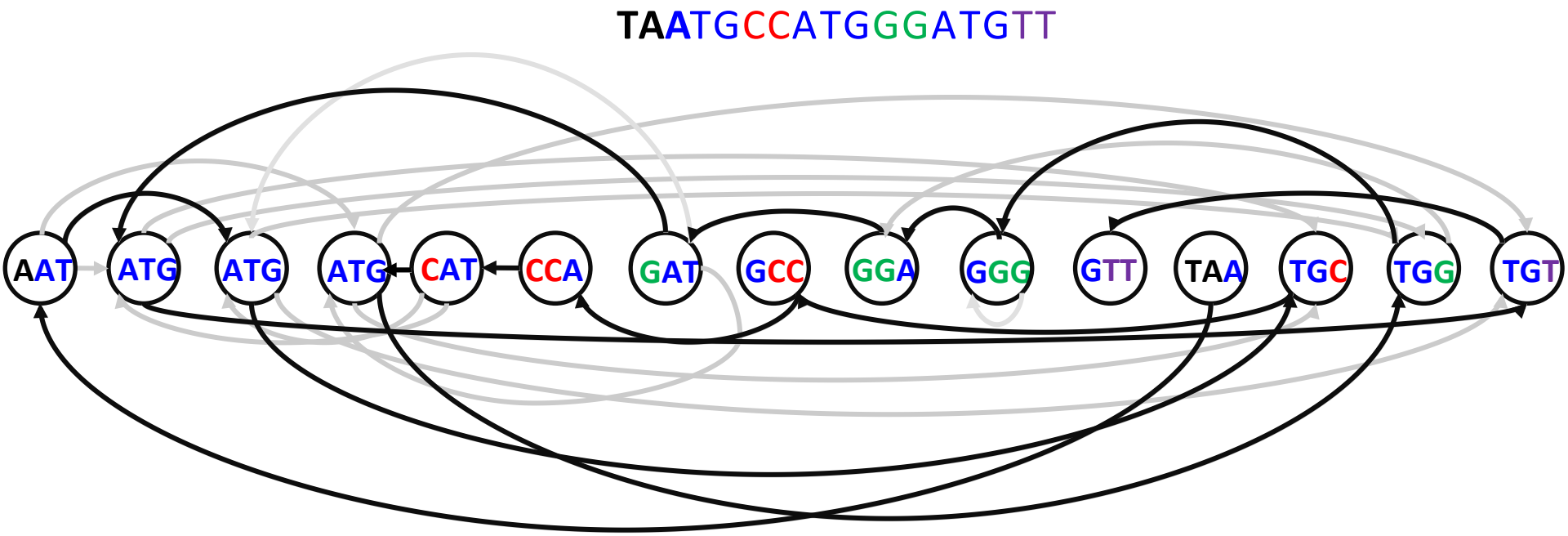
A Path Turns into a Graph



Can we still find the genome path in this graph?

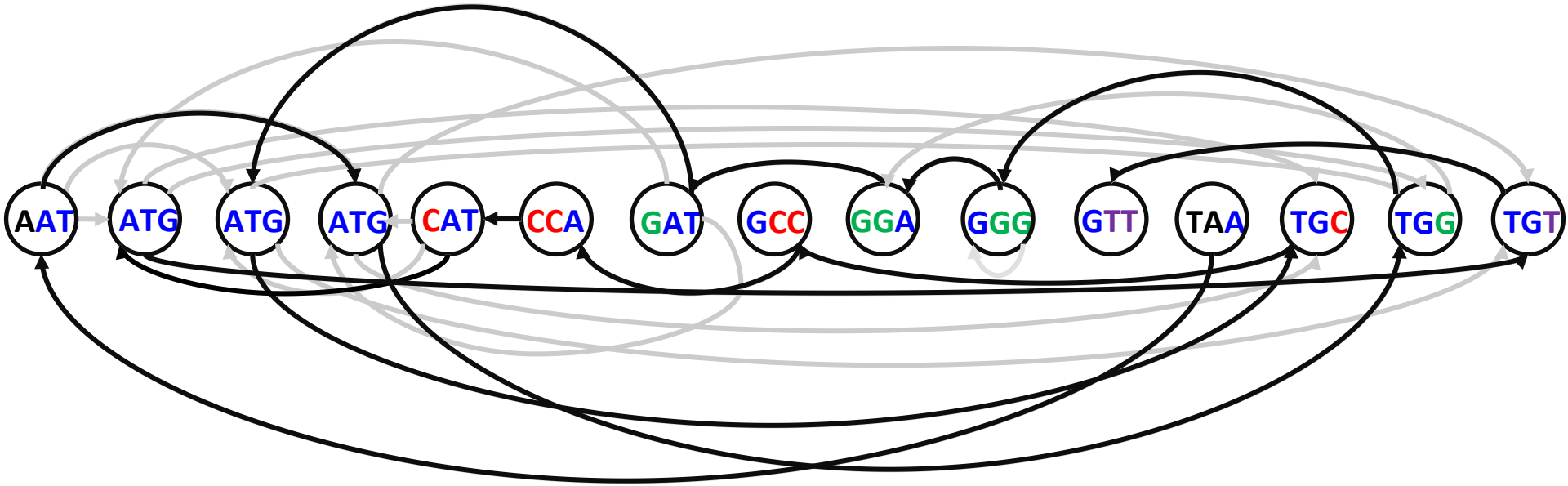
Where Is the Genomic Path?

A **Hamiltonian path:** a path that visits each node in a graph exactly once.

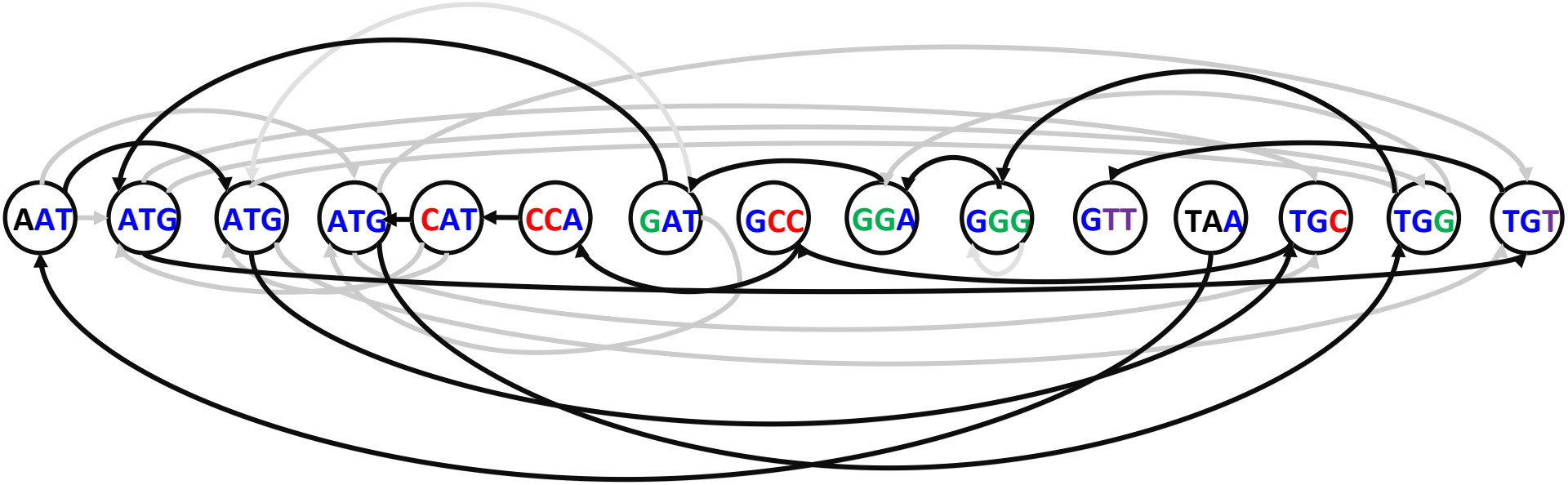


What are we trying to find in this graph?

TAATGGGATGCCATGTT



TAATGCCATGGGATGTT



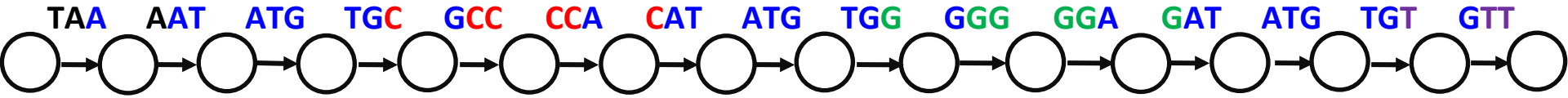
Outline

- What Is Genome Sequencing?
- Exploding Newspapers
- The String Reconstruction Problem
- String Reconstruction as a Hamiltonian Path Problem
- String Reconstruction as an Eulerian Path Problem
- Similar Problems with Different Fates
- De Bruijn Graphs
- Euler's Theorem
- Assembling Read-Pairs
- De Bruijn Graphs Face Harsh Realities of Assembly

A Slightly Different Path

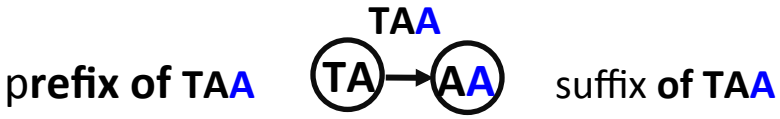
TAATGCCATGGGATGTT

Instead of representing 3-mers as nodes



Let's represent 3-mers as edges

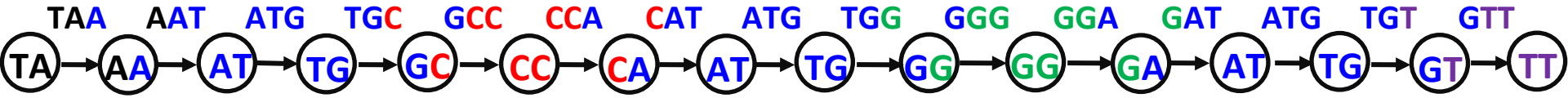
How do we label the starting and ending nodes of an edge?



Labeling Nodes in the New Path

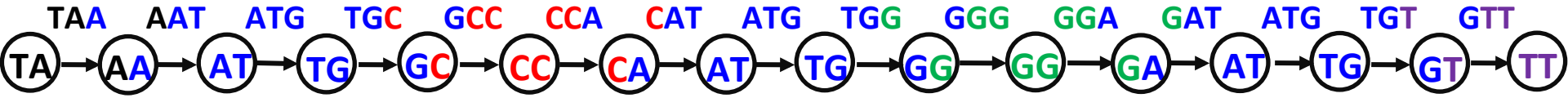
TAATGCCATGGGATGTT

3-mers as nodes



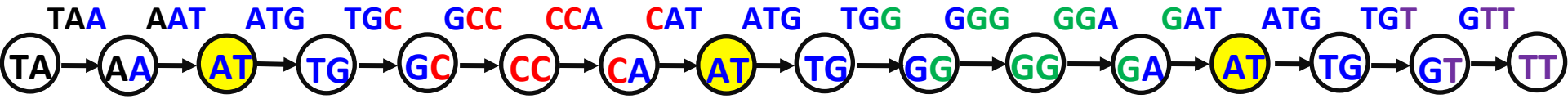
3-mers as edges and 2-mers as nodes

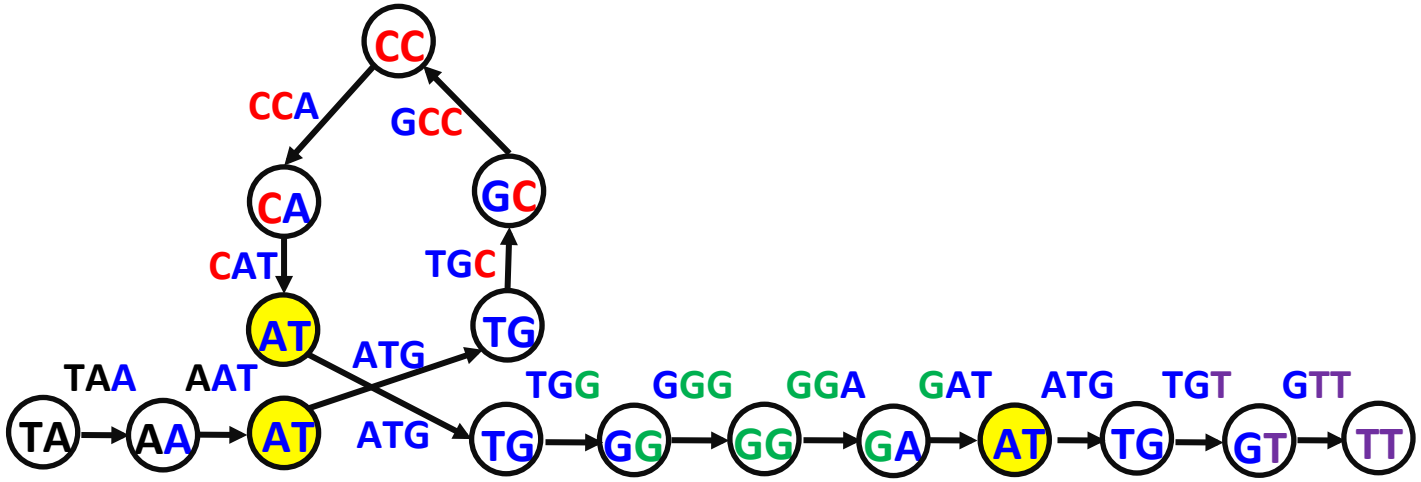
Labeling Nodes in the New Path



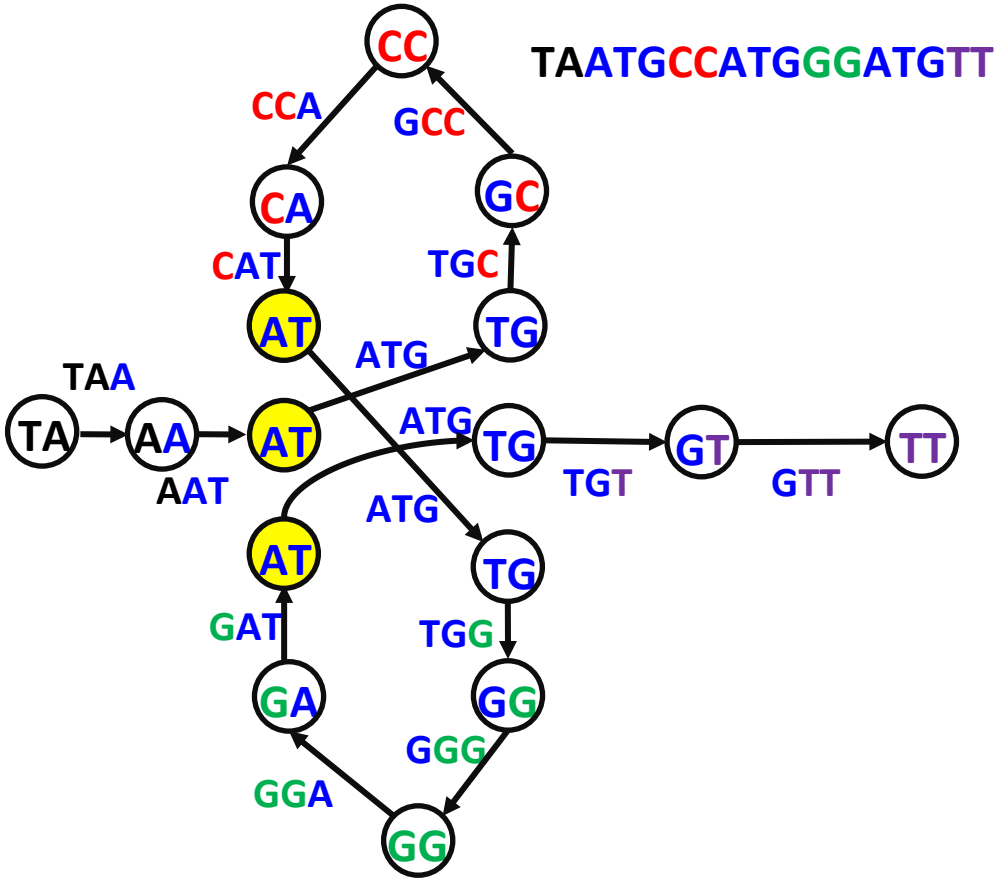
3-mers as edges and 2-mers as nodes

Gluing Identically Labeled Nodes

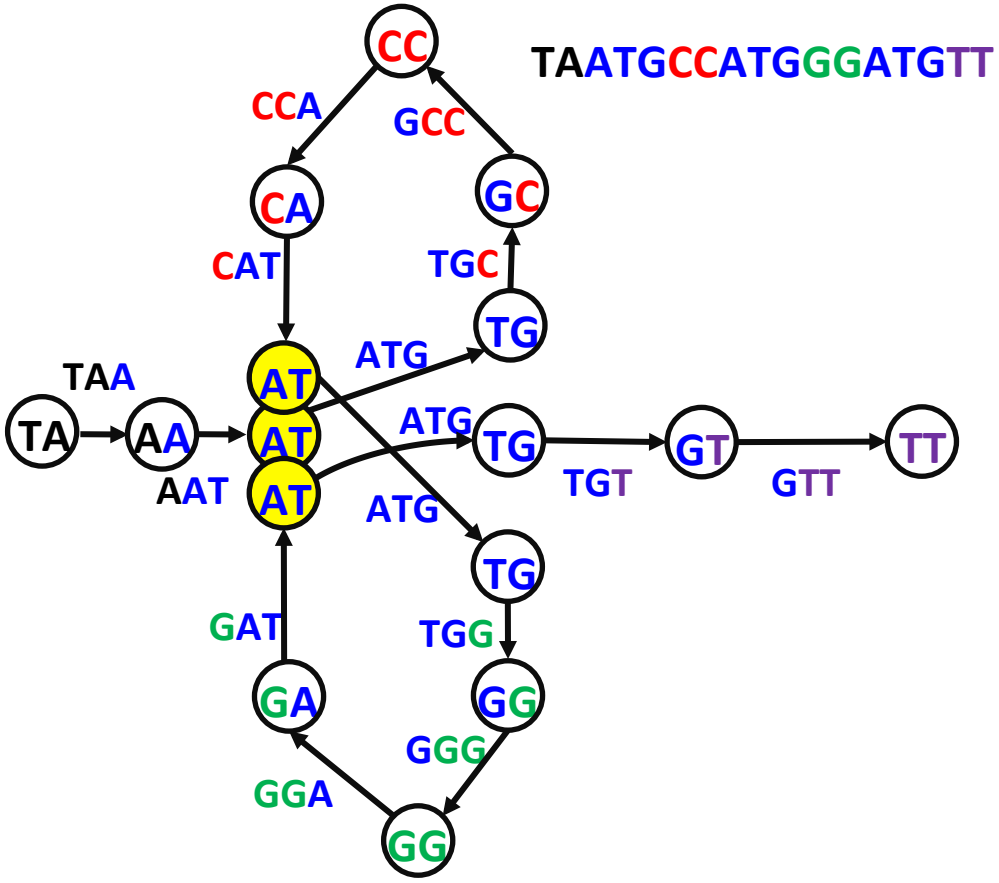




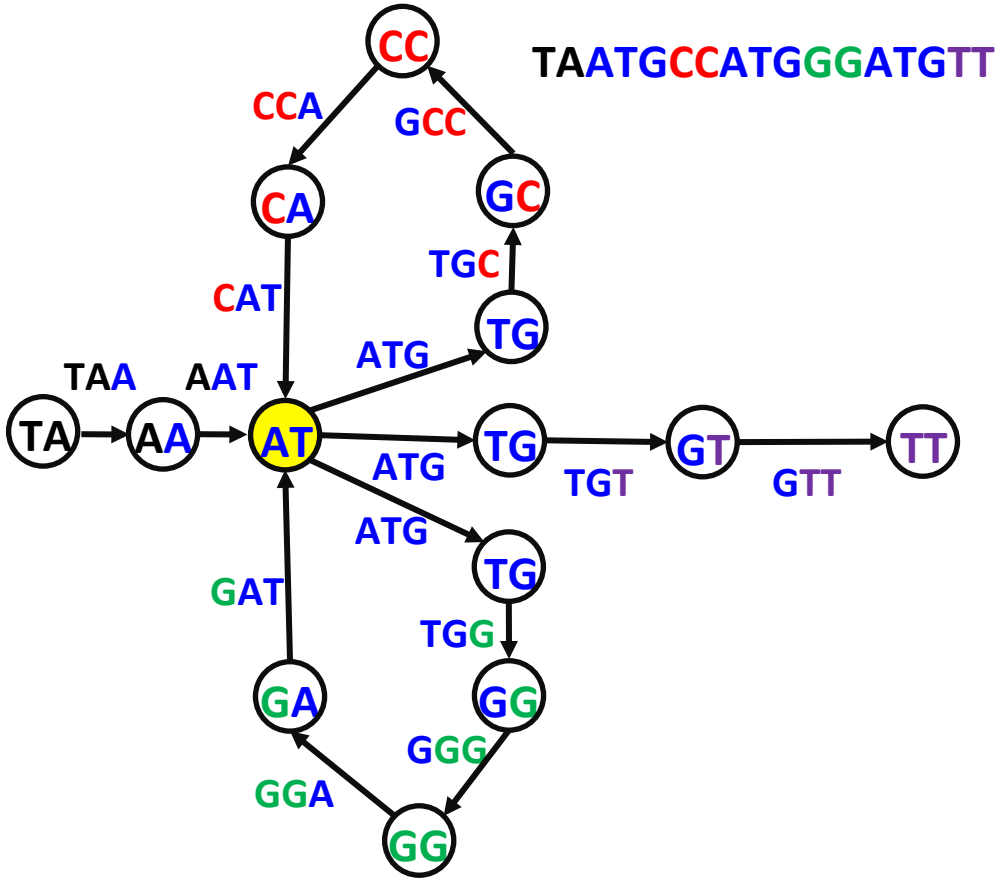
Gluing Identically Labeled Nodes



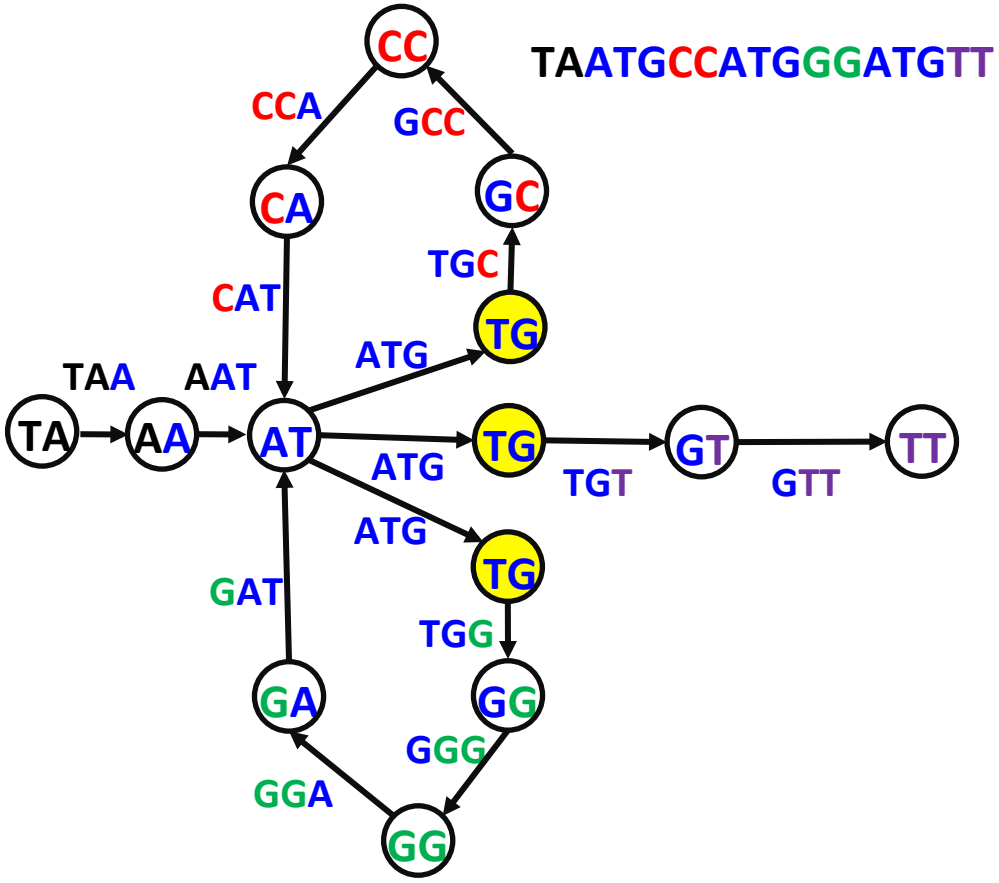
Gluing Identically Labeled Nodes



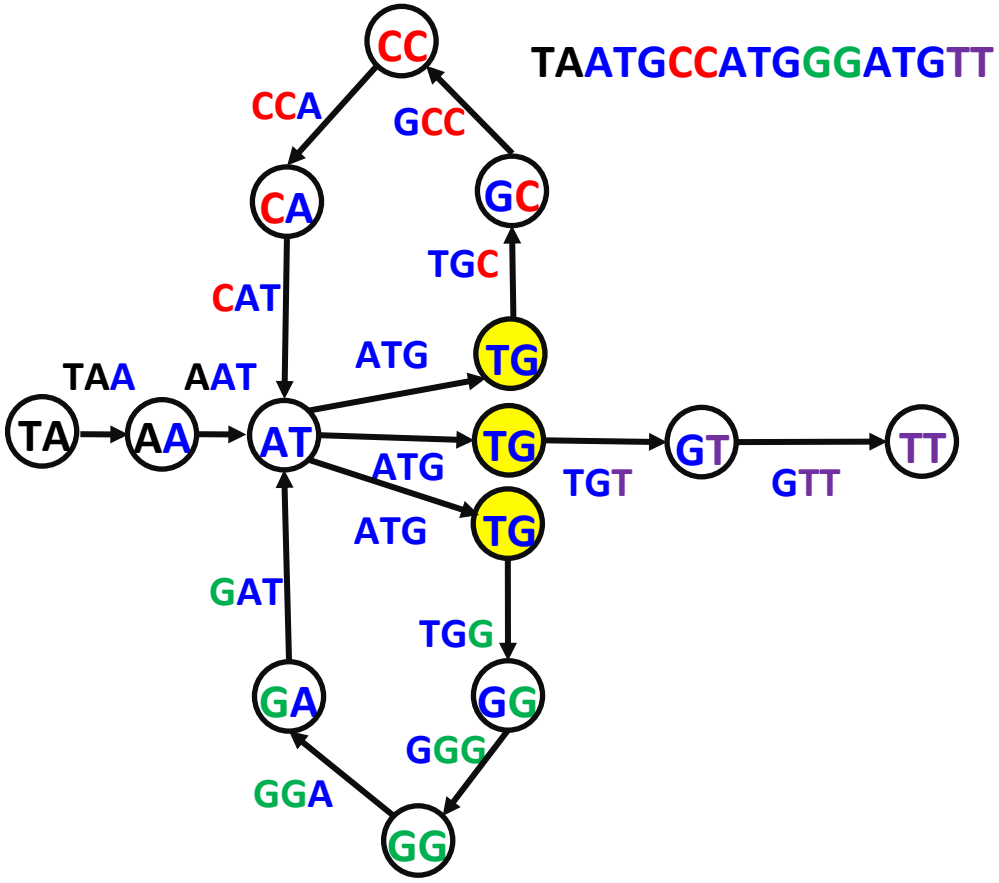
Gluing Identically Labeled Nodes



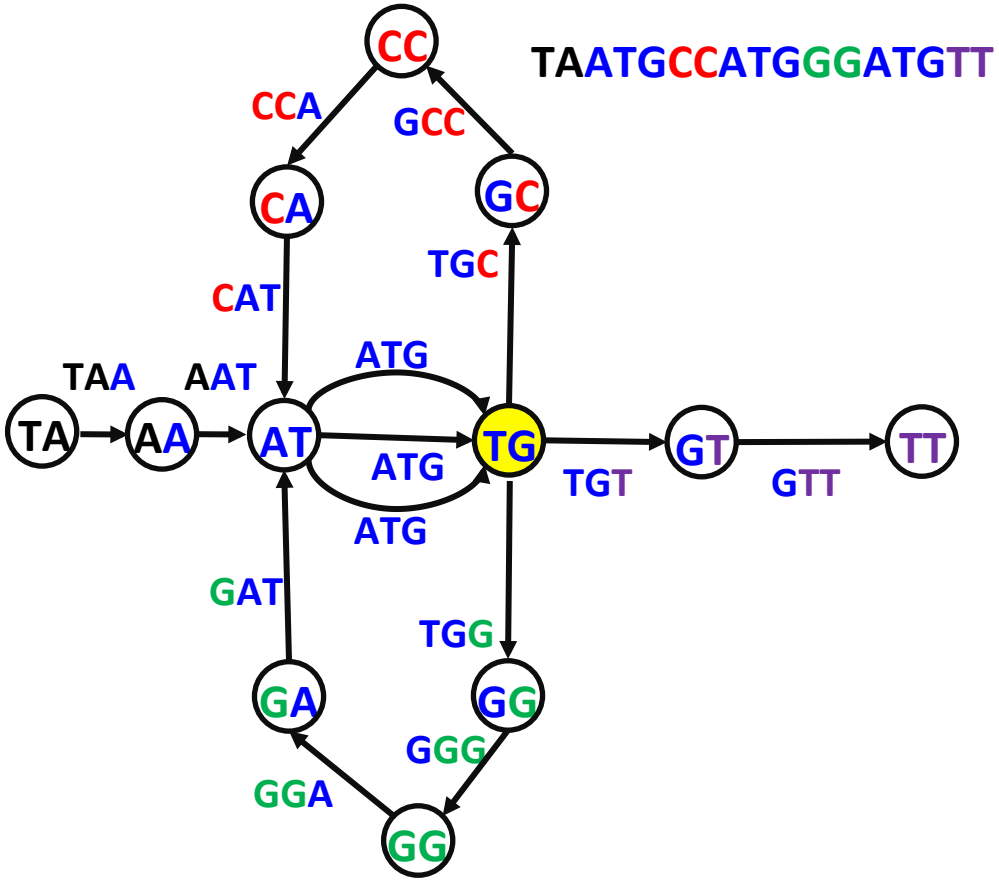
Gluing Identically Labeled Nodes



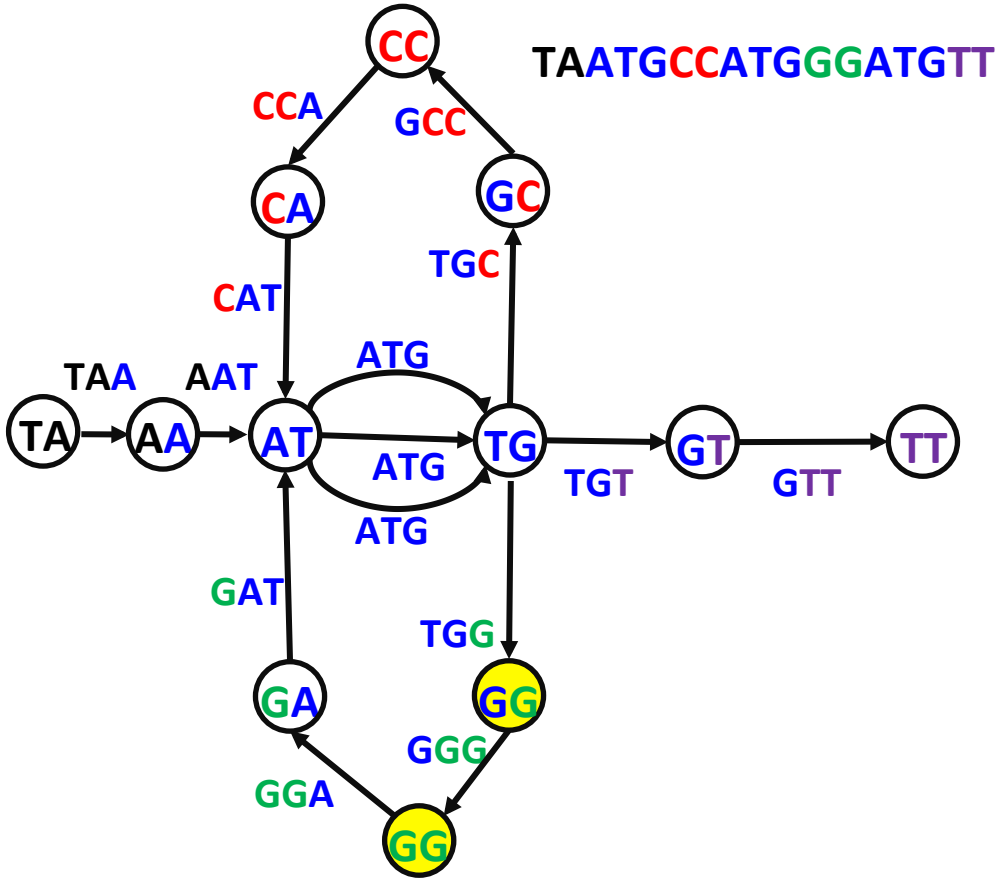
Gluing Identically Labeled Nodes



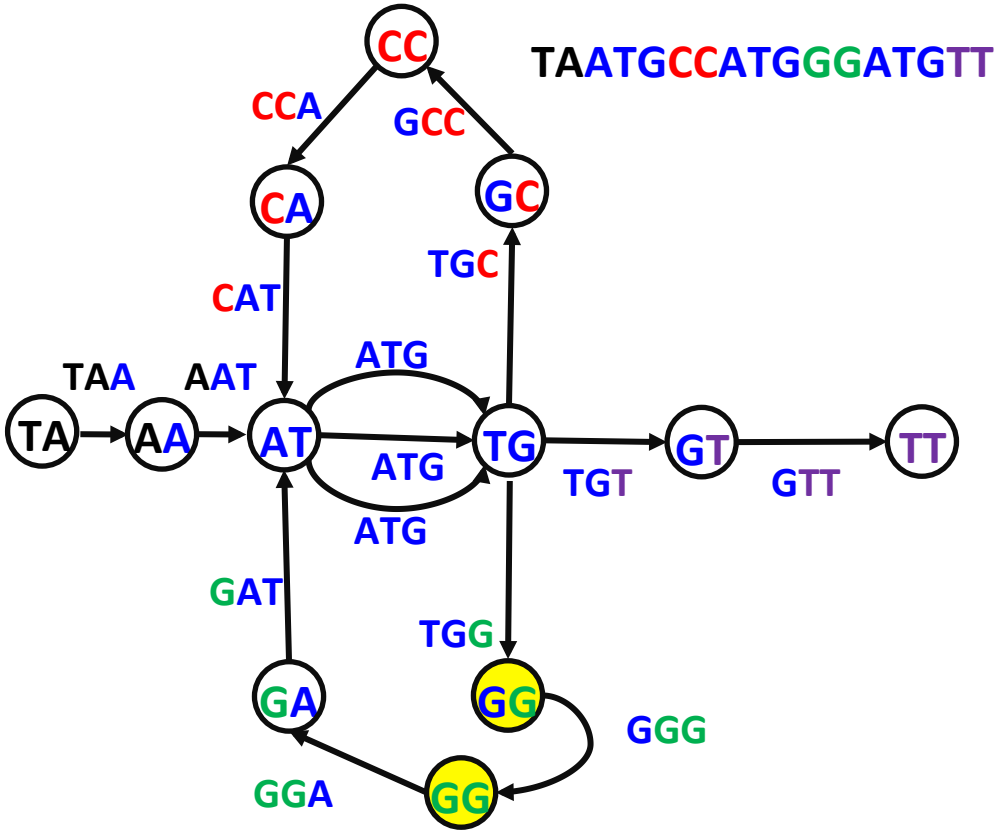
Gluing Identically Labeled Nodes



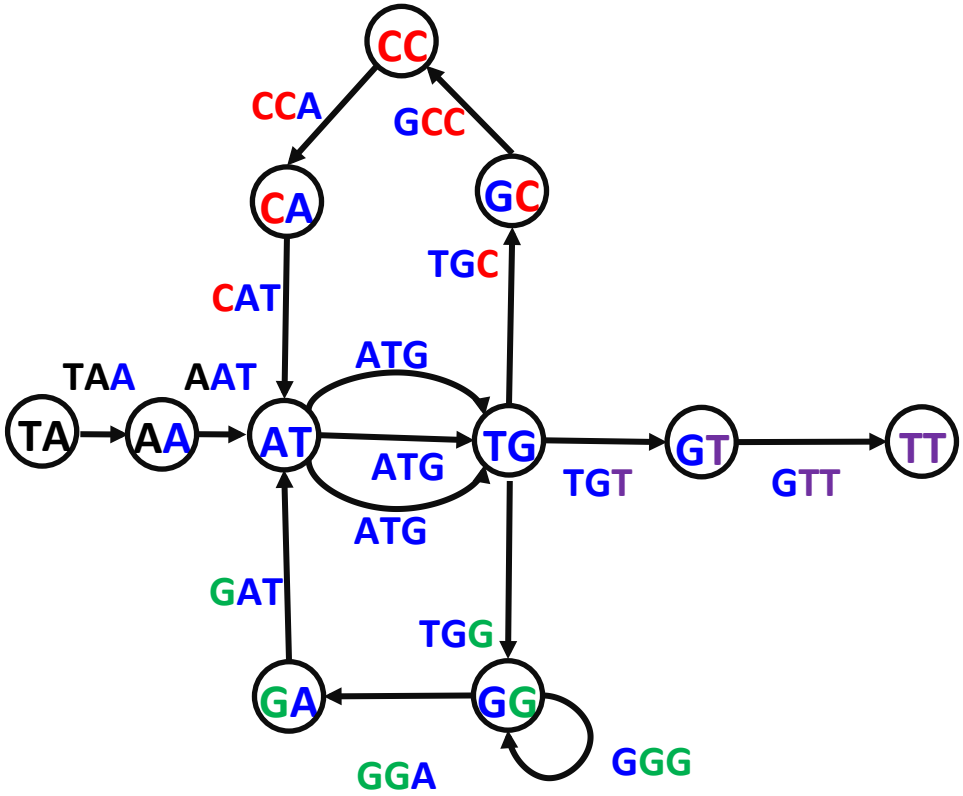
Gluing Identically Labeled Nodes



Gluing Identically Labeled Nodes



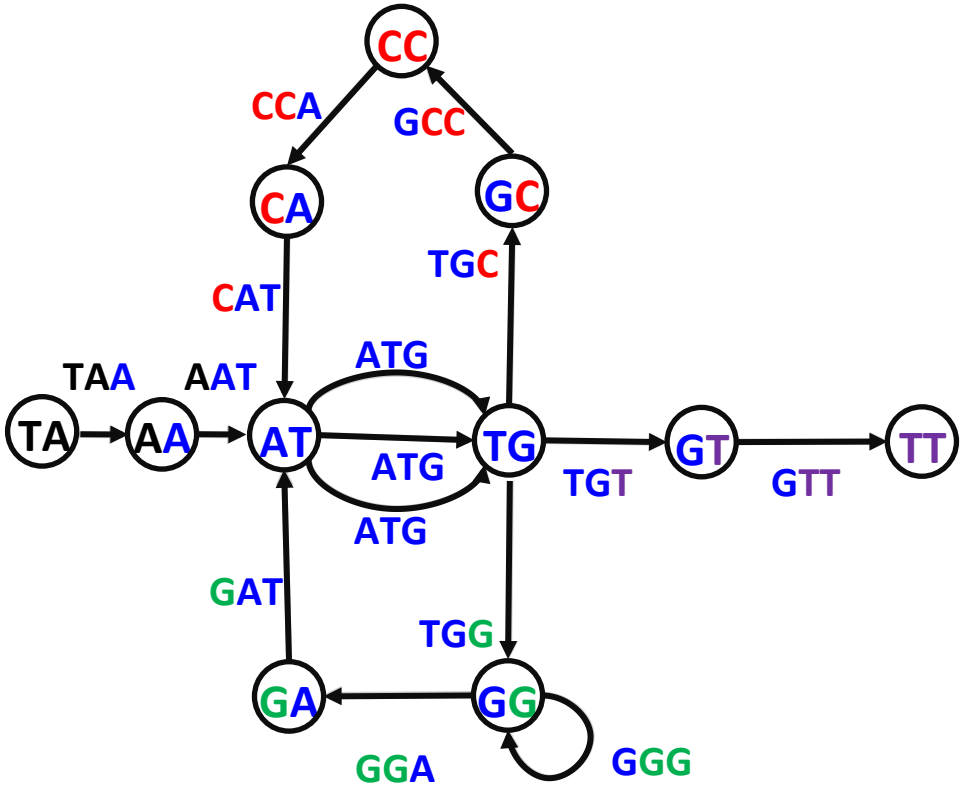
De Bruijn Graph of TAATGCCATGGGATGTT



Where is the Genome hiding in this graph?

It Was Always There!

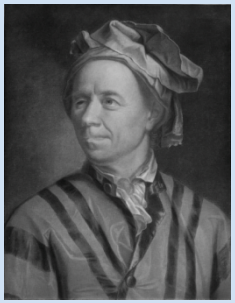
TAATGCCATGGGATGTT



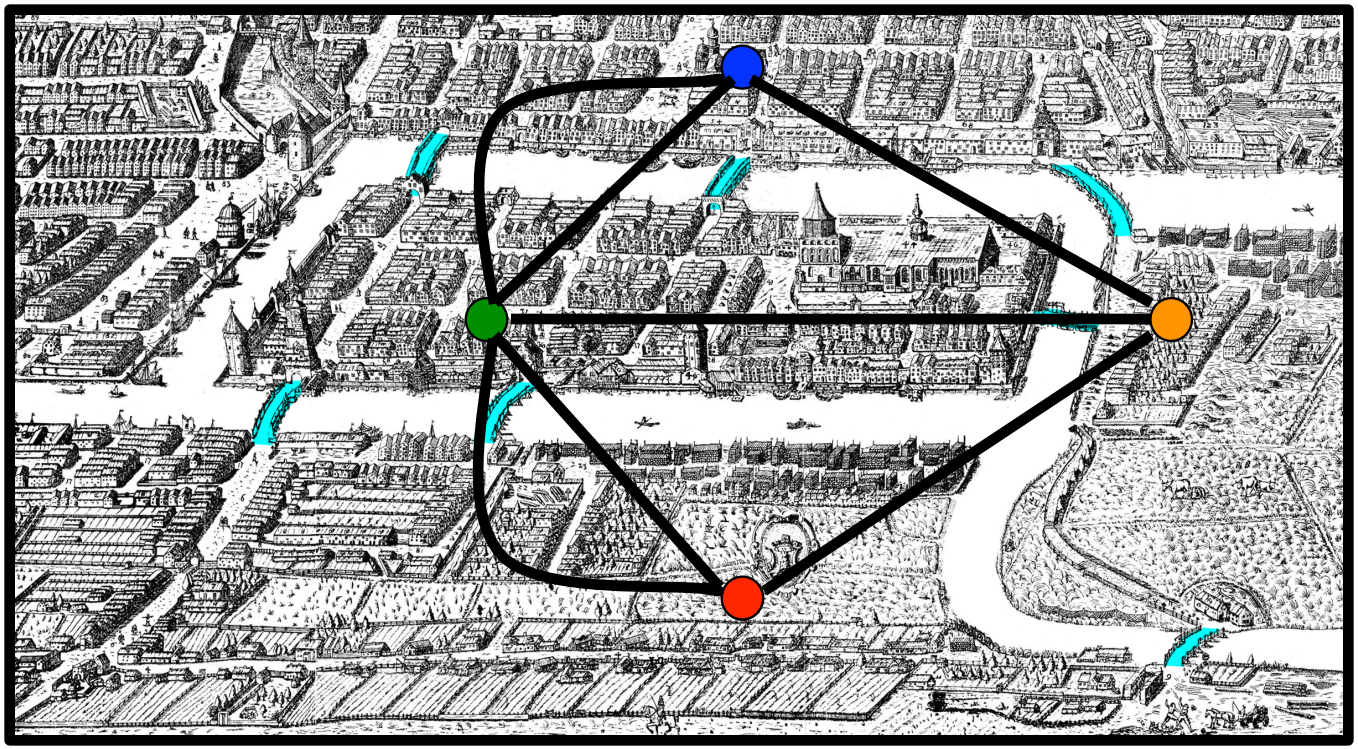
An Eulerian **path** in a graph is a path that visits each edge exactly once.

Eulerian Path Problem

Eulerian Path Problem. Find an Eulerian path in a graph.



- Input. A graph.
- Output. A path visiting every edge in the graph exactly once.

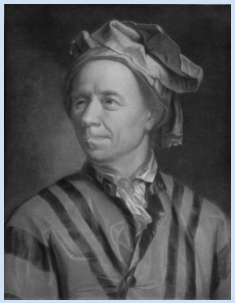


Outline

- What Is Genome Sequencing?
- Exploding Newspapers
- The String Reconstruction Problem
- String Reconstruction as a Hamiltonian Path Problem
- String Reconstruction as an Eulerian Path Problem
- Similar Problems with Different Fates
- De Bruijn Graphs
- Euler's Theorem
- Assembling Read-Pairs
- De Bruijn Graphs Face Harsh Realities of Assembly

Eulerian Versus Hamiltonian Paths

Eulerian Path Problem. Find an Eulerian path in a graph.

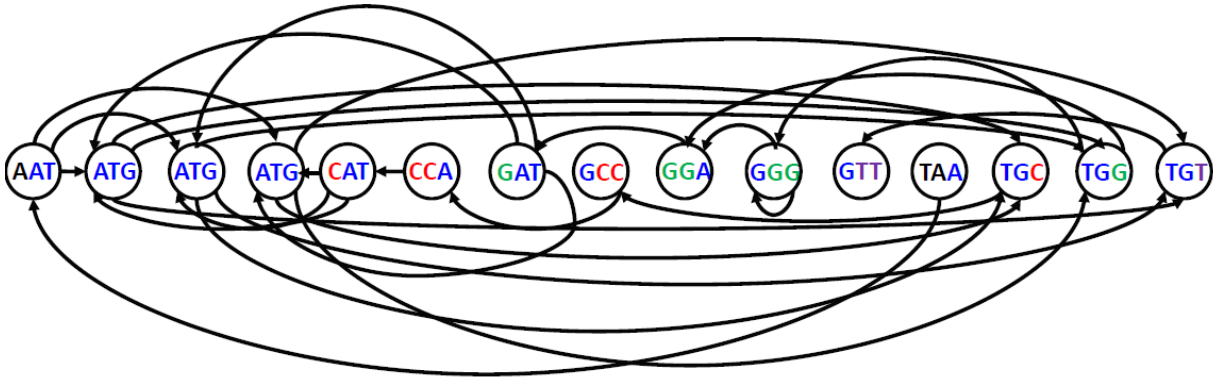


- Input. A graph.
- Output. A path visiting every edge in the graph exactly once.

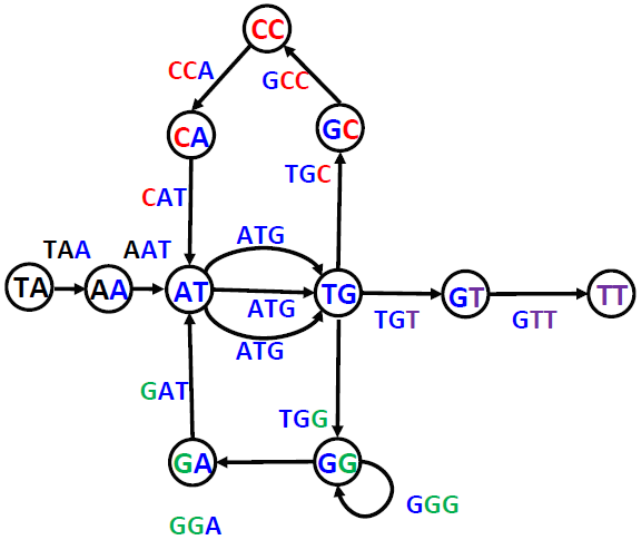
Hamiltonian Path Problem. Find a Hamiltonian path in a graph.

- Input. A graph.
- Output. A path visiting every node in the graph exactly once.

What Problem Would You Prefer to Solve?

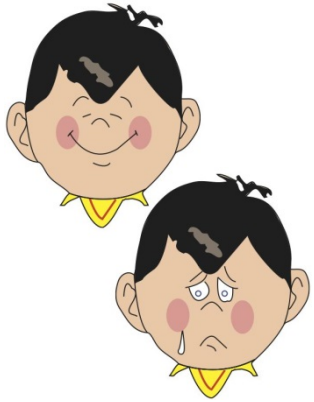


Hamiltonian Path Problem



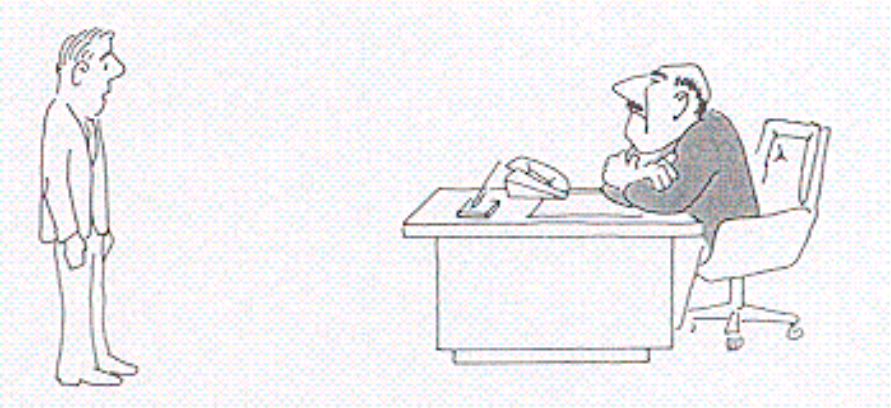
Eulerian Path Problem

While Euler solved the Eulerian Path Problem (even for a city with a million bridges), nobody has developed a fast algorithm for the Hamiltonian Path Problem yet.



NP-Complete Problems

- The Hamiltonian Path Problem belongs to a collection containing thousands of computational problems for which no fast algorithms are known.



"I can't find an efficient algorithm, I must not be working hard enough."

From Garey and Johnson. Computers and Intractability. 1979

Change of Attitude

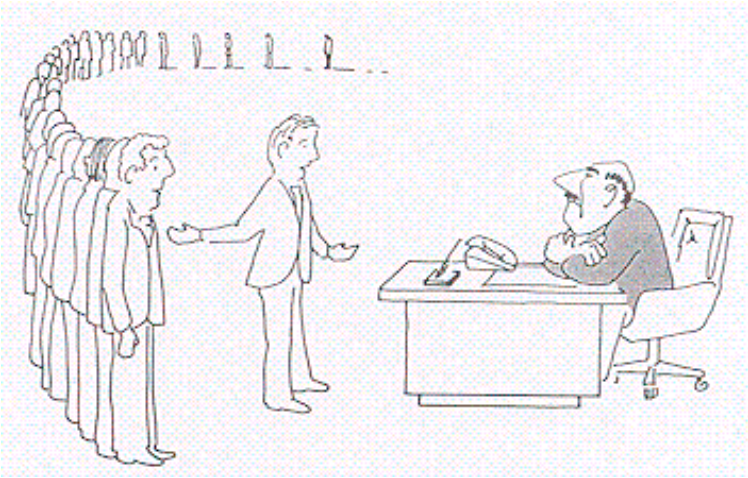
That would be an excellent argument, but the question of whether or not NP-Complete problems can be solved efficiently is one of seven **Millennium Problems** in mathematics.

"I can't find an efficient algorithm, because no such algorithm is possible."

From Garey and Johnson. Computers and Intractability. 1979

The Modern State of Affairs

NP-Complete problems are all equivalent: find an efficient solution to one, and you have an efficient solution to them all.



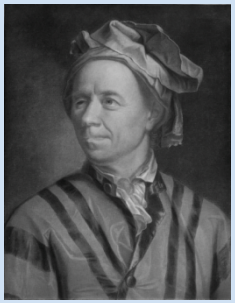
"I can't find an efficient algorithm, but neither can all these famous people."

Outline

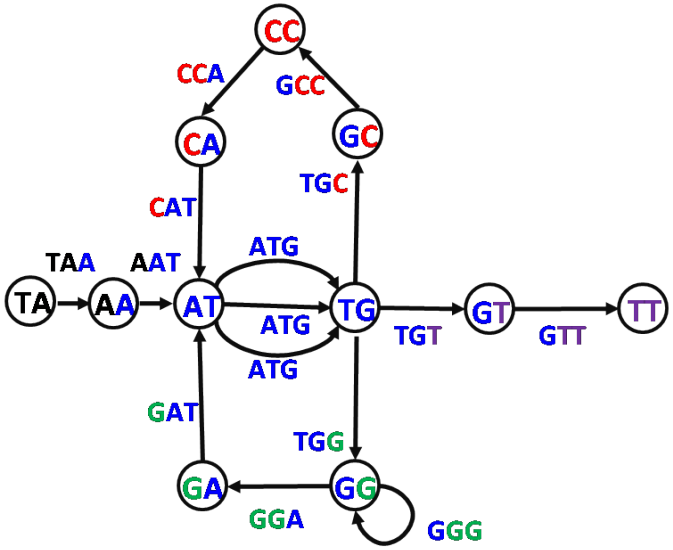
- What Is Genome Sequencing?
- Exploding Newspapers
- The String Reconstruction Problem
- String Reconstruction as a Hamiltonian Path Problem
- String Reconstruction as an Eulerian Path Problem
- Similar Problems with Different Fates
- De Bruijn Graphs
- Euler's Theorem
- Assembling Read-Pairs
- De Bruijn Graphs Face Harsh Realities of Assembly

Eulerian Path Problem

Eulerian Path Problem. Find an Eulerian path in a graph.

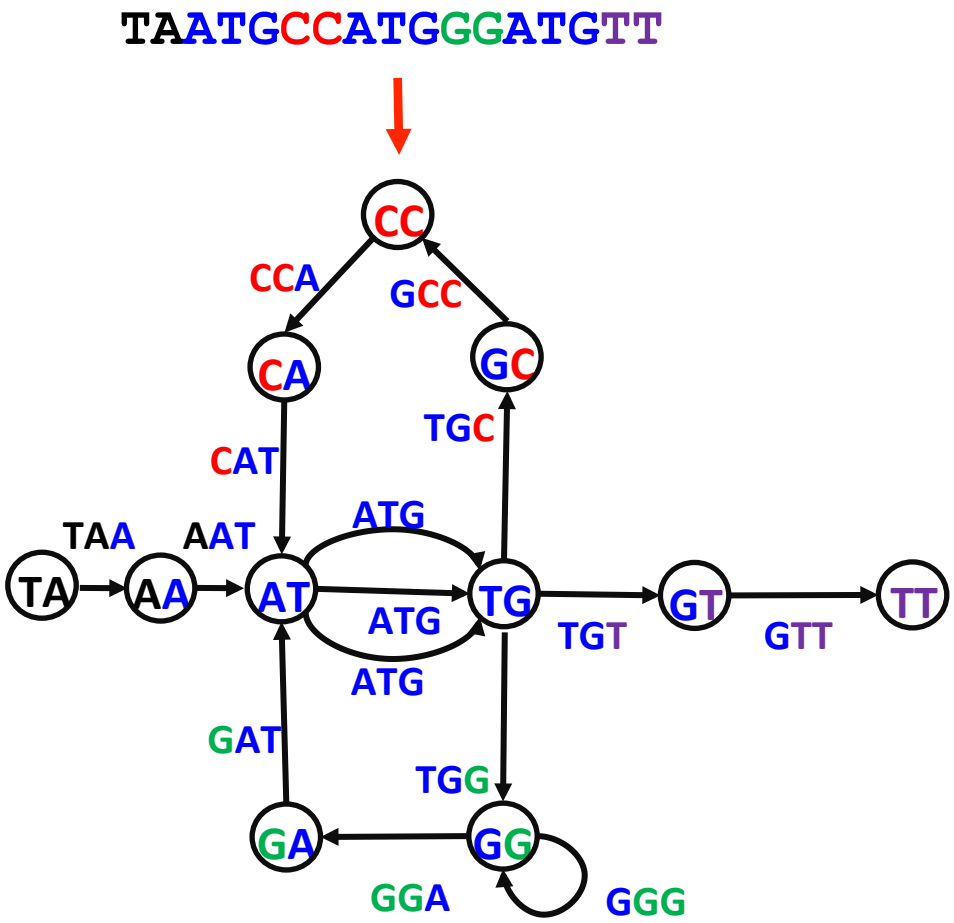


- Input. A graph.
- Output. A path visiting every edge in the graph exactly once.

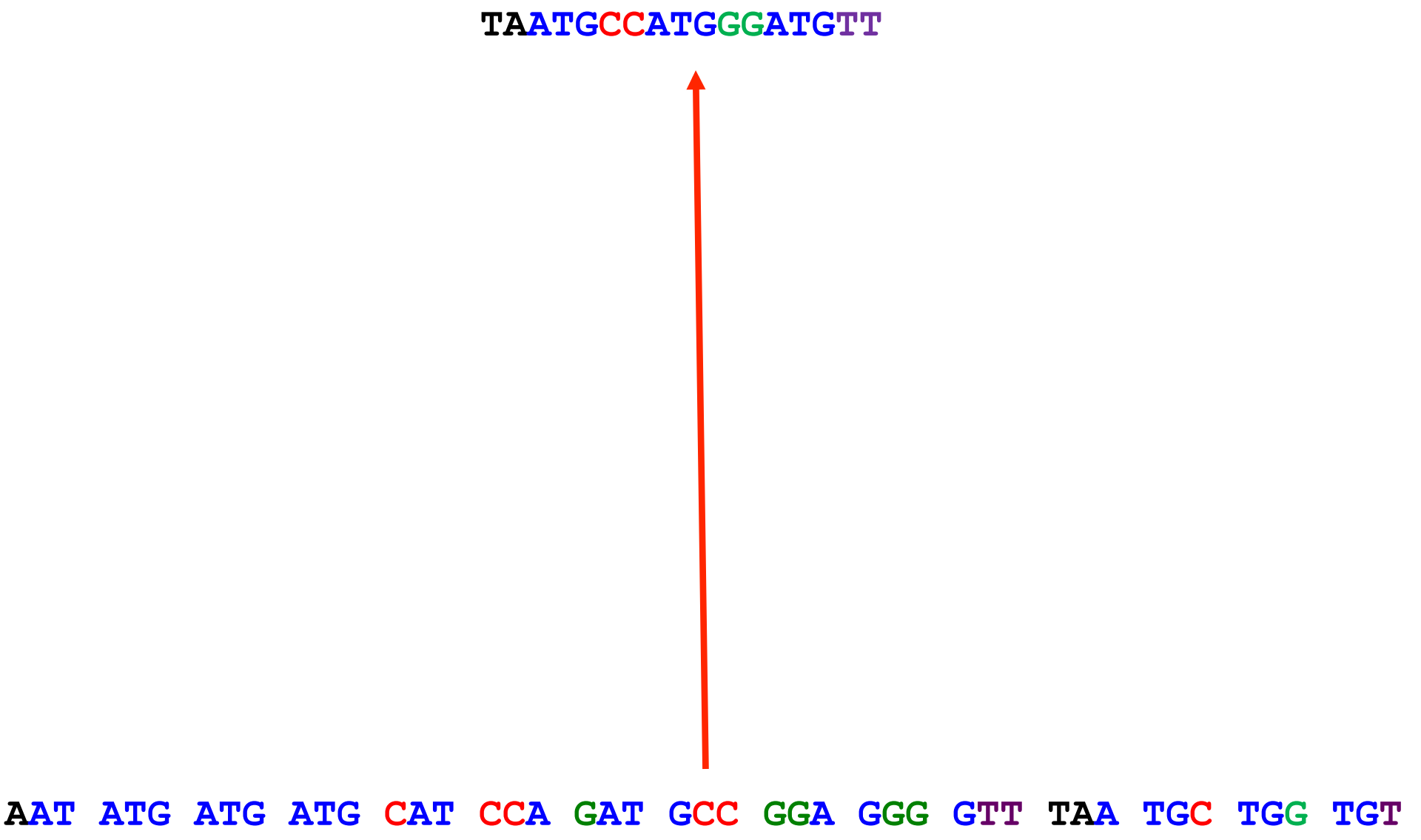


We constructed the de Bruijn graph from Genome, but in reality, Genome is unknown!

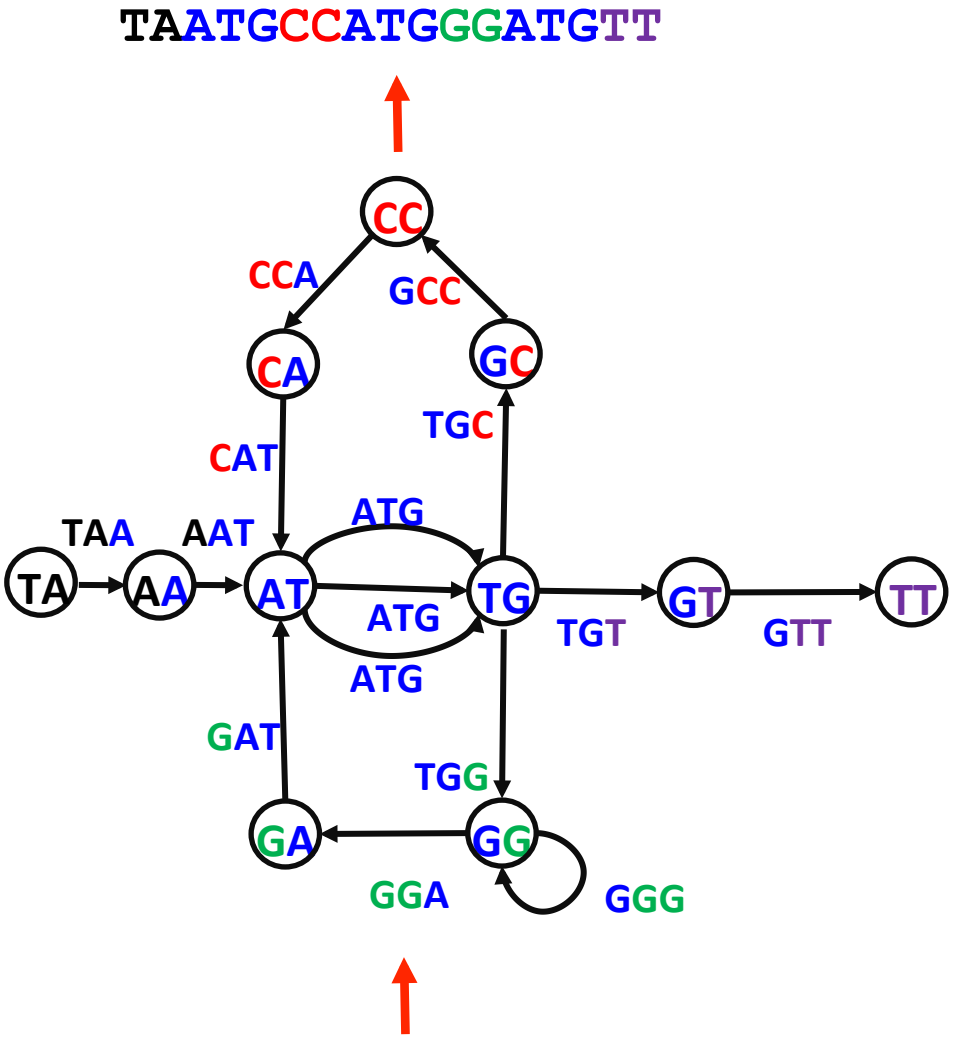
What We Have Done: From Genome to de Bruijn Graph



What We Want: From Reads (k-mers) to Genome



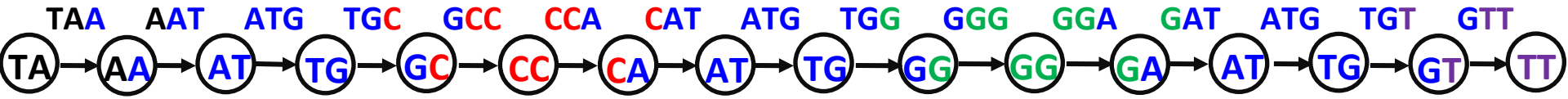
What We will Show: From Reads to de Bruijn Graph to Genome



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

Constructing de Bruijn Graph when Genome Is Known

TAATGCCATGGGATGTT



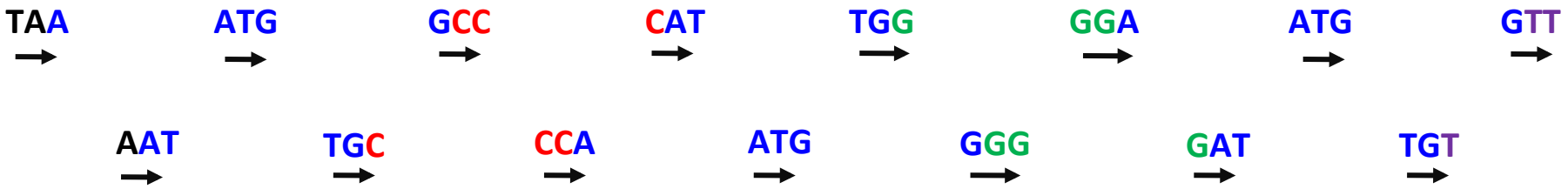
Constructing de Bruijn when Genome Is Unknown

TAA ATG GCC CAT TGG GGA ATG GTT

AAT TGC CCA ATG GGG GAT TGT

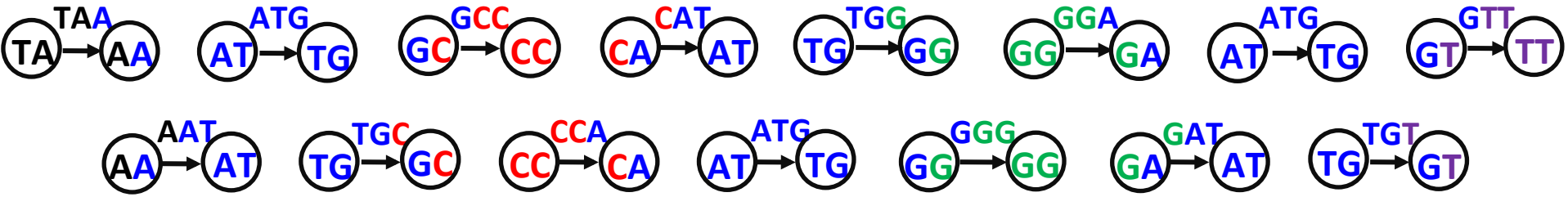
Composition$_3$(**TAATGCCATGGGATGTT**)

Representing Composition as a Graph Consisting of Isolated Edges



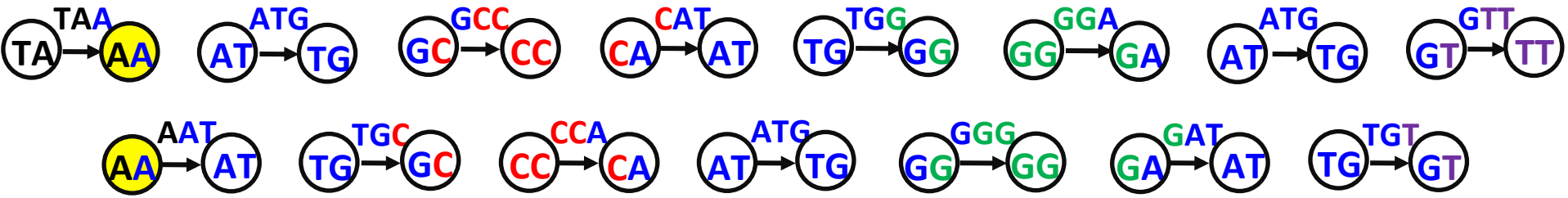
Composition_3**(TAATGCCATGGGATGTT)**

Constructing de Bruijn Graph from k-mer Composition

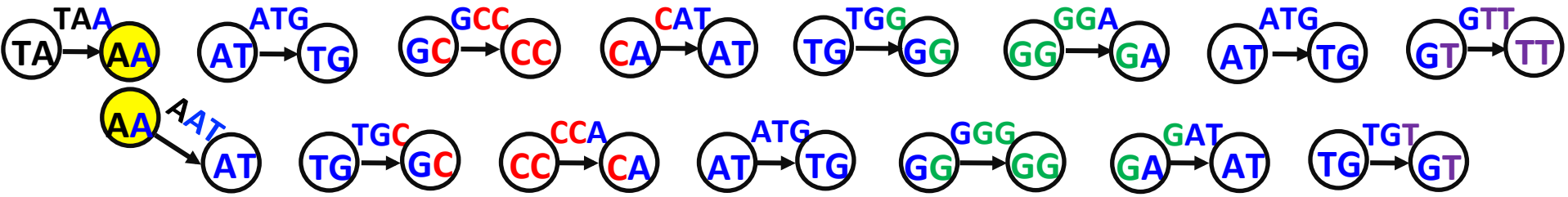


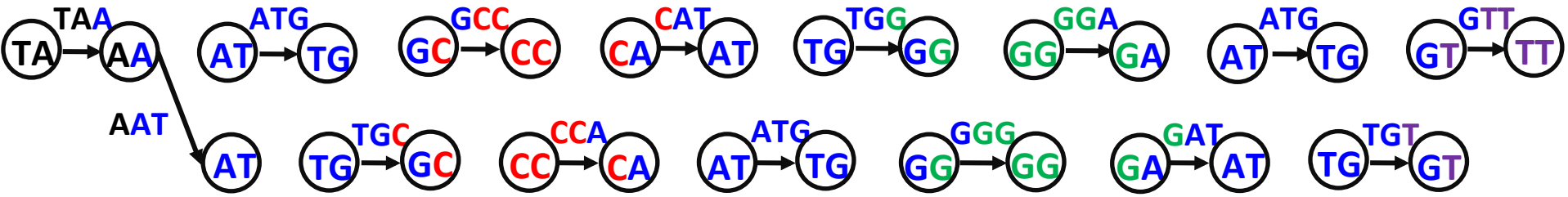
Composition$_3$(**TAATGCCATGGGATGTT**)

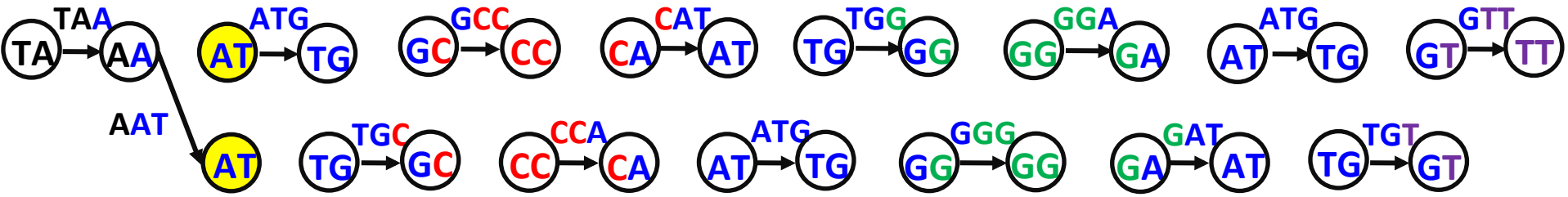
Gluing Identically Labeled Nodes

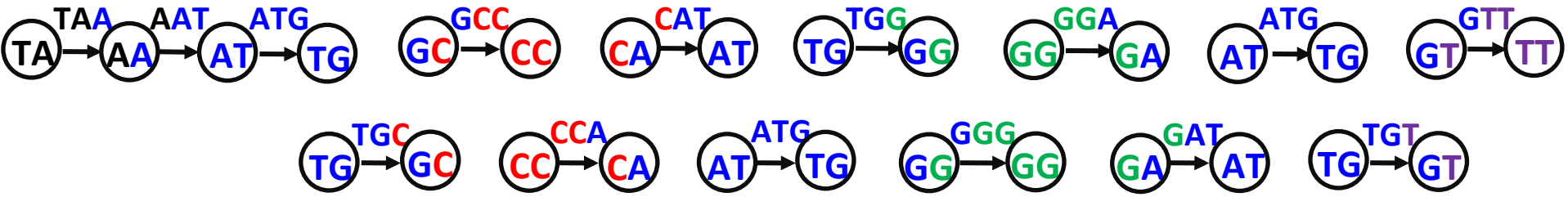


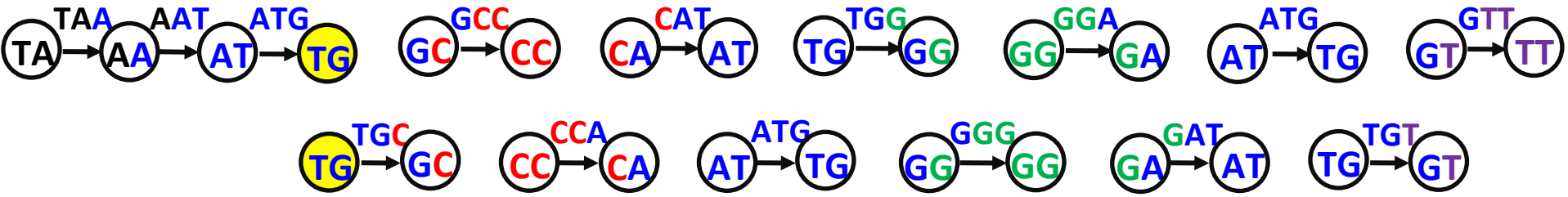
Gluing Identically Labeled Nodes

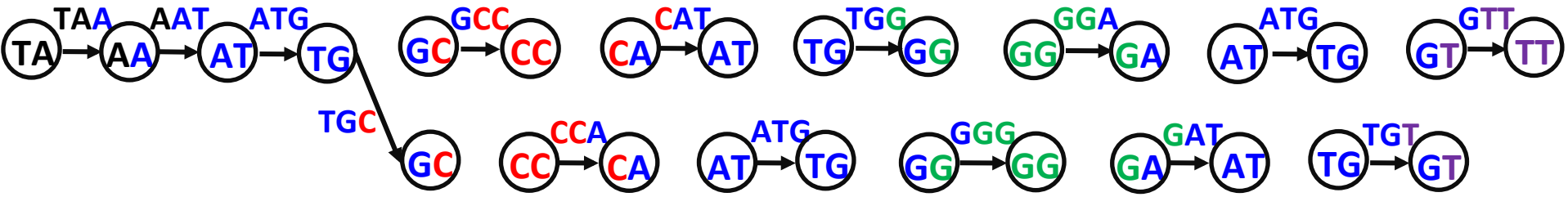


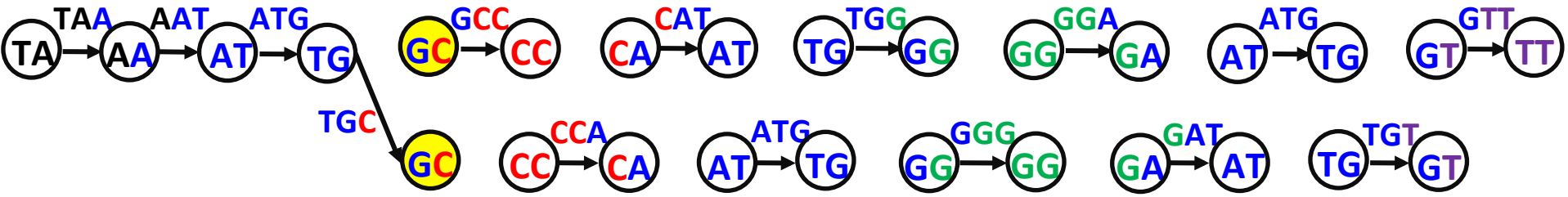

TA
TAA
AA
AAT
AT
ATG
TG
GC
GCC
CC
CA
CAT
AT
TG
TGG
GG
GG
GGA
GA
AT
ATG
TG
GT
GTT
TT
TG
TGC
GC
CC
CCA
CA
AT
ATG
TG
GG
GGG
GG
GA
GAT
AT
TG
TGT
GT

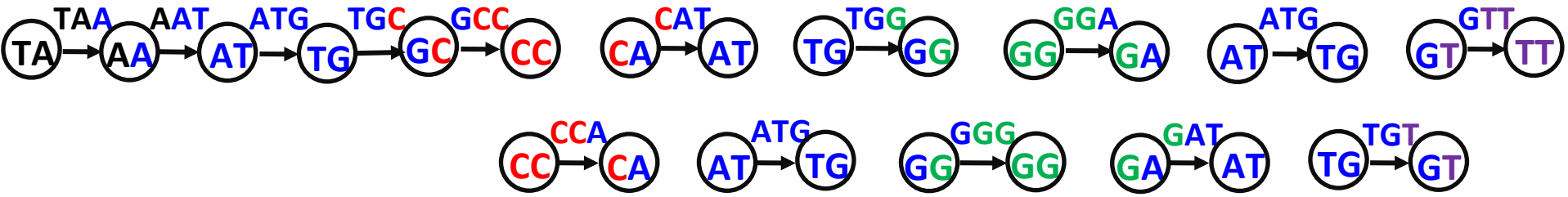

TA
TAA
AA
AAT
AT
ATG
TG
GC
GCC
CC
CA
CAT
AT
TG
TGG
GG
GG
GGA
GA
AT
ATG
TG
GT
GTT
TT
AT
TG
TGC
GC
CC
CCA
CA
AT
ATG
TG
GG
GGG
GG
GA
GAT
AT
TG
TGT
GT

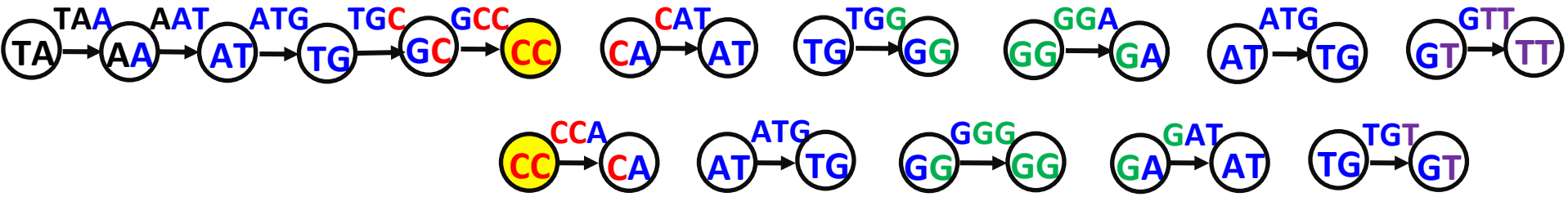

TAA
AAT
ATG
TA
AA
AT
TG
GCC
GC
CC
CAT
CA
AT
TGG
TG
GG
GGA
GG
GA
ATG
AT
TG
GTT
GT
TT
TGC
TG
GC
CCA
CC
CA
ATG
AT
TG
GGG
GG
GG
GAT
GA
AT
TGT
TG
GT

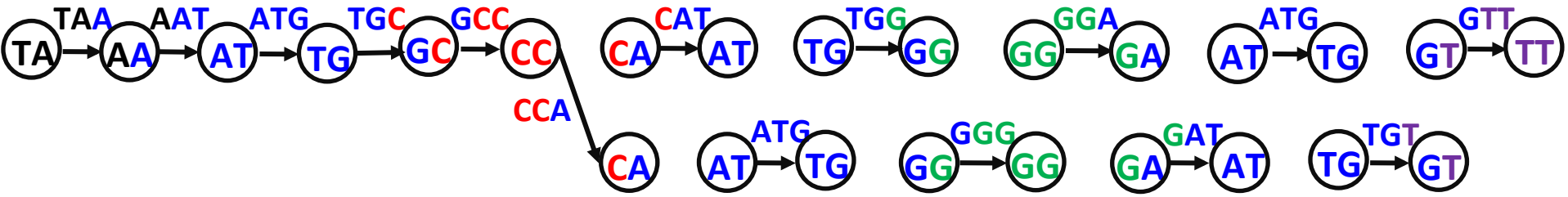

TAA
AAT
ATG
TA
AA
AT
TG
GCC
GC
CC
CAT
CA
AT
TGG
TG
GG
GGA
GG
GA
ATG
AT
TG
GTT
GT
TT
TGC
TG
GC
CCA
CC
CA
ATG
AT
TG
GGG
GG
GG
GAT
GA
AT
TGT
TG
GT

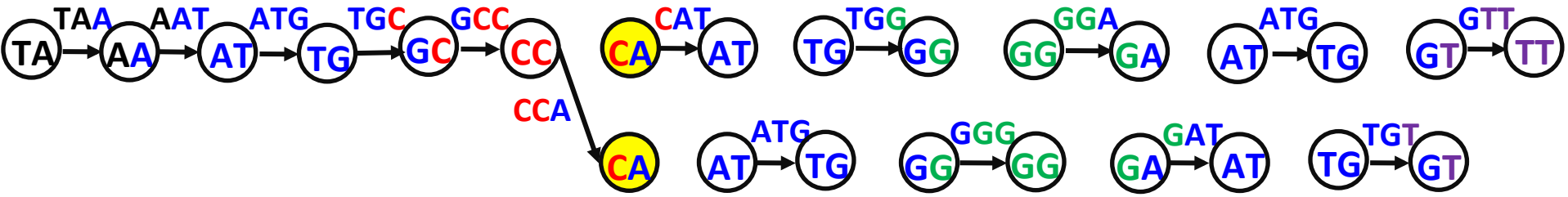

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TAA
AAT
ATG
GCC
CAT
TGG
GGA
ATG
GTT
TGC
CCA
ATG
GGG
GAT
TGT

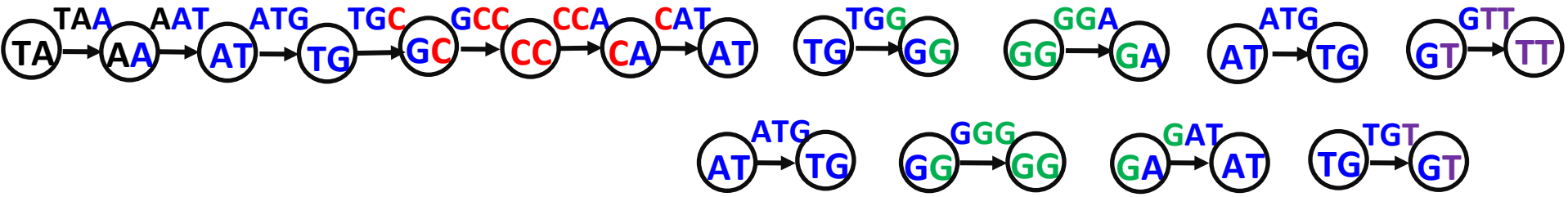

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TAA
AAT
ATG
GCC
CAT
TGG
GGA
ATG
GTT
TGC
CCA
ATG
GGG
GAT
TGT

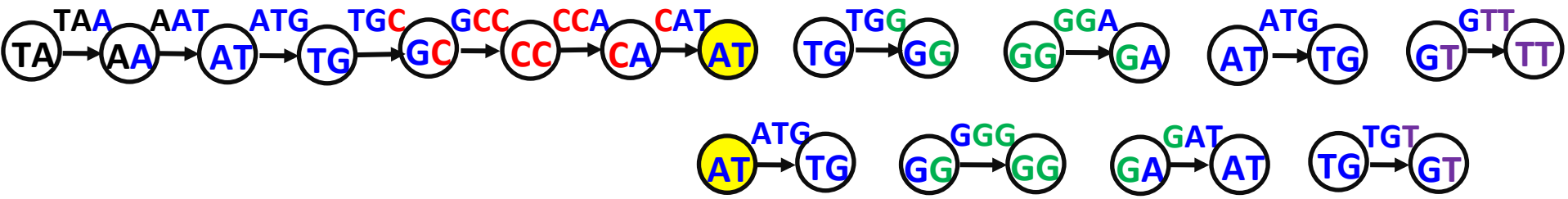

TAA
AAT
ATG
TGC
GCC
TA
AA
AT
TG
GC
CC
CAT
CA
AT
TGG
TG
GG
GGA
GG
GA
ATG
AT
TG
GTT
GT
TT
CCA
CC
CA
ATG
AT
TG
GGG
GG
GG
GAT
GA
AT
TGT
TG
GT

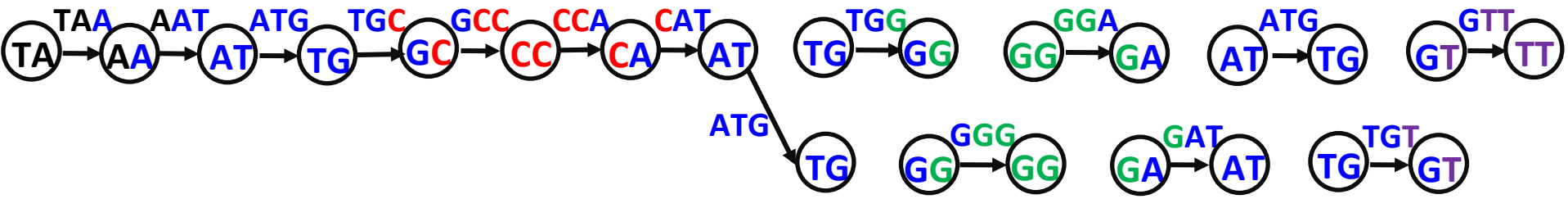

TA
AA
AT
TG
GC
CC
TAA
AAT
ATG
TGC
GCC
CA
AT
CAT
TG
GG
TGG
GG
GA
GGA
AT
TG
ATG
GT
TT
GTT
CC
CA
CCA
AT
TG
ATG
GG
GG
GGG
GA
AT
GAT
TG
GT
TGT

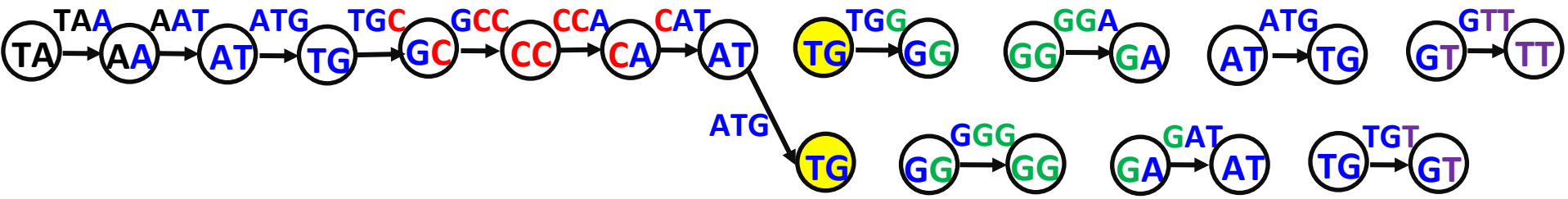

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
TAA
AAT
ATG
TGC
GCC
CAT
TGG
GGA
ATG
GTT
CCA
CA
AT
TG
GG
GG
GA
AT
TG
GT
ATG
GGG
GAT
TGT

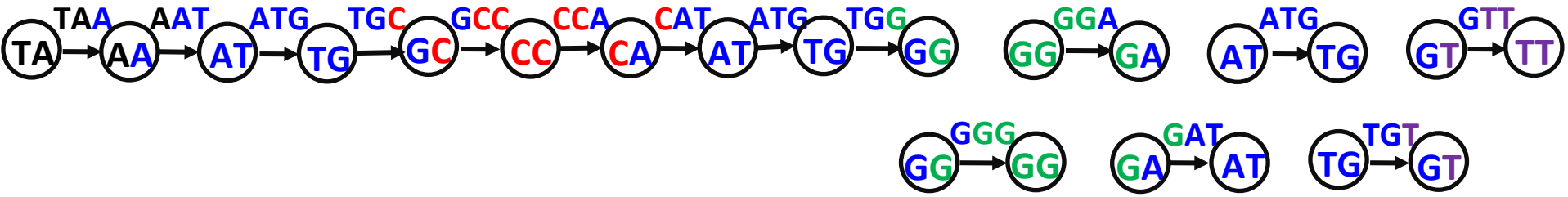

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
TAA
AAT
ATG
TGC
GCC
CAT
TGG
GGA
ATG
GTT
CCA
CA
AT
TG
GG
GG
GA
AT
TG
GT
ATG
GGG
GAT
TGT

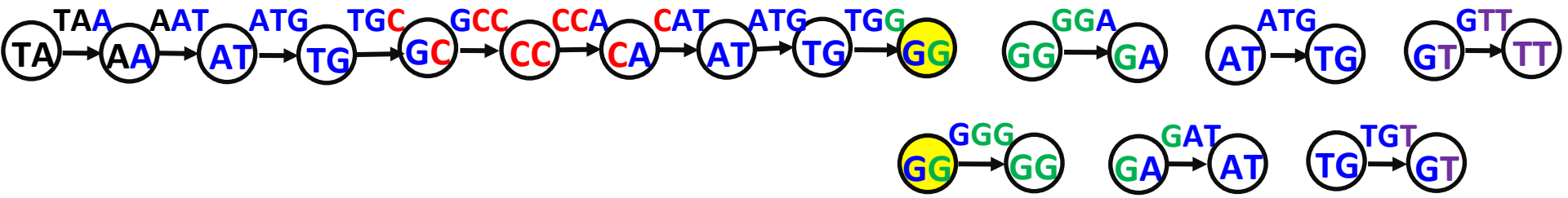

TAA
AAT
ATG
TGC
GCC
CCA
CAT
TA
AA
AT
TG
GC
CC
CA
AT
TGG
TG
GG
GGA
GG
GA
ATG
AT
TG
GTT
GT
TT
ATG
AT
TG
GGG
GG
GG
GAT
GA
AT
TGT
TG
GT

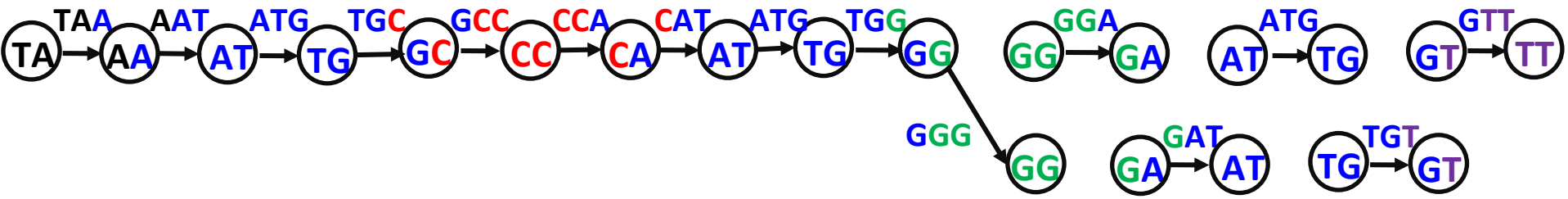

TAA
AAT
ATG
TGC
GCC
CCA
CAT
TA
AA
AT
TG
GC
CC
CA
AT
TGG
TG
GG
GGA
GG
GA
ATG
AT
TG
GTT
GT
TT
ATG
AT
TG
GGG
GG
GG
GAT
GA
AT
TGT
TG
GT

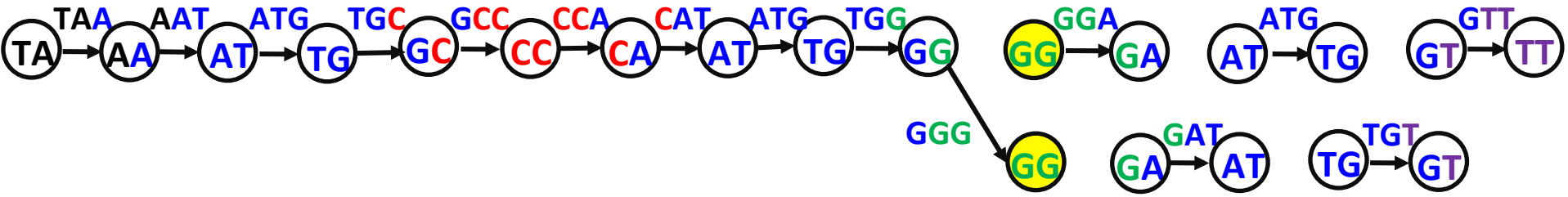

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
TAA
AAT
ATG
TGC
GCC
CCA
CAT
TGG
GGA
ATG
GTT
ATG
TG
GG
GG
GA
AT
TG
GT
GGG
GAT
TGT

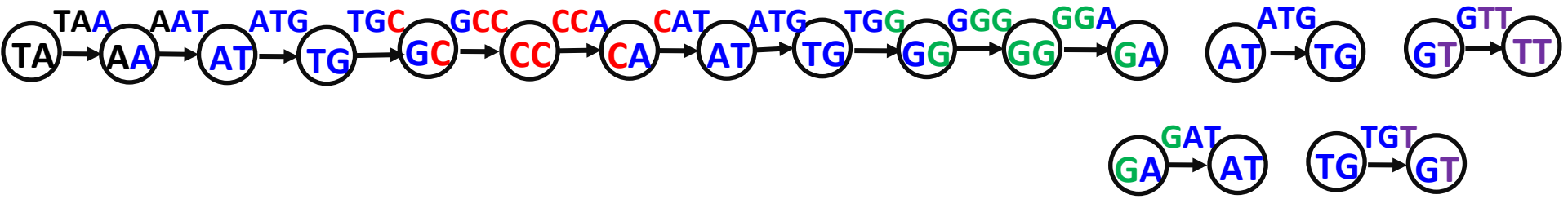

TAA
AAT
ATG
TGC
GCC
CCA
CAT
TA
AA
AT
TG
GC
CC
CA
AT
TGG
TG
GG
GGA
GG
GA
ATG
AT
TG
GTT
GT
TT
ATG
TG
GGG
GG
GG
GAT
GA
AT
TGT
TG
GT

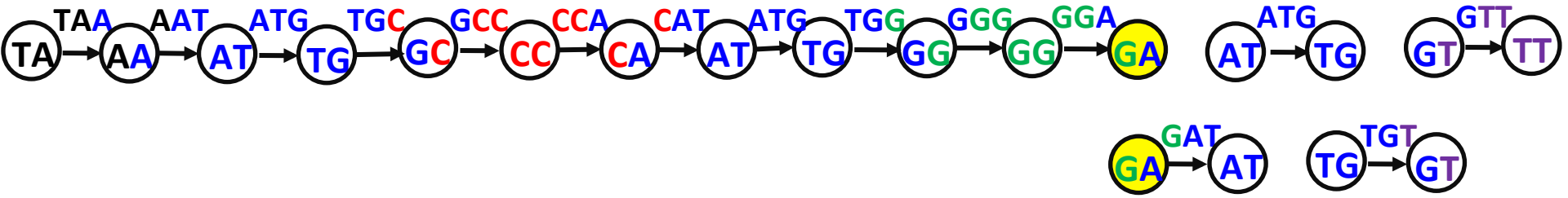

TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGA
ATG
GTT
GGG
GAT
TGT
TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
GG
GG
GA
AT
TG
GT

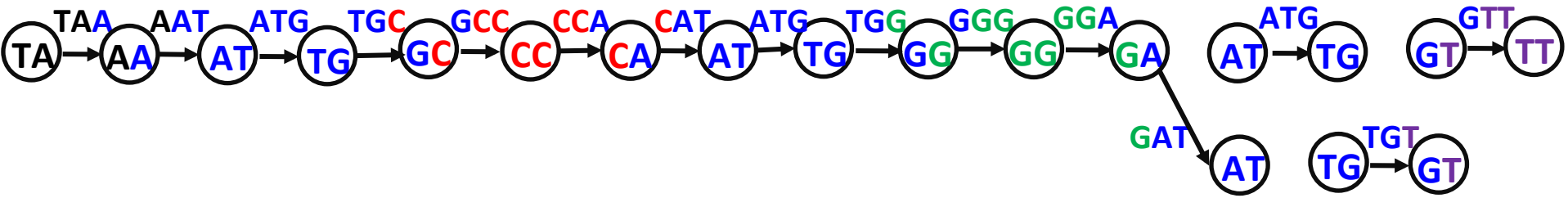

TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GGA
GG
GA
ATG
AT
TG
GTT
GT
TT
GGG
GG
GG
GAT
GA
AT
TGT
TG
GT

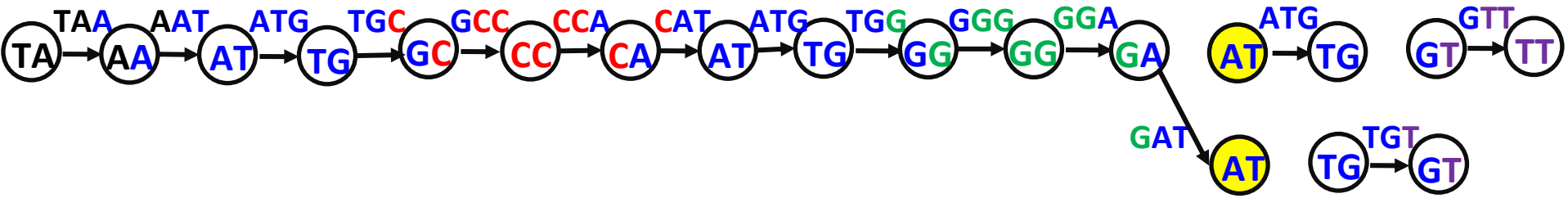

TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGA
ATG
GTT
TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
GGG
GG
GAT
GA
AT
TGT
TG
GT

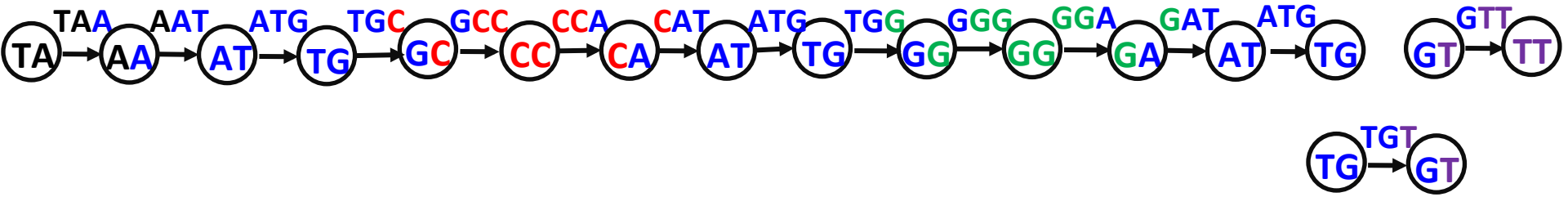

TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGA
ATG
GTT
TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
GGG
GG
GAT
GA
AT
TGT
TG
GT

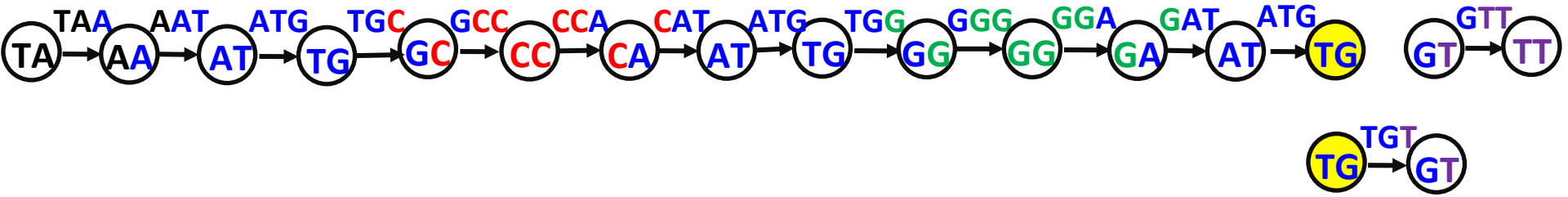

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
GA
AT
TG
GT
TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGG
GGA
ATG
GTT
GAT
TGT

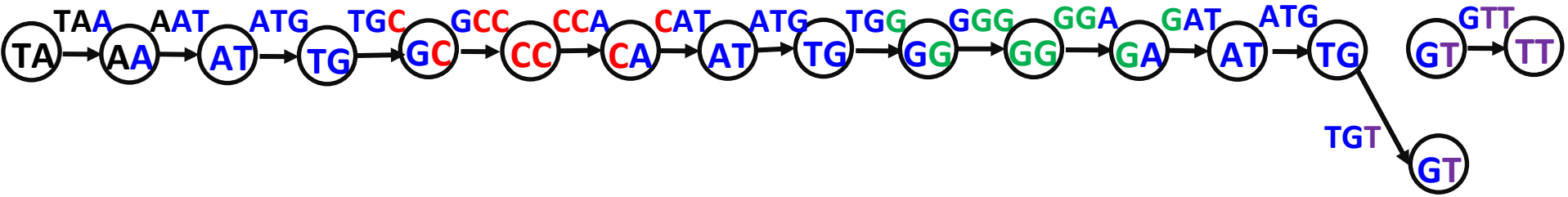

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGG
GGA
AT
TG
ATG
GT
TT
GTT
GA
AT
GAT
TG
GT
TGT

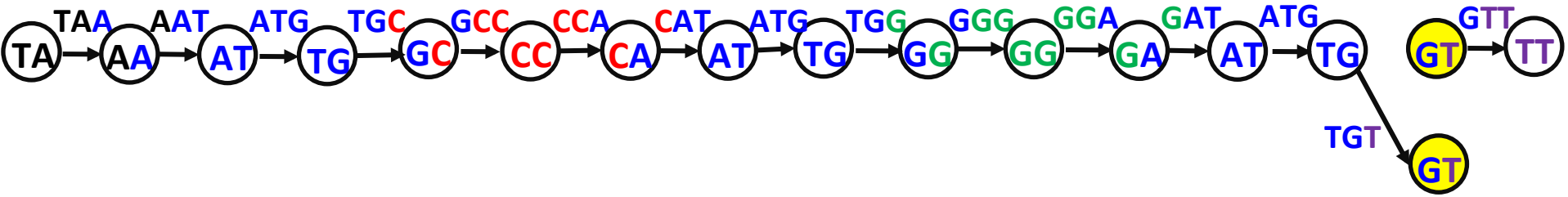

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
AT
TG
GT
TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGG
GGA
ATG
GTT
GAT
TGT


TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
AT
TG
GT
TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGG
GGA
ATG
GTT
GAT
TGT

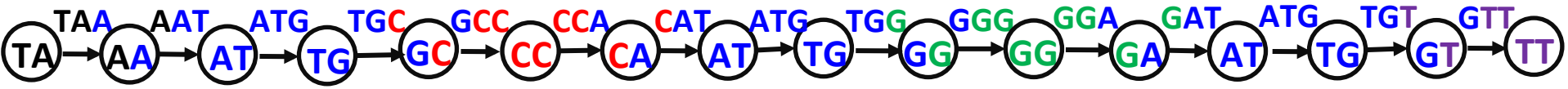

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
TG
GT
TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGG
GGA
GAT
ATG
GTT
TGT


TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
TG
GT
TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGG
GGA
GAT
ATG
GTT
TGT

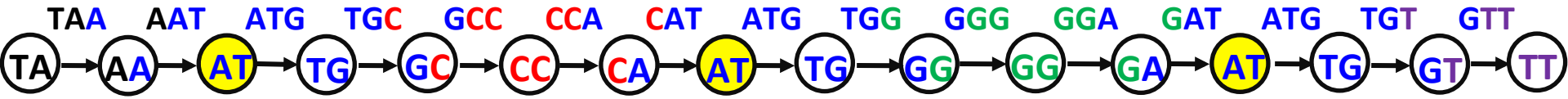

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
TGT
GT
TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGG
GGA
GAT
ATG
GTT

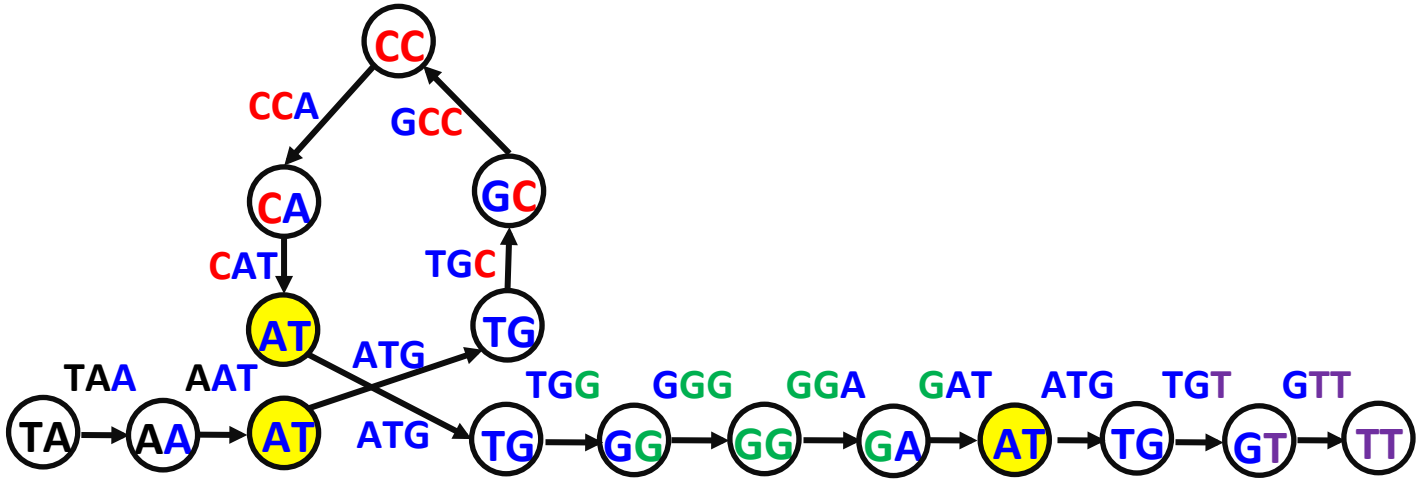

TA
AA
AT
TG
GC
CC
CA
AT
TG
GG
GG
GA
AT
TG
GT
TT
GT
TAA
AAT
ATG
TGC
GCC
CCA
CAT
ATG
TGG
GGG
GGA
GAT
ATG
GTT
TGT

We Are Not Done with Gluing Yet

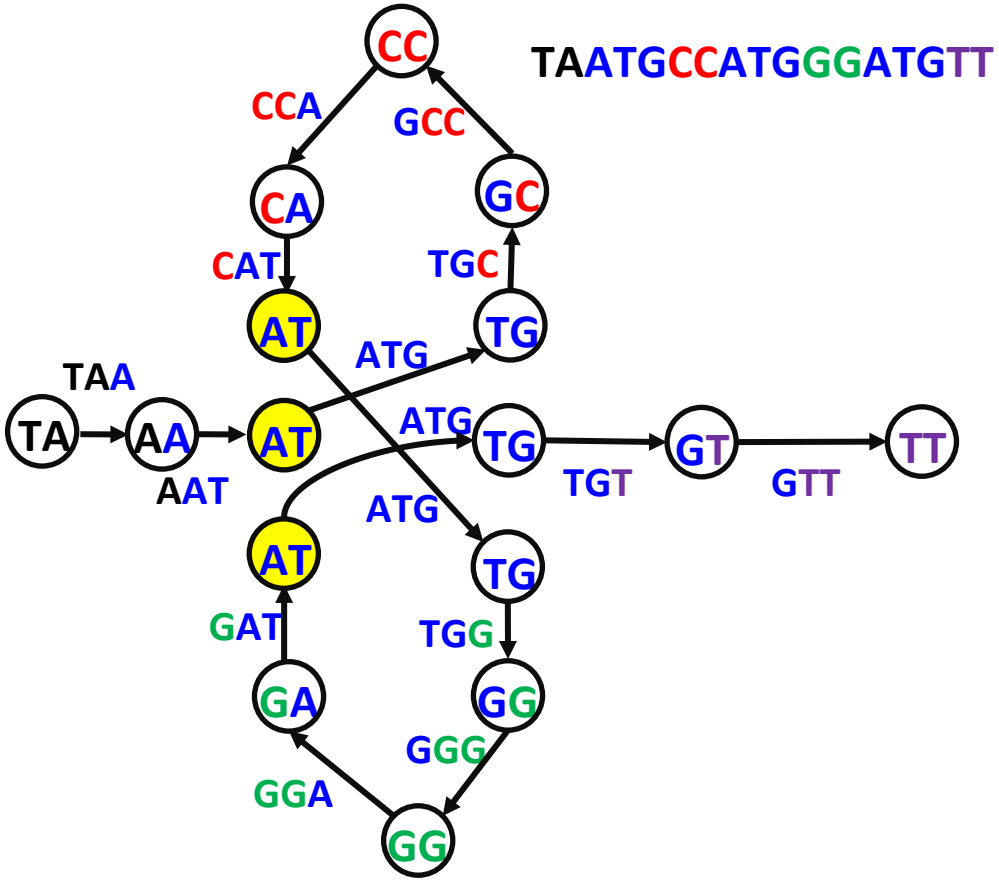


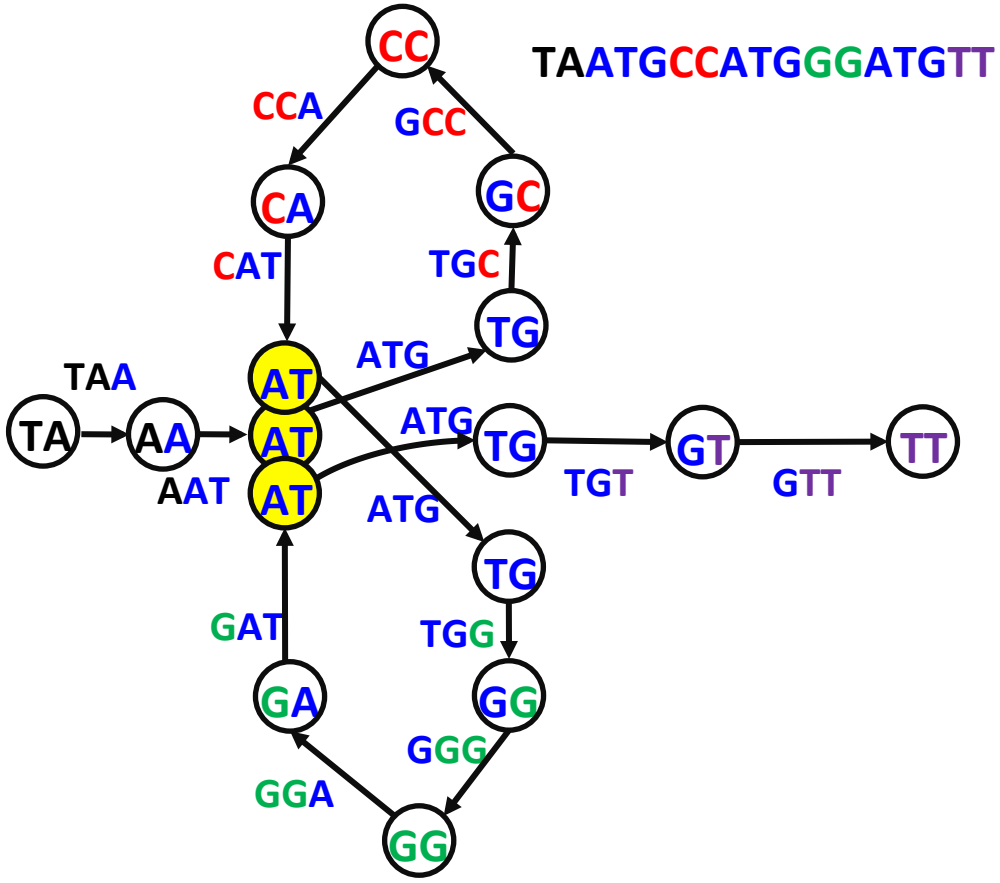
Gluing Identically Labeled Nodes



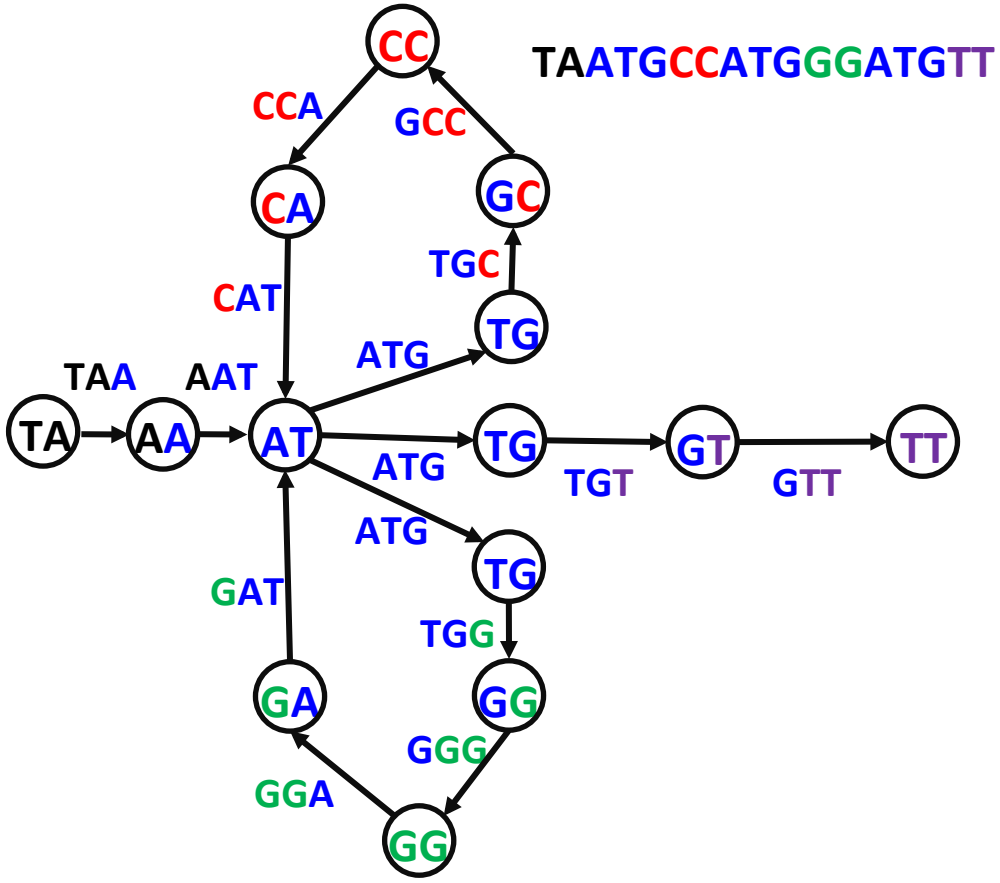


Gluing Identically Labeled Nodes

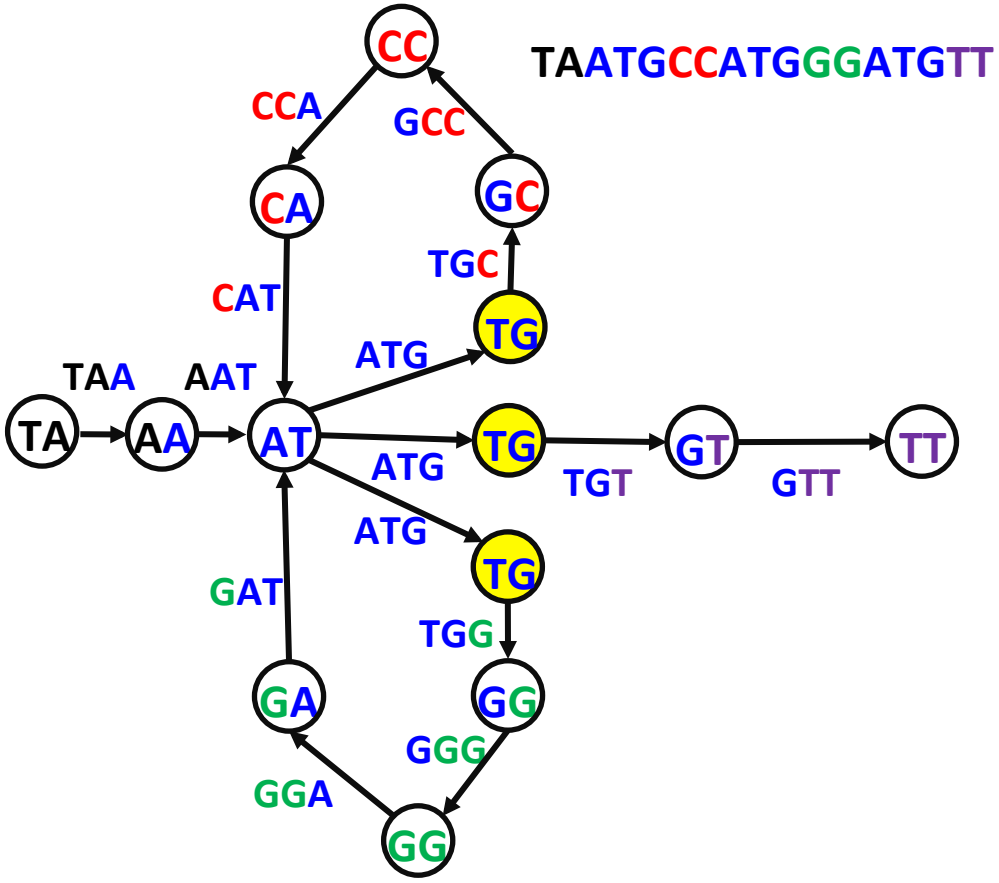



TAATGCCATGGGATGTT
CC
CCA
GCC
CA
GC
CAT
TGC
TG
ATG
TAA
AT
TA
AA
AT
ATG
TG
GT
TT
TGT
GTT
AAT
AT
ATG
TG
GAT
TGG
GA
GG
GGG
GGA
GG

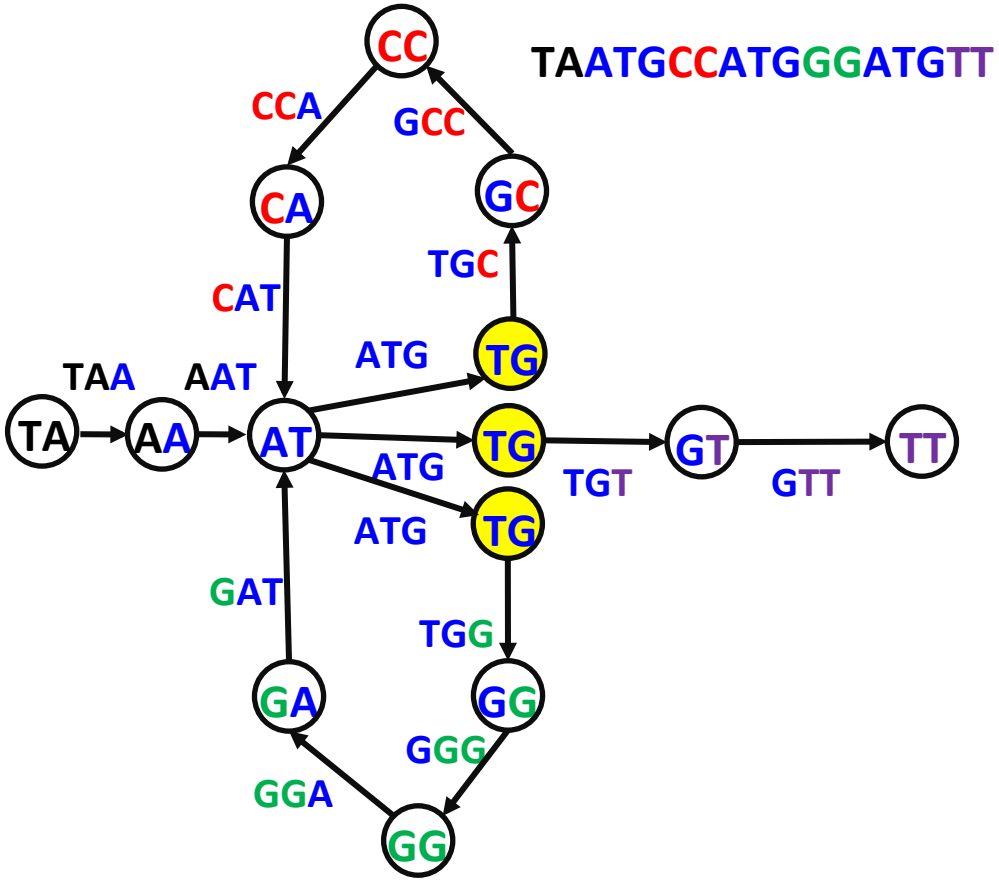
Gluing Identically Labeled Nodes



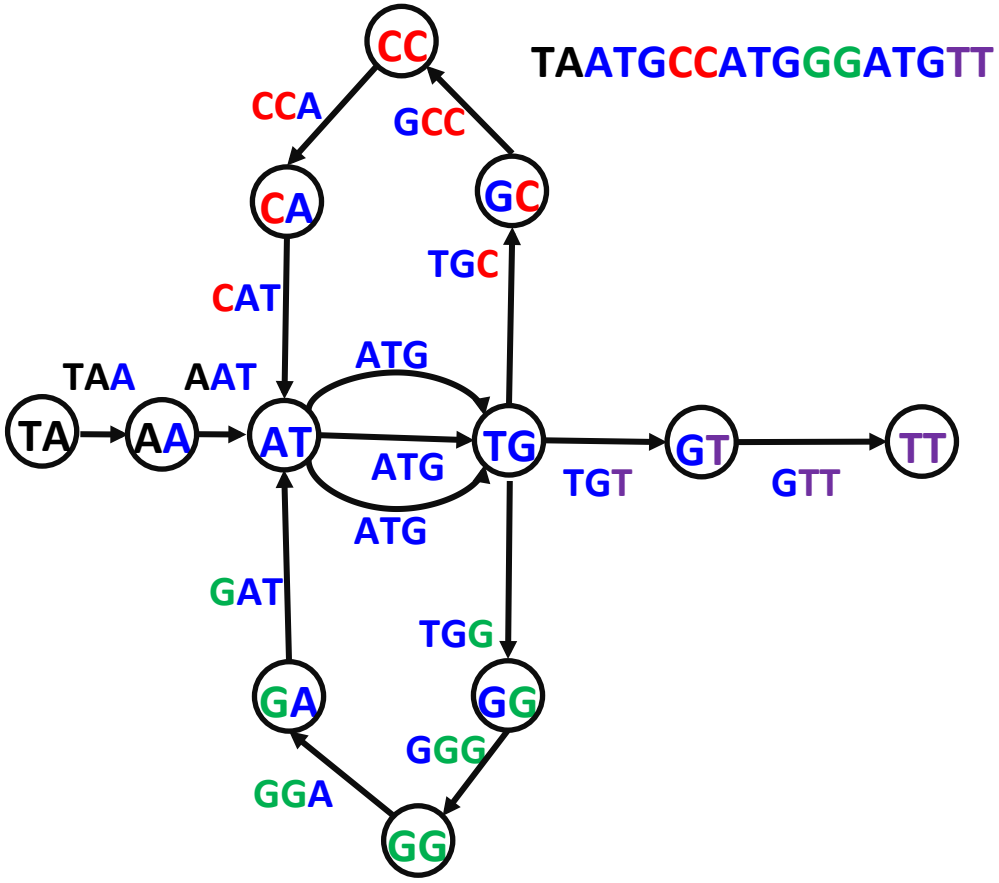
Gluing Identically Labeled Nodes



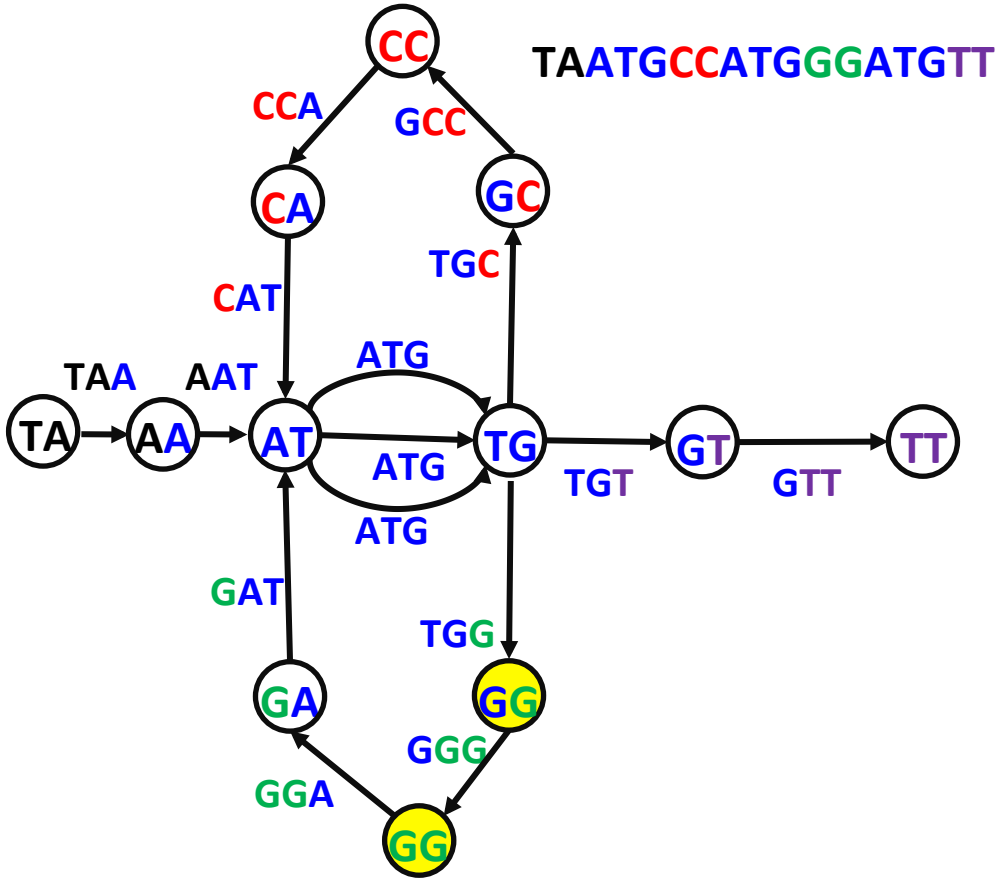
Gluing Identically Labeled Nodes



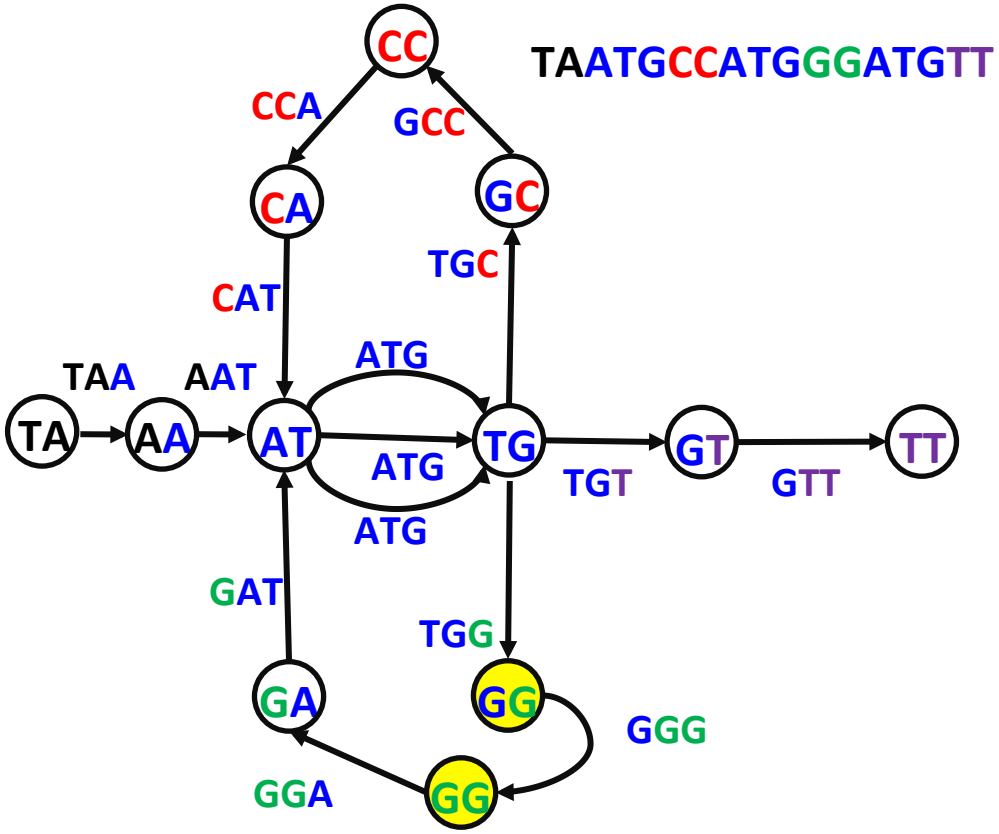
Gluing Identically Labeled Nodes



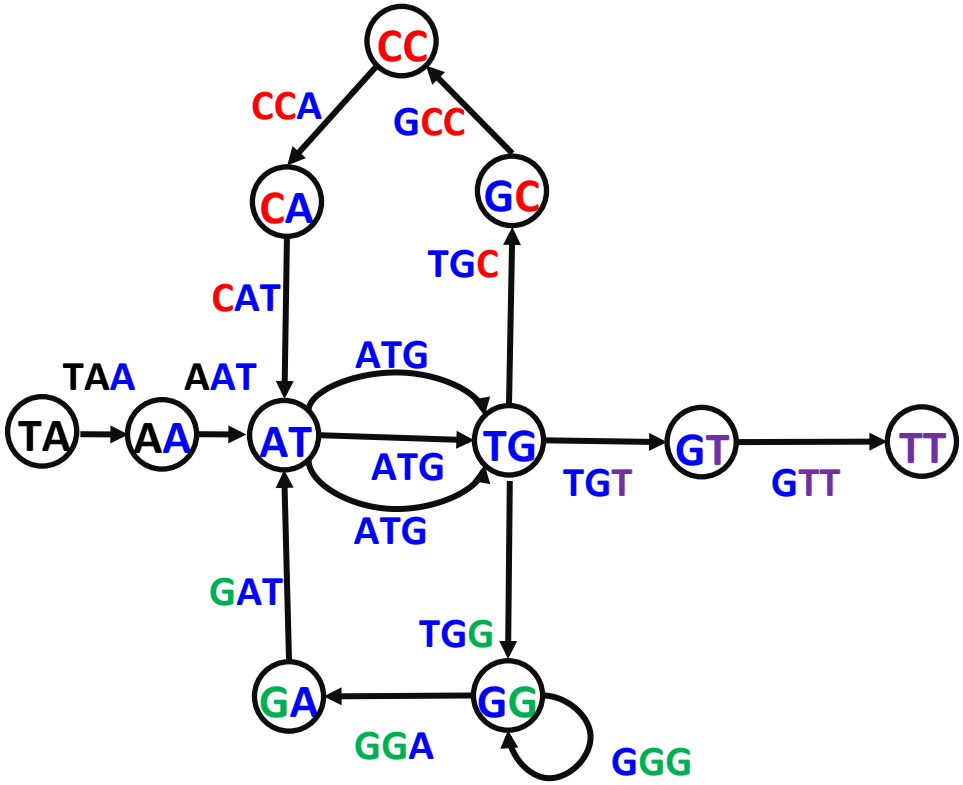
Gluing Identically Labeled Nodes



Gluing Identically Labeled Nodes



The Same de Bruijn Graph: DeBruin(Genome)=DeBruin(Genome Composition)

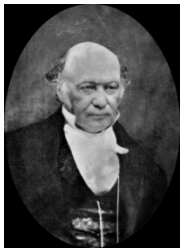
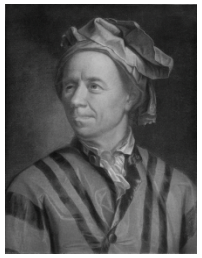


Constructing de Bruijn Graph

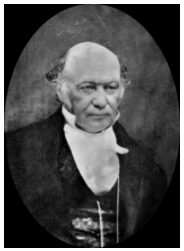
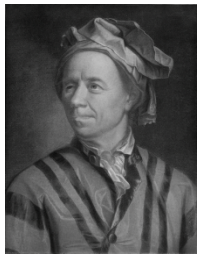
De Bruijn graph of a collection of *k*-mers:

- Represent every *k*-mer as an edge between its prefix and suffix
- Glue **ALL** nodes with identical labels.

```
DeBruijn(k-mers)
  form a node for each (k-1)-mer from k-mers
  for each k-mer in k-mers
     connect its prefix node with its suffix node by an edge
```

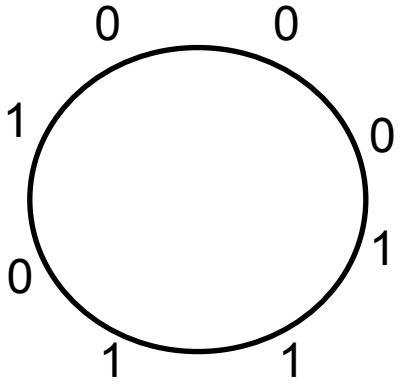
From Hamilton  to Euler  to de Bruijn 

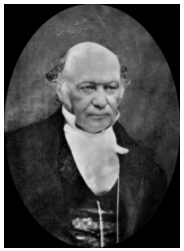
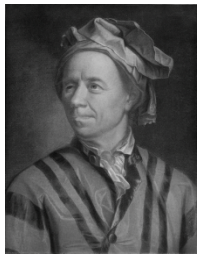
Universal String Problem (De Bruijn, 1946). Find a circular string containing each binary k-mer exactly once.

From Hamilton  to Euler  to de Bruijn 

Universal String Problem (De Bruijn, 1946). Find a circular string containing each binary k-mer exactly once.

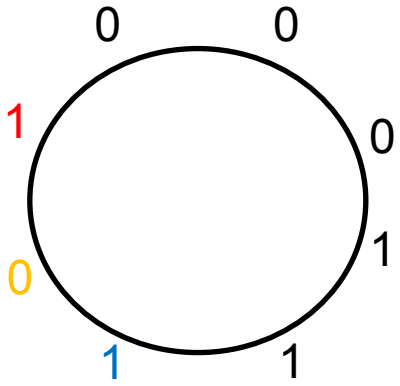
000 001 010 011 100 101 110 111

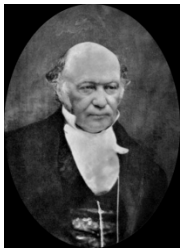
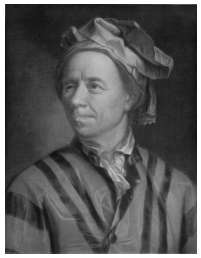


From Hamilton  to Euler  to de Bruijn 

Universal String Problem (Nicolaas de Bruijn, 1946). Find a circular string containing each binary k-mer exactly once.

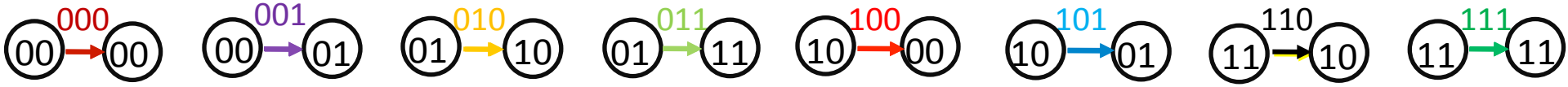
000 001 010 011 100 101 110 111

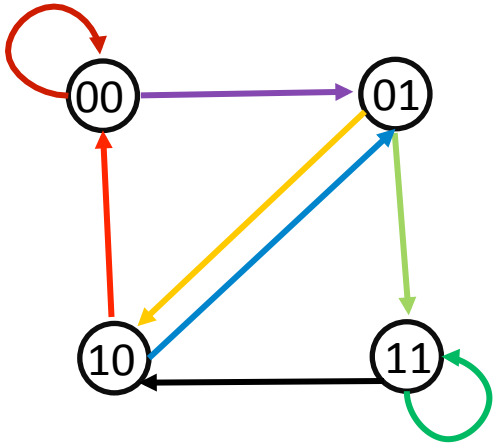


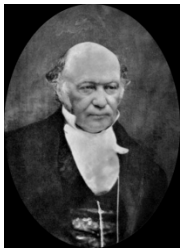
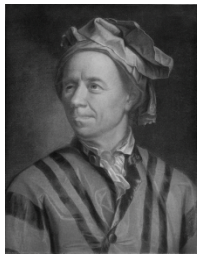
From Hamilton  to Euler  to de Bruijn 

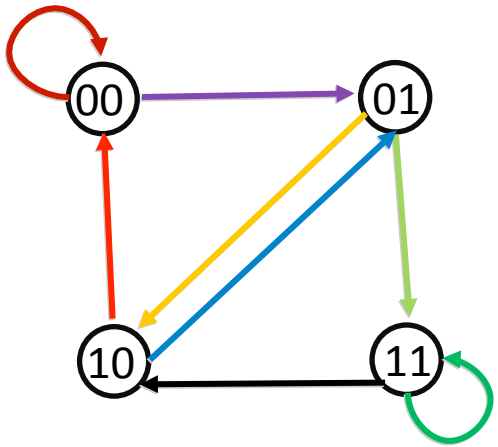
Universal String Problem (Nicolaas de Bruijn, 1946). Find a circular string containing each binary k-mer exactly once.

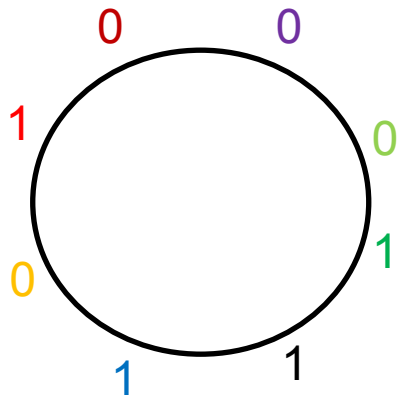
000 001 010 011 100 101 110 111



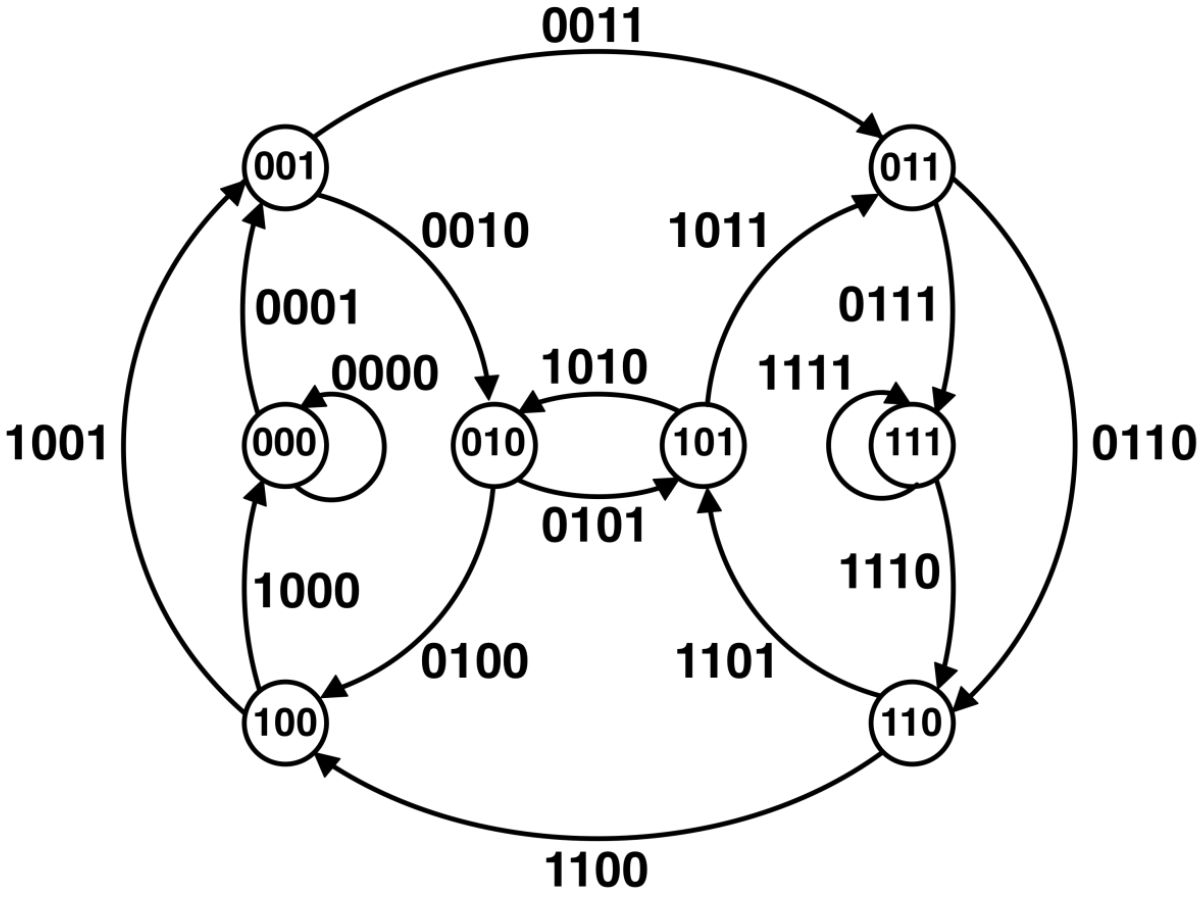


From Hamilton  to Euler  to de Bruijn 





De Bruijn Graph for 4-Universal String



Does it have an Eulerian cycle? If yes, how can we find it?

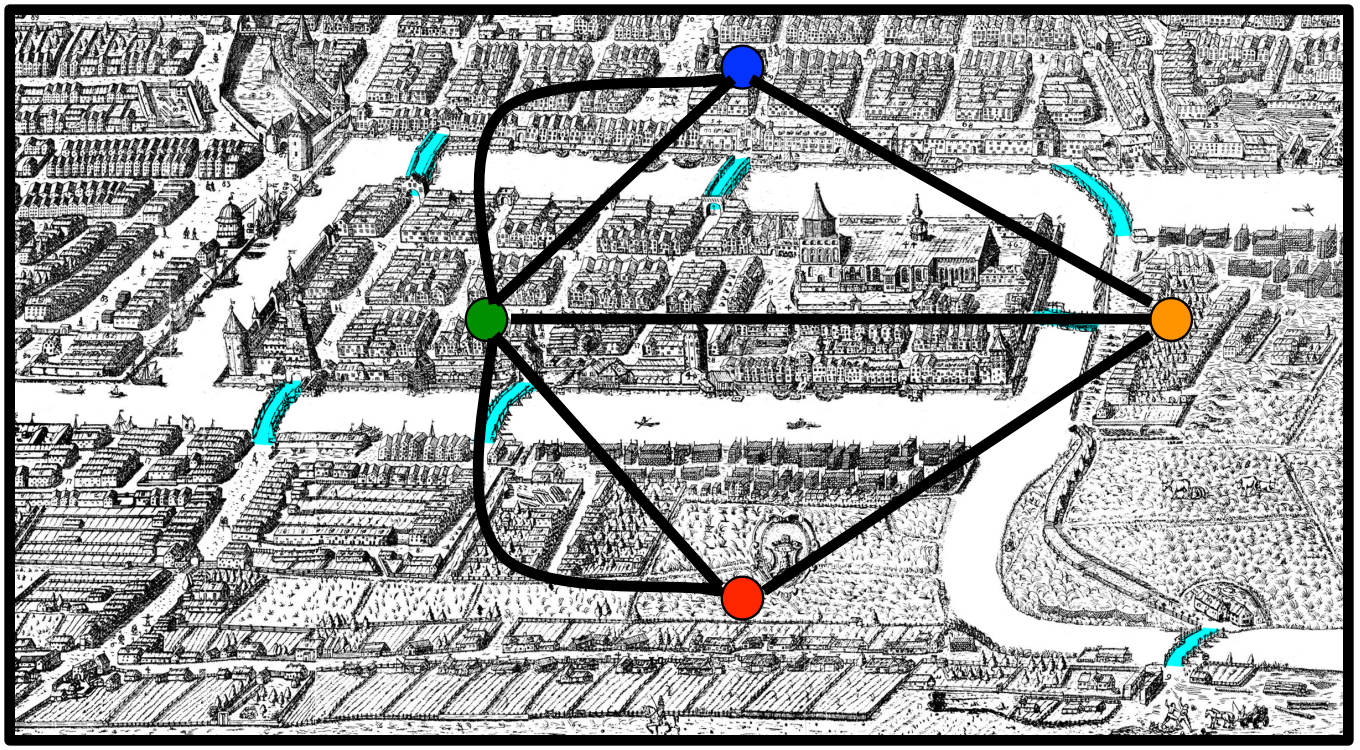
Outline

- What Is Genome Sequencing?
- Exploding Newspapers
- The String Reconstruction Problem
- String Reconstruction as a Hamiltonian Path Problem
- String Reconstruction as an Eulerian Path Problem
- Similar Problems with Different Fates
- De Bruijn Graphs
- Euler's Theorem
- Assembling Read-Pairs
- De Bruijn Graphs Face Harsh Realities of Assembly

Eulerian CYCLE Problem

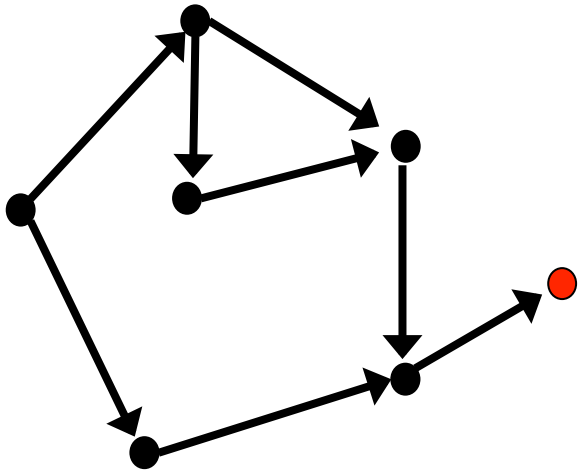
Eulerian CYCLE Problem. Find an Eulerian cycle in a graph.

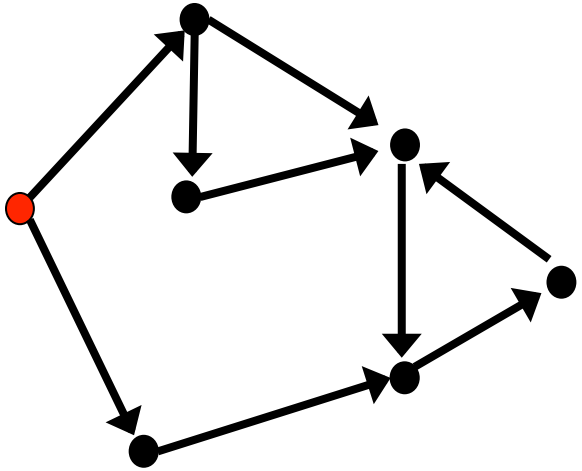
- Input. A graph.
- Output. A cycle visiting every edge in the graph exactly once.

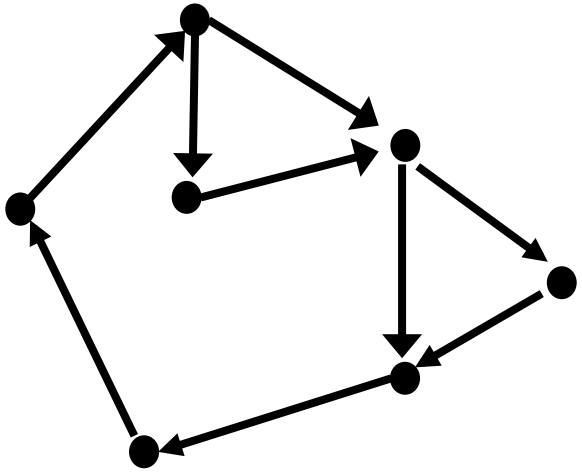


A Graph is **Eulerian** if It Contains an Eulerian Cycle.

Is this graph Eulerian?

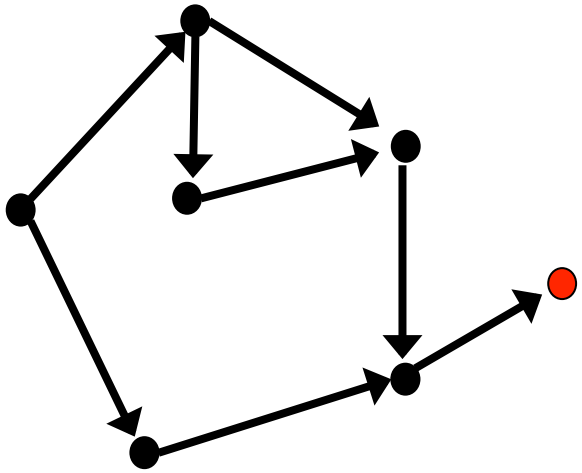


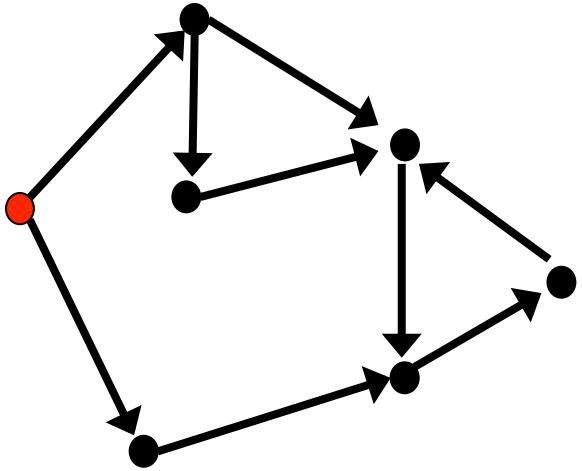


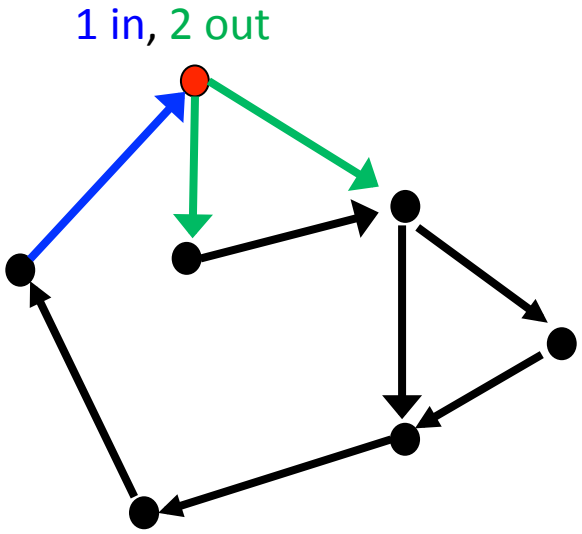


A Graph is **Eulerian** if It Contains an Eulerian Cycle.

Is this graph Eulerian?

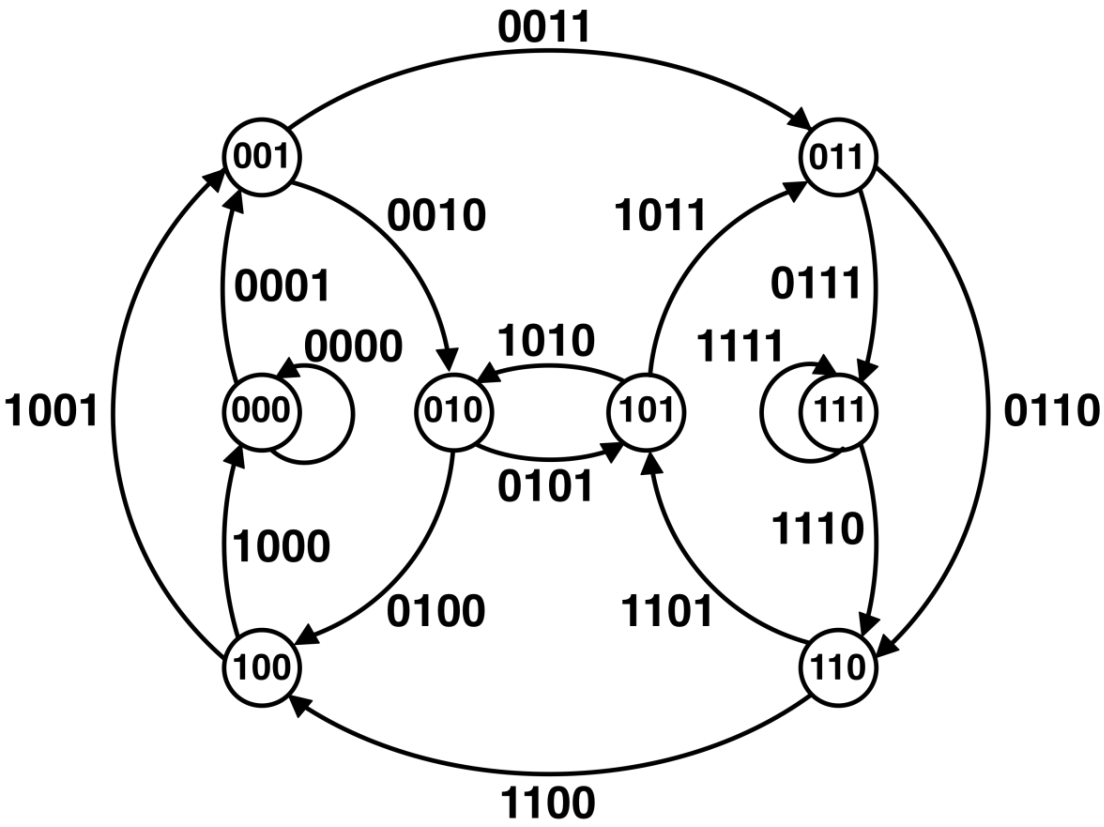






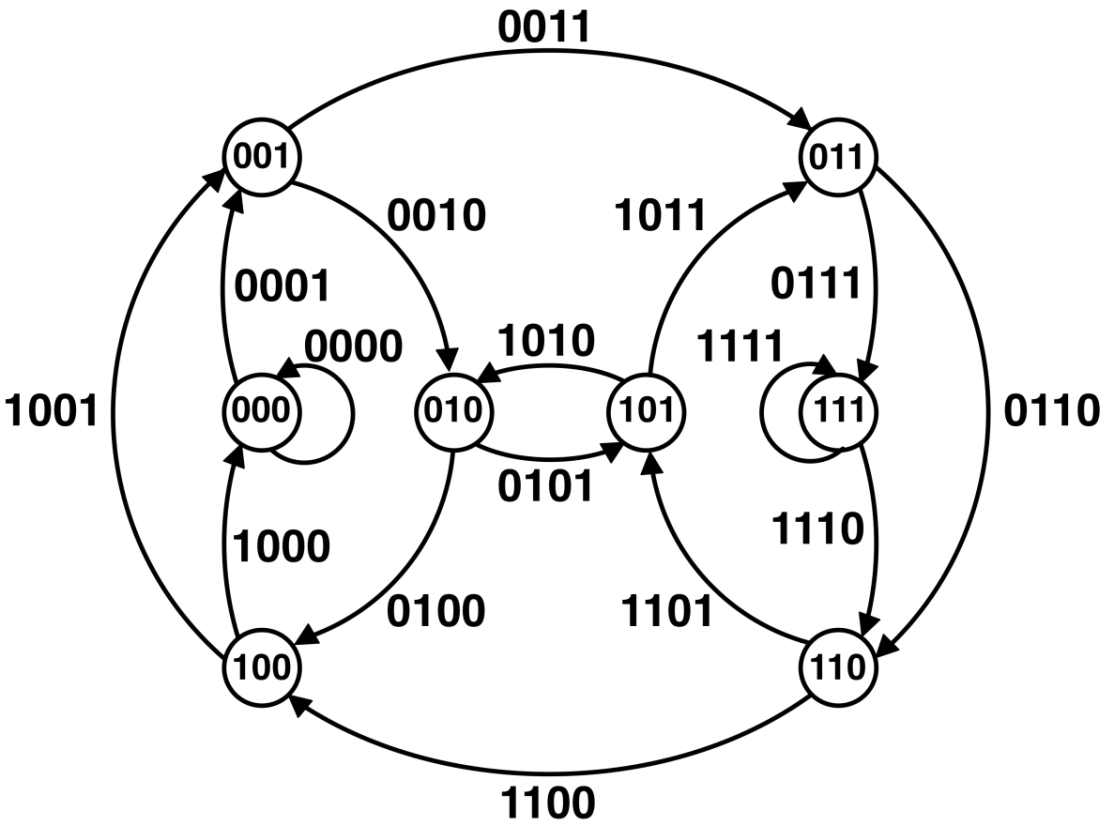
A graph is balanced if indegree = outdegree for each node

Is the Graph for 4-Universal String Balanced?



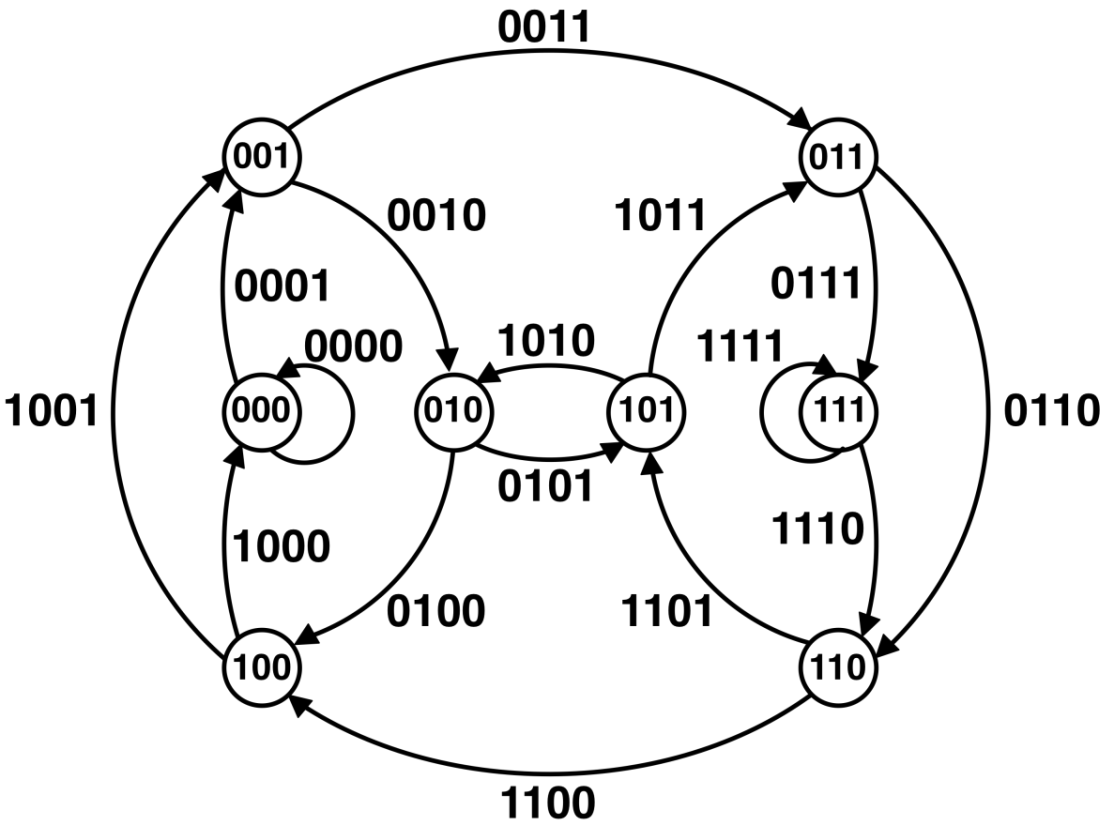
Euler's Theorem

- Every Eulerian graph is balanced



Euler's Theorem

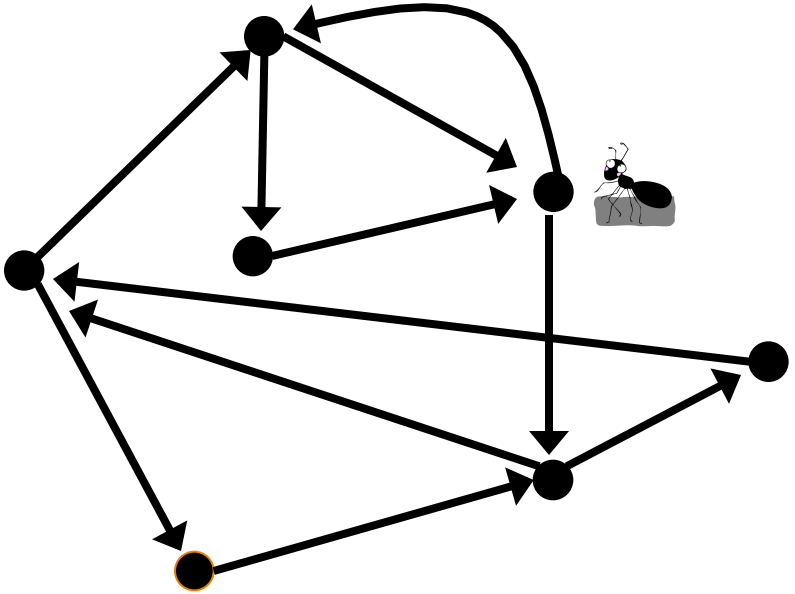
- Every Eulerian graph is balanced
- **Every balanced* graph is Eulerian**



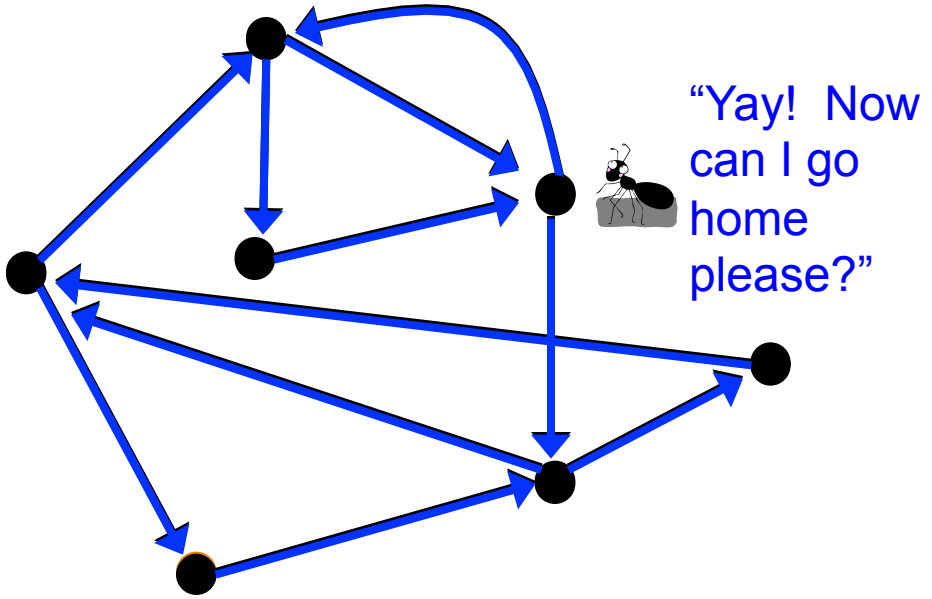
(*) and strongly connected, of course!

Recruiting an Ant to Prove Euler's Theorem

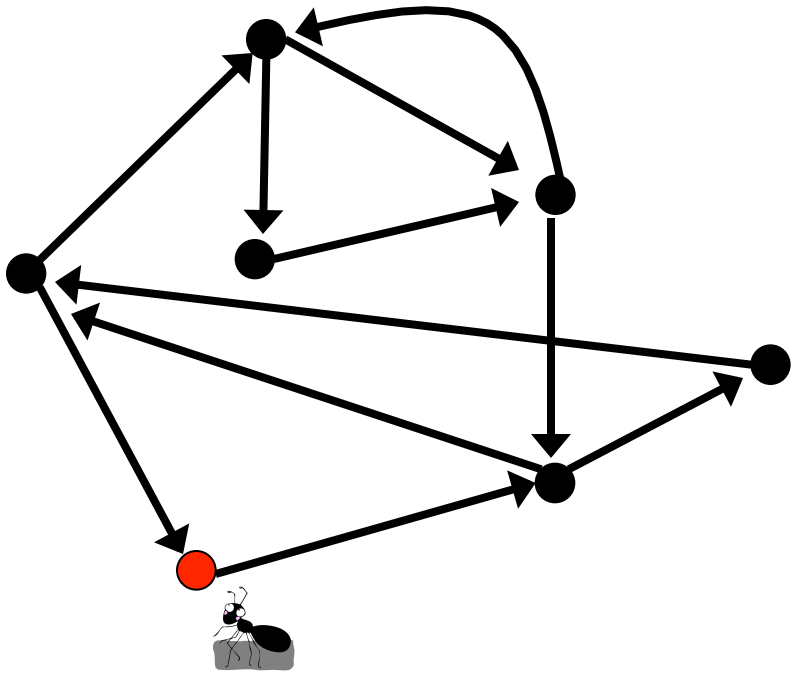
Let an ant randomly walk through the graph.
The ant cannot use the same edge twice!



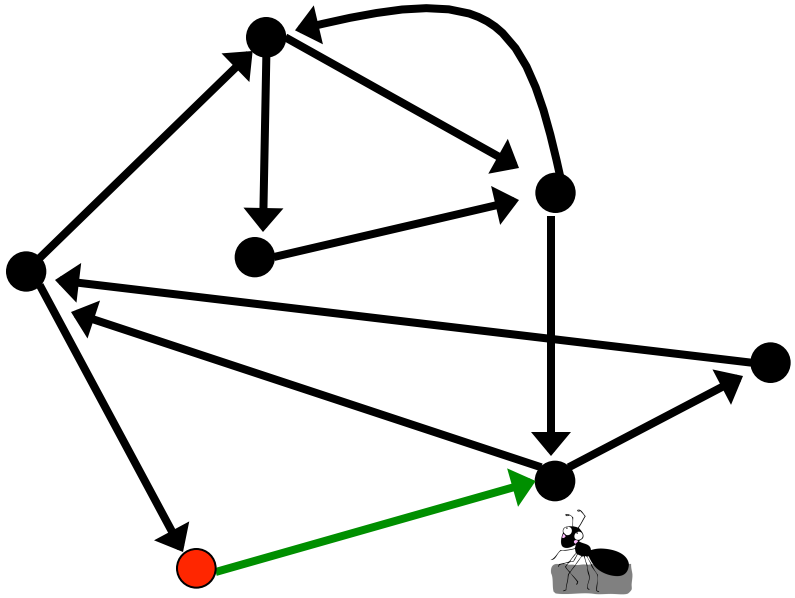
If Ant Was a Genius…



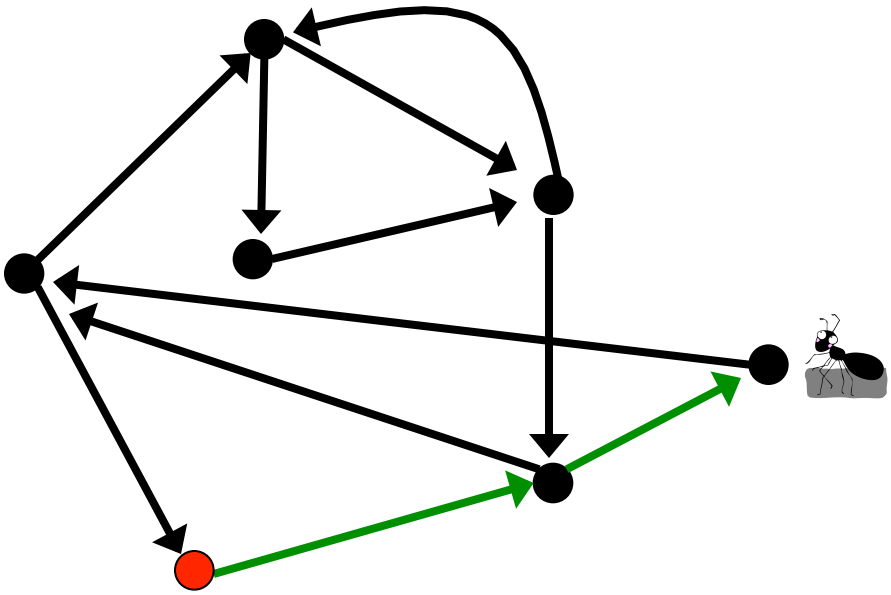
A Less Intelligent Ant Would Randomly Choose a Node and Start Walking…



Walking…

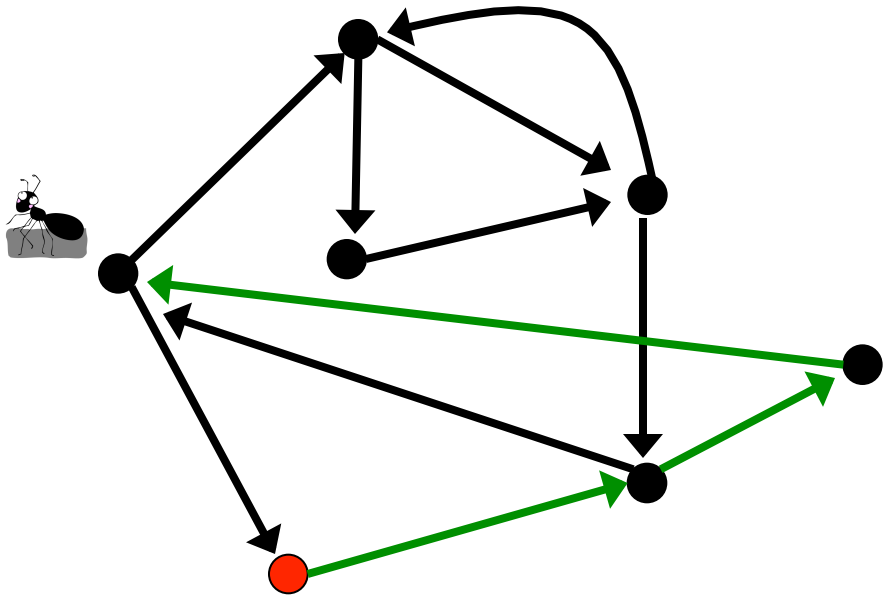


Walking… and Walking…

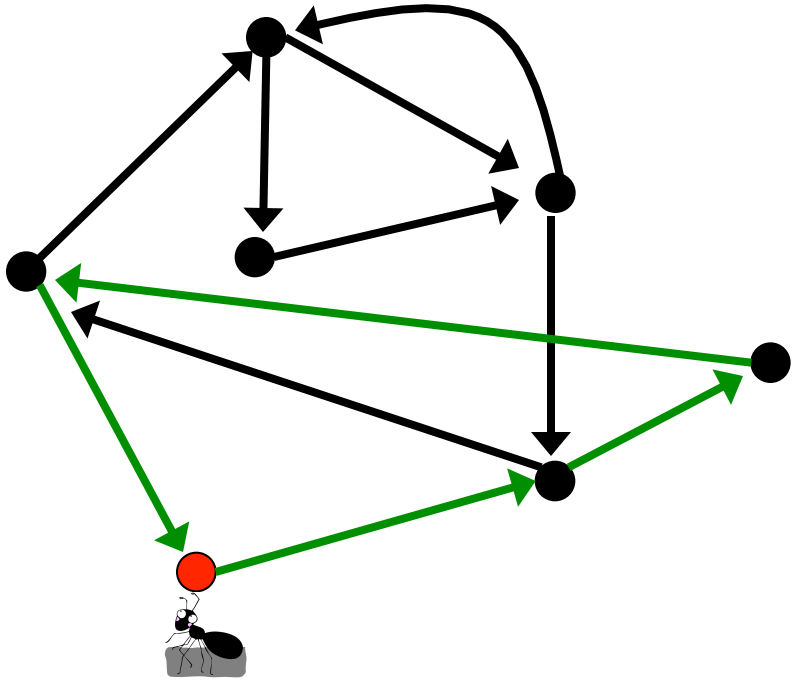


Walking… and Walking… and Walking…

Can it get stuck? **In what node?**

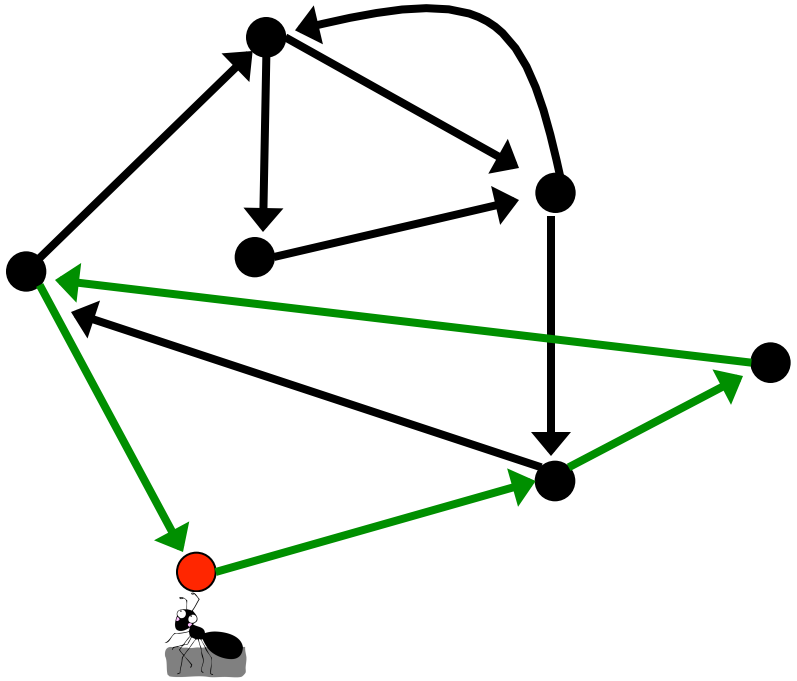


The Ant Can Only Get Stuck at the Starting Node



The Ant Has Completed a Cycle BUT has not Proven Euler's theorem yet…

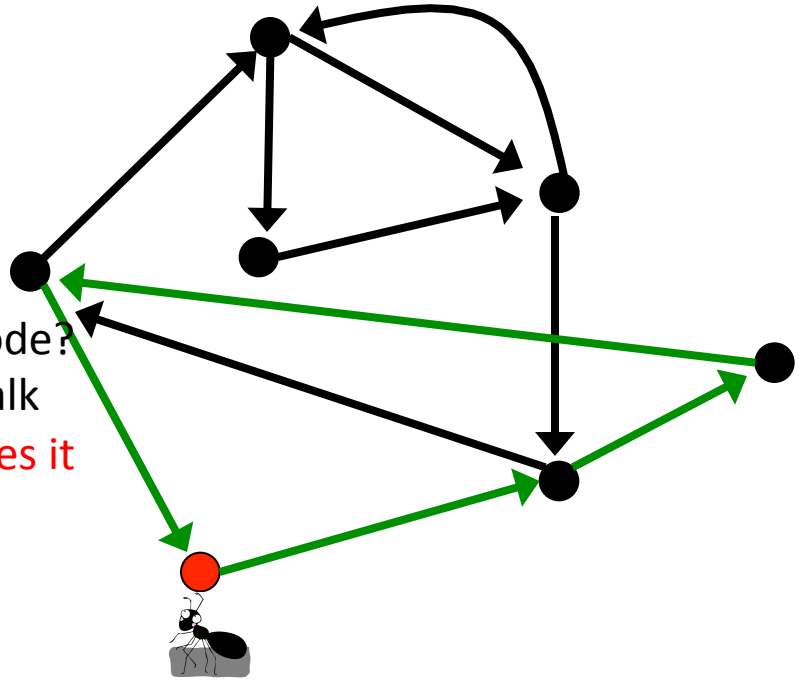
The constructed cycle is not Eulerian. **Can we enlarge it?**



Let's Start at a Different Node in the Green Cycle

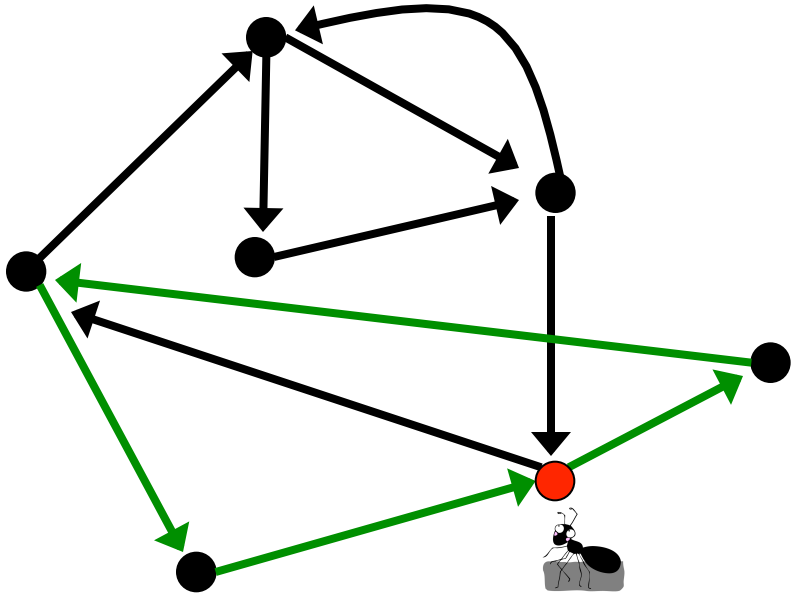
Let's start at a node with still unexplored edges.

"Why should I start at a different node? Backtracking? I'm not evolved to walk backwards! And what difference does it make???"



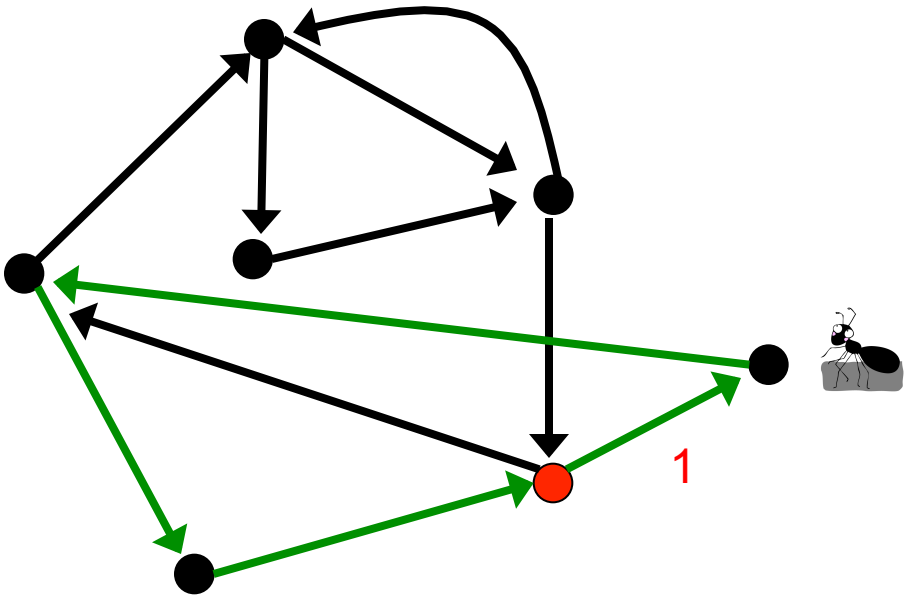
New Instructions for the Ant:

Starting at a node that has an unused edge, traverse the already constructed (green cycle) and return back to the starting node.



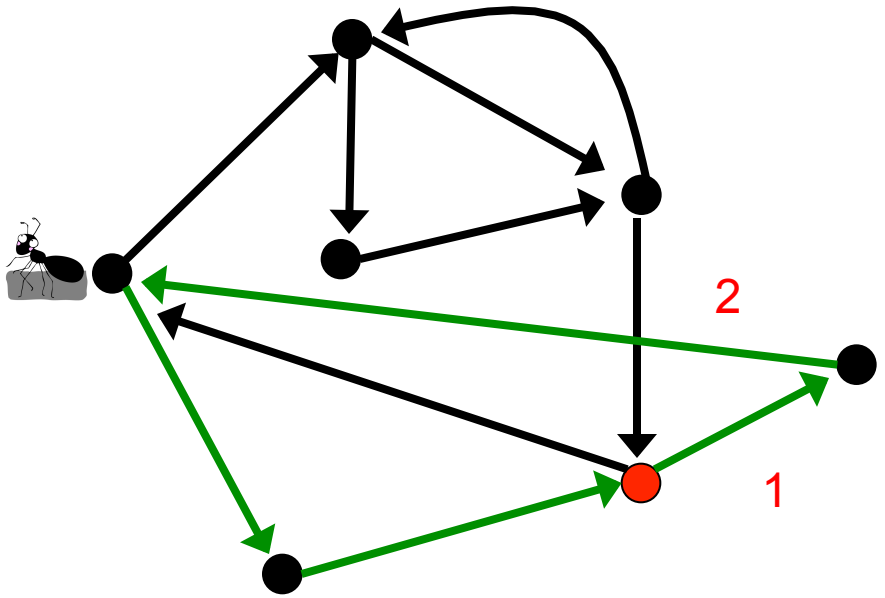
An Ant Traversing Previously Constructed Cycle

Starting at a node that has an unused edge, traverse the already constructed (green cycle) and return back to the starting node.



An Ant Traversing Previously Constructed Cycle

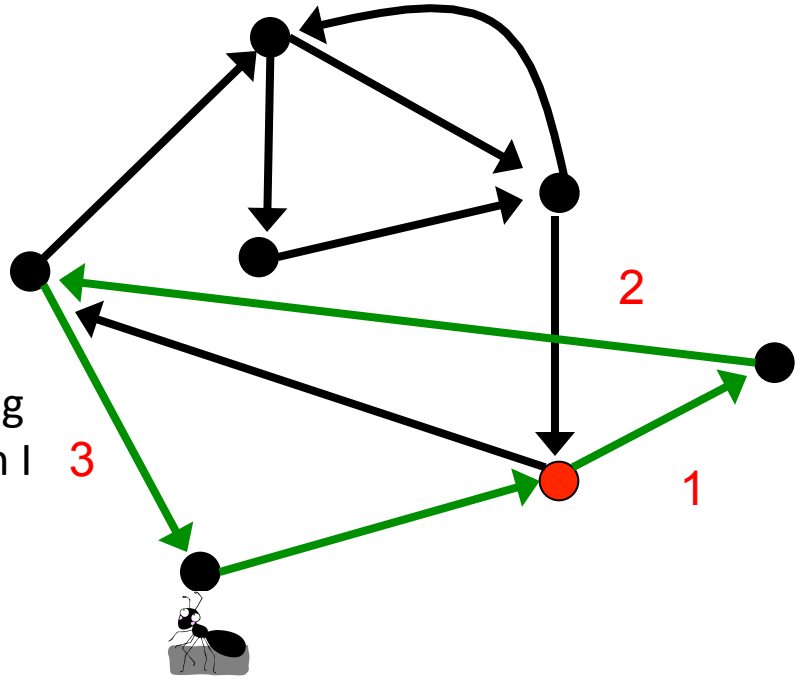
Starting at a node that has an unused edge, traverse the already constructed (green cycle) and return back to the starting node.



An Ant Traversing Previously Constructed Cycle

Starting at a node that has an unused edge, traverse the already constructed (green cycle) and return back to the starting node.

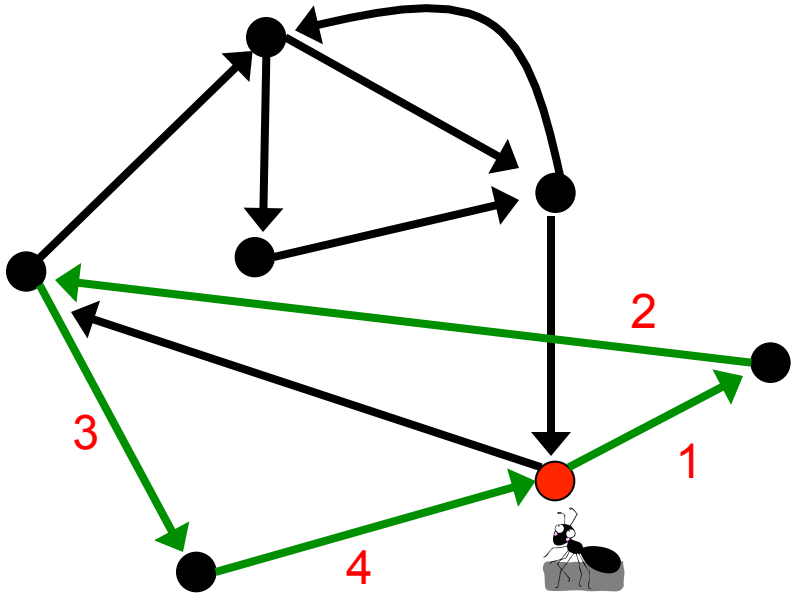
"Why do I have to walk along the same cycle again??? Can I see something new?"



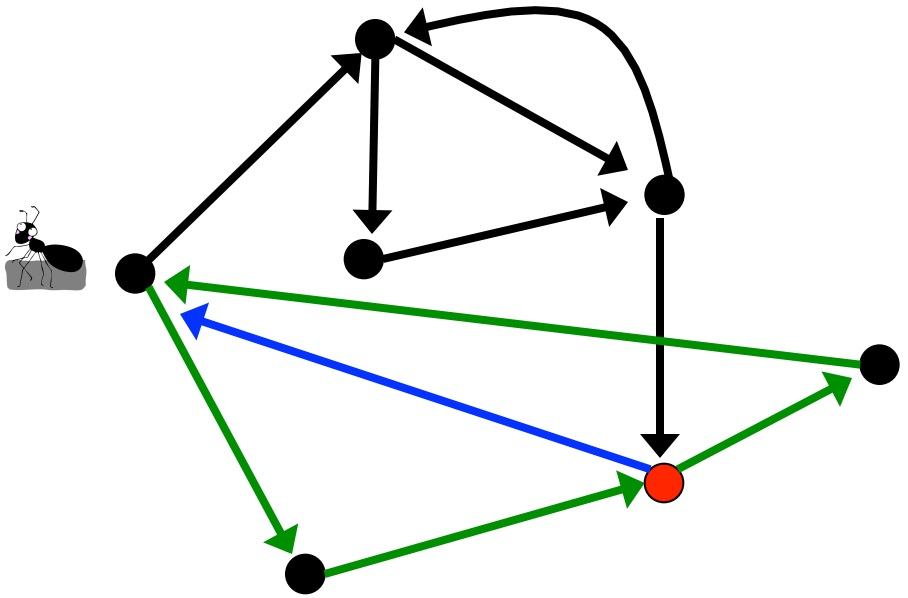
I Returned Back BUT… I Can Continue Walking!

Starting at a node that has an unused edge, traverse the already constructed (green cycle) and return back to the starting node.

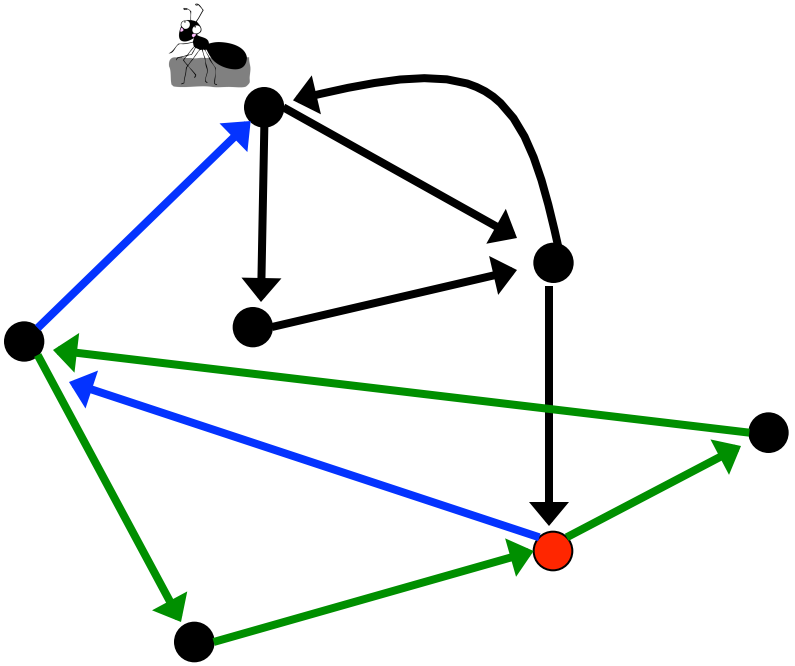
After completing the cycle, start random exploration of still untraversed edges in the graph.



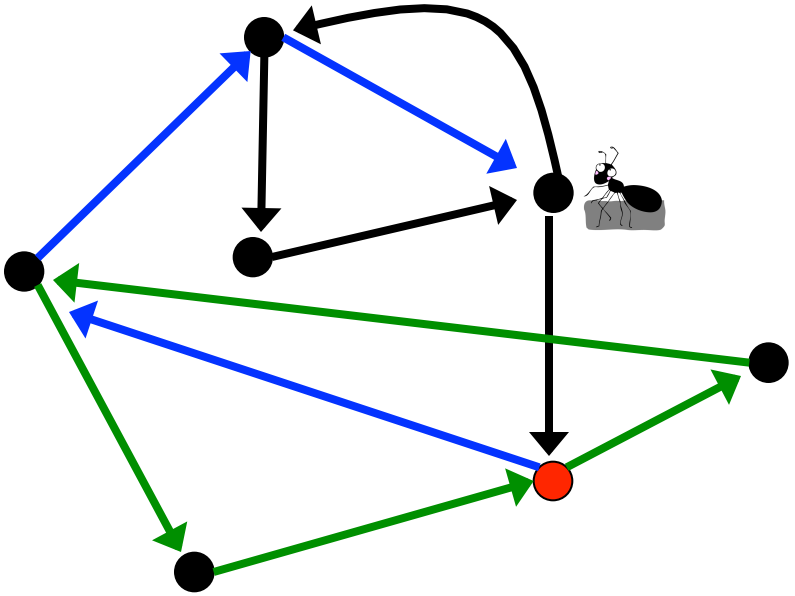
Enlarging the Previously Constructed Cycle



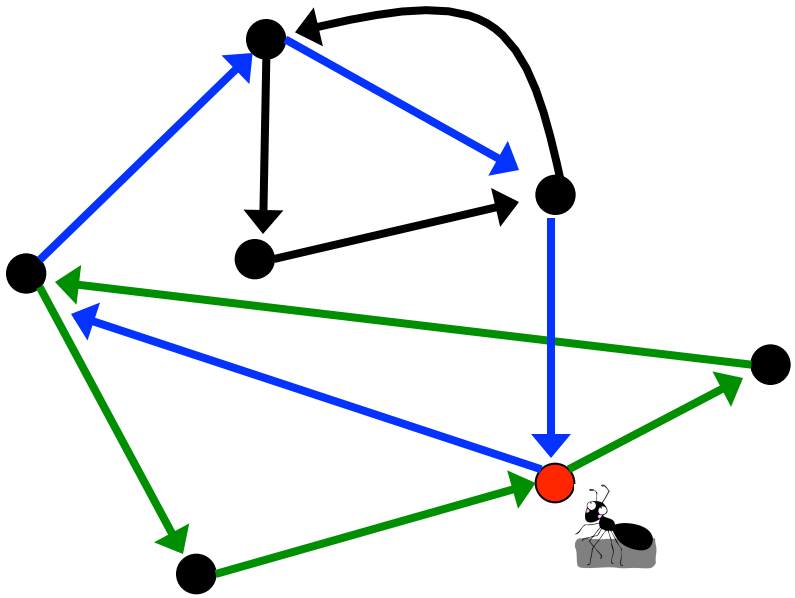
Enlarging the Previously Constructed Cycle



Enlarging the Previously Constructed Cycle



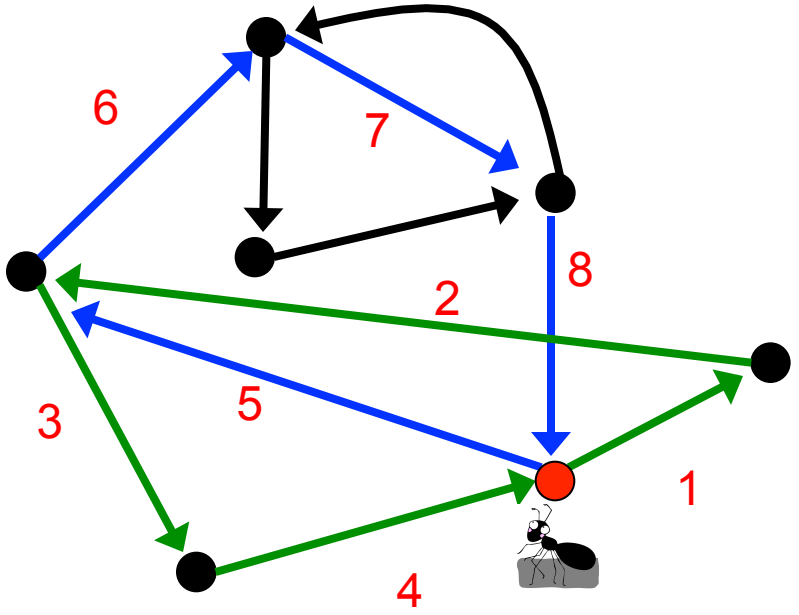
Enlarging the Previously Constructed Cycle



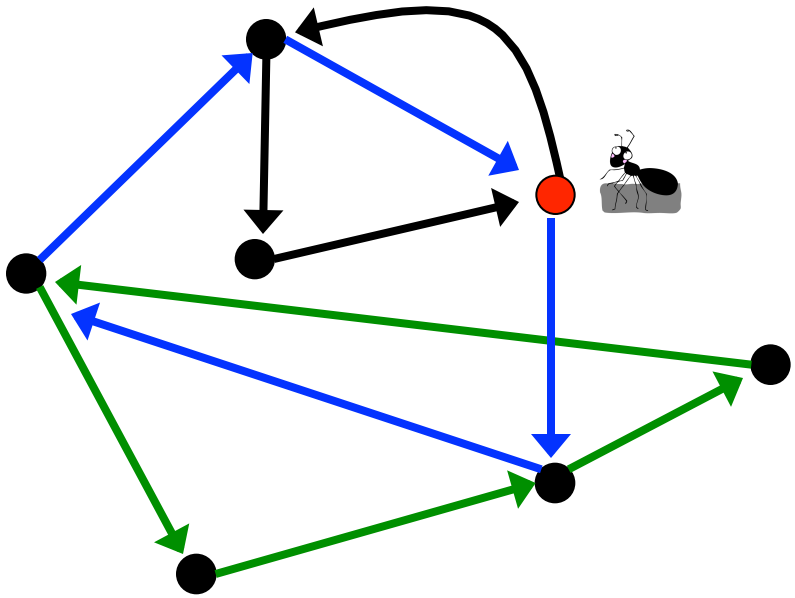
Stuck Again!

No Eulerian cycle yet… can we enlarge the green-blue cycle?

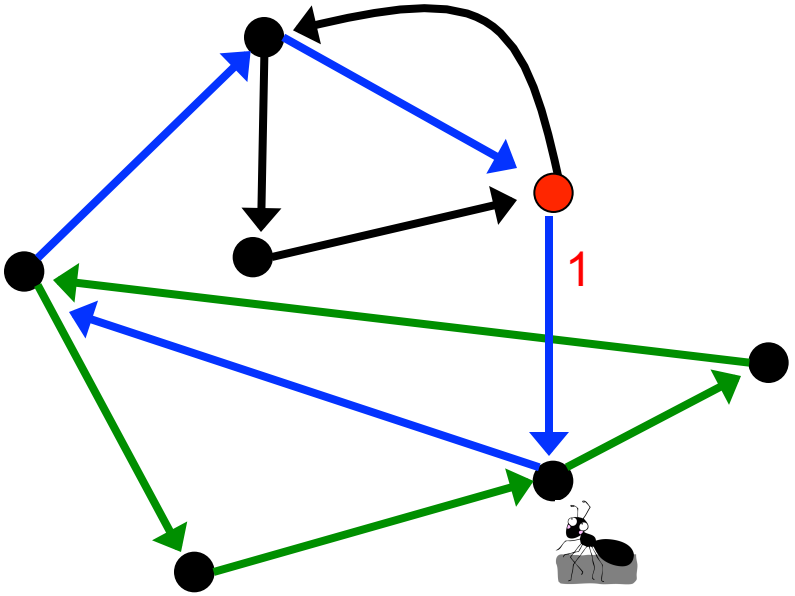
The ant should walk along the constructed cycle starting at yet another node. Which one?



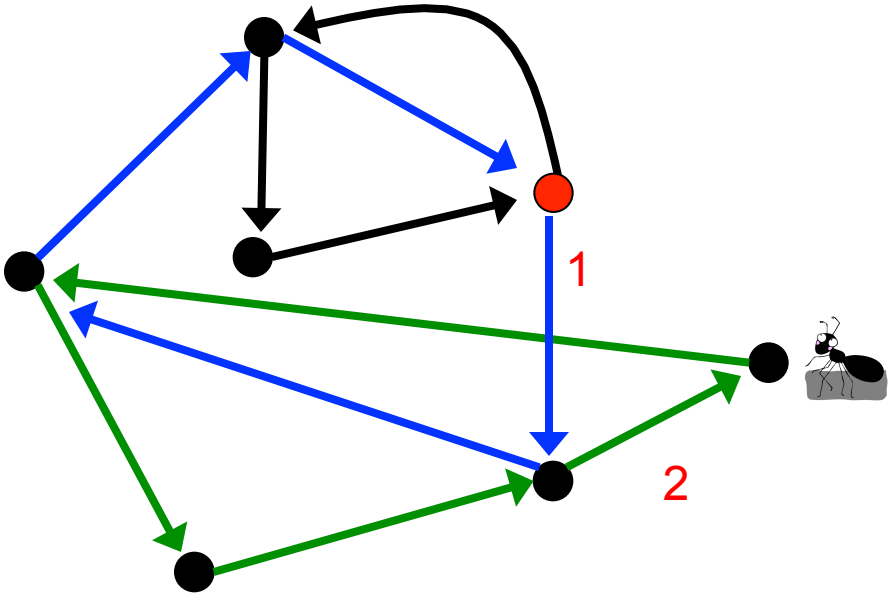
Starting at a New Node, Again…



Traversing the Previously Constructed Green-Blue Cycle

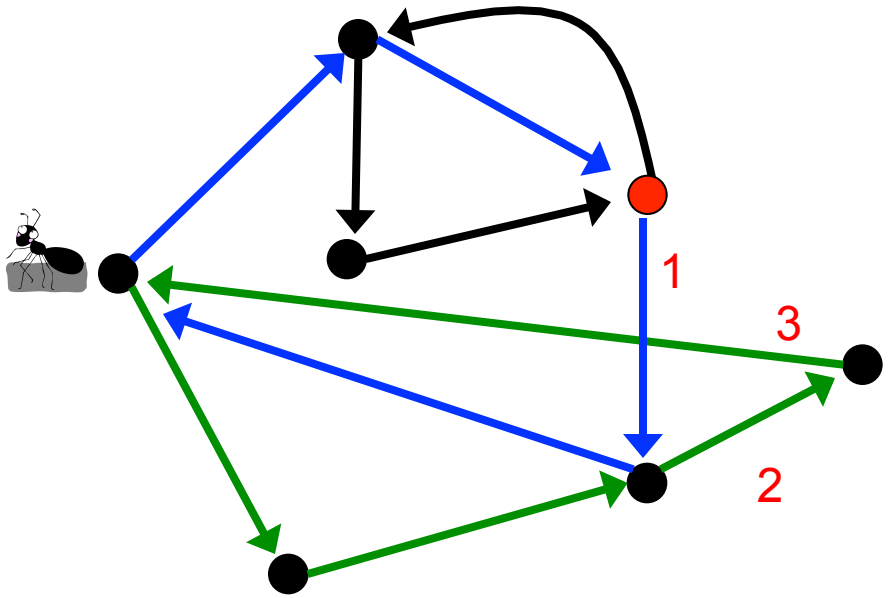


Traversing the Previously Constructed Green-Blue Cycle

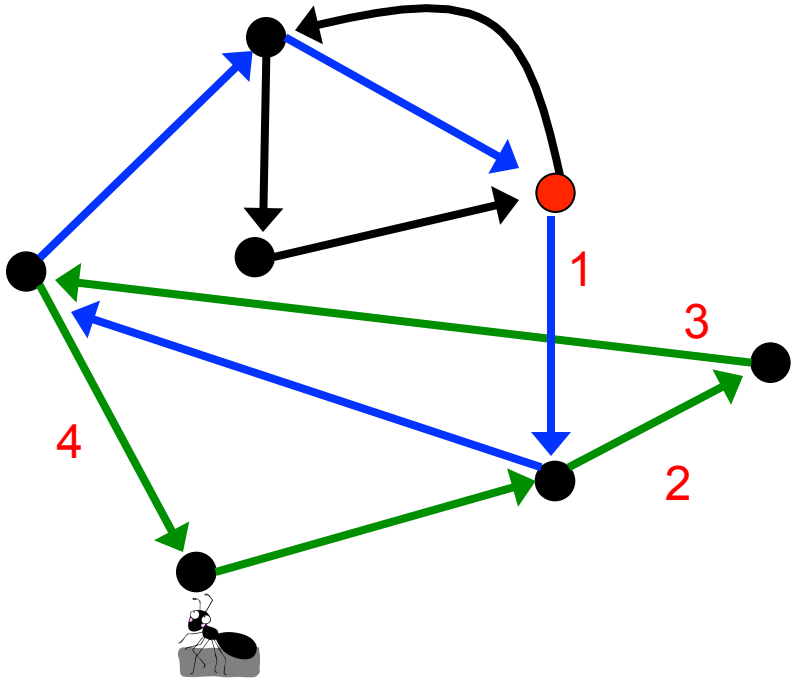


Traversing the Previously Constructed Green-Blue Cycle

"I hate to traverse the same cycle! What difference does it make where I start my walk???

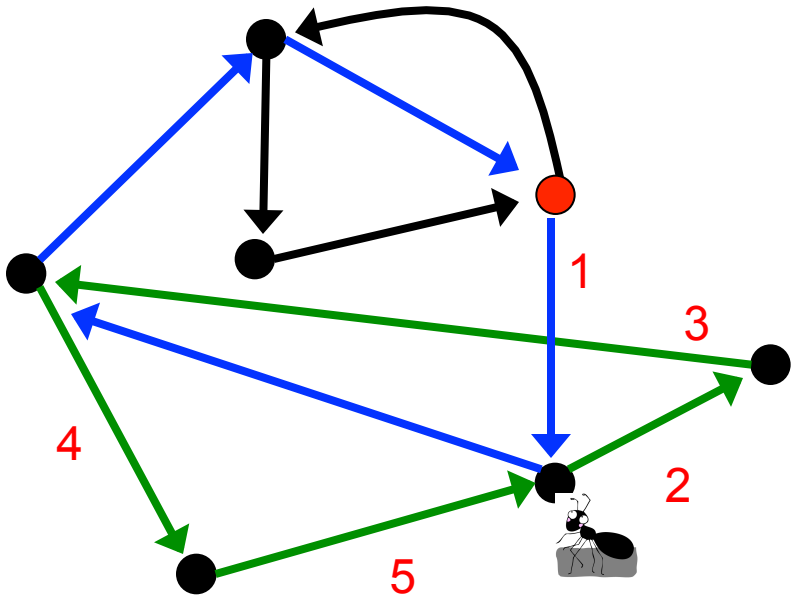


Traversing the Previously Constructed Green-Blue Cycle

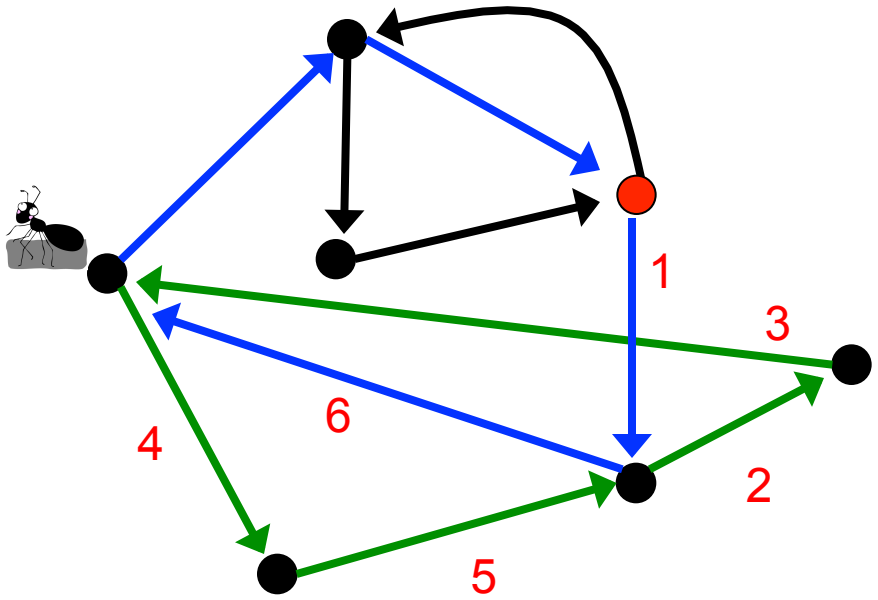


"These instructions are stupid…"

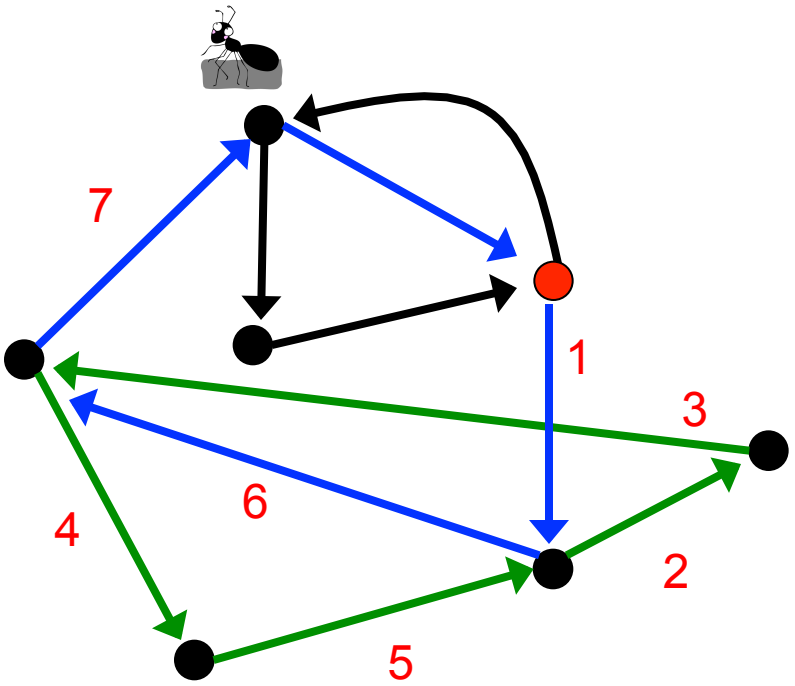
Traversing the Previously Constructed Green-Blue Cycle



Traversing the Previously Constructed Green-Blue Cycle

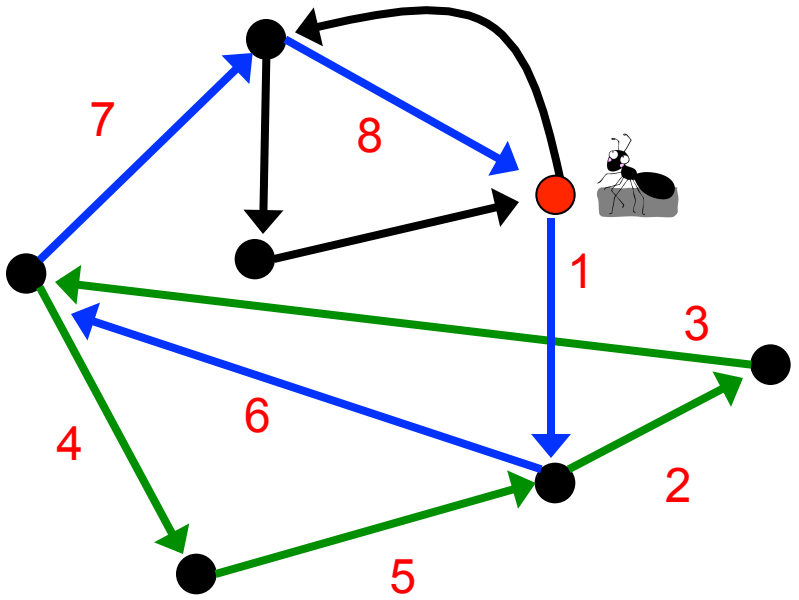


Traversing the Previously Constructed Green-Blue Cycle

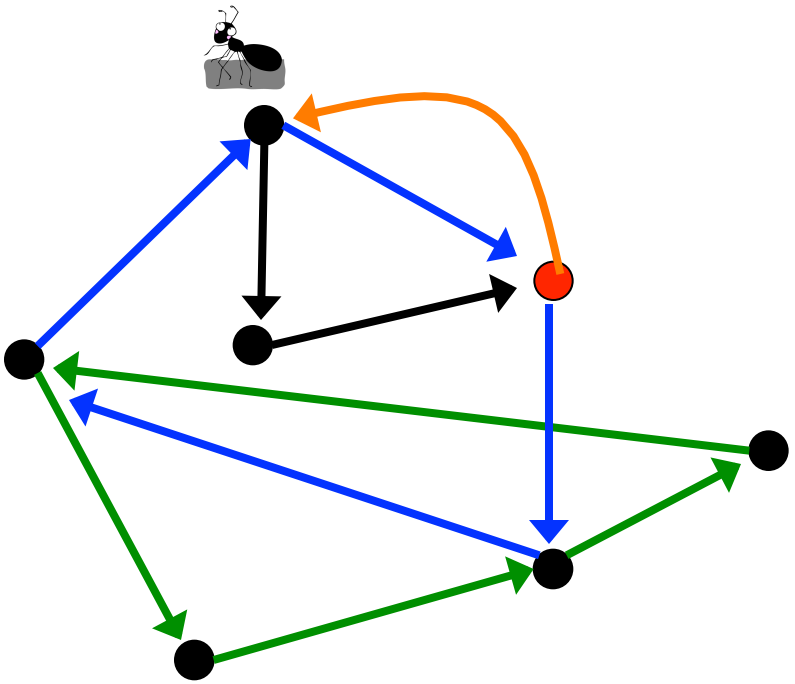


I Returned Back BUT… I Can Continue Walking!

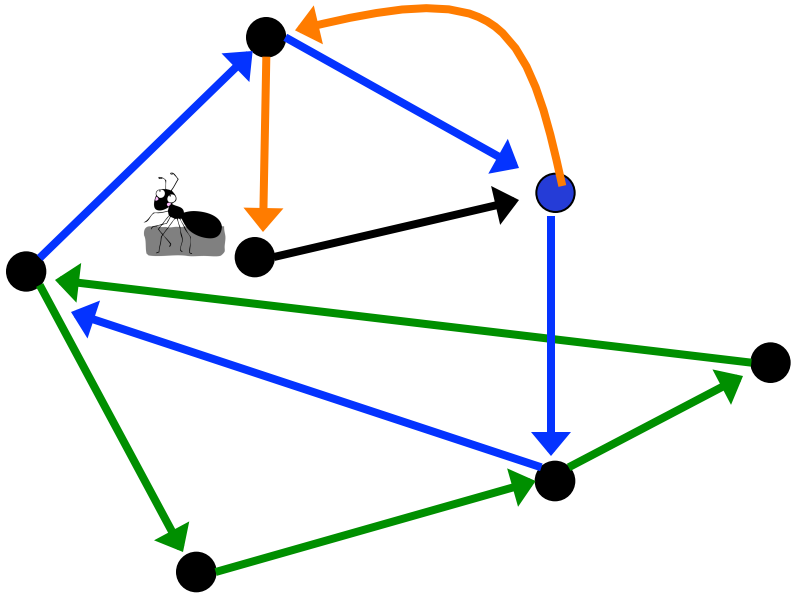
"Hmm, maybe these
instructions were no
that stupid…"



Enlarging the Green-Blue Cycle

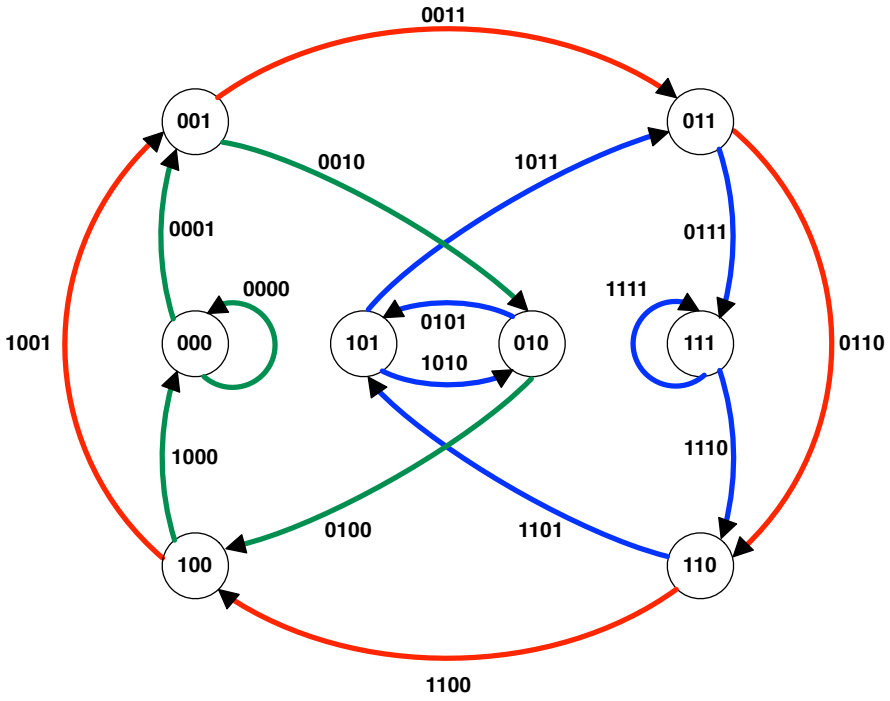


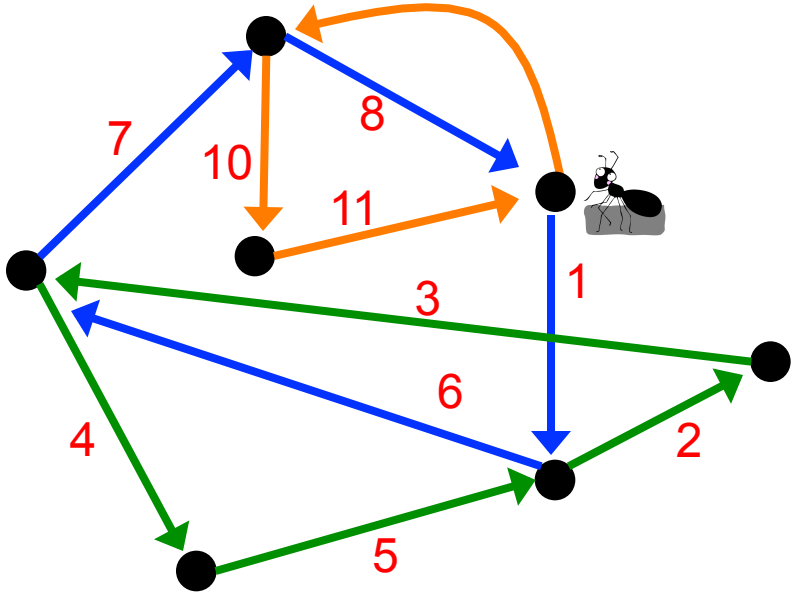
Enlarging the Green-Blue Cycle



I Proved Euler's Theorem! Can I Go Home Please?

EulerianCycle(Balanced*Graph*)
  form a *Cycle* by randomly walking in *BalancedGraph* (avoiding already visited edges)
    while *Cycle* is not Eulerian
      select a node newStart in Cycle with still unexplored outgoing edges
      form a *Cycle'* by traversing *Cycle* from newStart and randomly walking afterwards
      Cycle ← *Cycle'*
  return *Cycle*

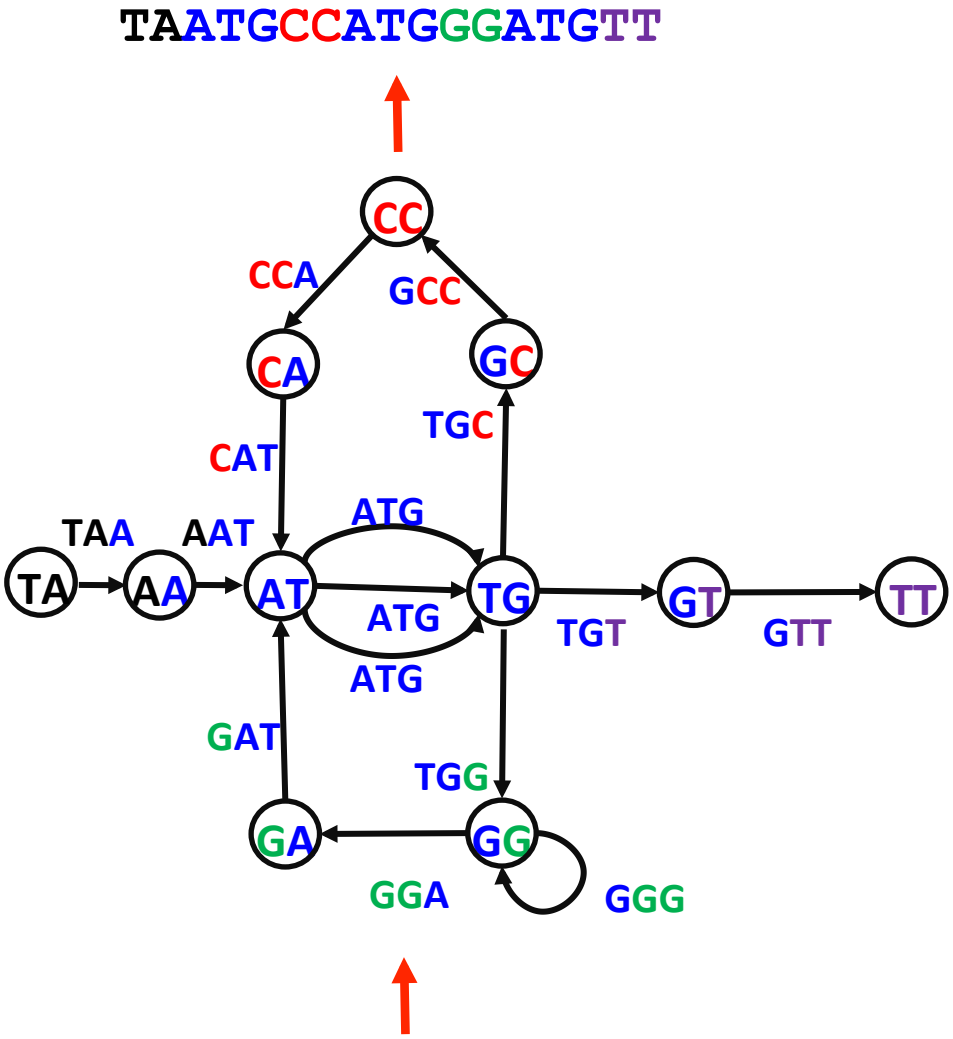




Outline

- What Is Genome Sequencing?
- Exploding Newspapers
- The String Reconstruction Problem
- String Reconstruction as a Hamiltonian Path Problem
- String Reconstruction as an Eulerian Path Problem
- Similar Problems with Different Fates
- De Bruijn Graphs
- Euler's Theorem
- Assembling Read-Pairs
- De Bruijn Graphs Face Harsh Realities of Assembly

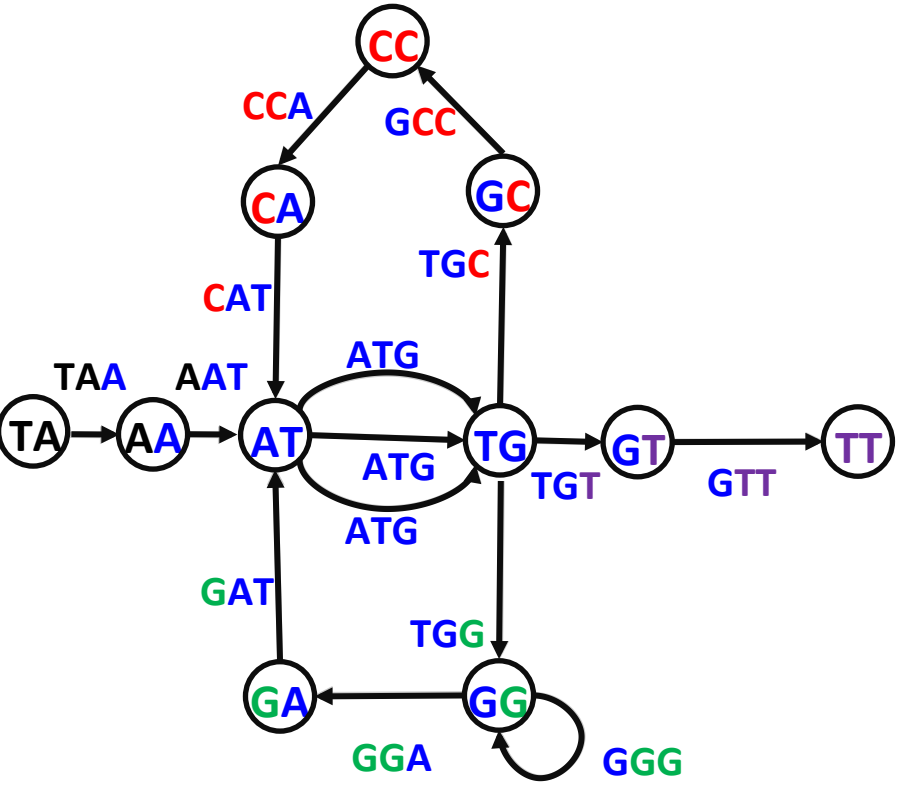
From Reads to de Bruijn Graph to Genome



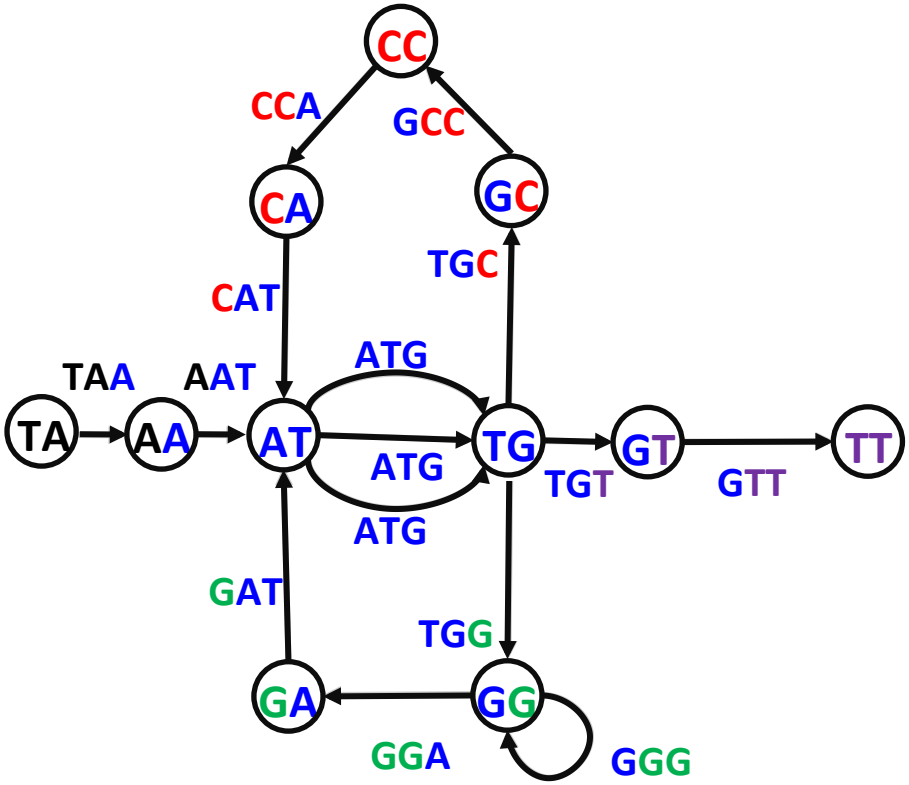
AAT ATG ATG ATG CAT CCA GAT GCC GGA GGG GTT TAA TGC TGG TGT

Multiple Eulerian Paths

TAATGCCATGGGATGTT

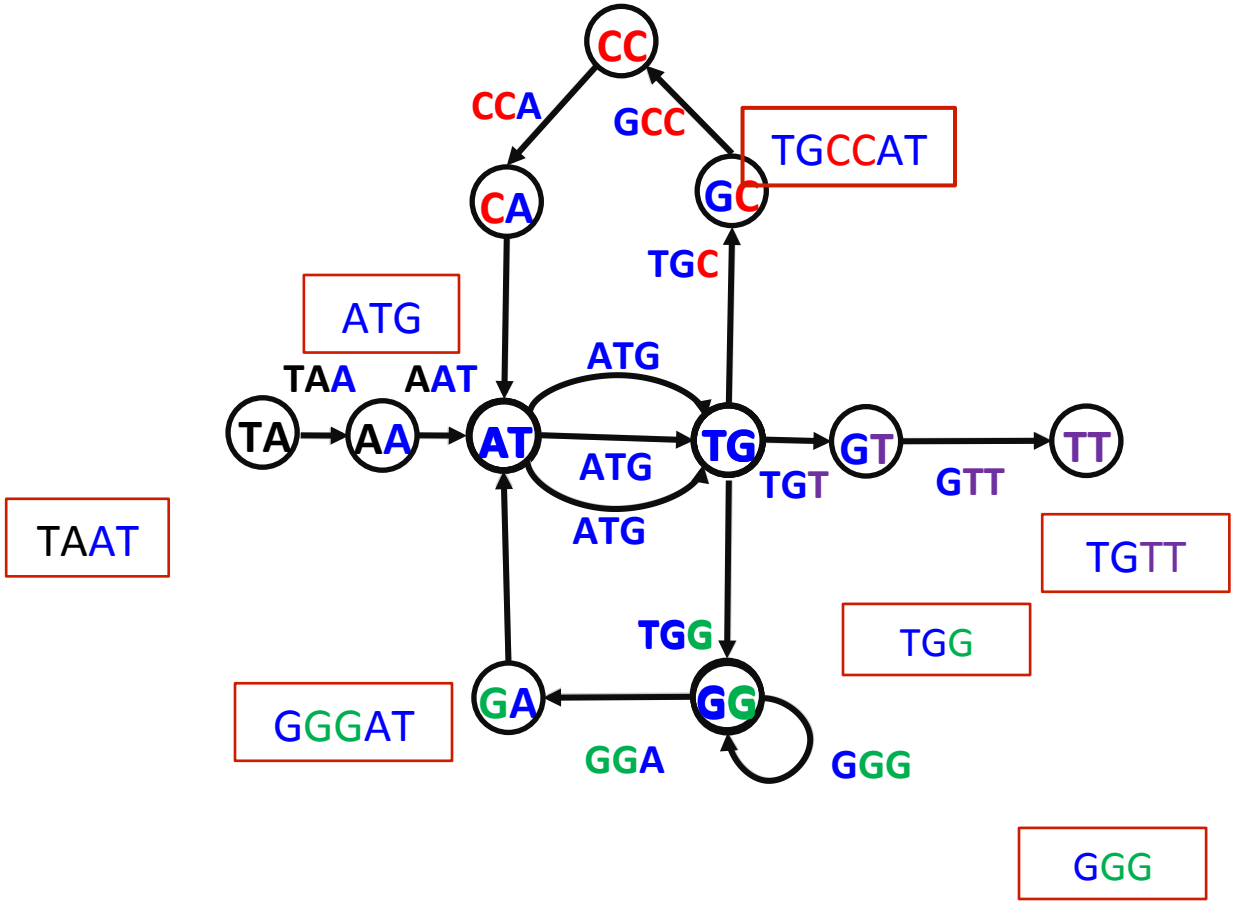


TAATGGGATGCCATGTT

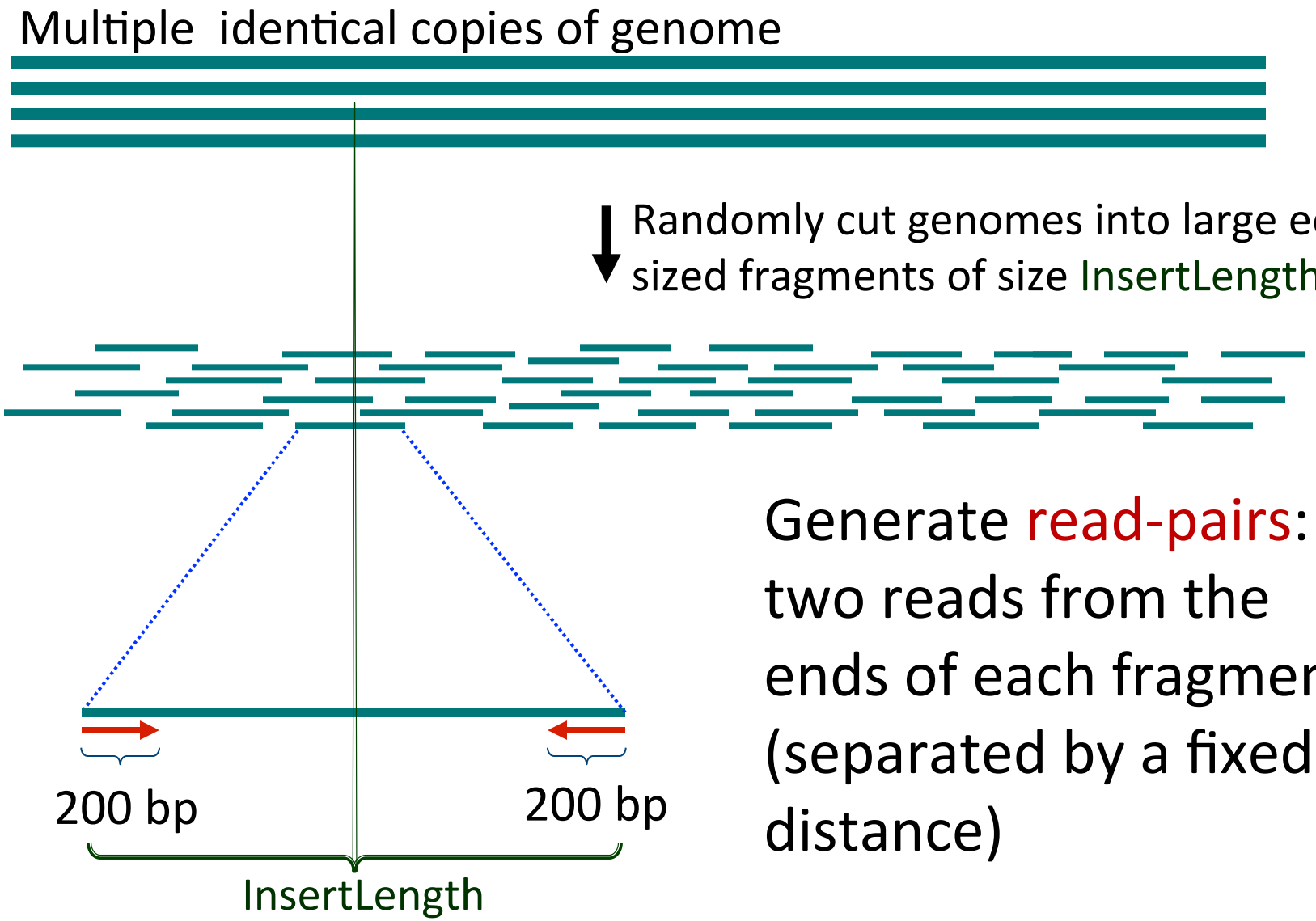


Breaking Genome into Contigs

TAATGCCATGGGATGTT

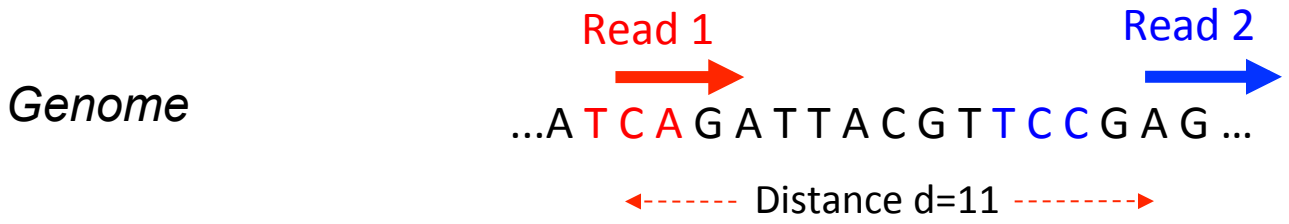


DNA Sequencing with Read-pairs



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

From *k*-mers to Paired *k*-mers



A paired k-mer is a pair of k-mers at a fixed distance d apart in Genome.
E.g. TCA and TCC are at distance d=11 apart.

Disclaimers:

1. In reality, Read1 and Read2 are typically sampled from different strands:
   (→ ..... ← rather than → ..... →)
2. In reality, the distance d between reads is measured with errors.

What is PairedComposition(**TAATGCCATGGGATGTT**)?

TAA GCC

paired 3-mer

What is PairedComposition(**TAATGCCATGGGATGTT**)?

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**

PairedComposition(**TAATGCCATGGGATGTT**)

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

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

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

Representing PairedComposition in lexicographic order

String Reconstruction from Read-Pairs Problem

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

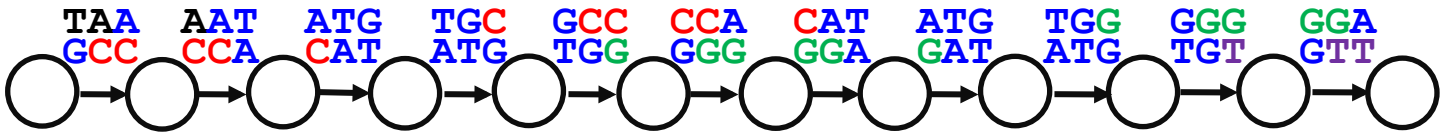
- **Input.** A collection of paired *k*-mers.
- **Output.** A string *Text* such that *PairedComposition*(*Text*) is equal to the collection of paired k-mers.

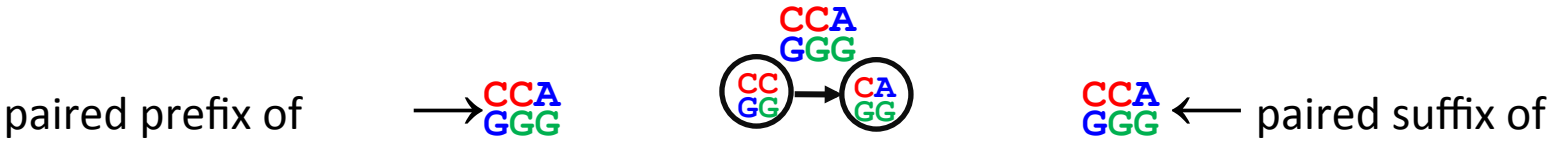
How Would de Bruijn Assemble Paired *k*-mers?

Paired de Bruijn Graphs

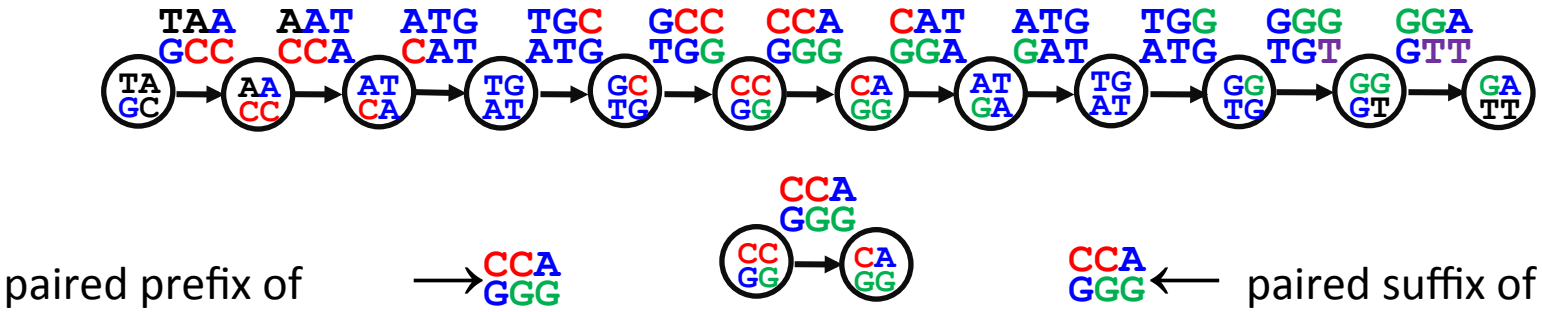
Representing Genome **TAATGCCATGGGATGTT** as a Path

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

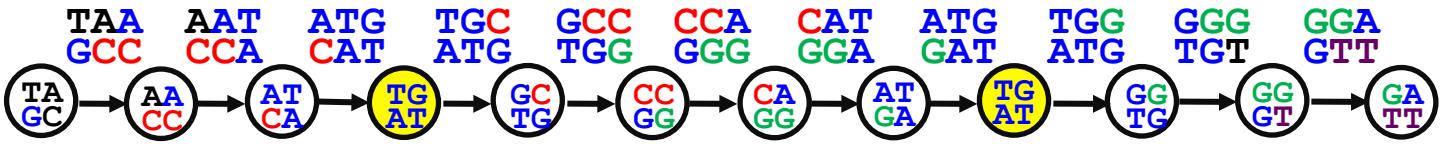




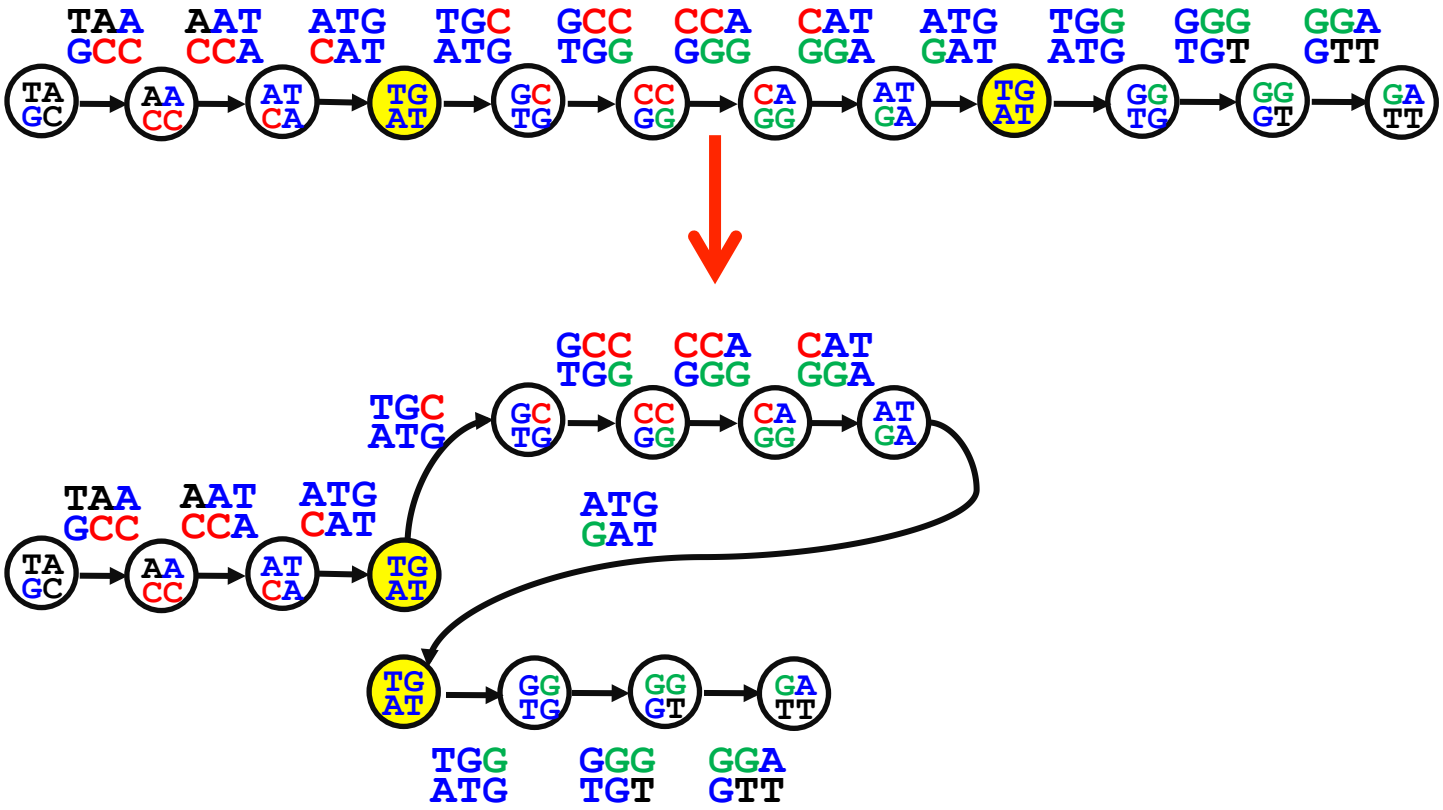
Labeling Nodes by Paired Prefixes and Suffixes



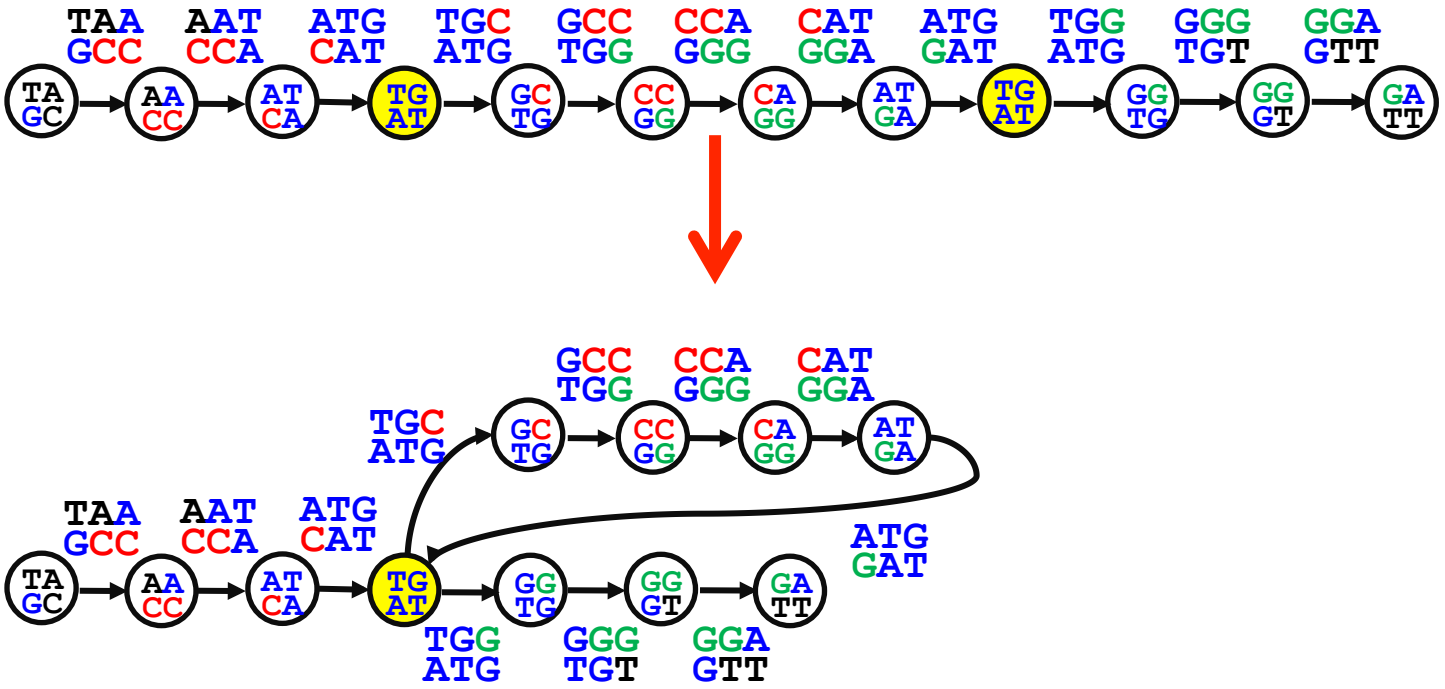
Glue nodes with identical labels



Glue nodes with identical labels

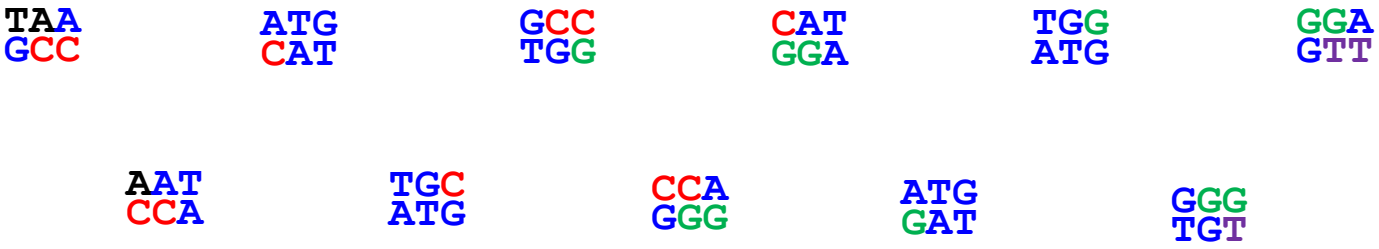


Glue nodes with identical labels

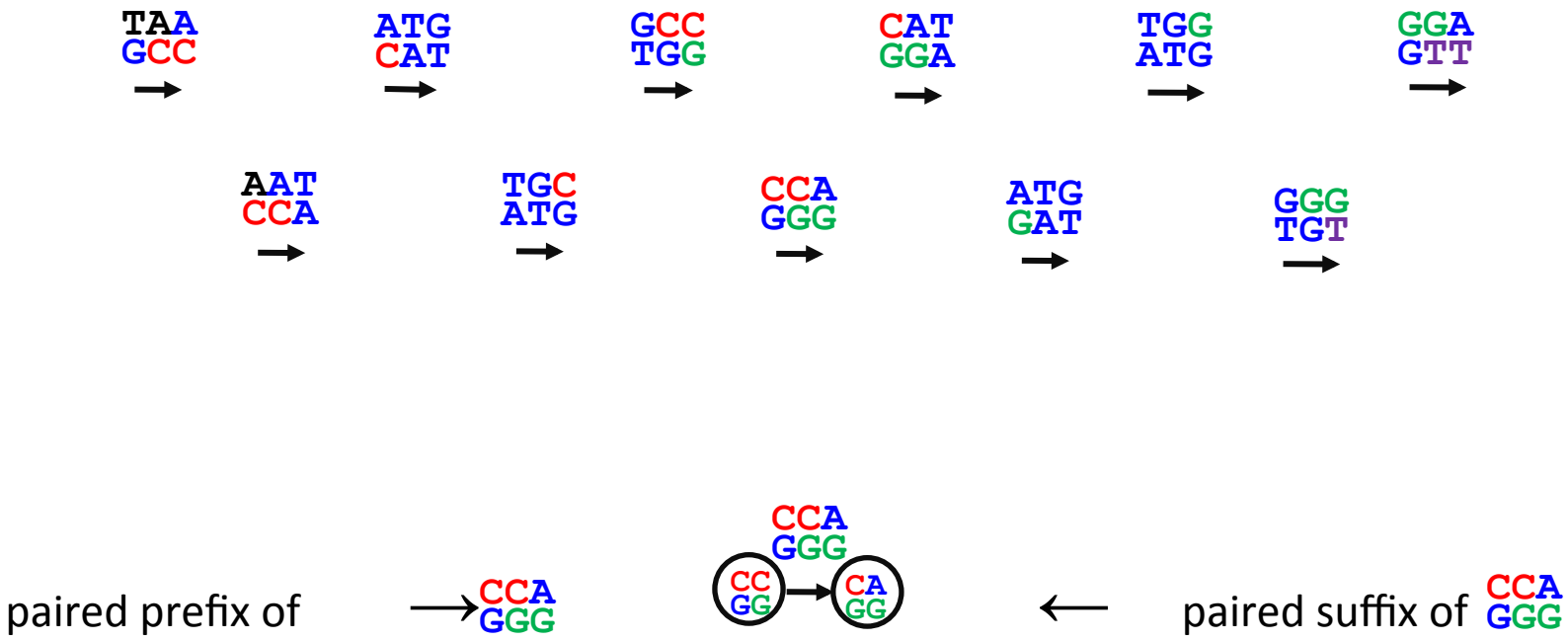


Paired de Bruijn Graph from the Genome

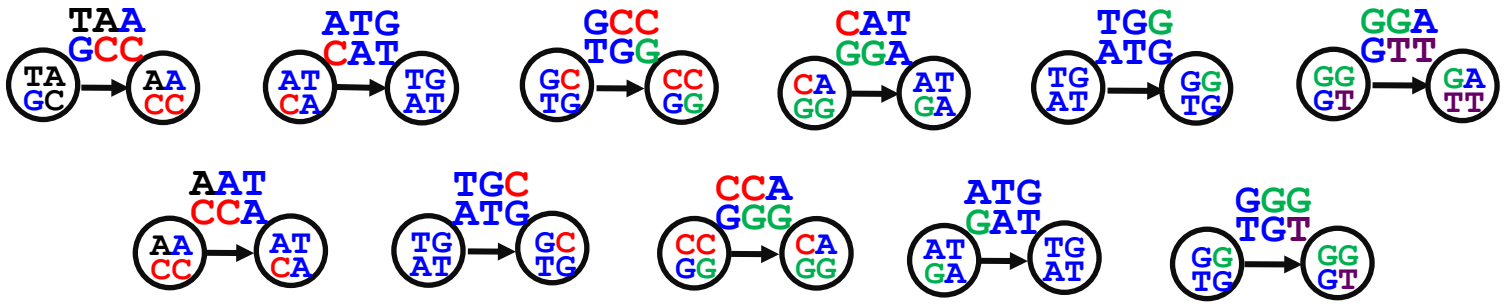
Constructing Paired de Bruijn Graph from paired k-mers

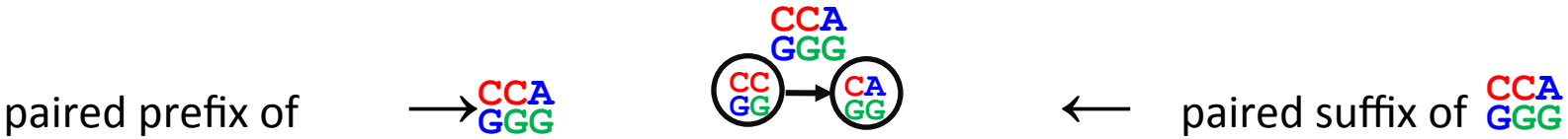


Constructing Paired de Bruijn Graph from paired k-mers

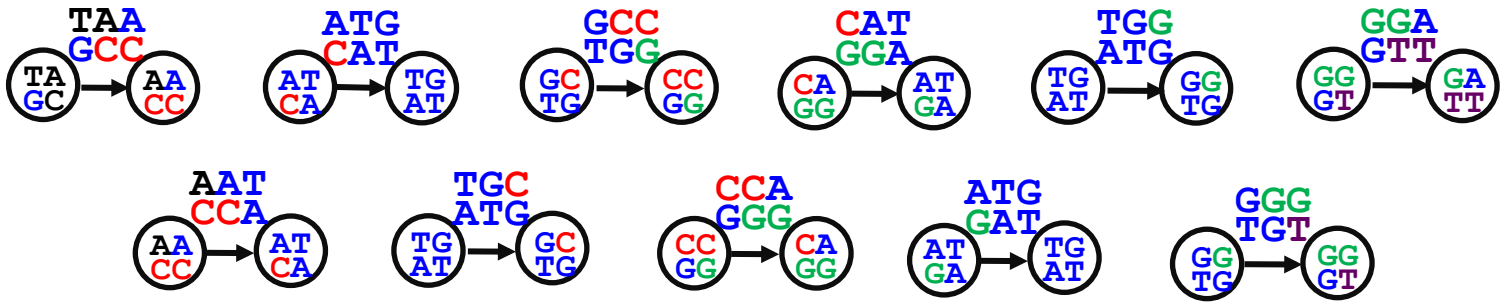


Constructing Paired de Bruijn Graph



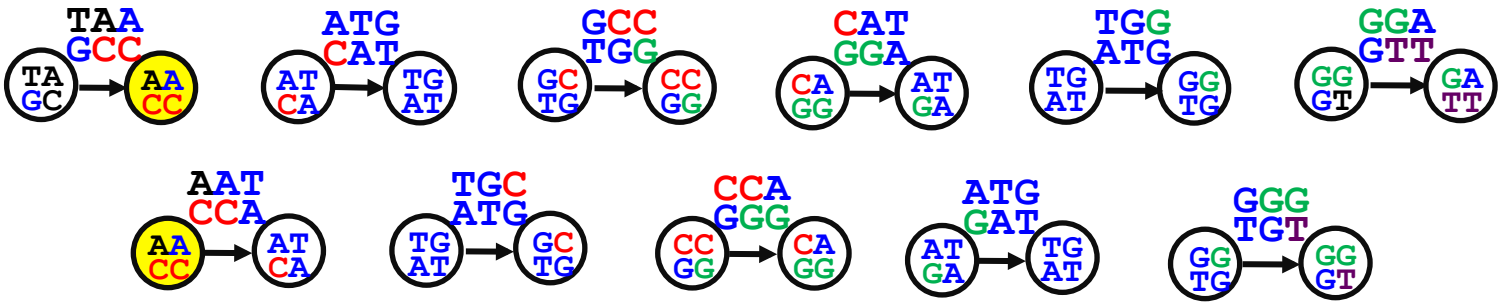


Constructing Paired de Bruijn Graph

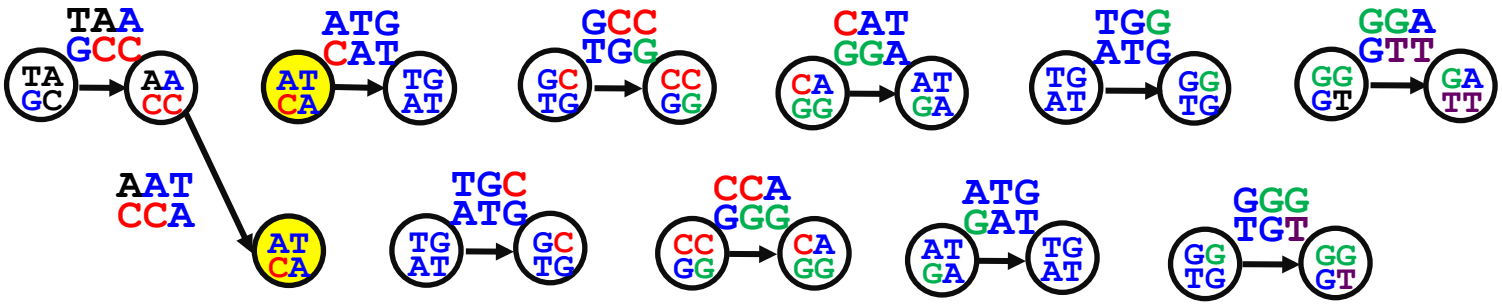


- **Paired de Bruijn graph for a collection of paired *k*-mers:**
  - Represent every paired *k*-mer as an edge between its paired prefix and paired suffix.
  - Glue **ALL** nodes with identical labels.

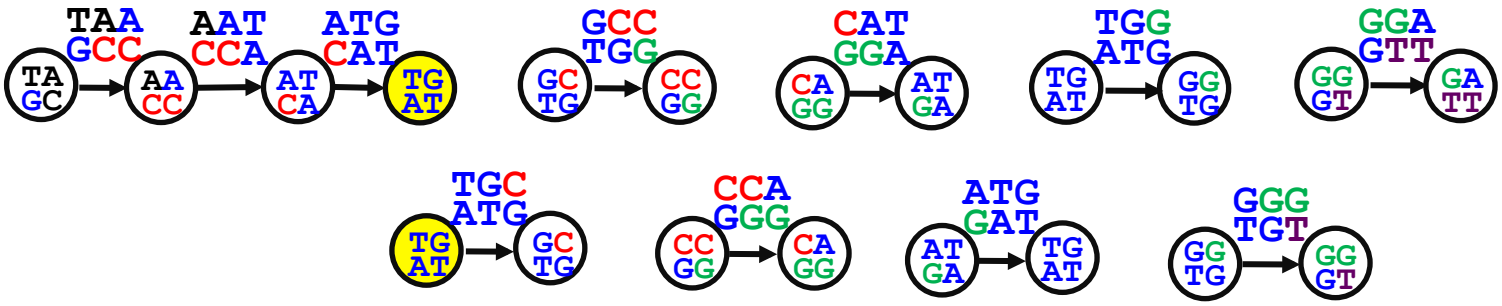
Constructing Paired de Bruijn Graph



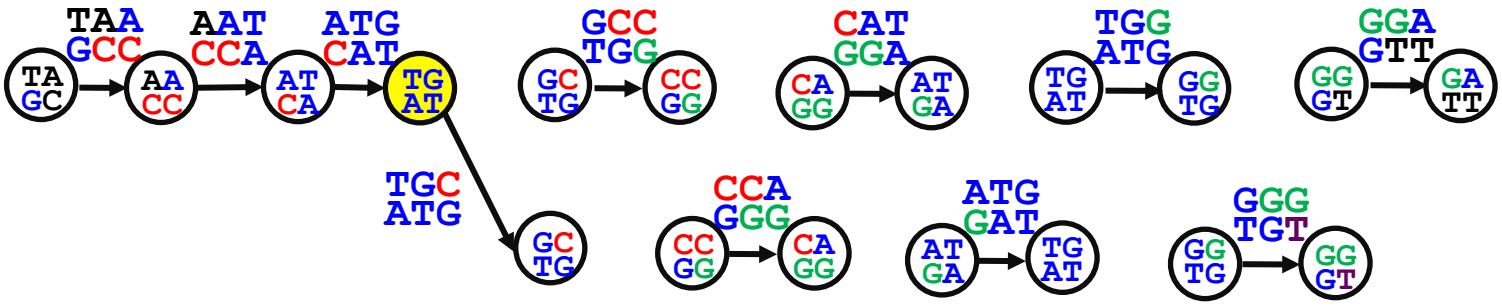
Constructing Paired de Bruijn Graph



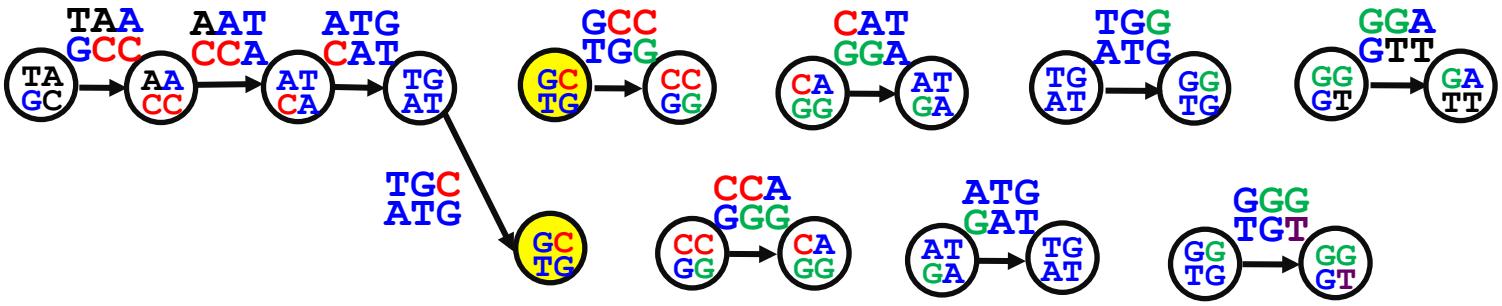
Constructing Paired de Bruijn Graph



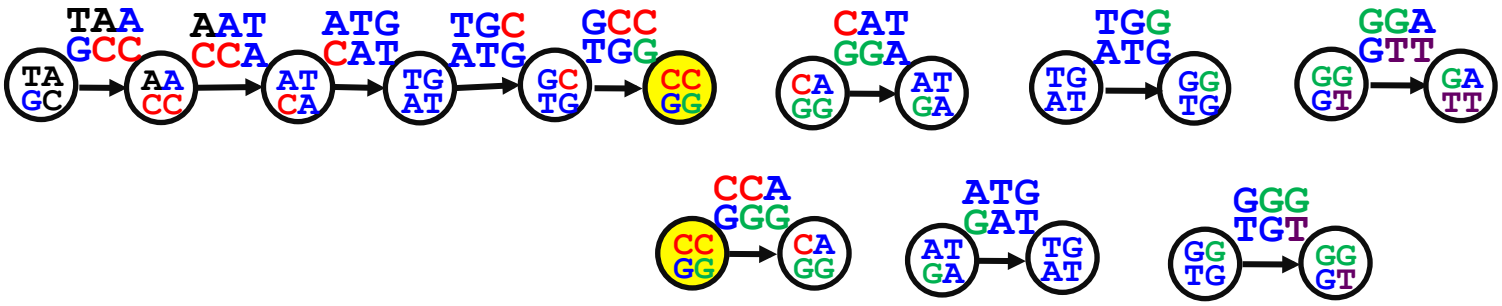
Constructing Paired de Bruijn Graph



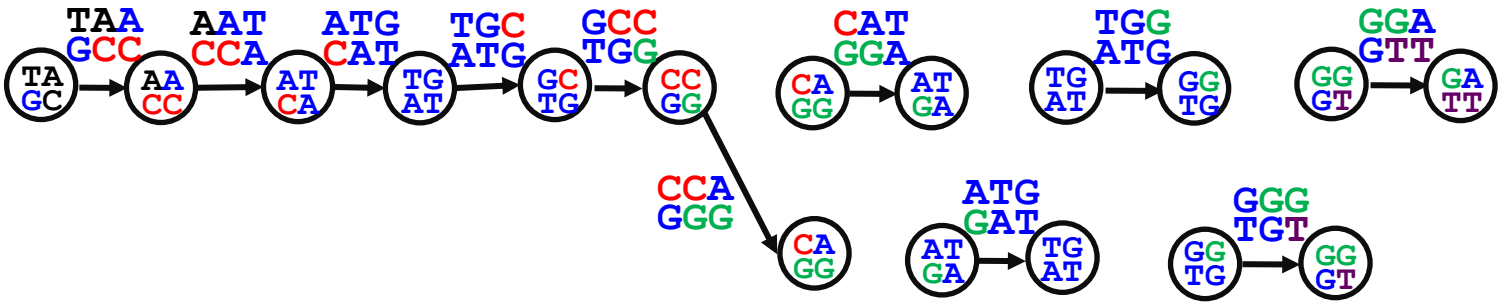
Constructing Paired de Bruijn Graph



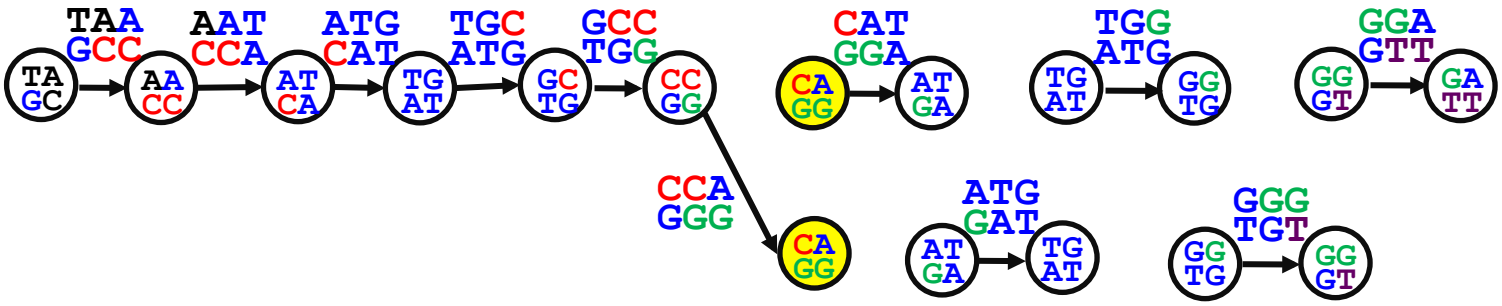
Constructing Paired de Bruijn Graph



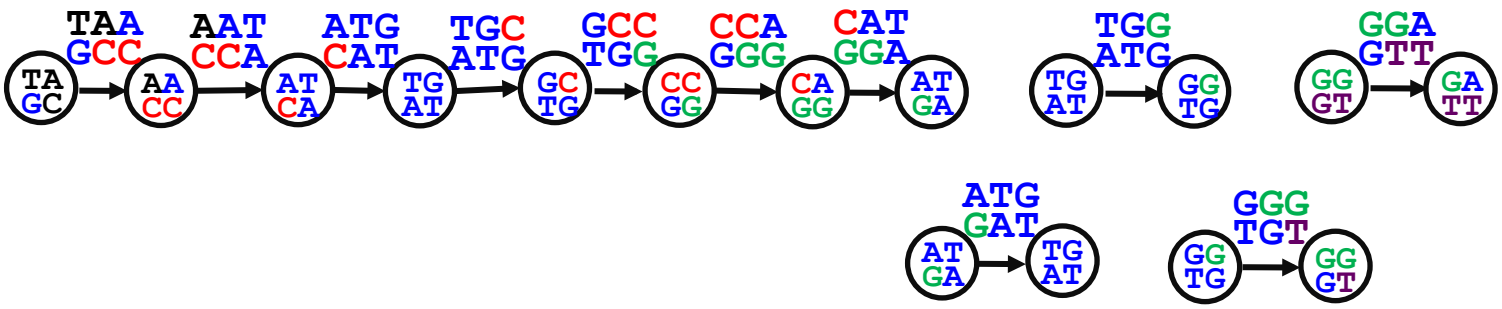
Constructing Paired de Bruijn Graph



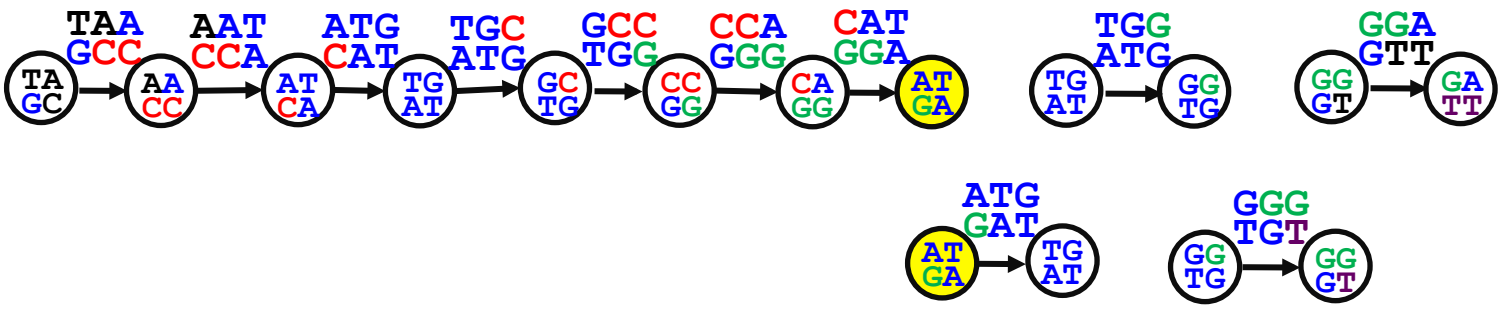
Constructing Paired de Bruijn Graph



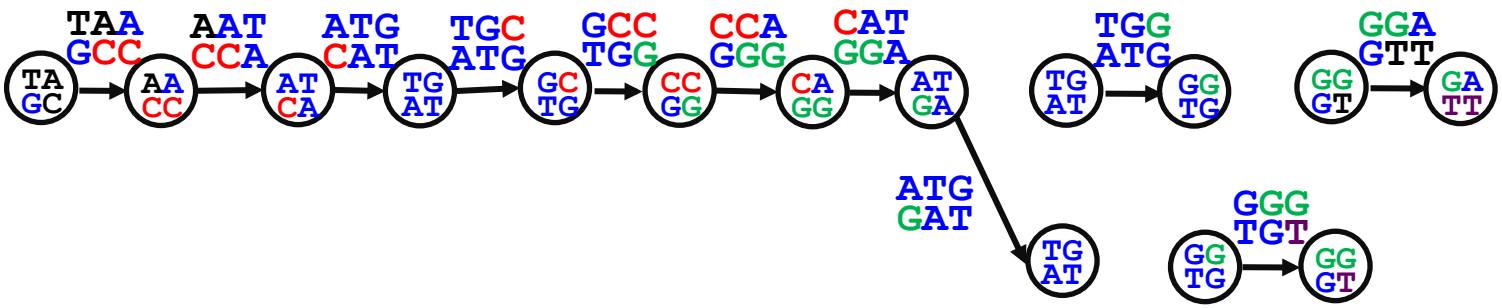
Constructing Paired de Bruijn Graph



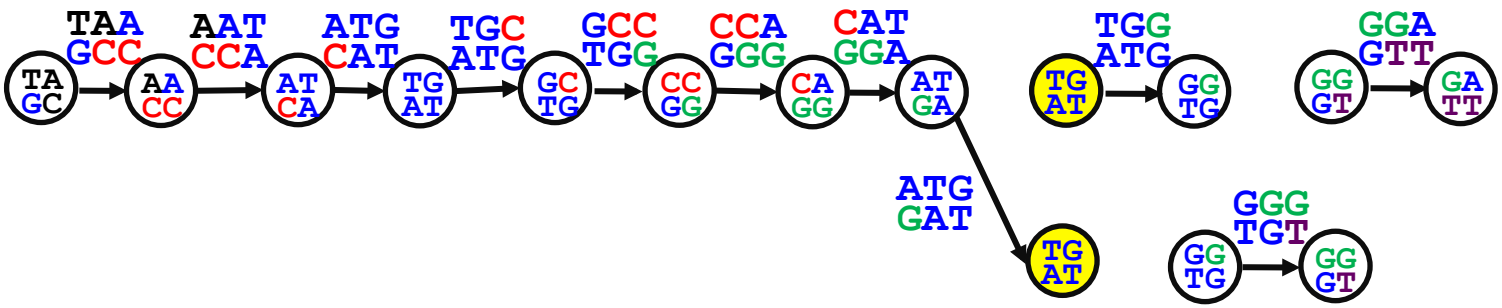
Constructing Paired de Bruijn Graph



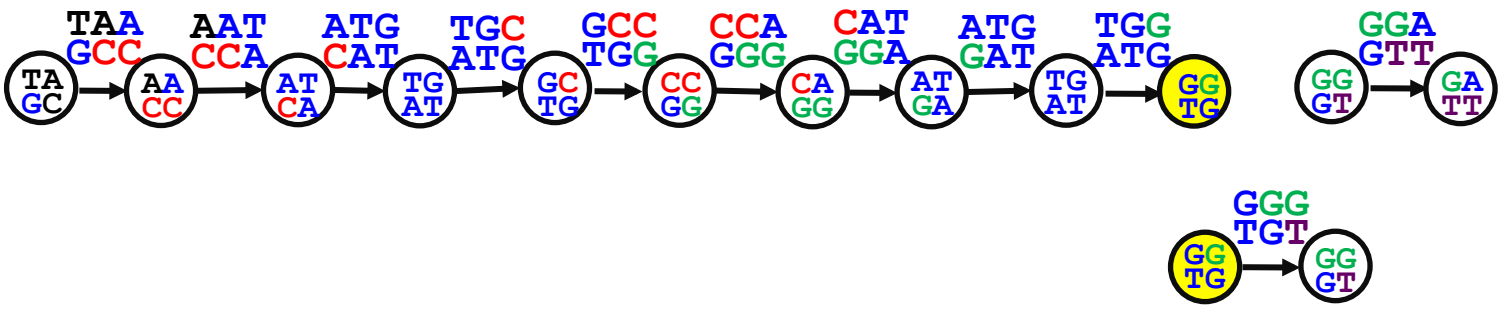
Constructing Paired de Bruijn Graph



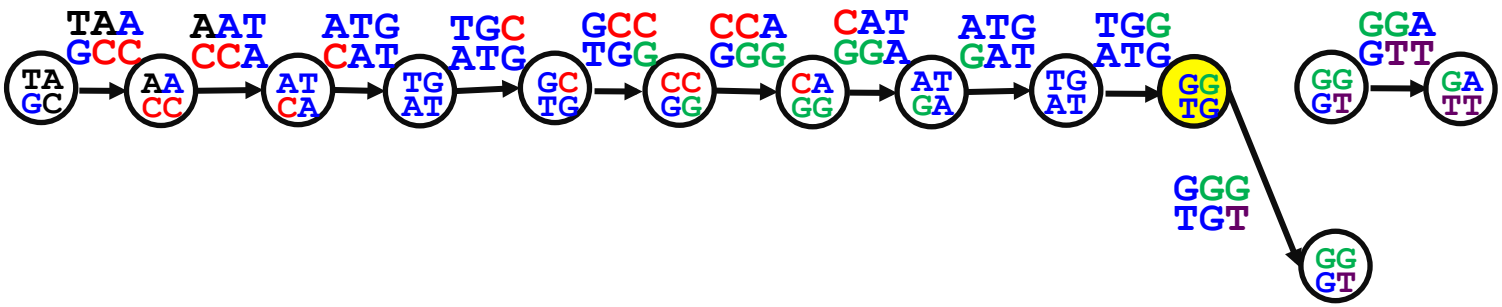
Constructing Paired de Bruijn Graph



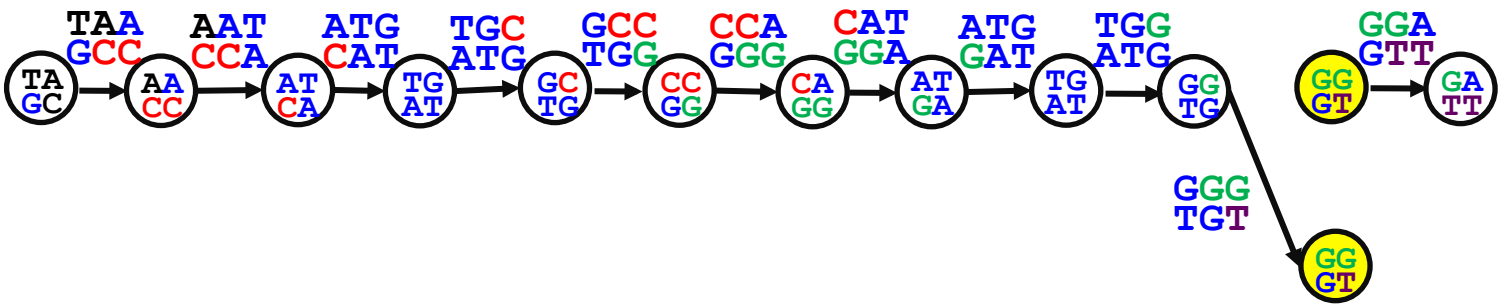
Constructing Paired de Bruijn Graph



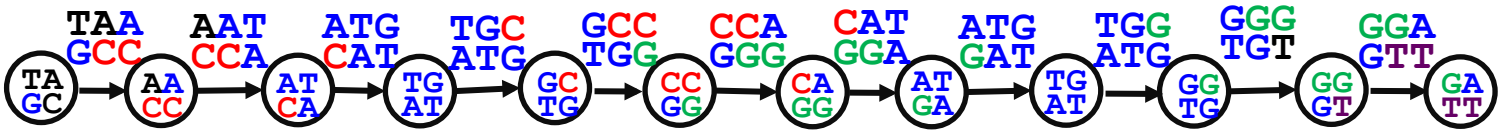
Constructing Paired de Bruijn Graph



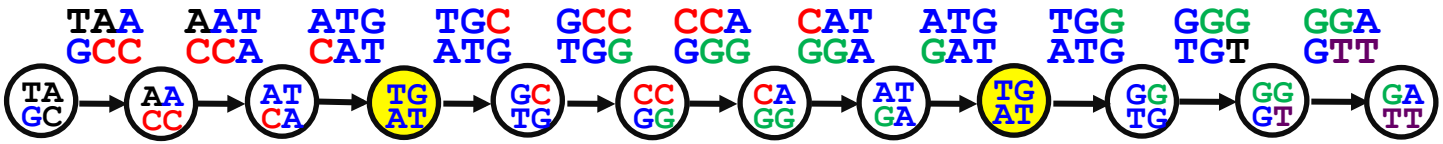
Constructing Paired de Bruijn Graph



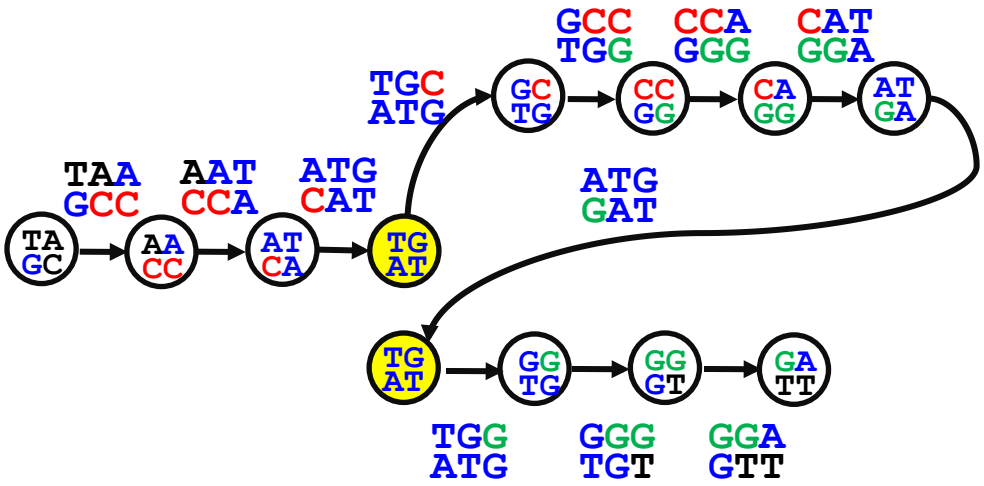
Constructing Paired de Bruijn Graph



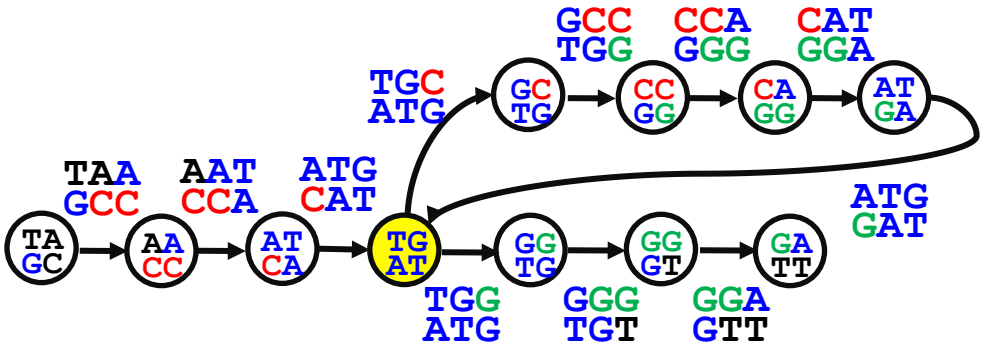
We Are Not Done with Gluing Yet



Constructing Paired de Bruijn Graph



Constructing Paired de Bruijn Graph



Paired de Bruijn Graph from read-pairs

Paired de Bruijn Graphs

- **Paired de Bruijn graph for a collection of paired *k*-mers:**
  - Represent every paired *k*-mer as an edge between its paired prefix and paired suffix.
  - Glue **ALL** nodes with identical labels.

Which Graph Represents a Better Assembly?

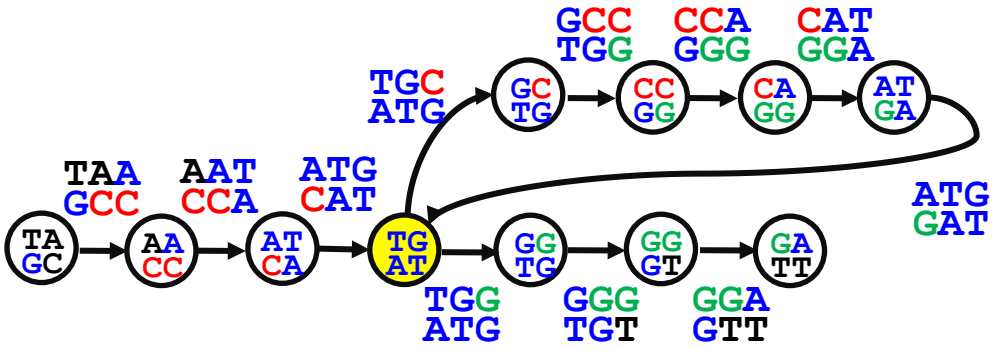
Unique genome reconstruction

TAATGCCATGGGATGTT

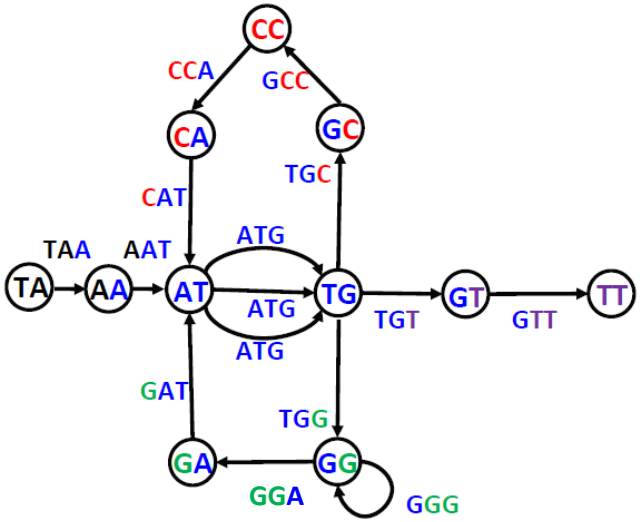
Multiple genome reconstructions

TAATGCCATGGGATGTT

TAATGGGATGCCATGTT



Paired de Bruijn Graph



De Bruijn Graph