ALGORITHMS AND DATA STRUCTURES CAPSTONE

Pavel Pevzner

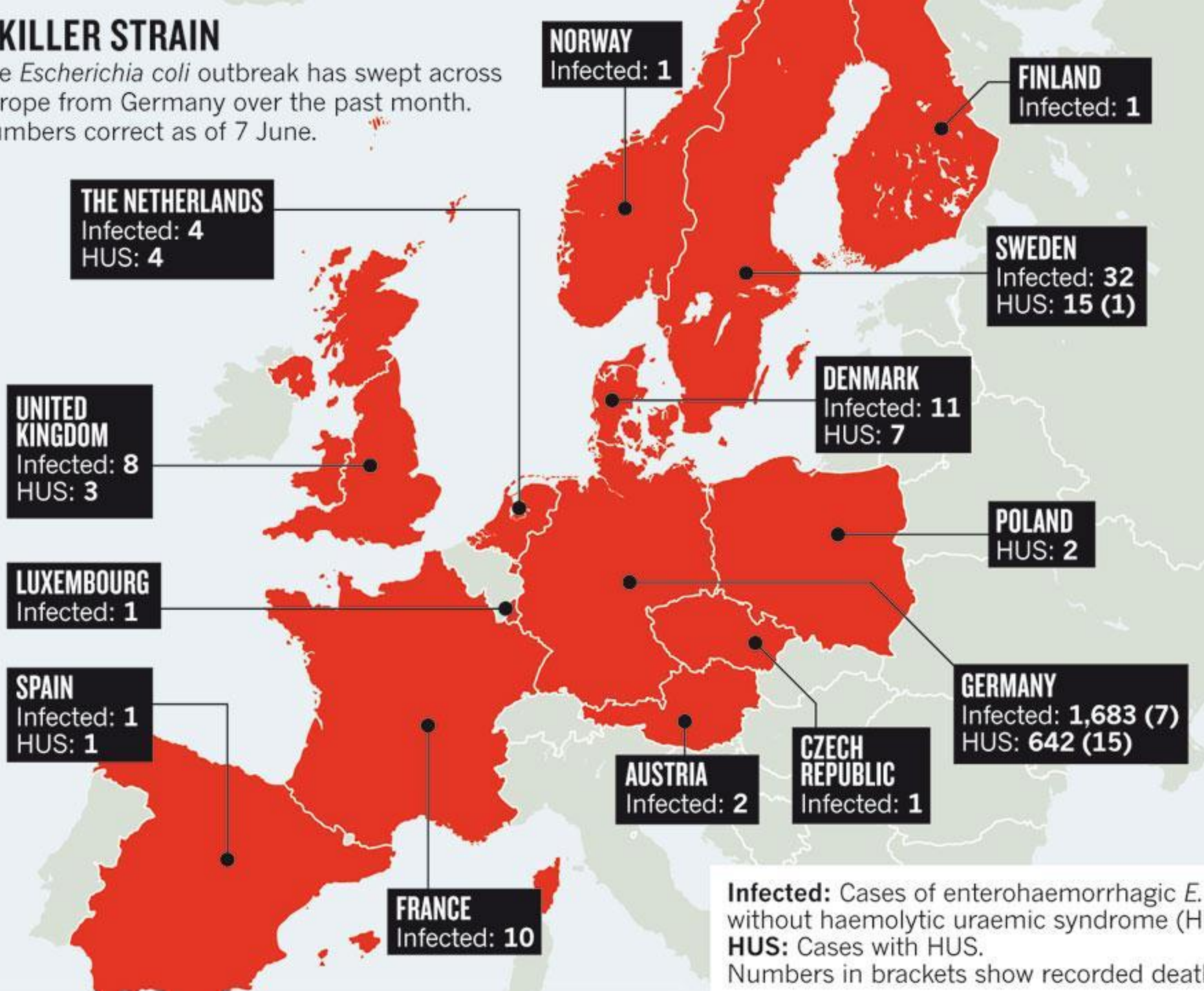
Department of Computer Science and Engineering
University of California at San Diego

Outline

- **2011 European *E. coli* outbreak**
- Assembling phage genome
- DNA arrays
- Assembling genomes from *k*-mers
- De Bruijn graphs
- Bridges of Königsberg and universal strings
- Euler theorem
- Splitting the genome into contigs
- From reads to read-pairs
- Genome assembly faces real sequencing data

A KILLER STRAIN

The *Escherichia coli* outbreak has swept across Europe from Germany over the past month. Numbers correct as of 7 June.



Infected: Cases of enterohaemorrhagic *E. coli* without haemolytic uraemic syndrome (HUS).
HUS: Cases with HUS.
Numbers in brackets show recorded deaths.


Berlin
12:32 PM
CNN SUNDAY
E. COLI KILLS 19 ACROSS EUROPE
More than 2,000 sickened in 12 countries
LIVE CNN
Unemployment rate rises to 9.1%
3:32 AM PT

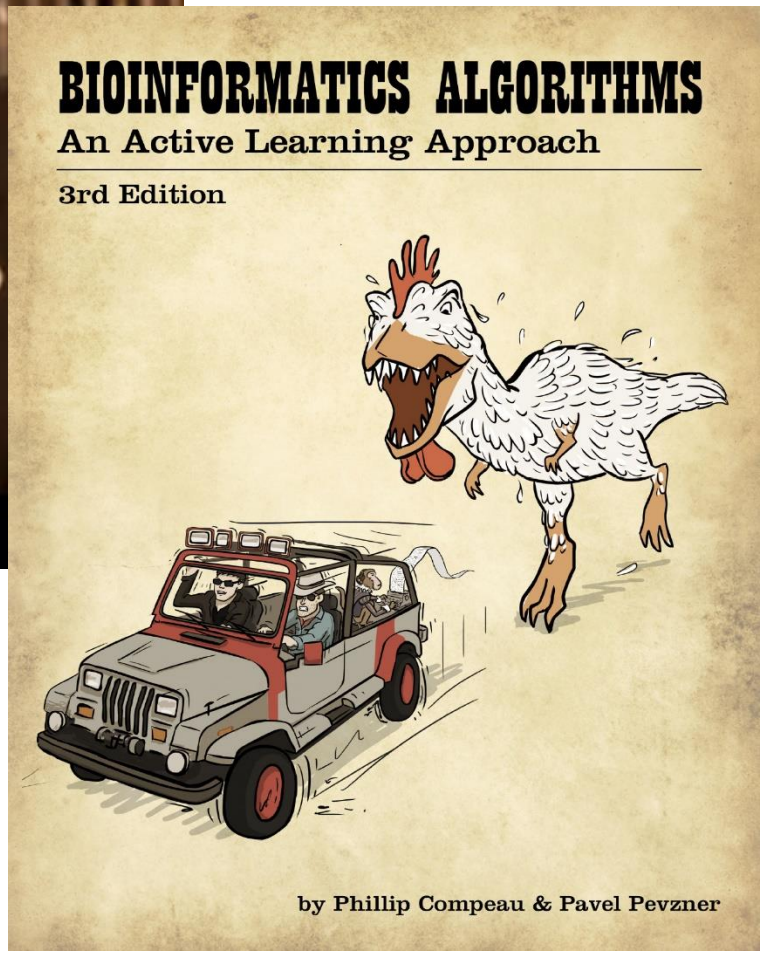

DEVELOPING STORY
JORG DEBATIN
Medical director, Hamburg Medical Center
LIVE
CNN

2015: German government who wrongly fingered Spanish cucumbers ordered to pay

How has *E. coli* become pathogenic?

- May 2011: a girl from Hamburg got bloody diarrhea after eating sprouts.
- Doctors suspected a common pathogenic *E. coli* strain but the blood sample did not pass the tests for known *E. coli* strains.

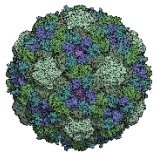
What is the genome of *E. coli* X*?*



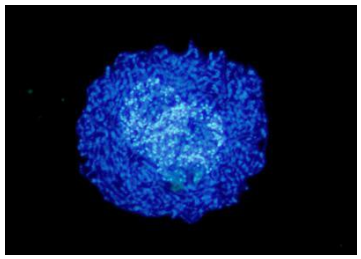
www.bioinformaticsalgorithms.org

towards assembly program

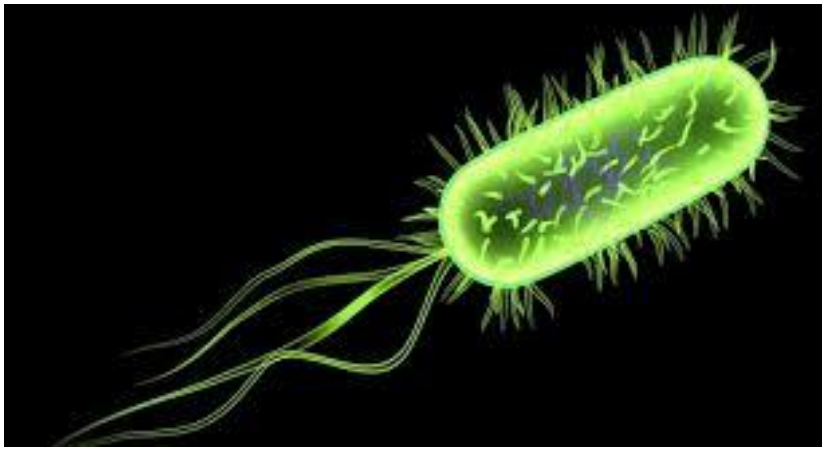
- **phi X174 virus**
  - 5,386-nucleotides
  - 11 genes



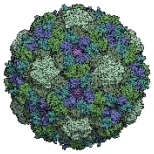
- ***N. deltocephalinicola*** bacterium
  - ≈110 thousand nucleotides
  - ≈140 genes



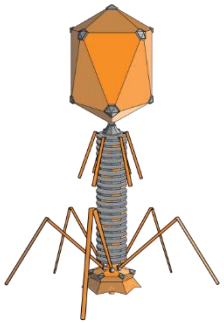
- ***E. coli* X** bacterium
  - ≈5 million nucleotides
  - ≈5 thousand genes



phi X174 phage



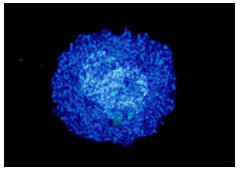
- Phages are bacterial viruses. They cannot replicate on their own and must infect bacteria to do so.
- Many phages are shaped like lunar landers, a design that helps them land on the cell wall of a bacterium and transmit their DNA into the bacterial genome



- Phage **phi X174** is the first sequenced genome, completed by Fred Sanger in 1977

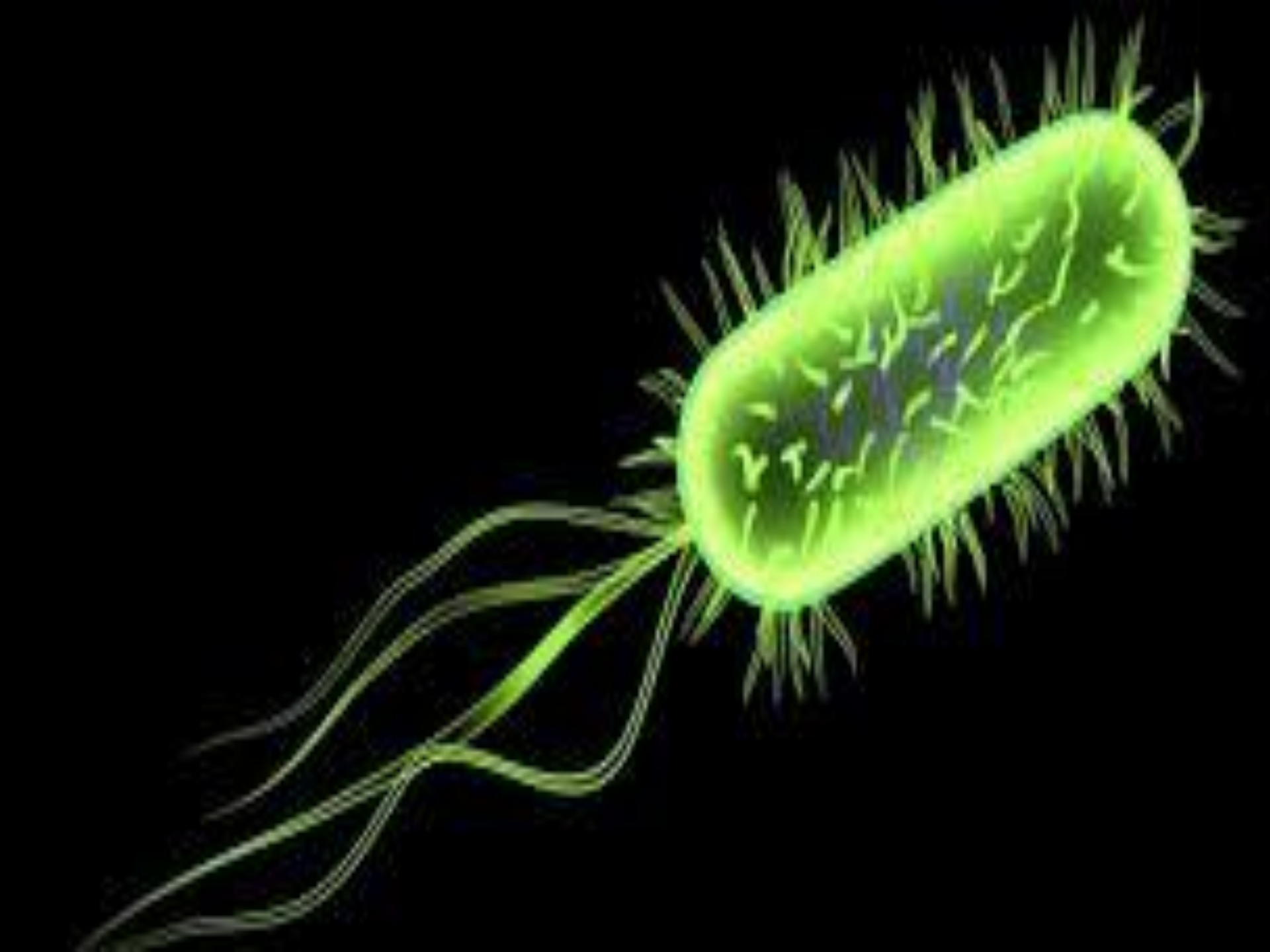
- **You will follow in the footsteps of Fred Sanger to assemble the phi X174 genome.**

The smallest bacterial genome



- *Nasuia deltocephalinicola* lives inside leafhoppers.
- Its sheltered life has allowed it to reduce its genome to only ≈110,000 nucleotides and ≈140 genes.

- It lacks some genes necessary for survival, but products of these genes are supplied by its insect host.
- *N. deltocephalinicola* is losing its bacterial identity and turning into a part of the insect's genome
- **You will follow in the footsteps of biologists who sequenced this genome in 2013**



Outline

- 2011 European *E. coli* outbreak
- **Assembling phage genome**
- DNA arrays
- Assembling genomes from *k*-mers
- De Bruijn graphs
- Bridges of Königsberg and universal strings
- Euler theorem
- Splitting the genome into contigs
- From reads to read-pairs
- Genome assembly faces real sequencing data

Assembling phage genome from error-free reads

- 1000 simulated error-free reads randomly drawn from 5,386-nucleotide long circular phi X174 genome.
- Each read is 100 nucleotides long.
- All genomes in this capstone will c
  nucleotide-long insertion.
- Your goal will be to figure out the se tag.

This is not a computational problem!

Genome Sequencing Problem.
Reconstruct a genome from reads.

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

Assembling phage genome from error-free reads

- 1000 simulated error-free reads randomly drawn from 5,386-nucleotide long circular phi X174 genome.
- Each read is 100 nucleotides long.
- All genomes in our will contain a tag, a 10-nucleotide-long insertion
- Your goal will be to figure out the sequences of the tag

What does it mean to "assemble" a genome?
Formulate a rigorous algorithmic problem that adequately models genome assembly.

Assembling phage genome from error-free reads

- 1000 simulated error-free reads randomly drawn from 5,386-nucleotide long circular phi X174 genome.
- Each read is 100 nucleotides long.
- All genomes in our will contain a tag, a 10-nucleotide-long insertion
- Your goal will be to figure out the sequences of the tag

Programming challenge: Assemble mutated phi X174 genome from simulated error-free reads and find out the inserted tag.

Assembling phage genome from error-prone reads

- 1000 simulated error-prone reads randomly drawn from 5,386-nucleotide long circular phi X174 genome.
- Each read is 100 nucleotides long.
- Each read has errors (substitutions of nucleotides) with probability 0.01 at each position.

What does it mean to "assemble" a genome from error-prone reads? Formulate a rigorous algorithmic problem that adequately models genome assembly from error-prone reads.

Assembling phage genome from error-prone reads

- 1000 simulated error-prone reads randomly drawn from 5,386-nucleotide long circular phi X174 genome.
- Each read is 100 nucleotides long.
- Each read has errors (substitutions of nucleotides) with probability 0.01 at each position.

Programming challenge: Assemble mutated phi X174 genome from simulated error-prone reads and find out the inserted tag.


@5107060064031704290740270206371 6
GeneChip®
phylochip
P/N: 510706
Lot #: 4027020
Exp. Date: 04/29/07
For Research Use Only
UNITED STATES OF AMERICA
IN GOD WE TRUST
LIBERTY
QUARTER DOLLAR

Outline

- 2011 European *E. coli* outbreak
- Assembling phage genome
- **DNA arrays**
- Assembling genomes from *k*-mers
- De Bruijn graphs
- Bridges of Königsberg and universal strings
- Euler theorem
- Splitting the genome into contigs
- From reads to read-pairs
- Genome assembly faces real sequencing data

Evolution of DNA sequencing technologies

- **1977:** Sanger assembled the phi X174 genome from 500-nucleotide long reads. Scaling it to the human genome would cost hundreds of billion dollars and would likely fail due to unresolved algorithmic challenges.
- **1984:** US government started to plan the Human Genome Project.
- **1988:** Radoje Drmanac, Andrey Mirzabekov, and Edwin Southern proposed **DNA chips.**



DNA arrays vs. Sanger sequencing

- **Sanger technology:** generate ***some*** long reads randomly sampled from the genome (read length ≈500)

- **DNA chip:** generate ***all*** short *k*-mers from a genome (*k*=10)

k-mer composition

Composition$_3$(TAATGCCATGGGATGTT)=

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

k-mer composition

Composition$_3$(TAATGCCATGGGATGTT)=

k-mer composition

Composition$_3$(TAATGCCATGGGATGTT)=

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

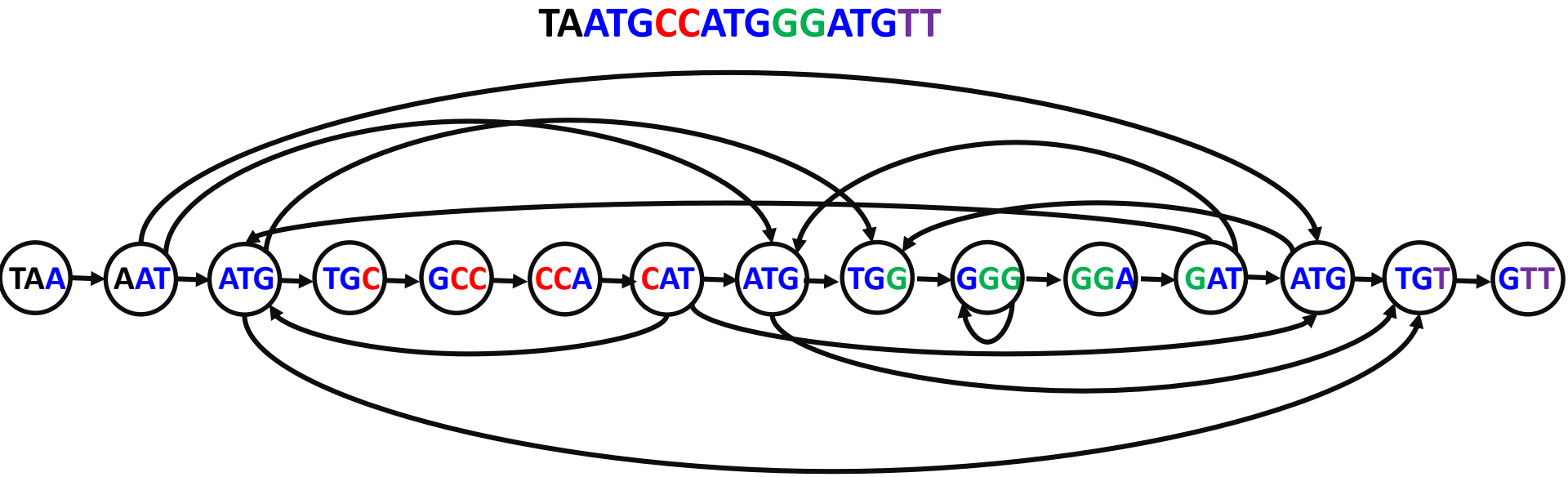
Connecting overlapping *k*-mers

Composition$_3$(TAATGCCATGGGATGTT)=

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

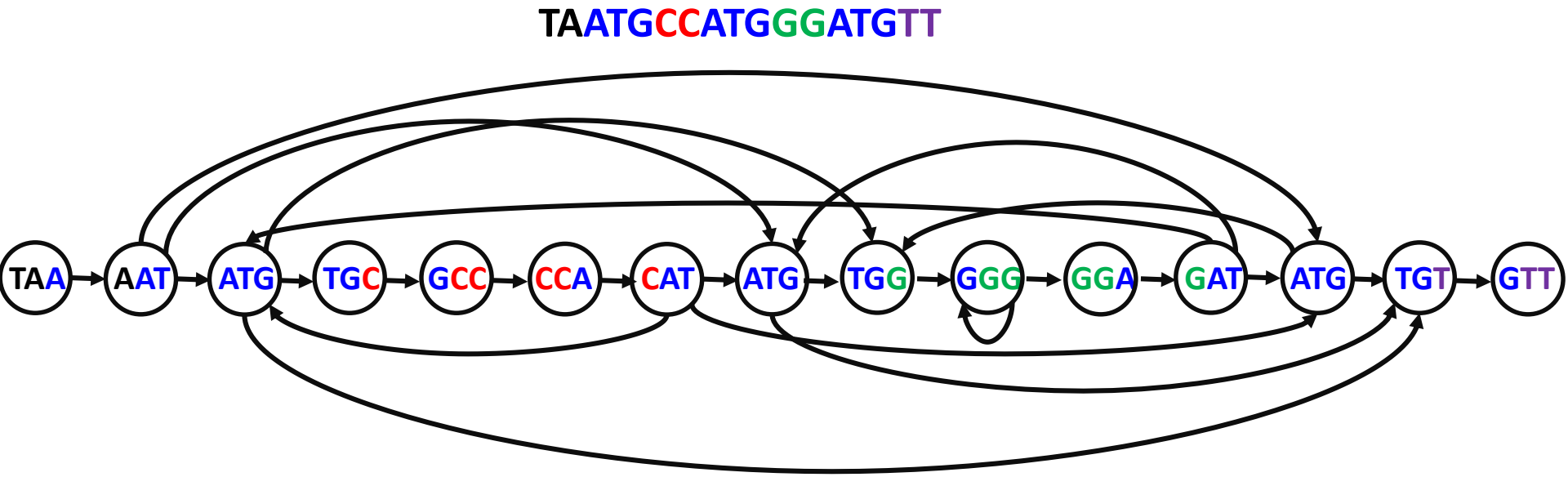
Connect *k*-mer$_1$ with *k*-mer$_2$ if suffix(*k*-mer$_1$)=prefix(*k*-mer$_2$).
e.g. TAA → AAT

Overlap graph



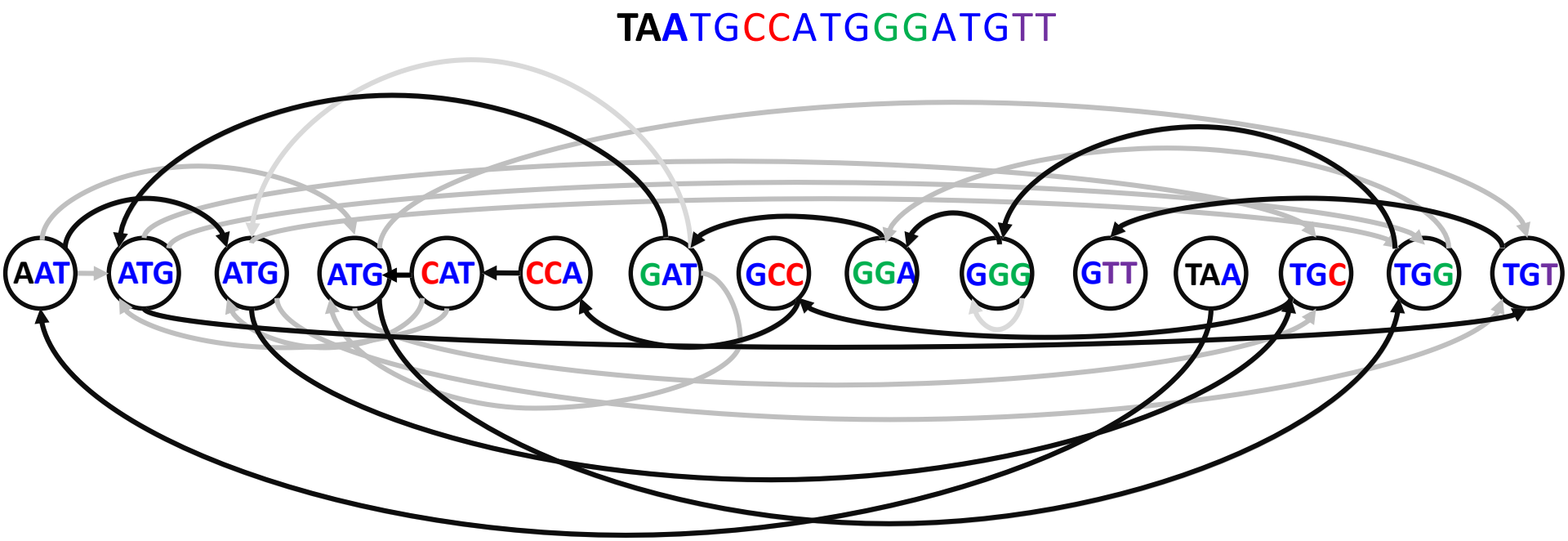
Connect k-mer$_1$ with k-mer$_2$ if suffix(k-mer$_1$)=prefix(k-mer$_2$).
e.g. TAA → AAT

Where is the genome path?



Searching for genome path

A **Hamiltonian path** that visits each vertex exactly once



What are we trying to find in this graph?

Outline

- 2011 European *E. coli* outbreak
- Assembling phage genome
- DNA arrays
- **Assembling genomes from *k*-mers**
- De Bruijn graphs
- Bridges of Königsberg and universal strings
- Euler theorem
- Splitting the genome into contigs
- From reads to read-pairs
- Genome assembly faces real sequencing data

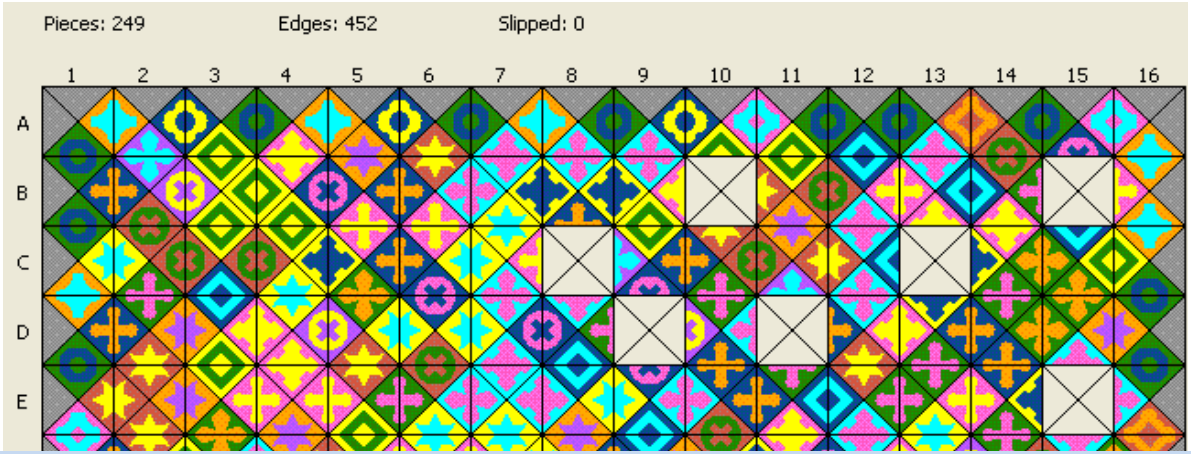

Triazzle
Brain Teaser Puzzle

Assembling phi X174 genome from *k*-mers

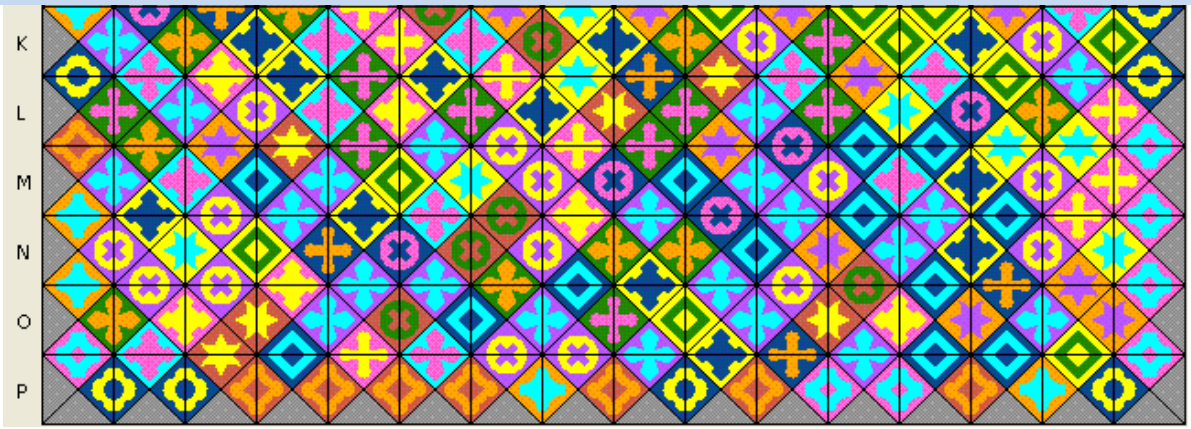
Problem: Assemble mutated phi X174 genome from all its *k*-mers and find the inserted tag.

EXERCISE BREAK: What is the minimum value of *k* for which phi X174 genome can be uniquely reconstructed from its *k*-mer composition?

Eternity II puzzle ($2,000,000 prize)



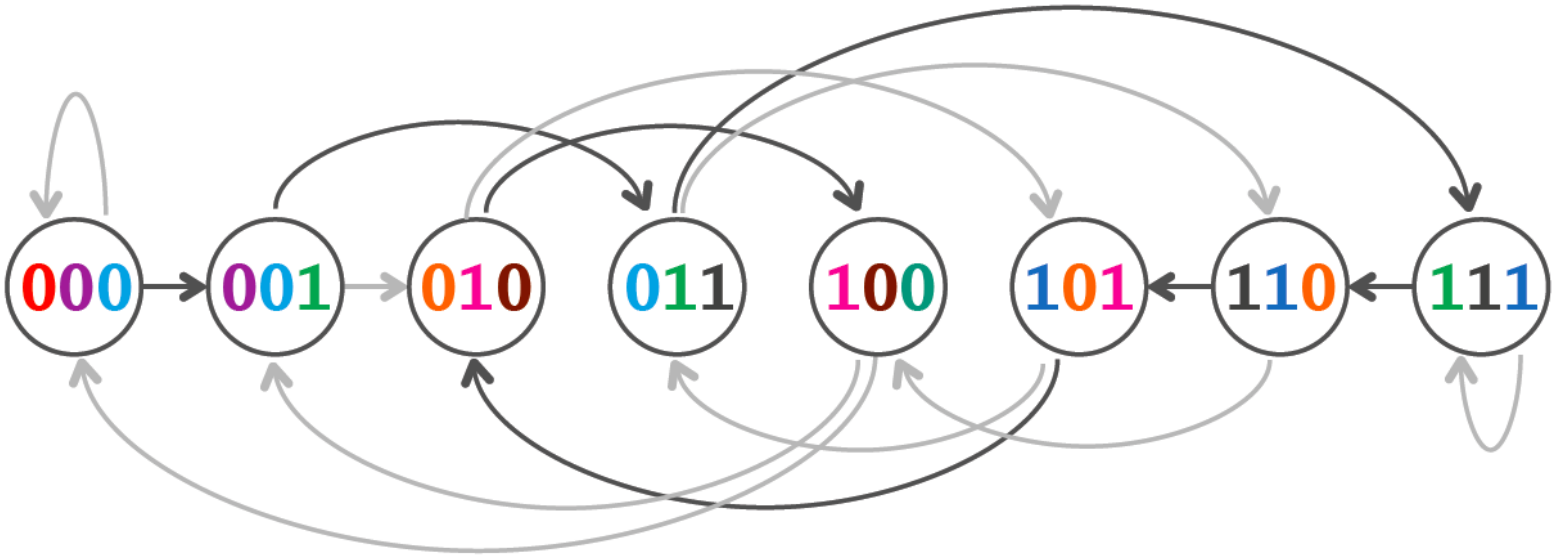
Puzzle Assembly Problem. Assemble a smaller version of the Eternity II puzzle that requires placing 25 square pieces into a 5-by-5 grid.



Universal strings

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

000 001 010 011 100 101 110 111



0001110100

Constructing a different graph

De Bruijn's idea: Construct a graph in which every *k*-mer corresponds to an **edge** rather than a vertex and where each *k*-universal string corresponds to an **Eulerian** path.

Eulerian Path Problem:
Construct an Eulerian path in a directed graph.

- **Input:** A directed graph.
- **Output:** A path visiting every edge in the graph exactly once (if such a path exists).

Outline

- 2011 European *E. coli* outbreak
- Assembling phage genome
- DNA arrays
- Assembling genomes from *k*-mers
- **De Bruijn graphs**
- Bridges of Königsberg and universal strings
- Euler theorem
- Splitting the genome into contigs
- From reads to read-pairs
- Genome assembly faces real sequencing data

Before: labeling vertices by *k*-mers

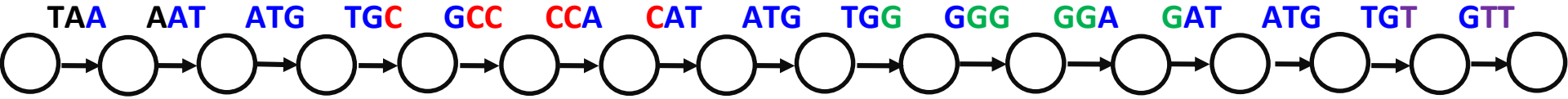
TAATGCCATGGGATGTT

3-mers as vertices

Now: labeling edges by *k*-mers

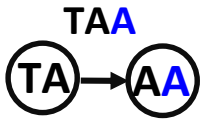
TAATGCCATGGGATGTT

3-mers as vertices

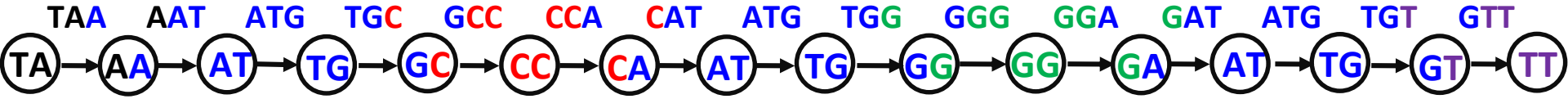


3-mers as edges

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

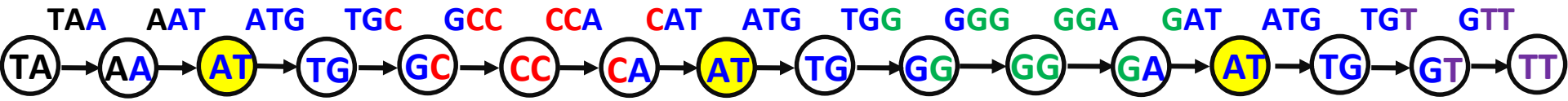
prefix of **TAA**  suffix of **TAA**

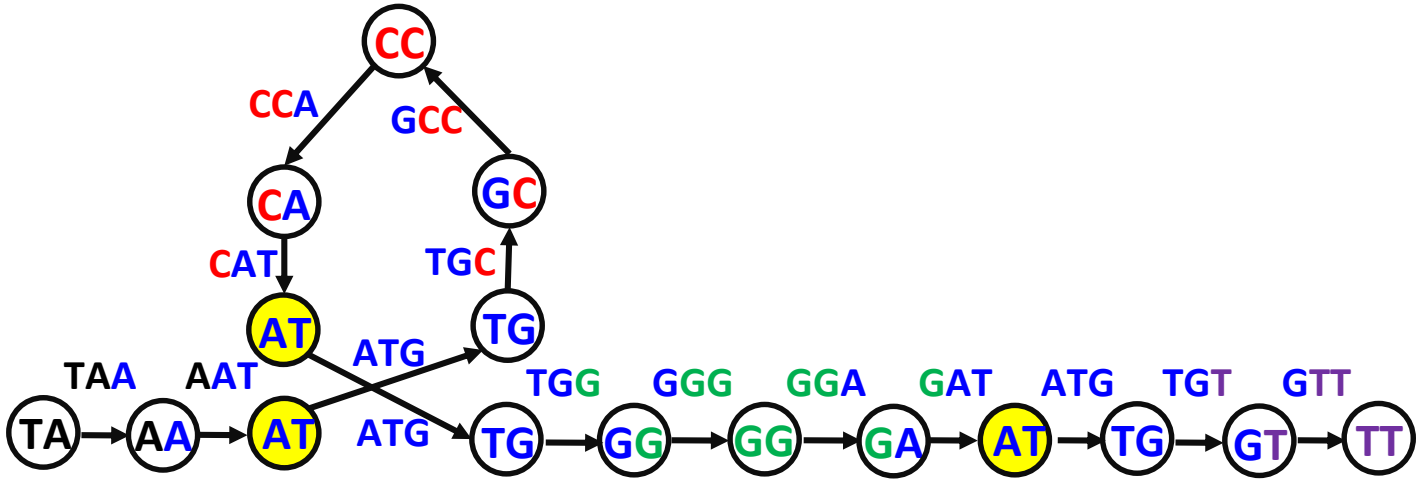
Labeling vertices by *(k-1)*-mers



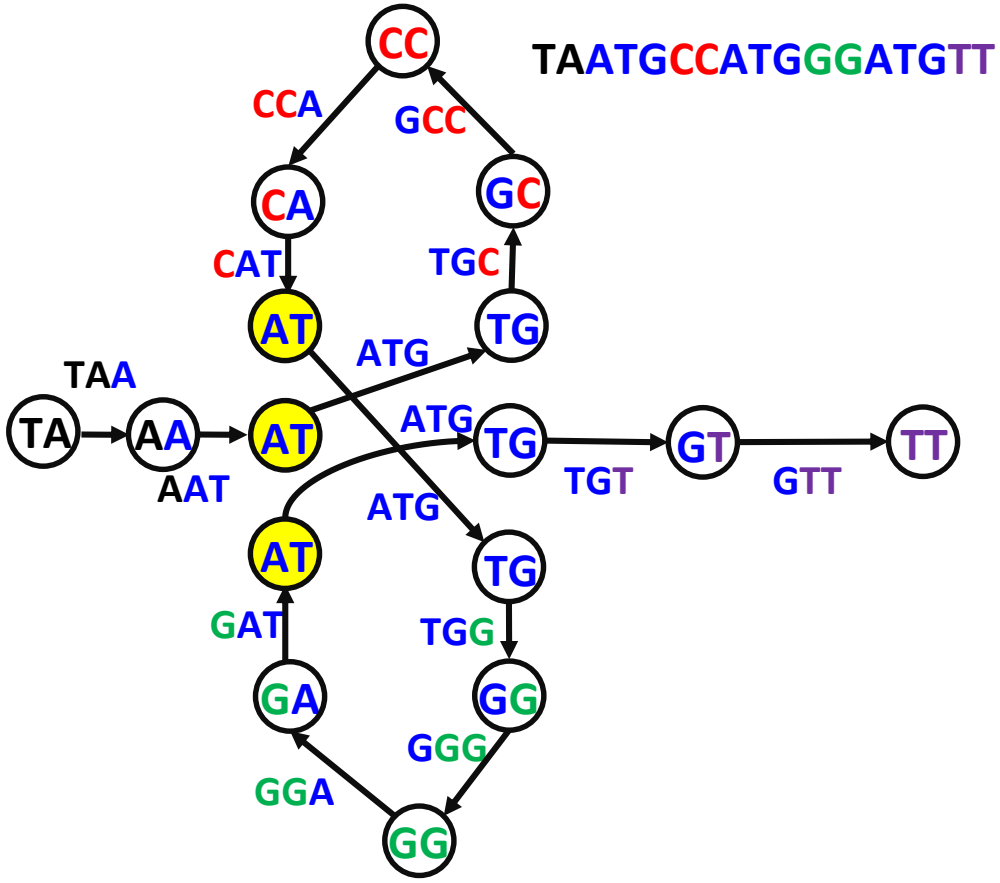
3-mers as edges and 2-mers as vertices

Gluing identically labeled vertices

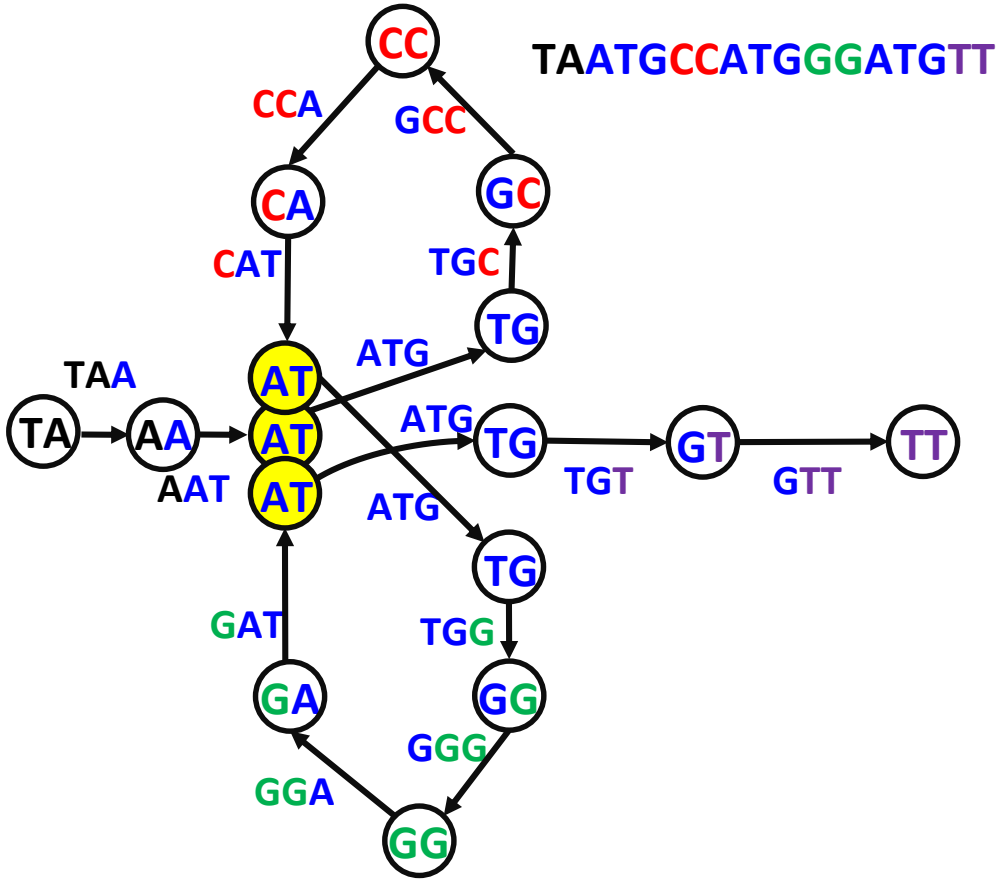




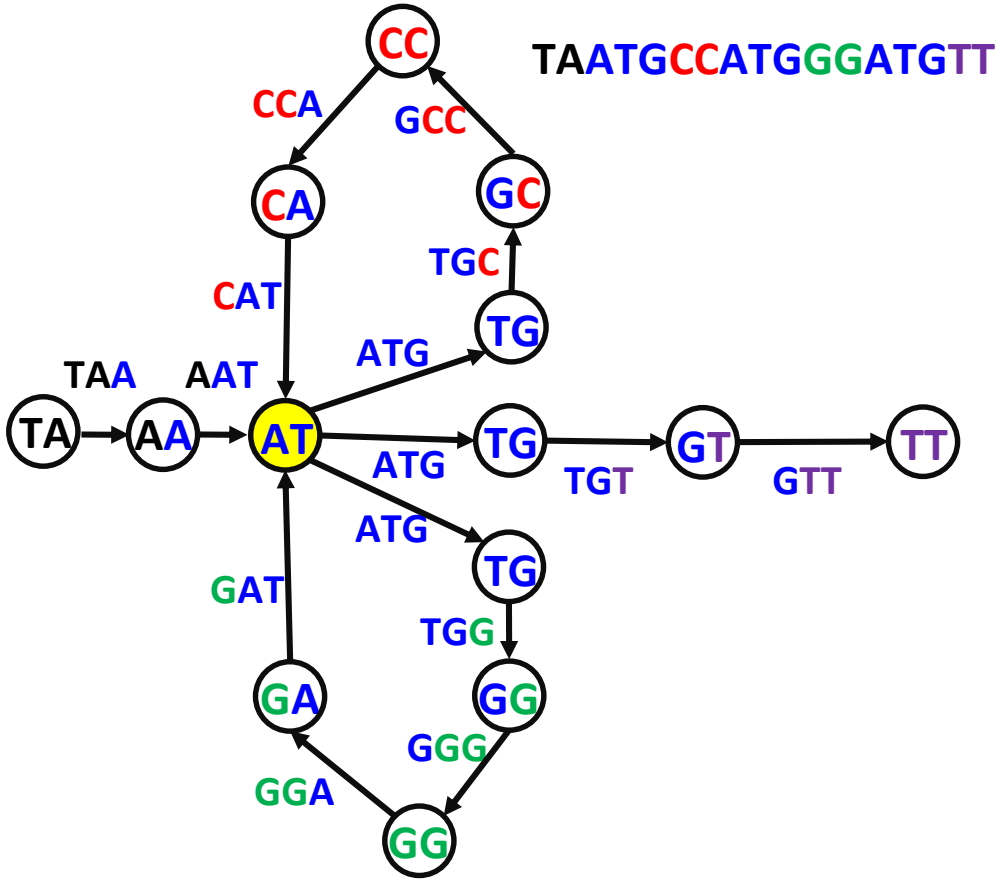
Gluing identically labeled vertices



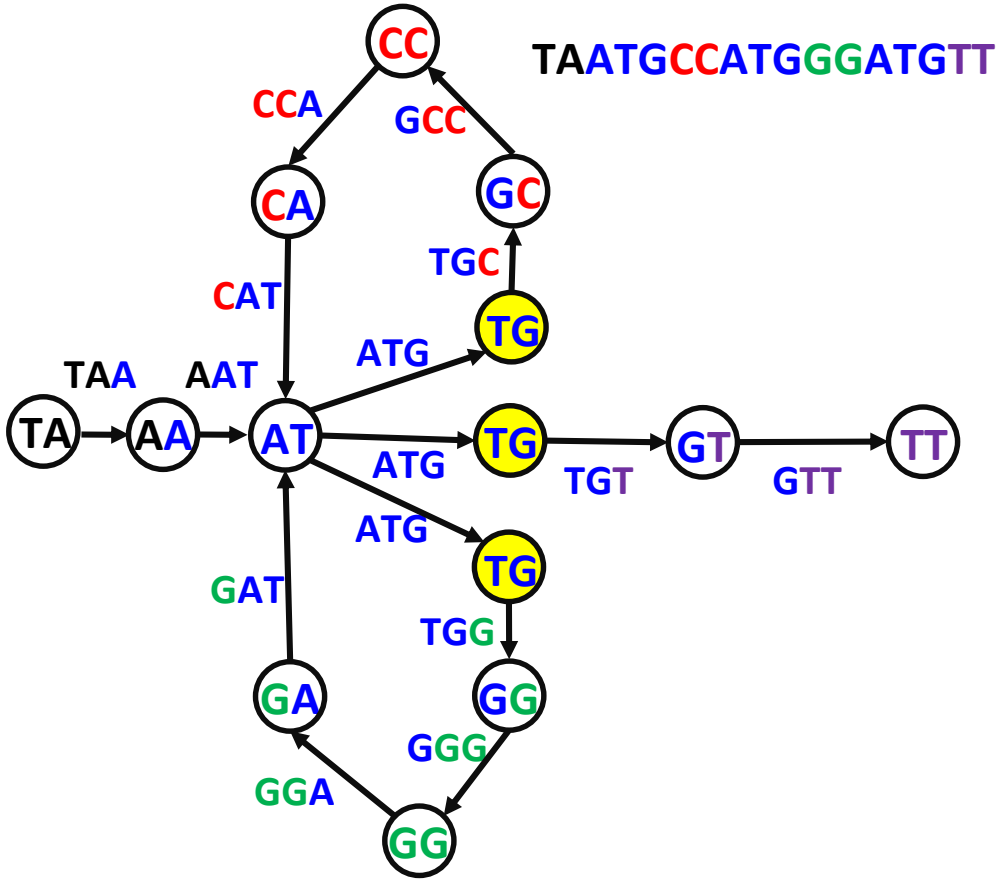
Gluing identically labeled vertices



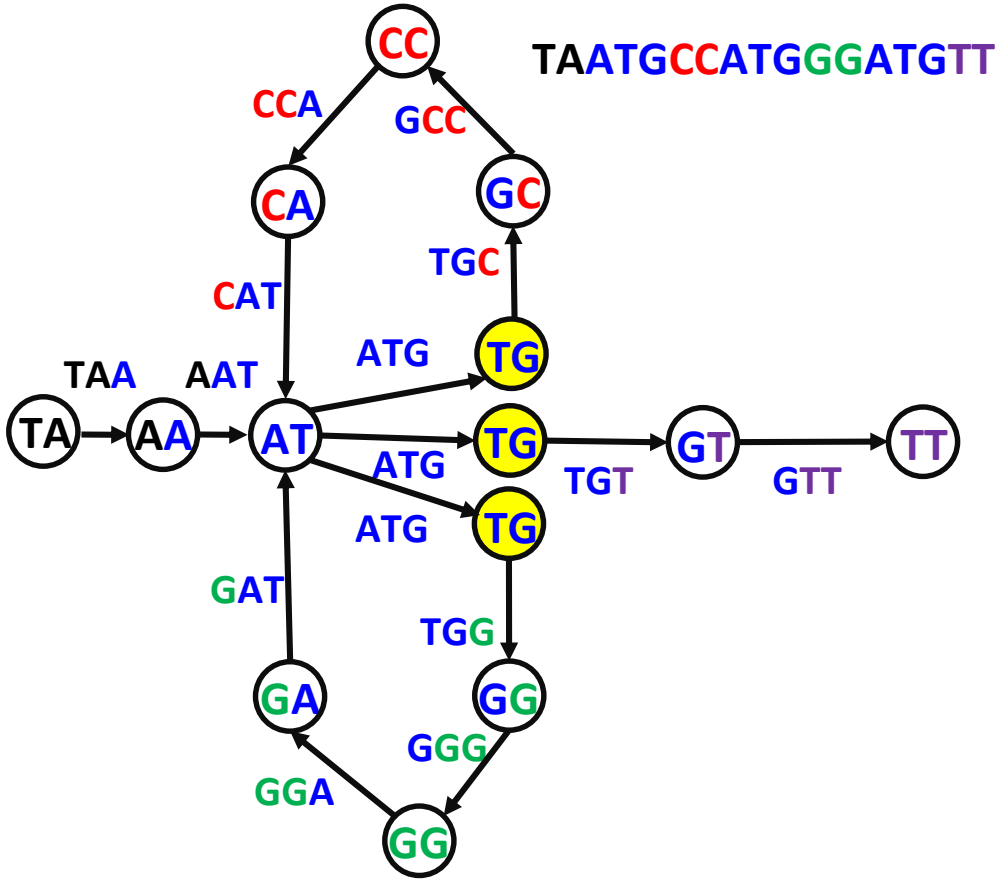
Gluing identically labeled vertices



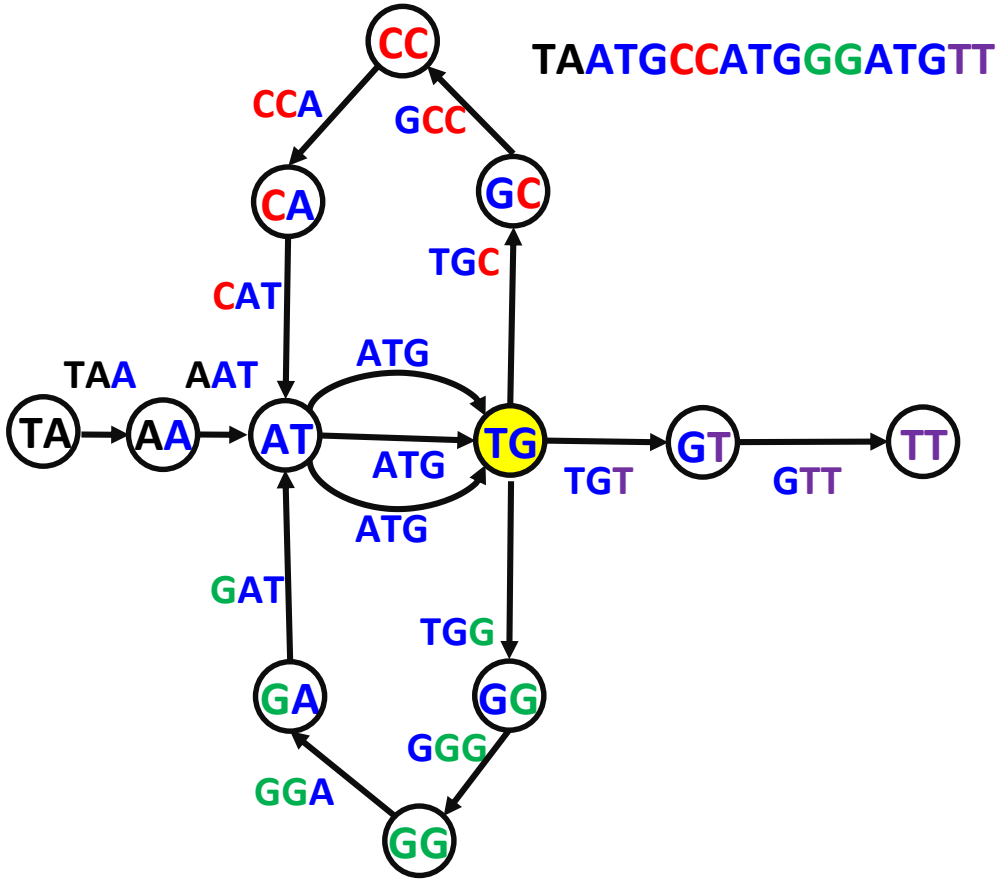
Gluing identically labeled vertices



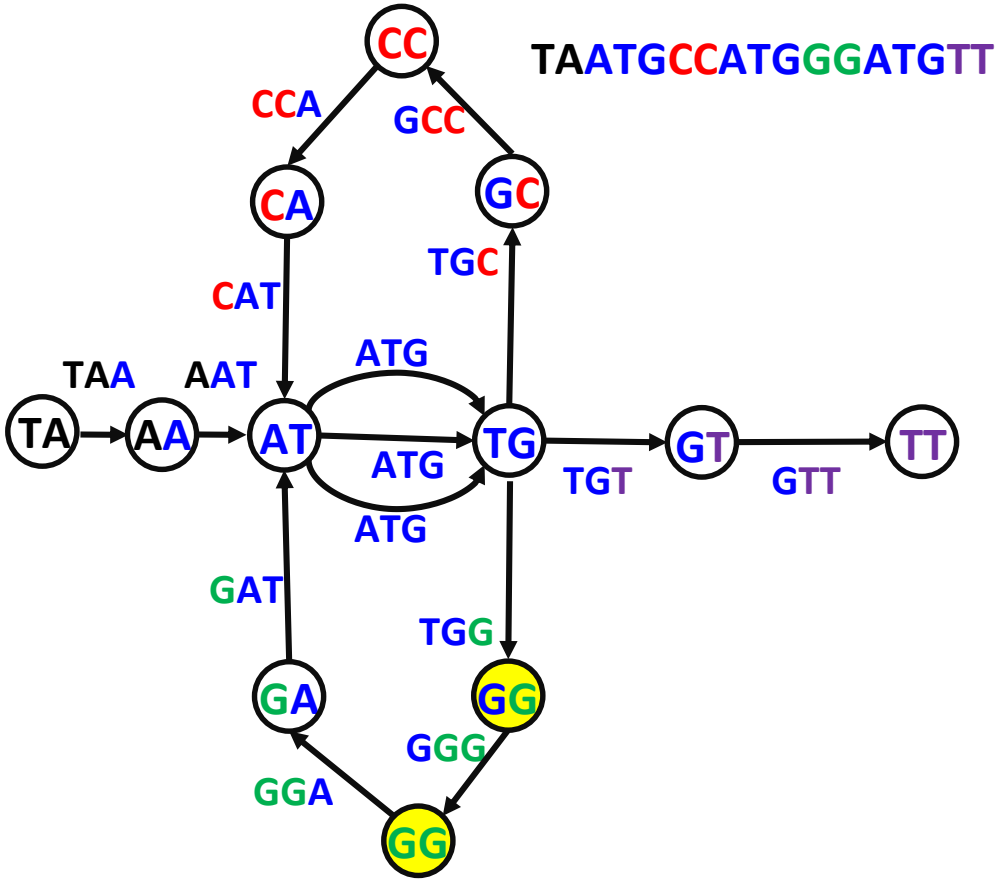
Gluing identically labeled vertices



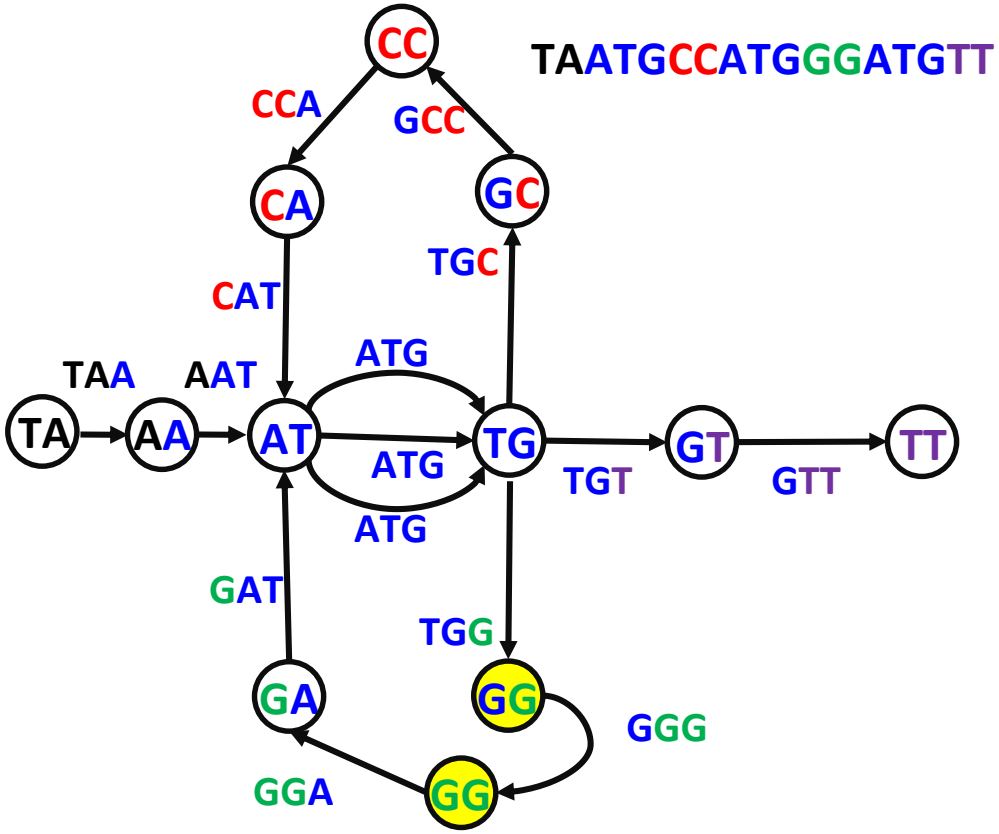
Gluing identically labeled vertices



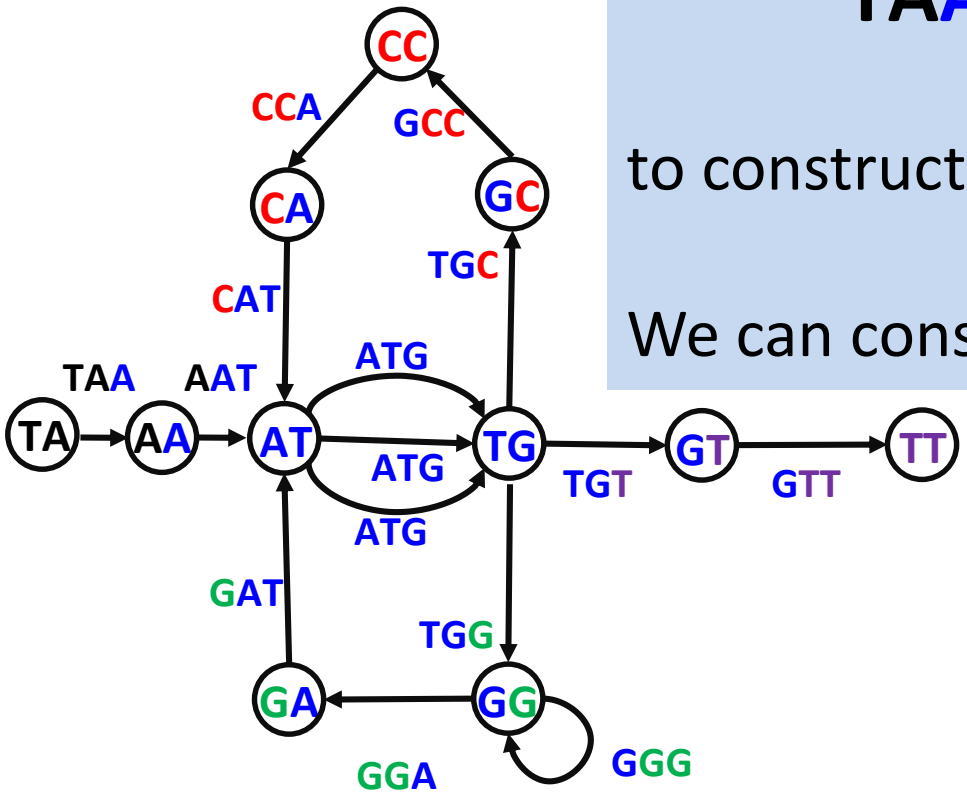
Gluing identically labeled vertices



Gluing identically labeled vertices



De Bruijn graph of TAATGCCATGGGATGTT



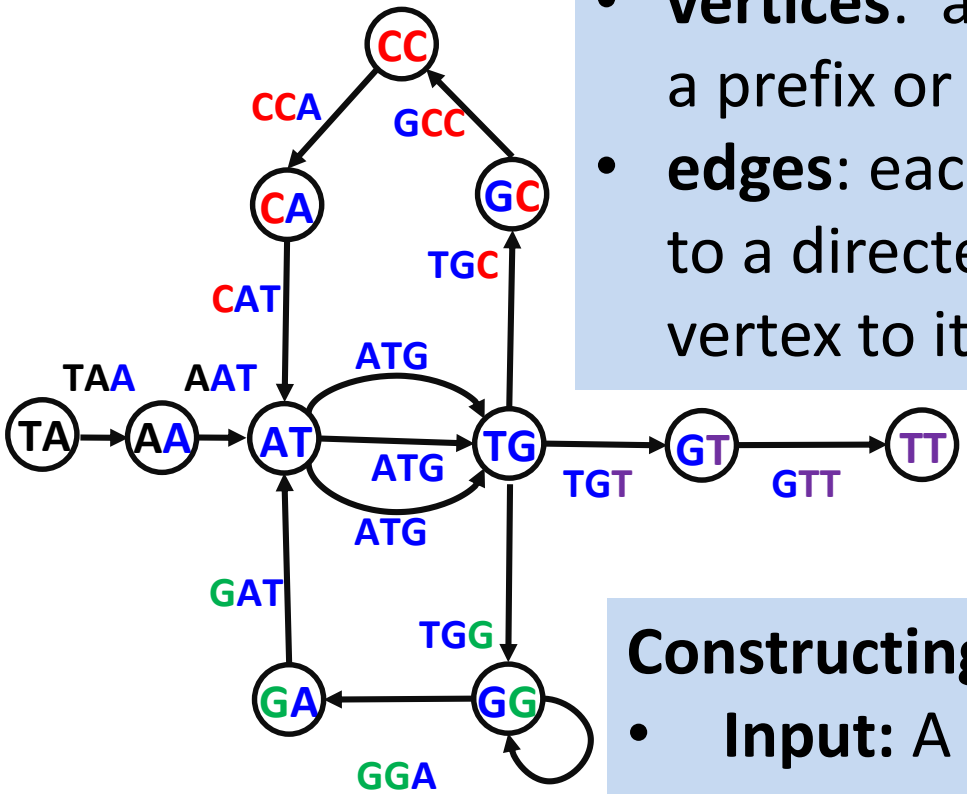
We don't need to know the string

TAATGCCATGGGATGTT

to construct the de Bruijn graph.

We can construct it from its 3-mers.

De Bruijn graph of *k*-mers



DeBRUIJN(*Patterns***)**: de Bruijn graph constructed for a set of *k*-mers *Patterns*:

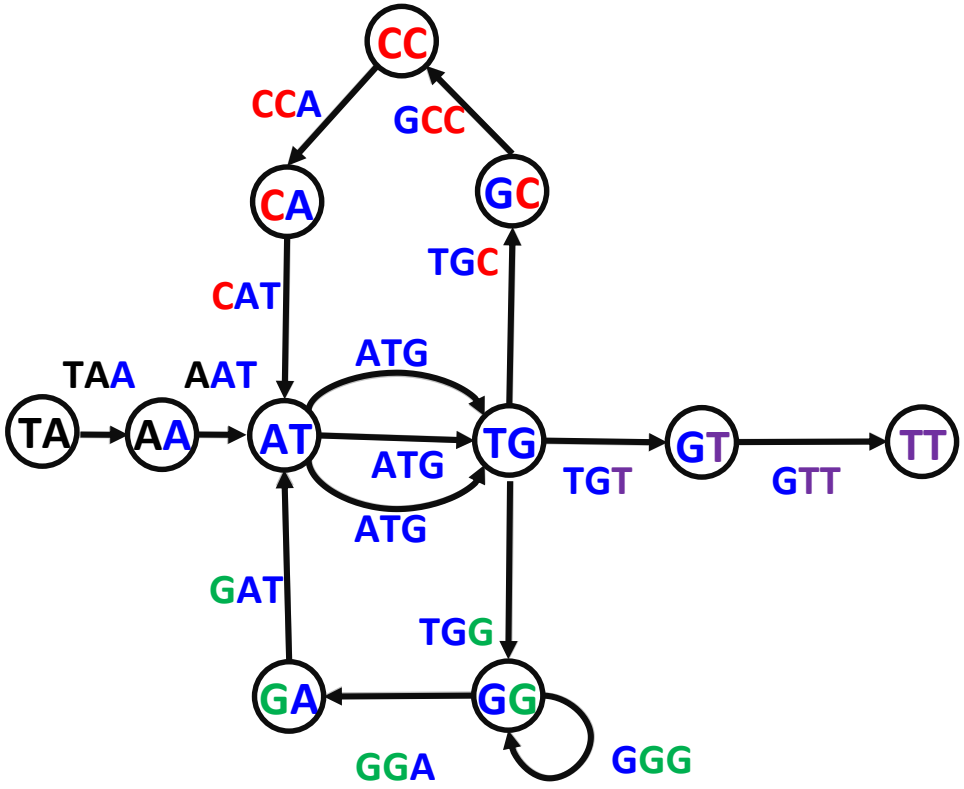
- **vertices**: all unique (*k*-1)-mers occurring as a prefix or suffix of *k*-mers in *Patterns*.
- **edges**: each *k*-mer in *Patterns* corresponds to a directed edge that connects its prefix vertex to its suffix vertex.

Constructing de Bruijn graph from *k*-mers.

- **Input:** A set of *k*-mers *Patterns.*
- **Output:** graph **DeBRUIJN(***Patterns***).**

Where is the *Genome* hiding in this graph?

TAATGCCATGGGATGTT

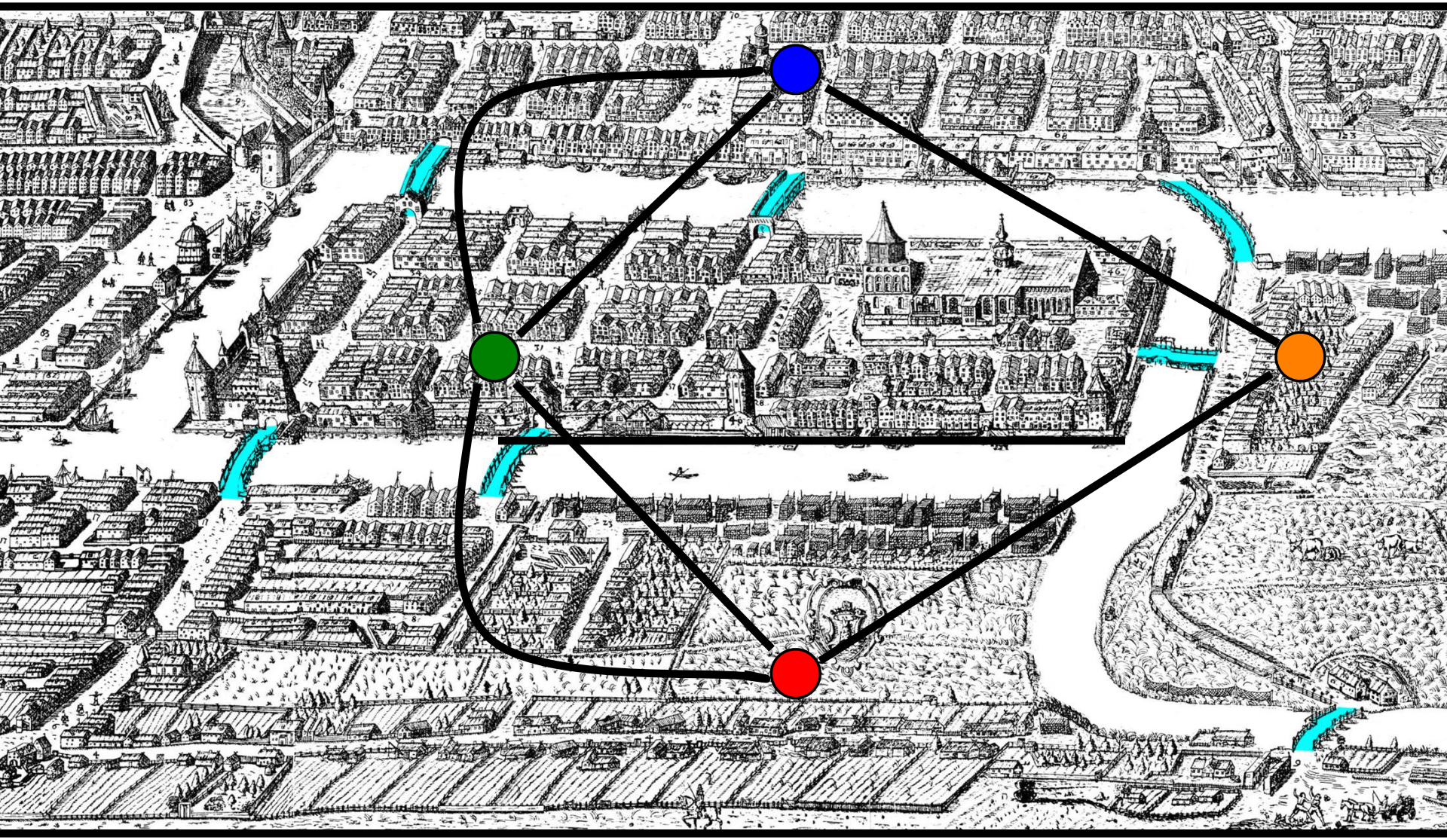


An Eulerian path that visits each edge exactly once.

Outline

- 2011 European *E. coli* outbreak
- Assembling phage genome
- DNA arrays
- Assembling genomes from *k*-mers
- De Bruijn graphs
- **Bridges of Königsberg and universal strings**
- Euler theorem
- Splitting the genome into contigs
- From reads to read-pairs
- Genome assembly faces real sequencing data

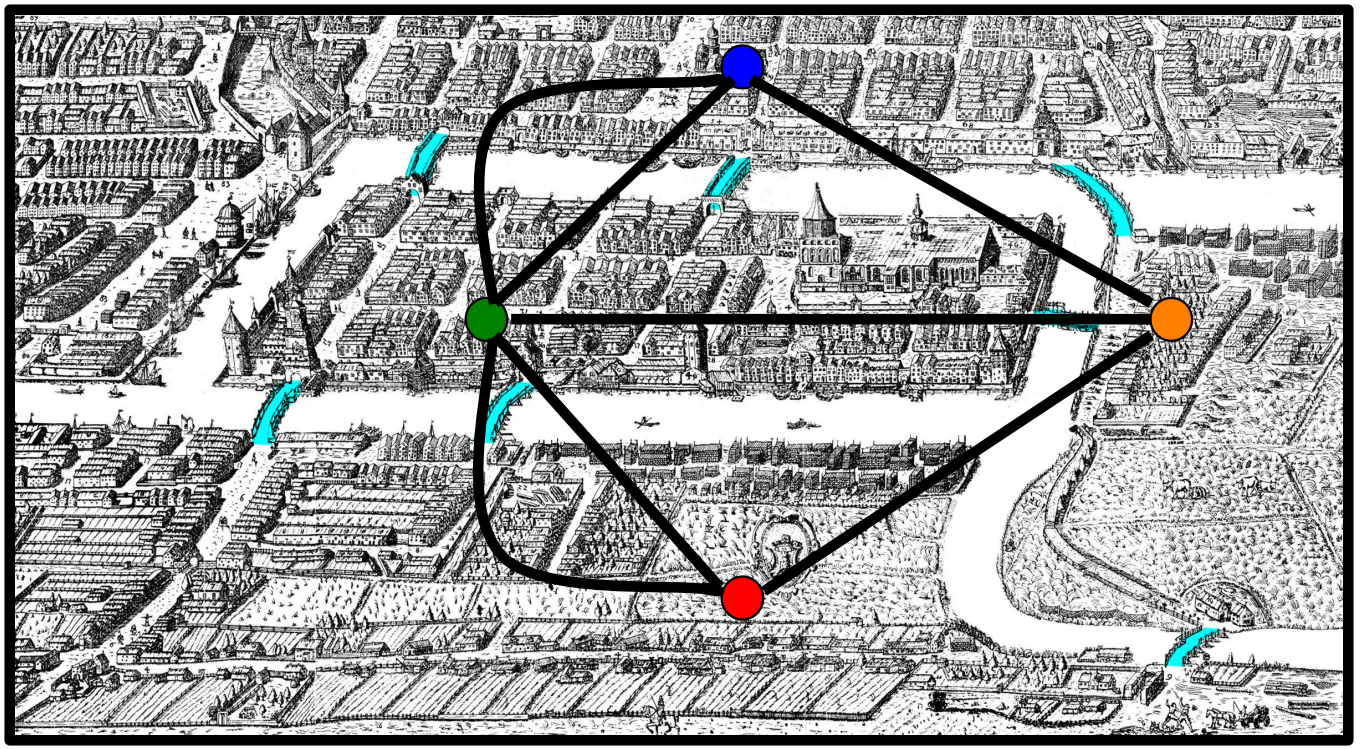
Bridges of Königsberg



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.



Eulerian versus Hamiltonian cycles

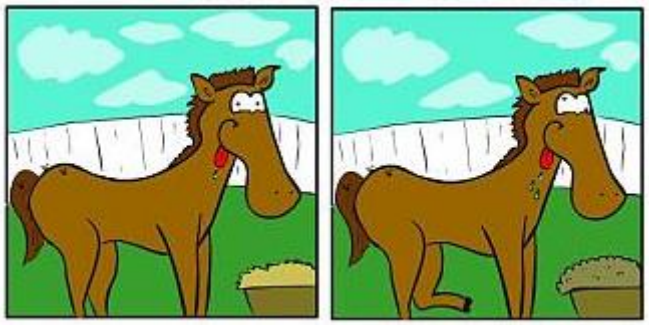
Eulerian Cycle Problem. Find an Eulerian cycle in a graph.

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

Hamiltonian Cycle Problem. Find a Hamiltonian cycle in a graph.

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

Find a difference!



Eulerian versus Hamiltonian cycles

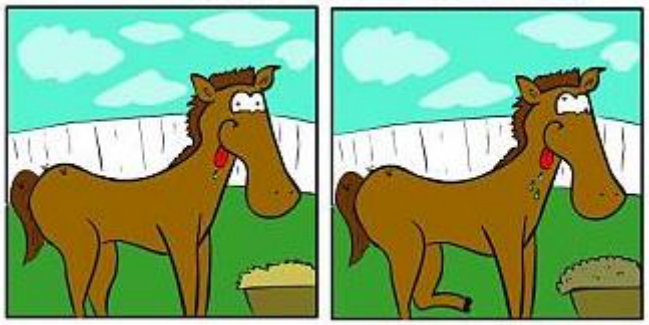
Eulerian Cycle Problem. Find an Eulerian cycle in a graph.

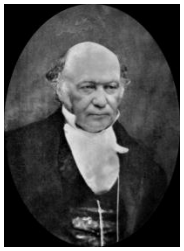
- **Input.** A graph.
- **Output.** A cycle visiting every **edge** in the graph exactly once.

Hamiltonian Cycle Problem. Find a Hamiltonian cycle in a graph.

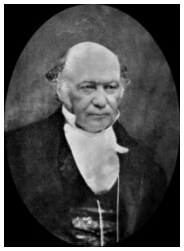
- **Input.** A graph.
- **Output.** A cycle visiting every **vertex** in the graph exactly once.

Find a difference!



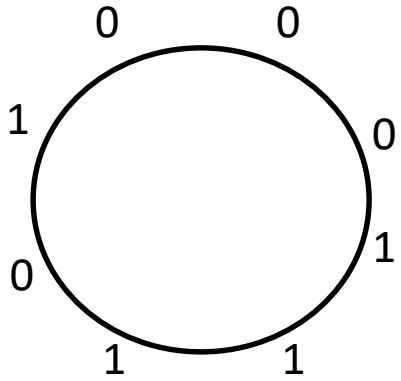
From Hamilton  to Euler  to de Bruijn 

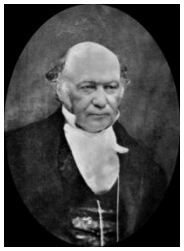
Universal Circular String Problem (De Bruijn, 1946). Find a circular string containing each binary *k*-mer exactly once.

From Hamilton  to Euler  to de Bruijn 

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

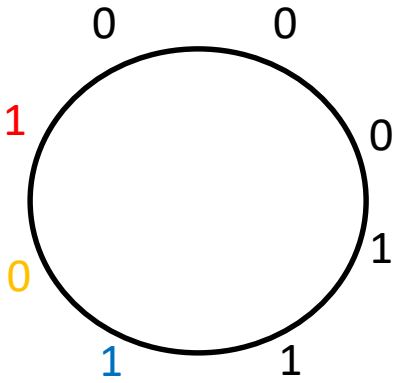
000 001 010 011 100 101 110 111

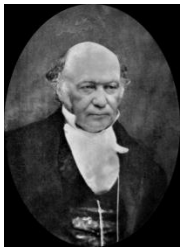


From Hamilton  to Euler  to de Bruijn 

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

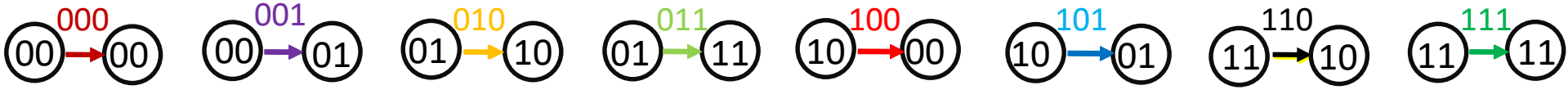
000 001 010 011 100 101 110 111

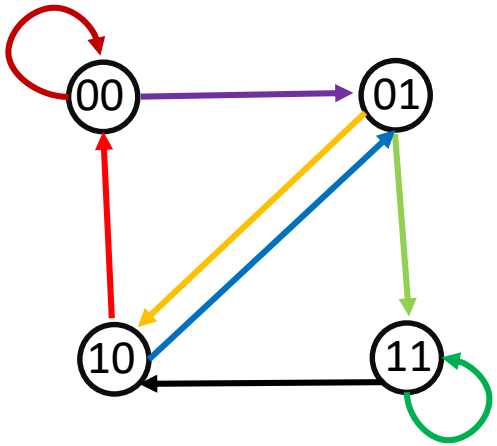


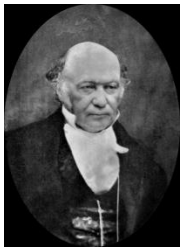
From Hamilton  to Euler  to de Bruijn 

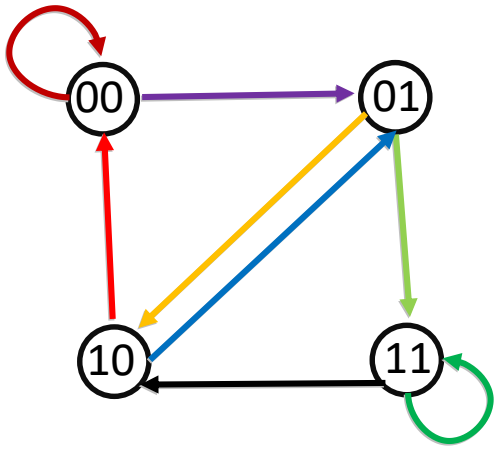
Universal Circular String Problem (de Bruijn, 1946). Find a circular string containing each binary *k*-mer exactly once.

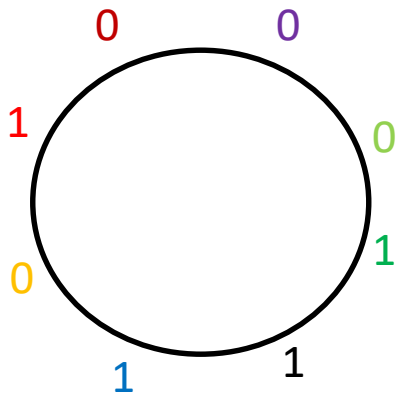
000 001 010 011 100 101 110 111



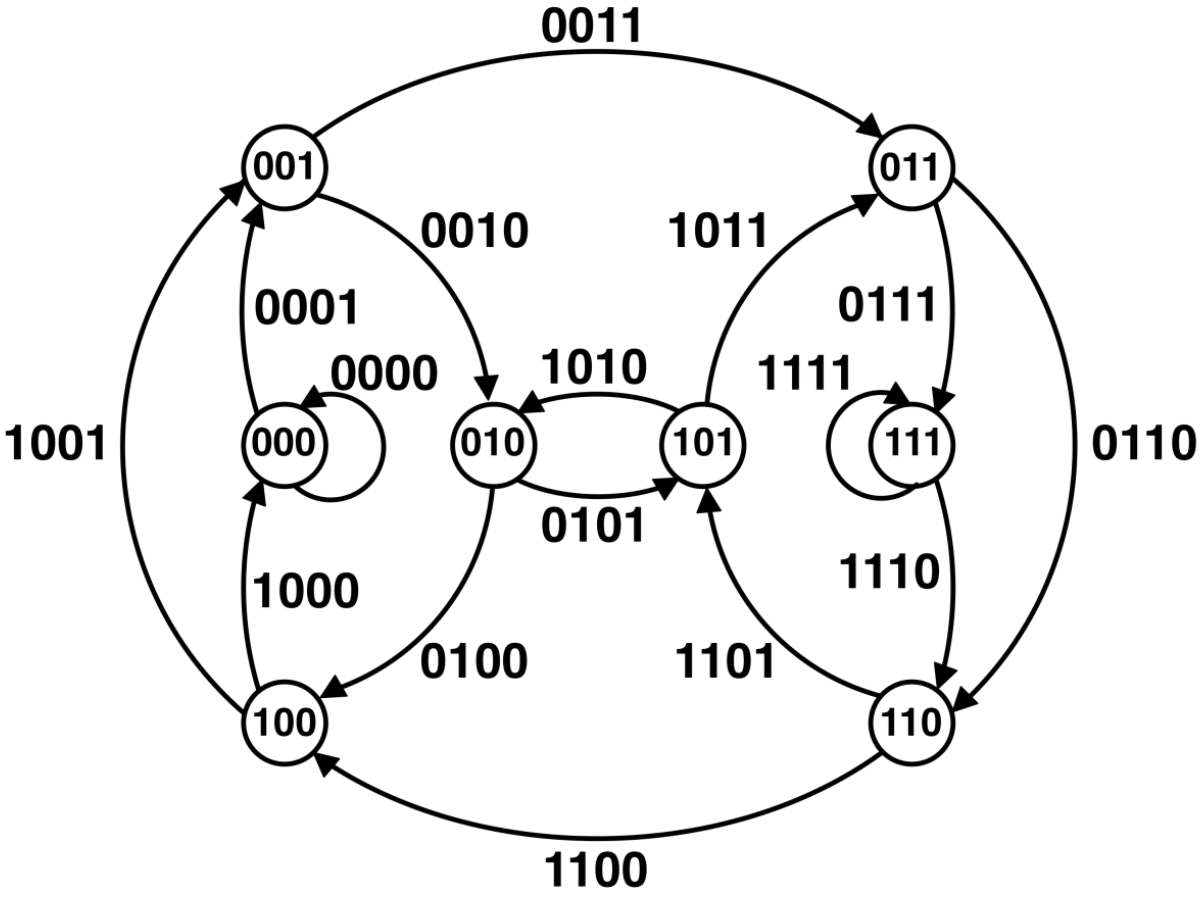


From Hamilton  to Euler  to de Bruijn 





De Bruijn graph for 4-universal strings

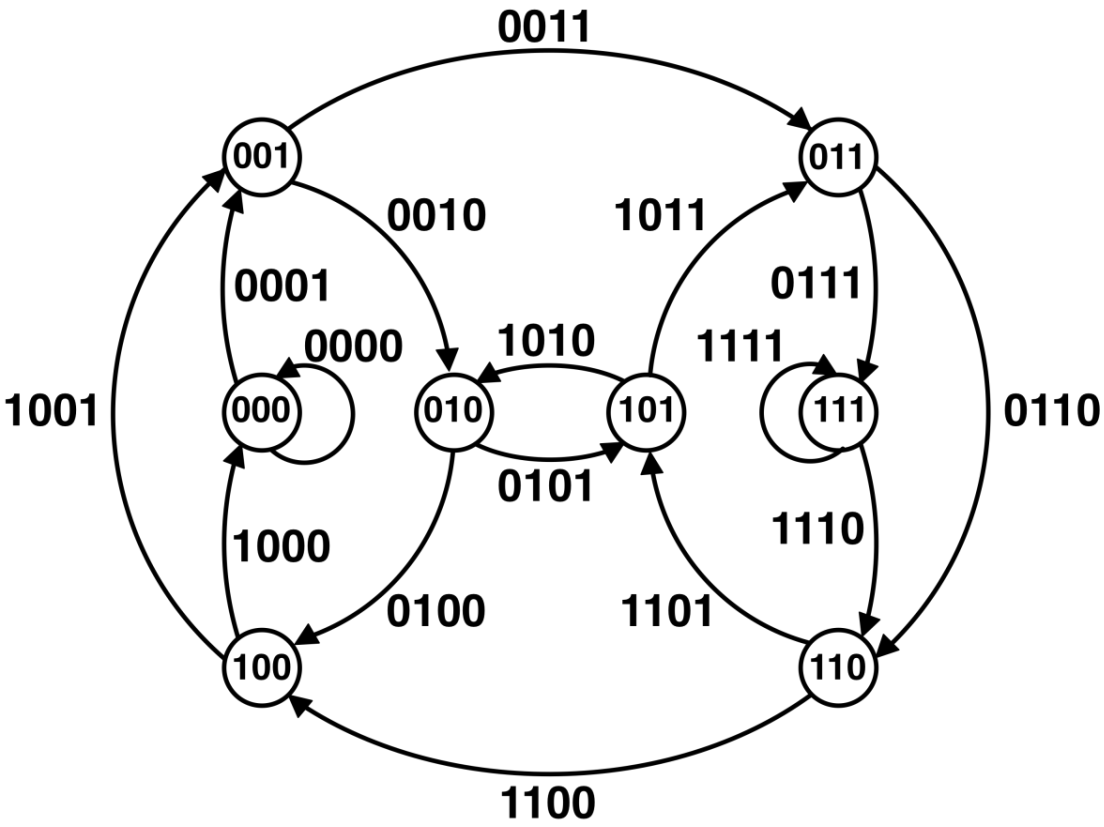


Does it have an Eulerian cycle? If yes, how can we find it?

Outline

- 2011 European *E. coli* outbreak
- Assembling phage genome
- DNA arrays
- Assembling genomes from *k*-mers
- De Bruijn graphs
- Bridges of Königsberg and universal strings
- **Euler theorem**
- Splitting the genome into contigs
- From reads to read-pairs
- Genome assembly faces real sequencing data

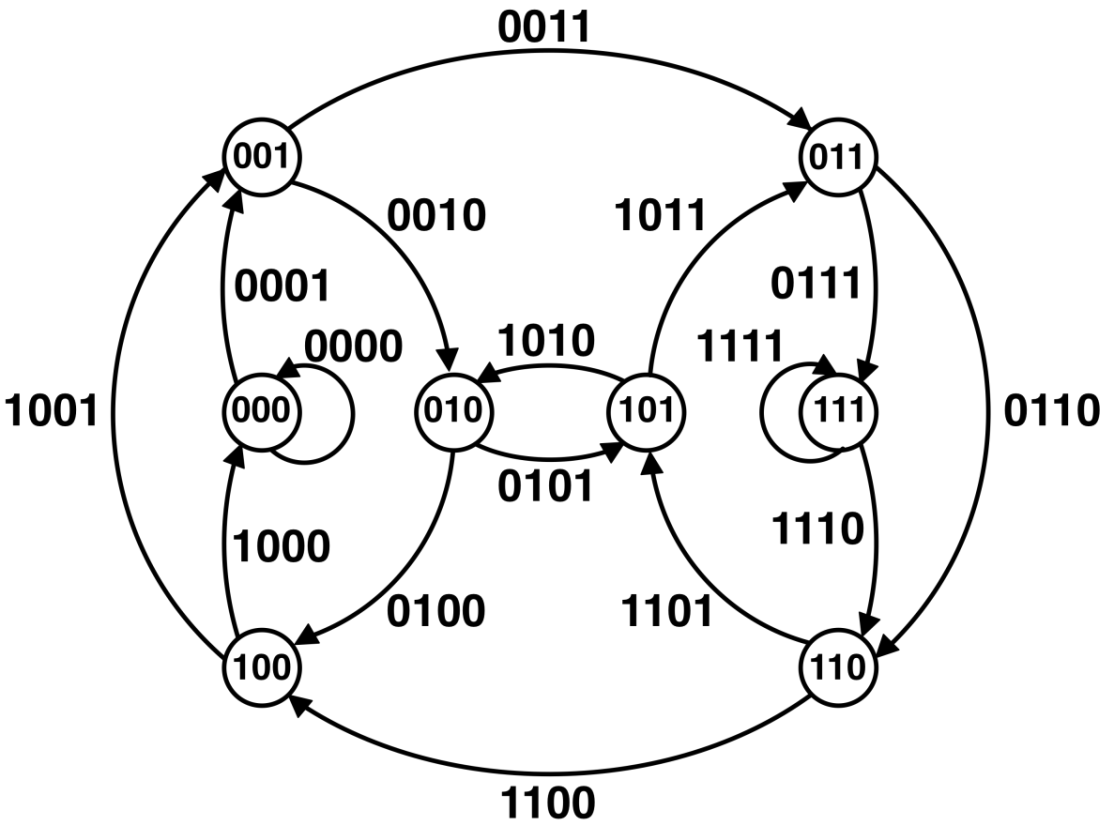
Is the graph for 4-universal strings balanced?



A graph is balanced if indegree = outdegree for each node

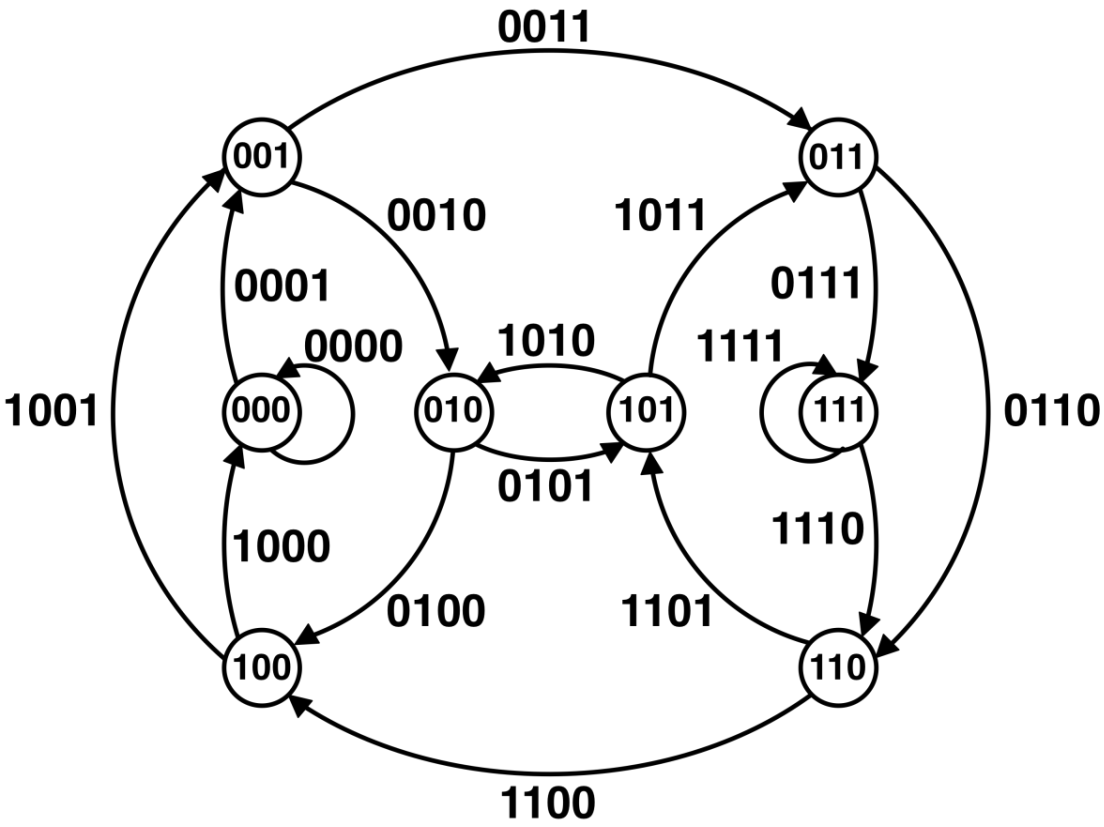
Euler's Theorem

- Every Eulerian graph is balanced



Euler's Theorem

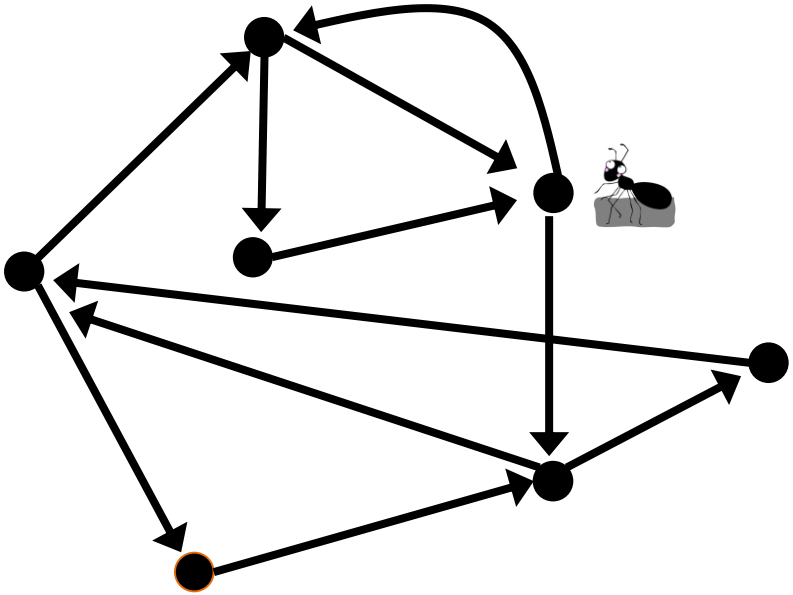
- Every Eulerian graph is balanced
- **Every balanced* graph is Eulerian**



(*) and strongly connected, of course!

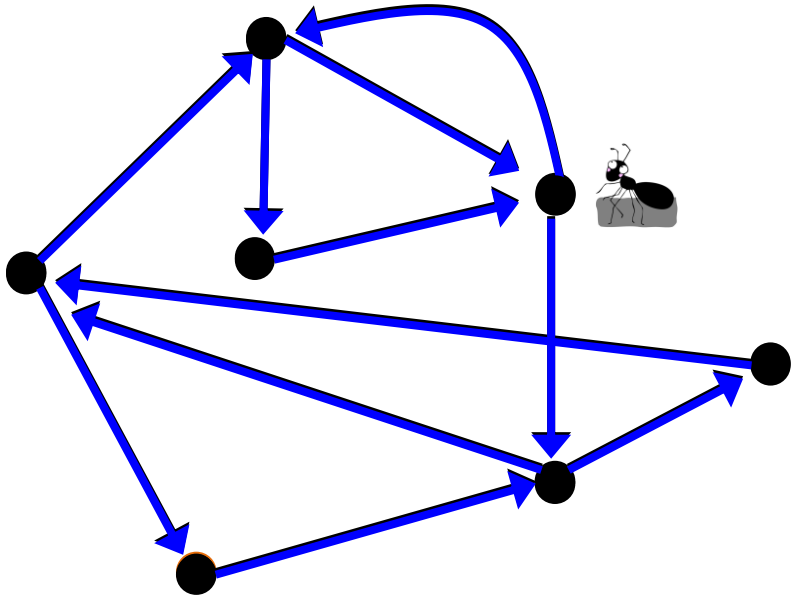
Recruiting an ant

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

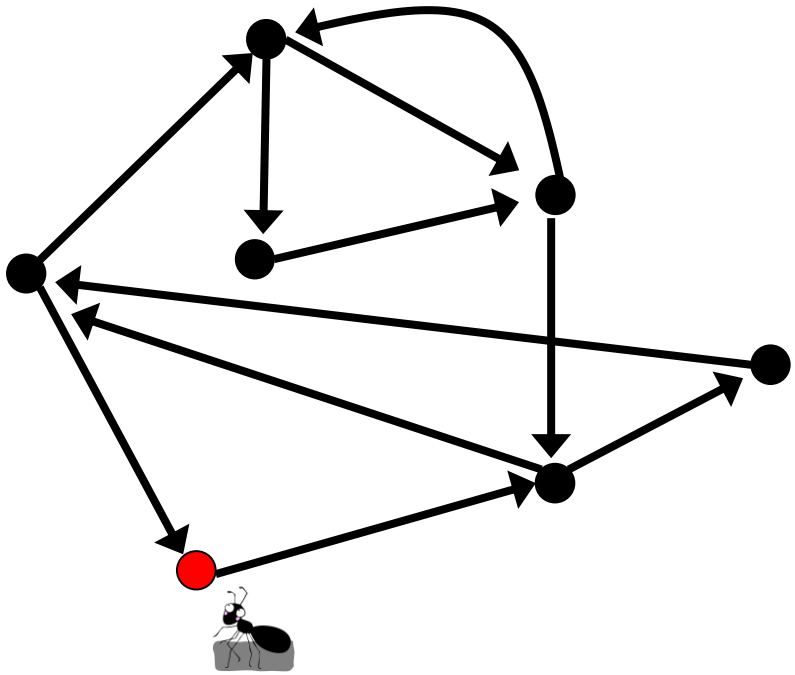


If ant was a genius…

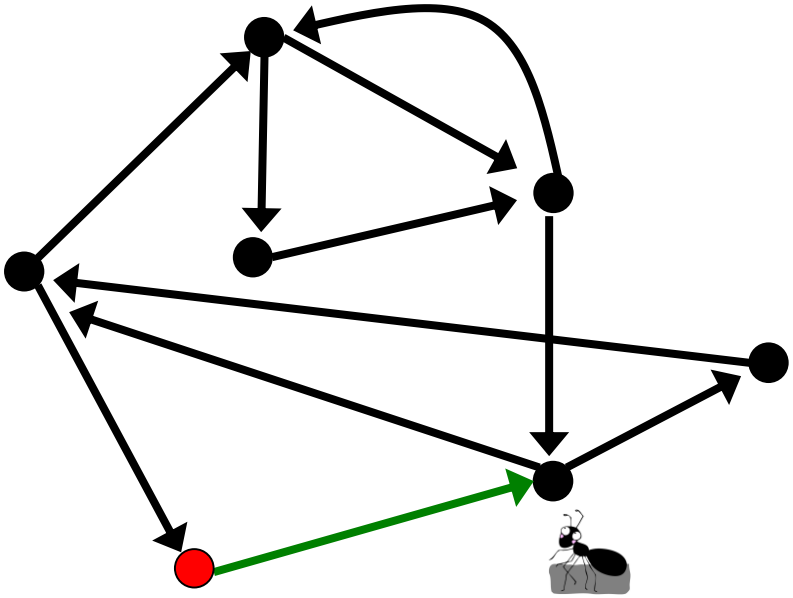
"Yay! Now can I go home please?"



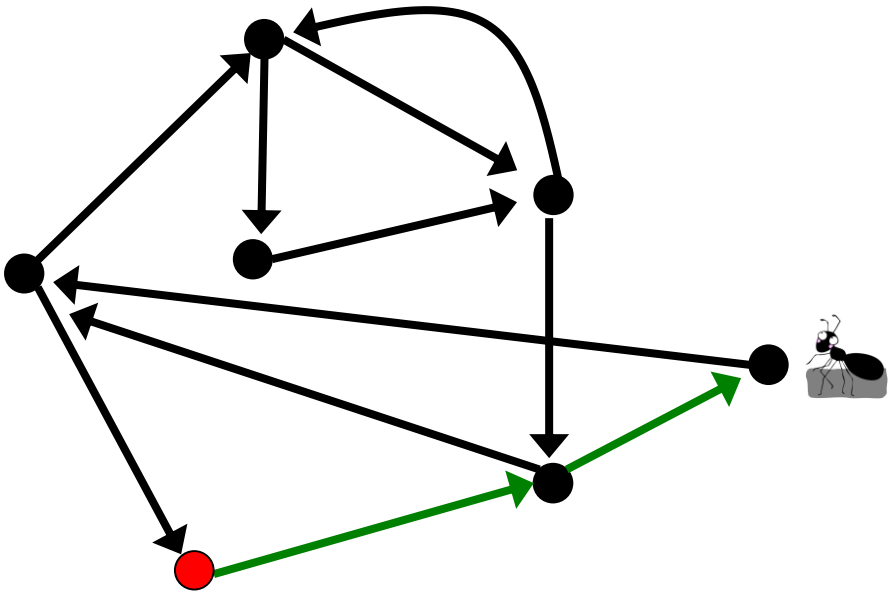
A less intelligent ant...



Walking…

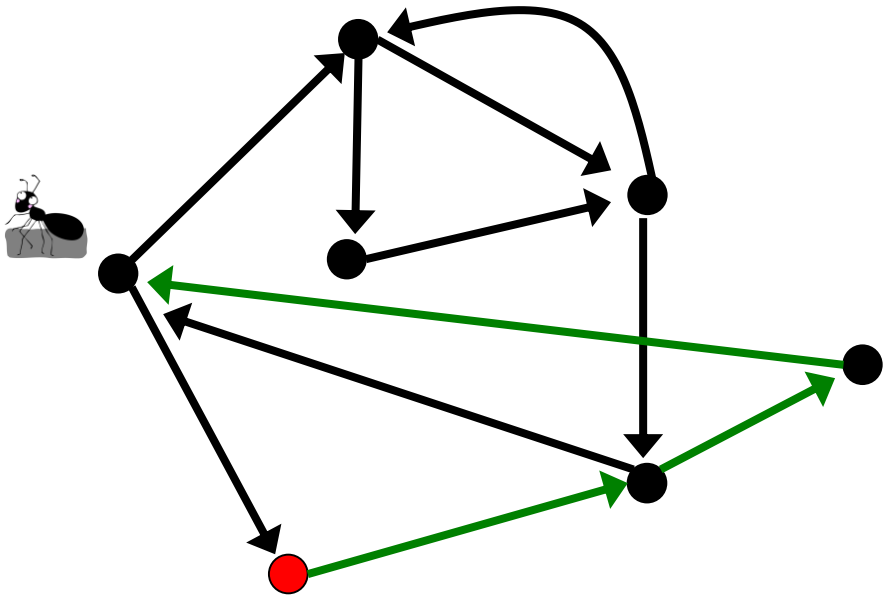


Walking… and walking…

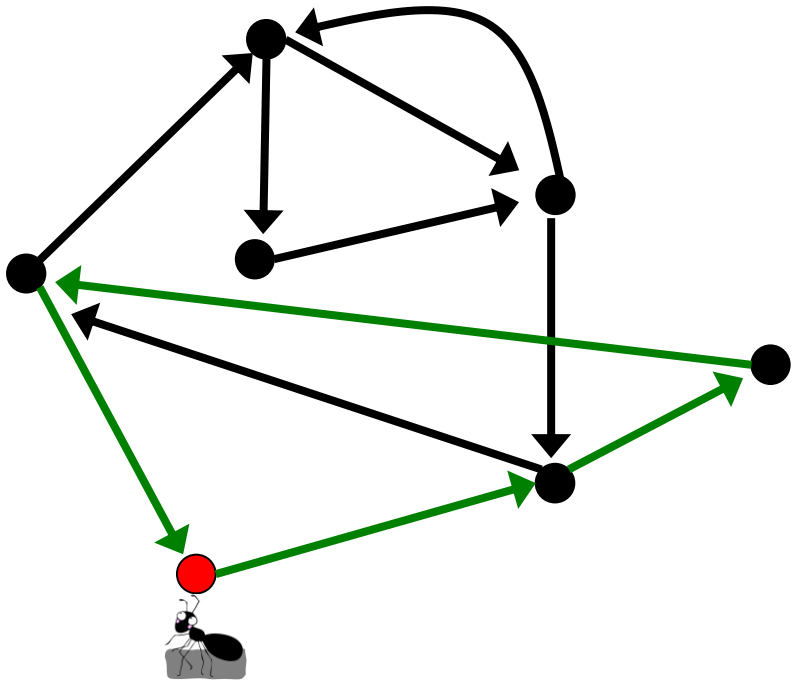


Walking… and walking… and walking…

Can it get stuck? In what vertex?

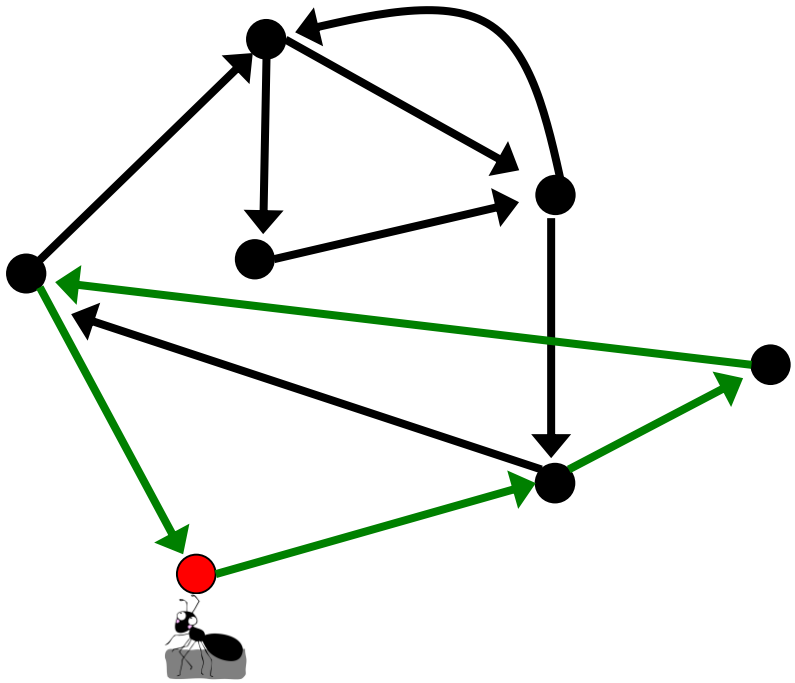


The ant can only get stuck at the starting vertex



The ant has completed a cycle BUT it is not Eulerian…

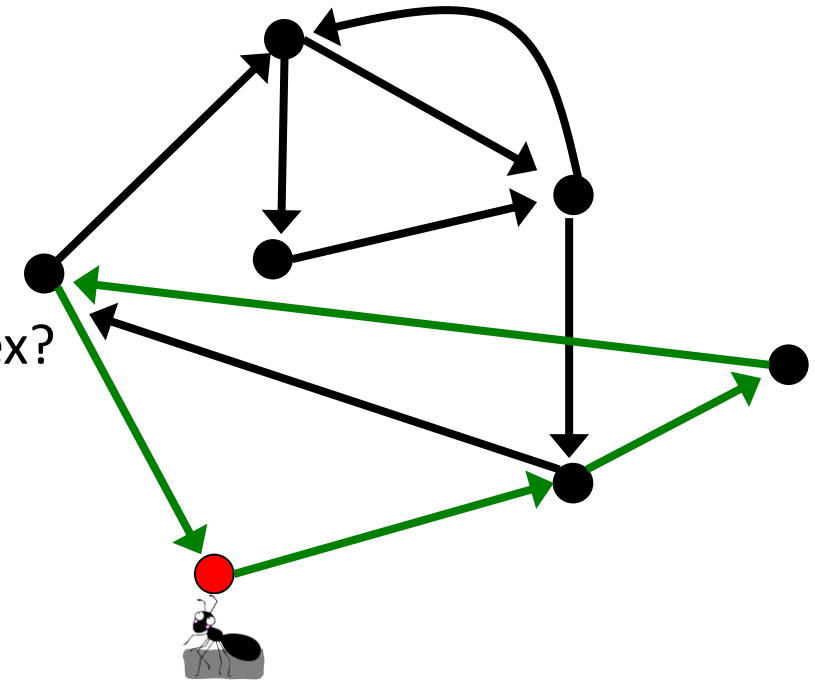
Can we enlarge this cycle?



Let's start at a different vertex in the green cycle
(with still unexplored edges)

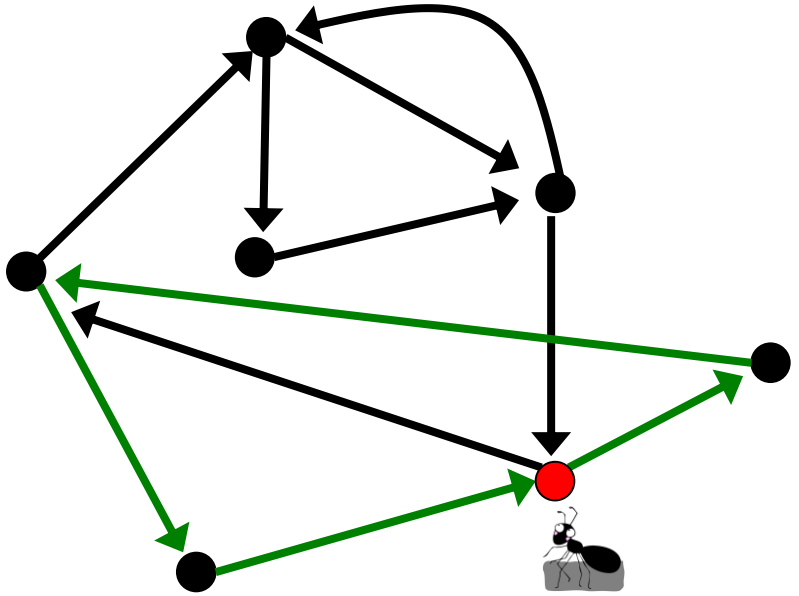
Why should I start at a different vertex?

What difference does it make???



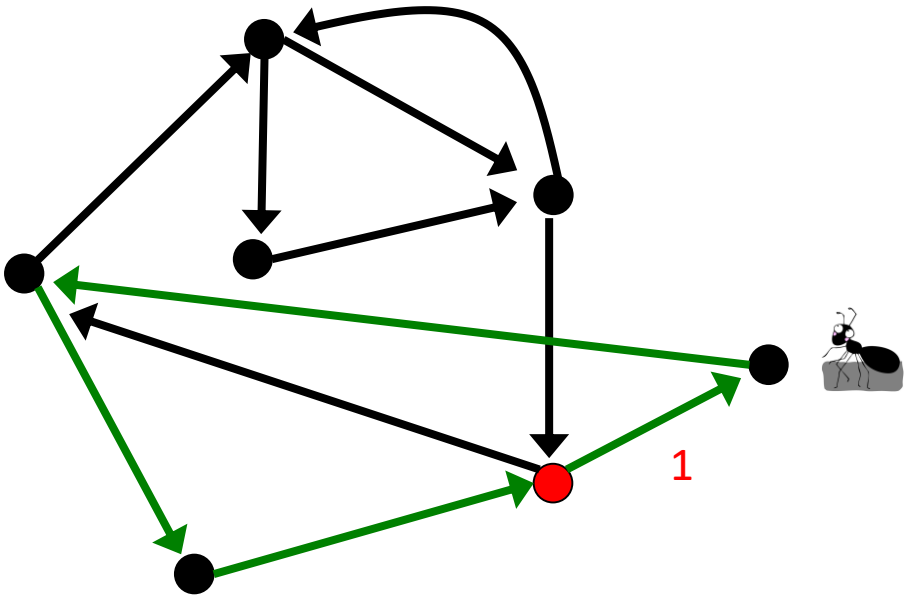
New walking instructions

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



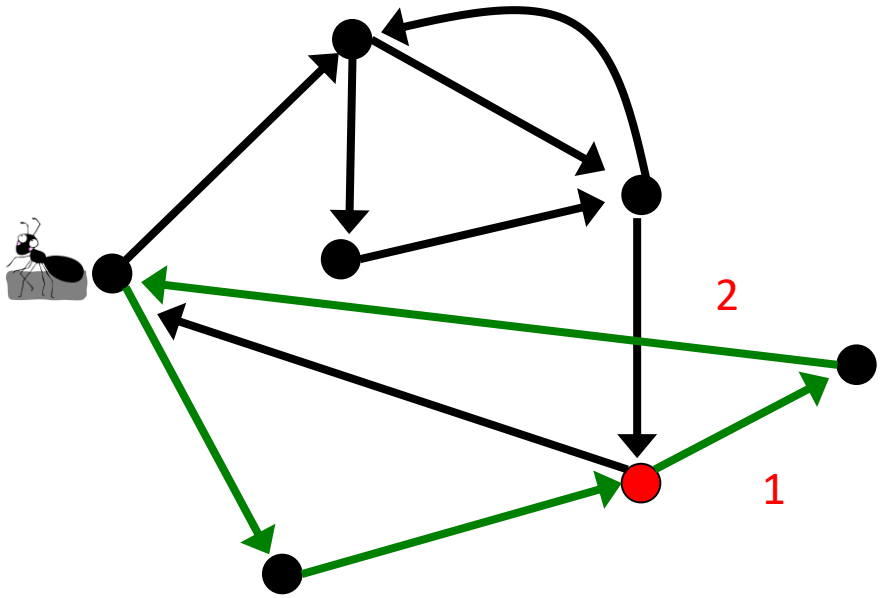
Traversing previously constructed cycle

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



Traversing previously constructed cycle

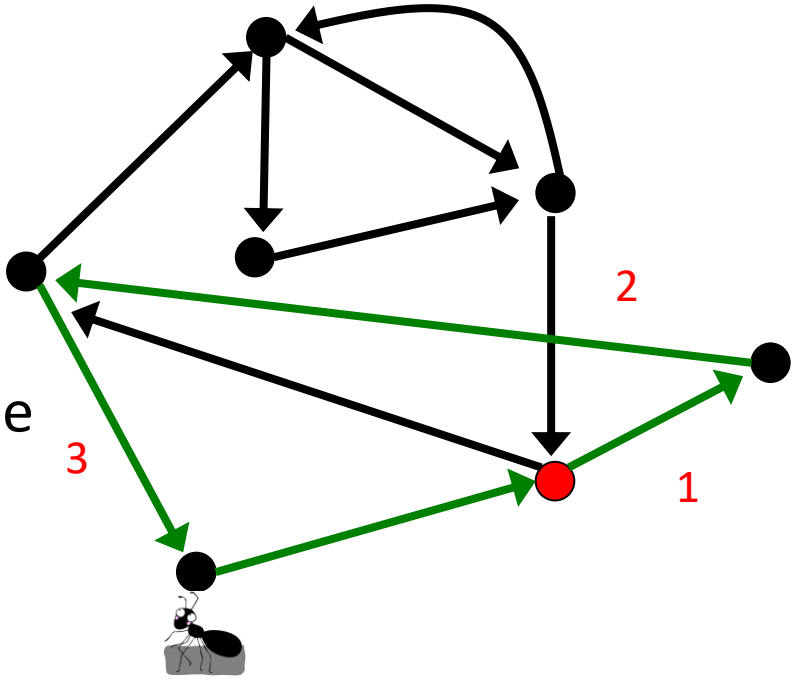
Starting at a vertex that has an unused edge, traverse the already constructed (green cycle) and return back to the starting vertex.



Traversing previously constructed cycle

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

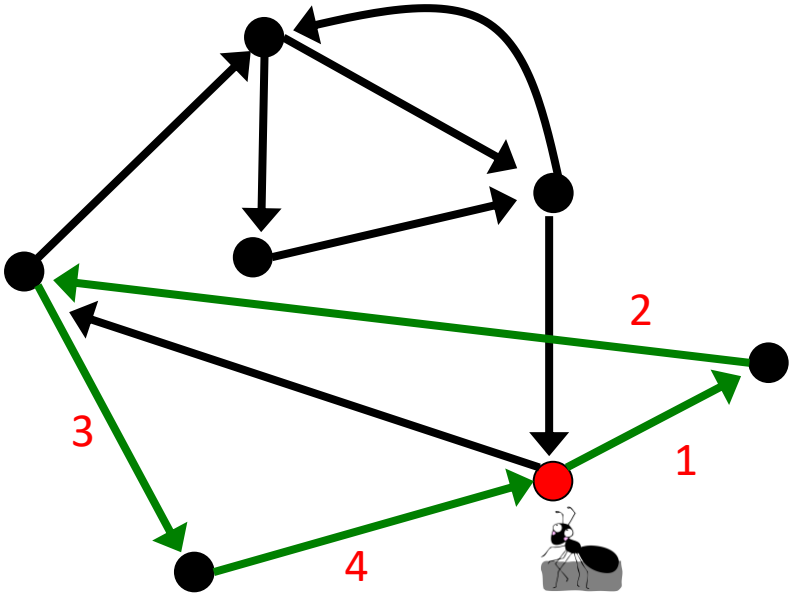
"Why do I have to walk along the same cycle again??? Can I see something new?"



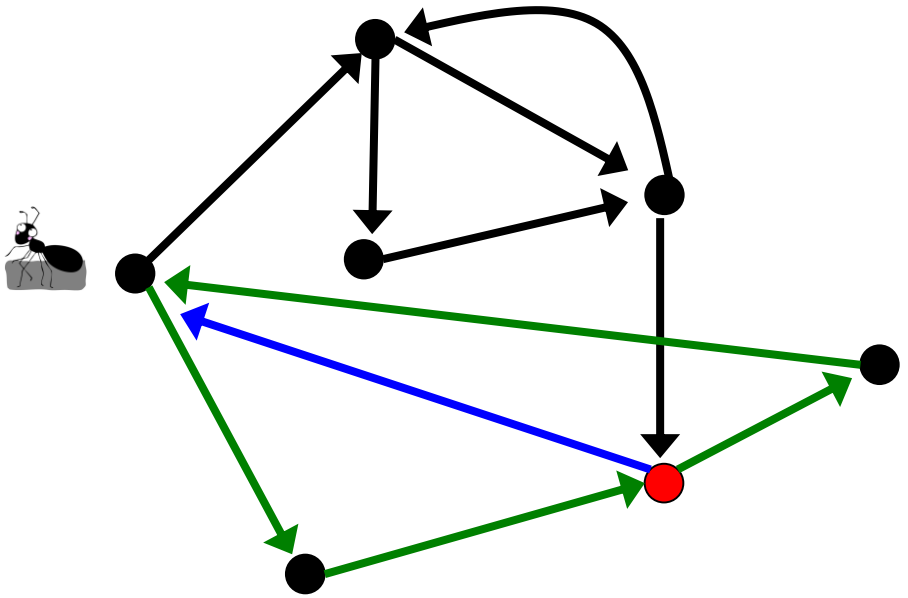
I returned back BUT… I can continue walking!

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

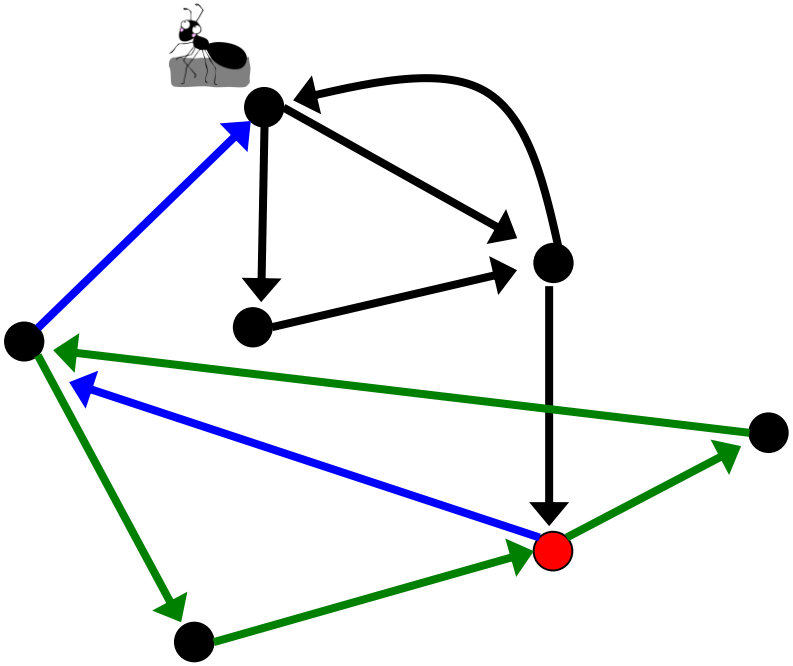
After completing the cycle, start random exploration of still untraversed edges in the graph.



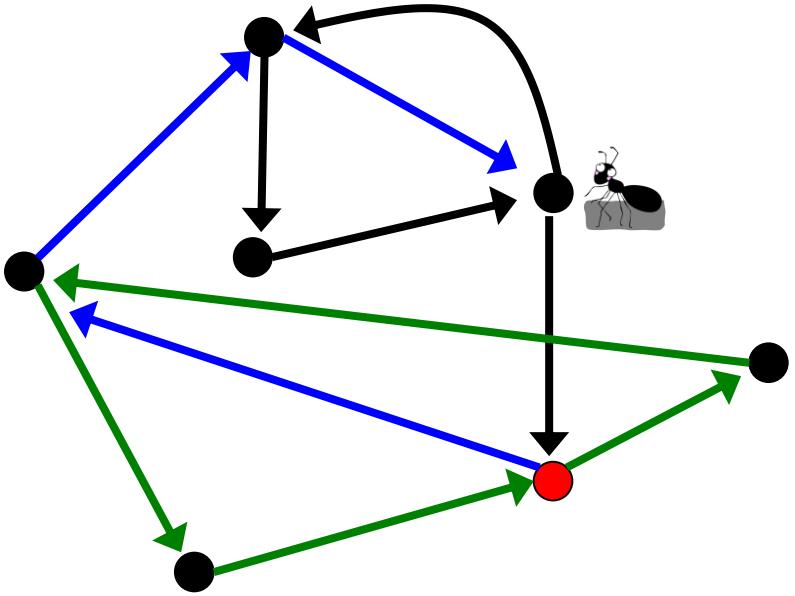
Enlarging the previously constructed cycle



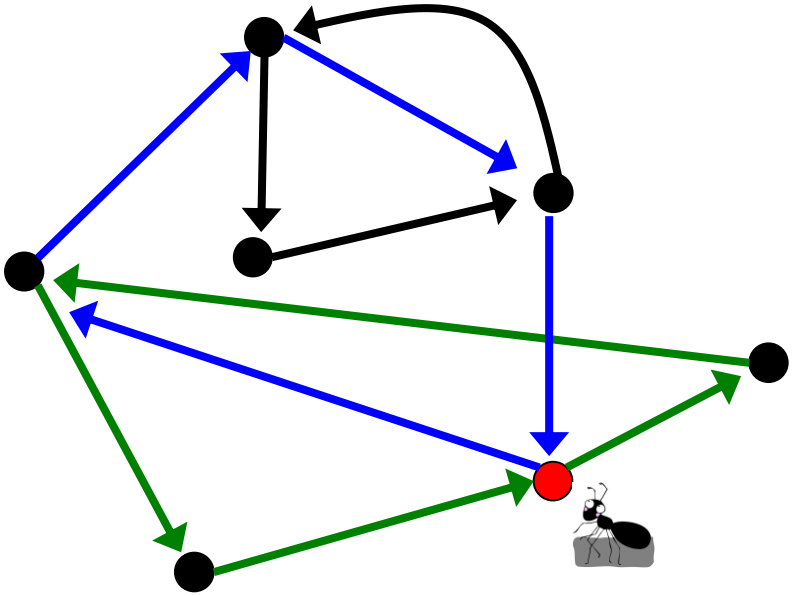
Enlarging the previously constructed cycle



Enlarging the previously constructed cycle



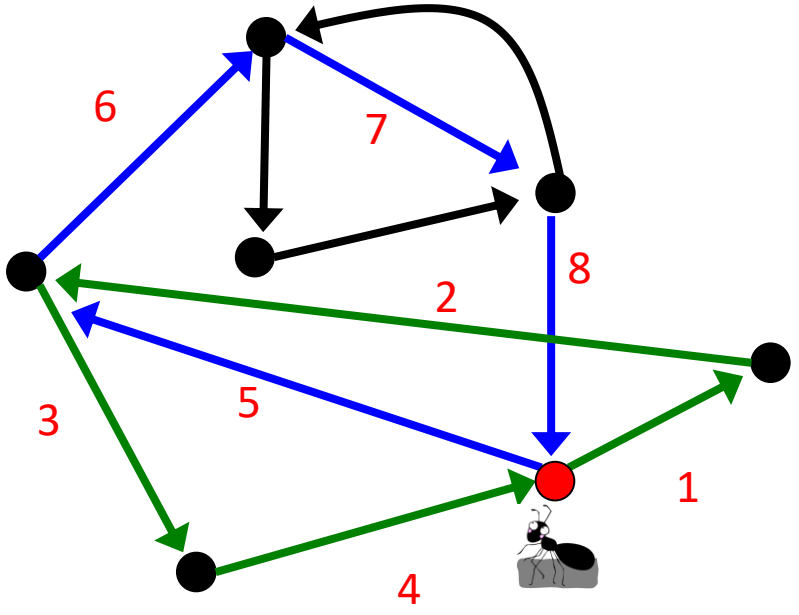
Enlarging the previously constructed cycle



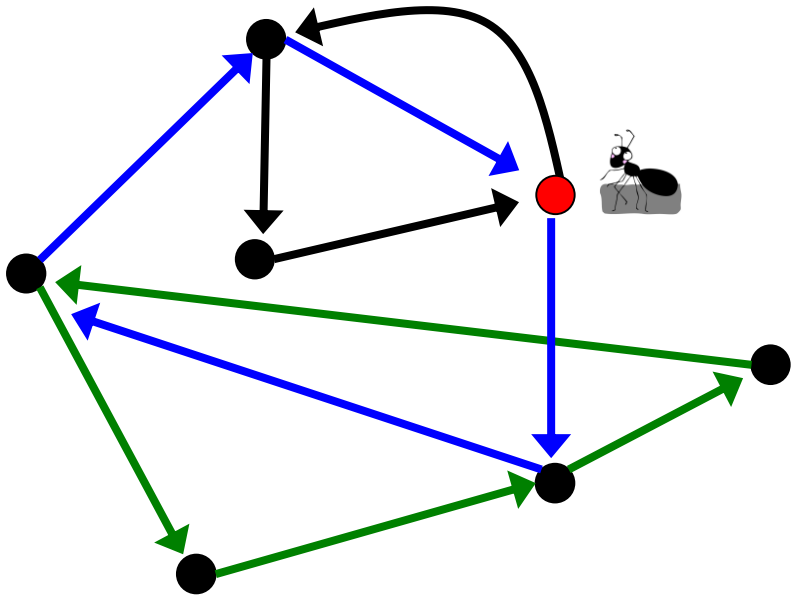
Stuck again!

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

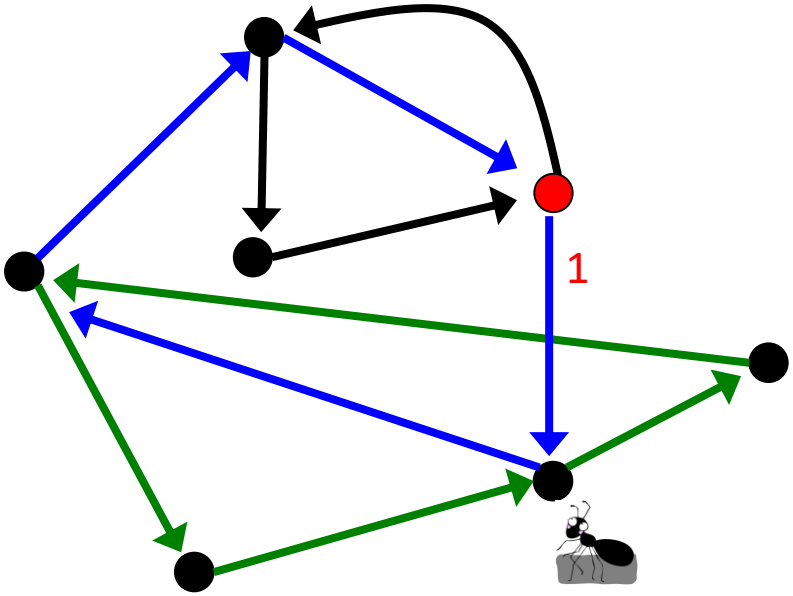
The ant should walk along the constructed green-blue cycle starting at yet another vertex. Which one?



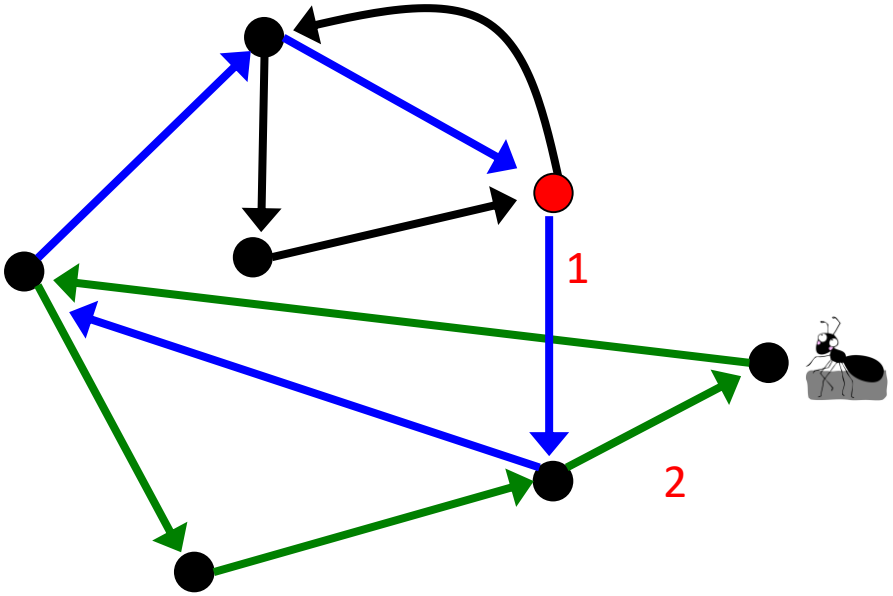
Starting at a new vertex, again…



Traversing the green-blue cycle again

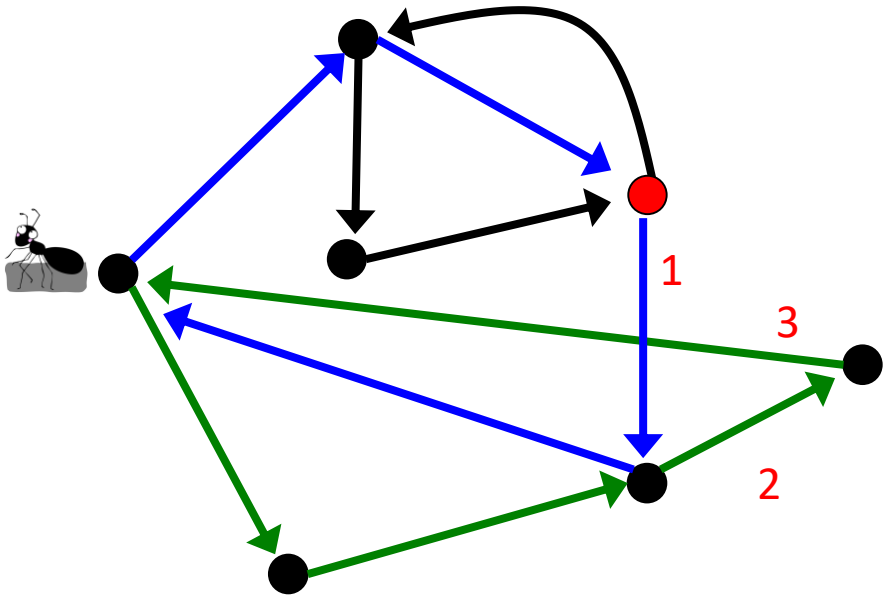


Traversing the green-blue cycle again

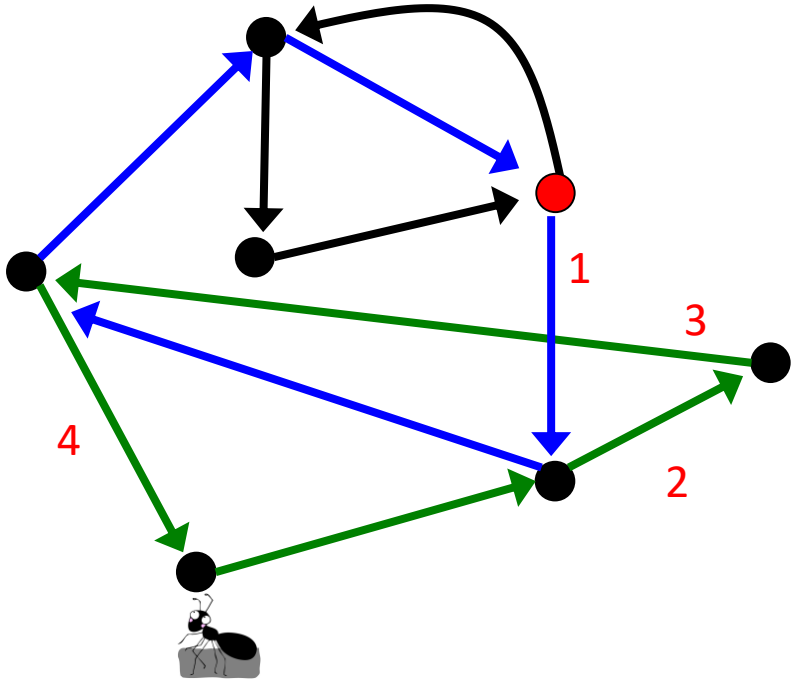


Traversing the green-blue cycle again

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

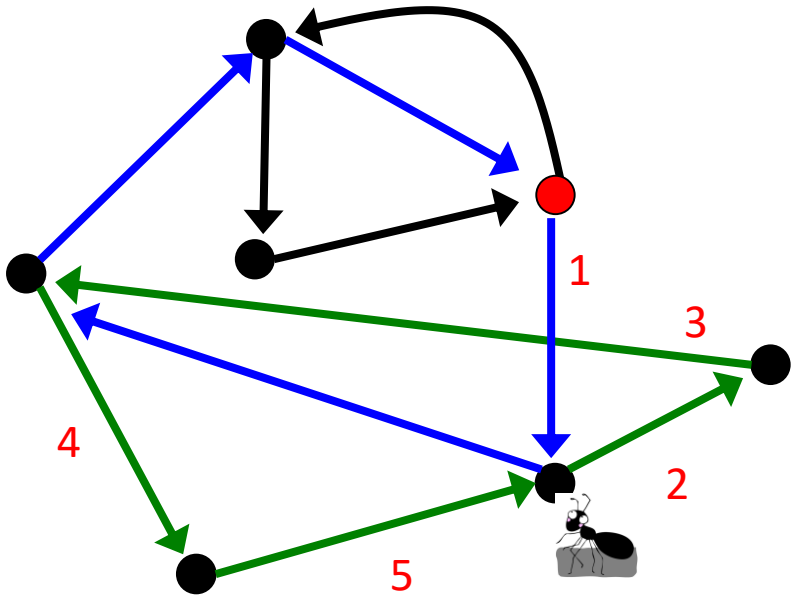


Traversing the green-blue cycle again

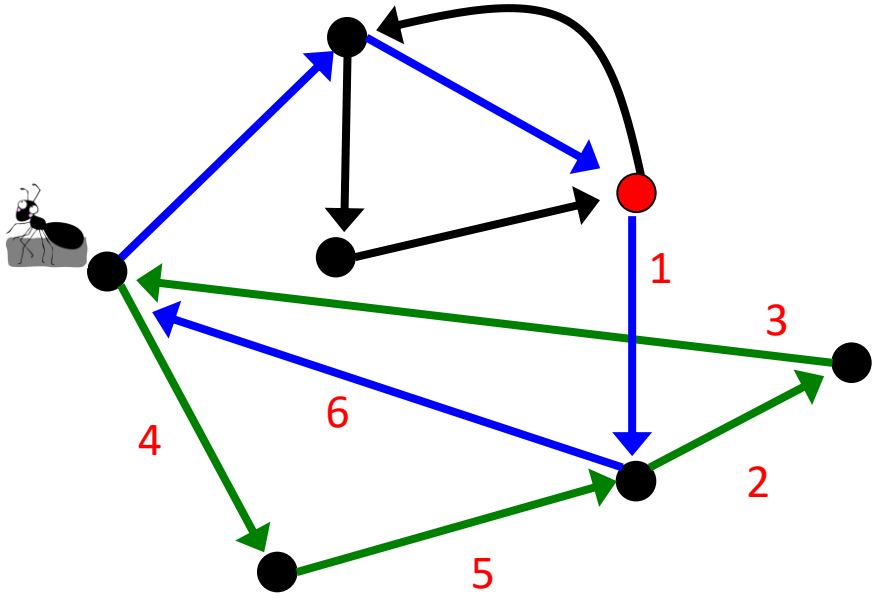


“These instructions are stupid…”

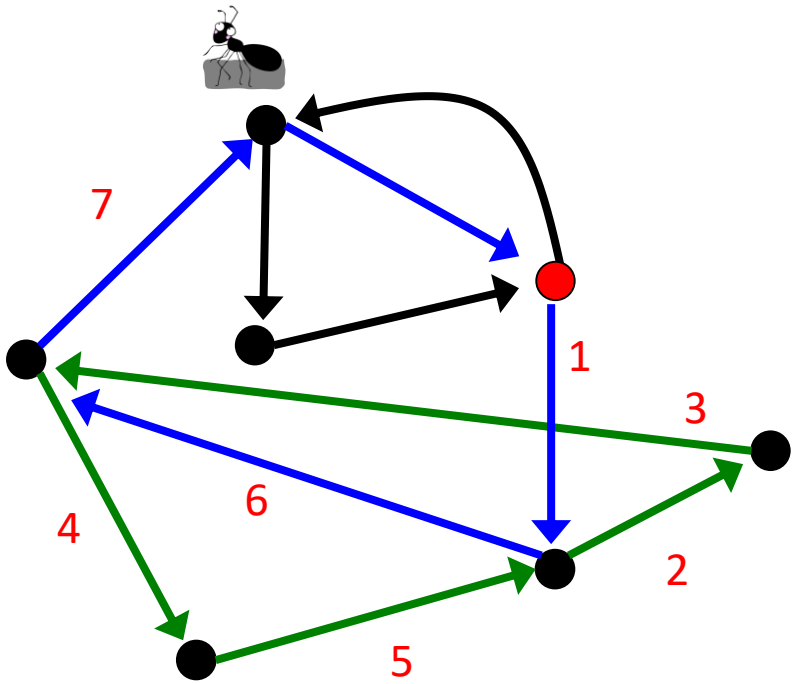
Traversing the green-blue cycle again



Traversing the green-blue cycle again

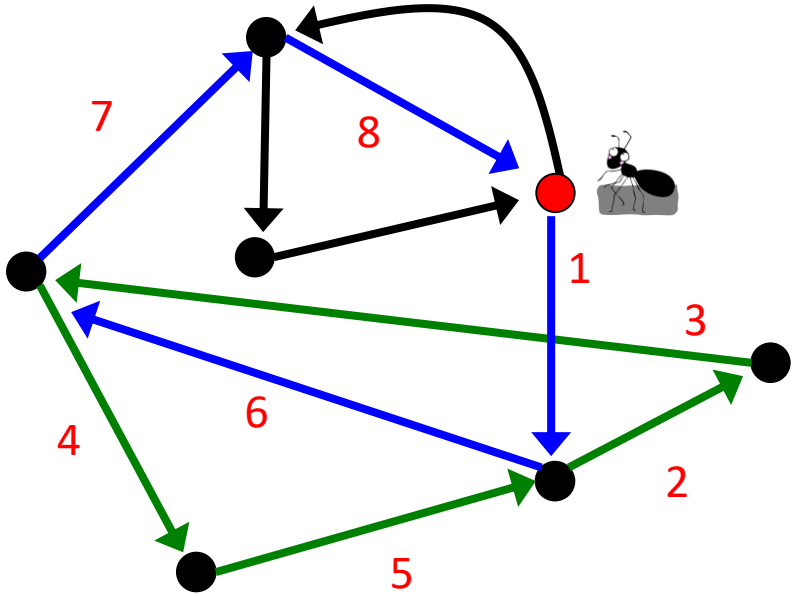


Traversing the green-blue cycle again

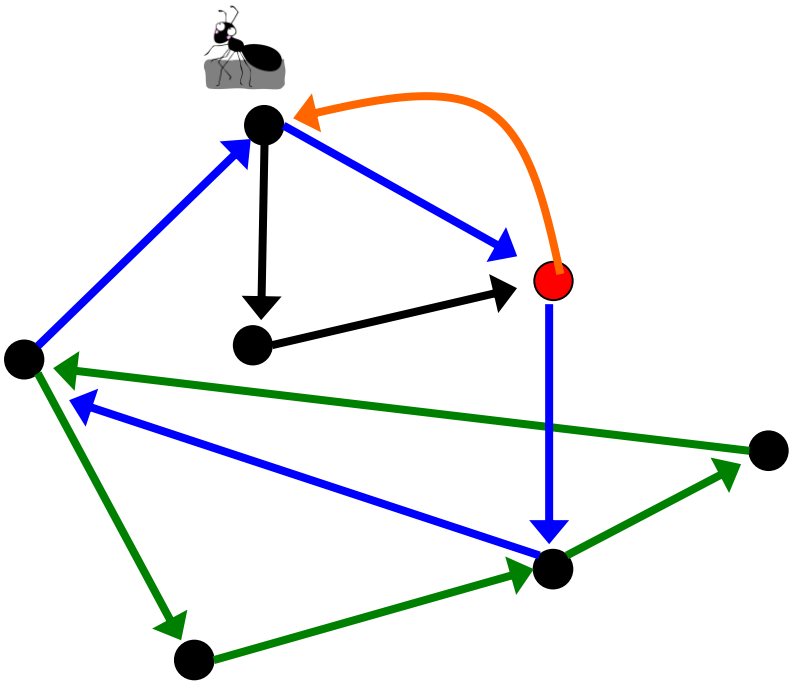


I returned back BUT… I can continue walking!

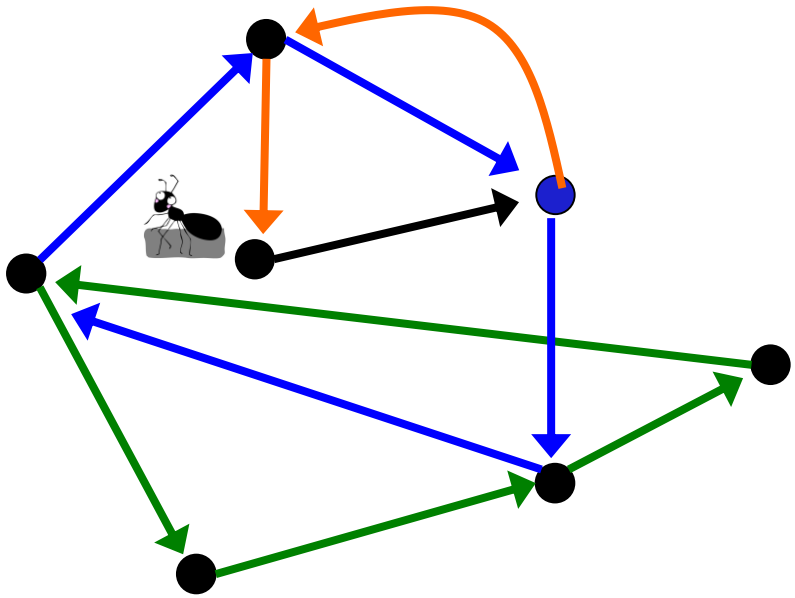
“Hmm, maybe these instructions were not that stupid…”



Enlarging the green-blue cycle

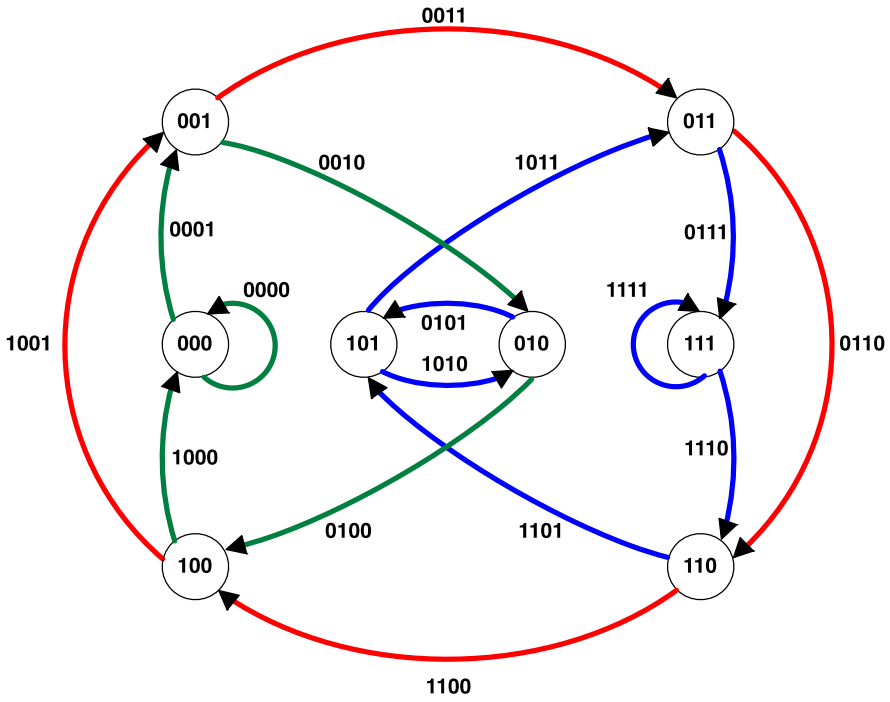


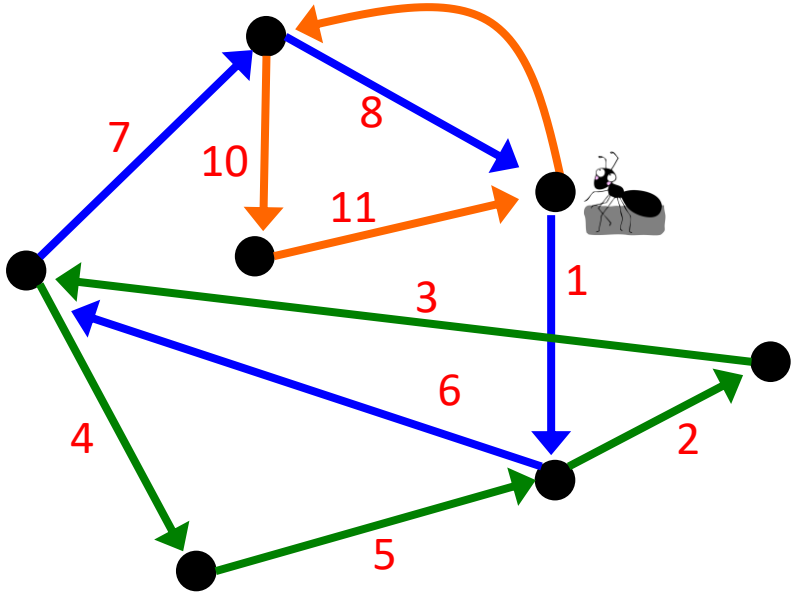
Enlarging the green-blue cycle



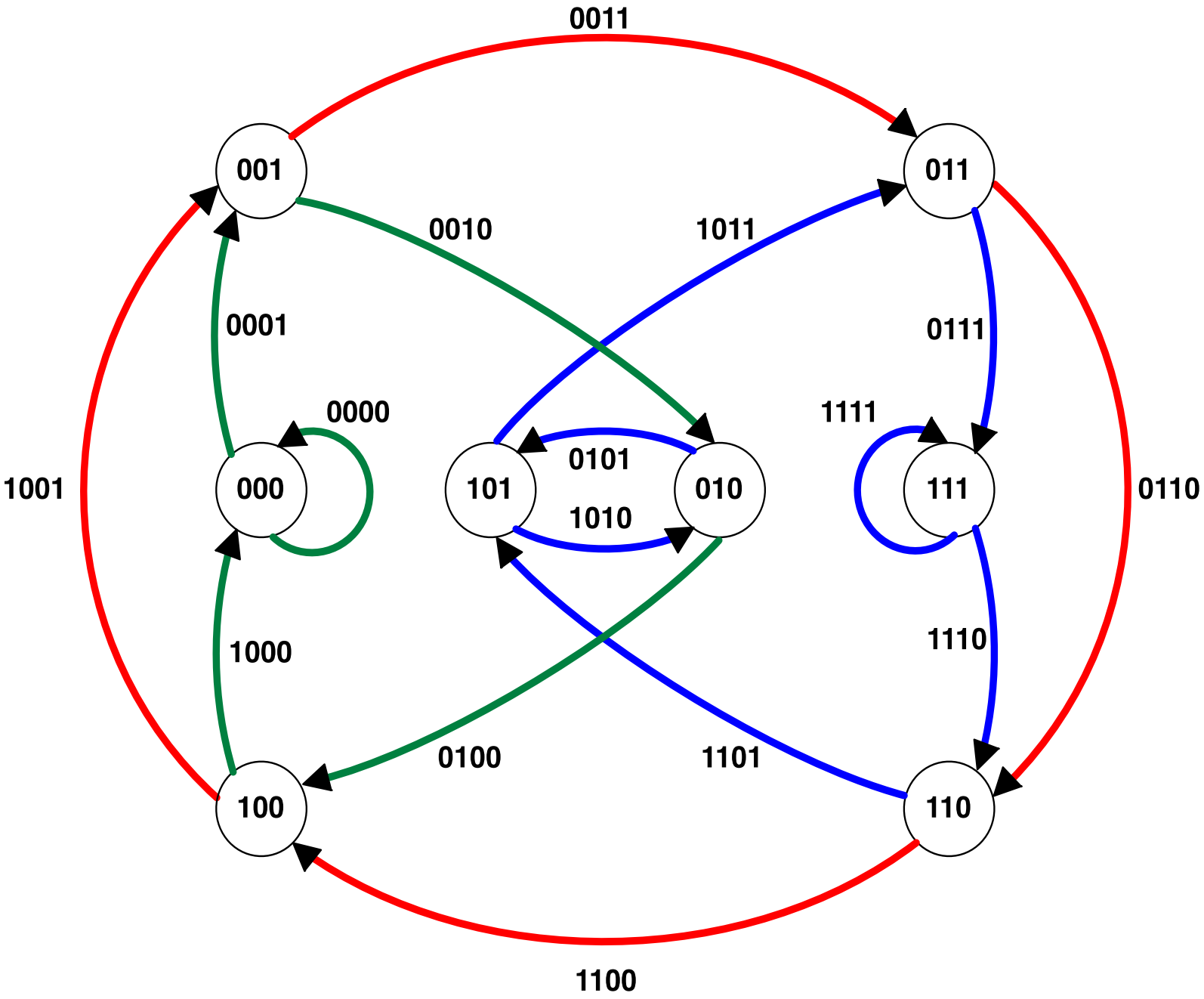
I proved Euler's Theorem! Can I go home please?

EulerianCycle(*BalancedGraph*)
  form a *Cycle* by randomly walking in *BalancedGraph* (dont visit the same edge twice!)
    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*





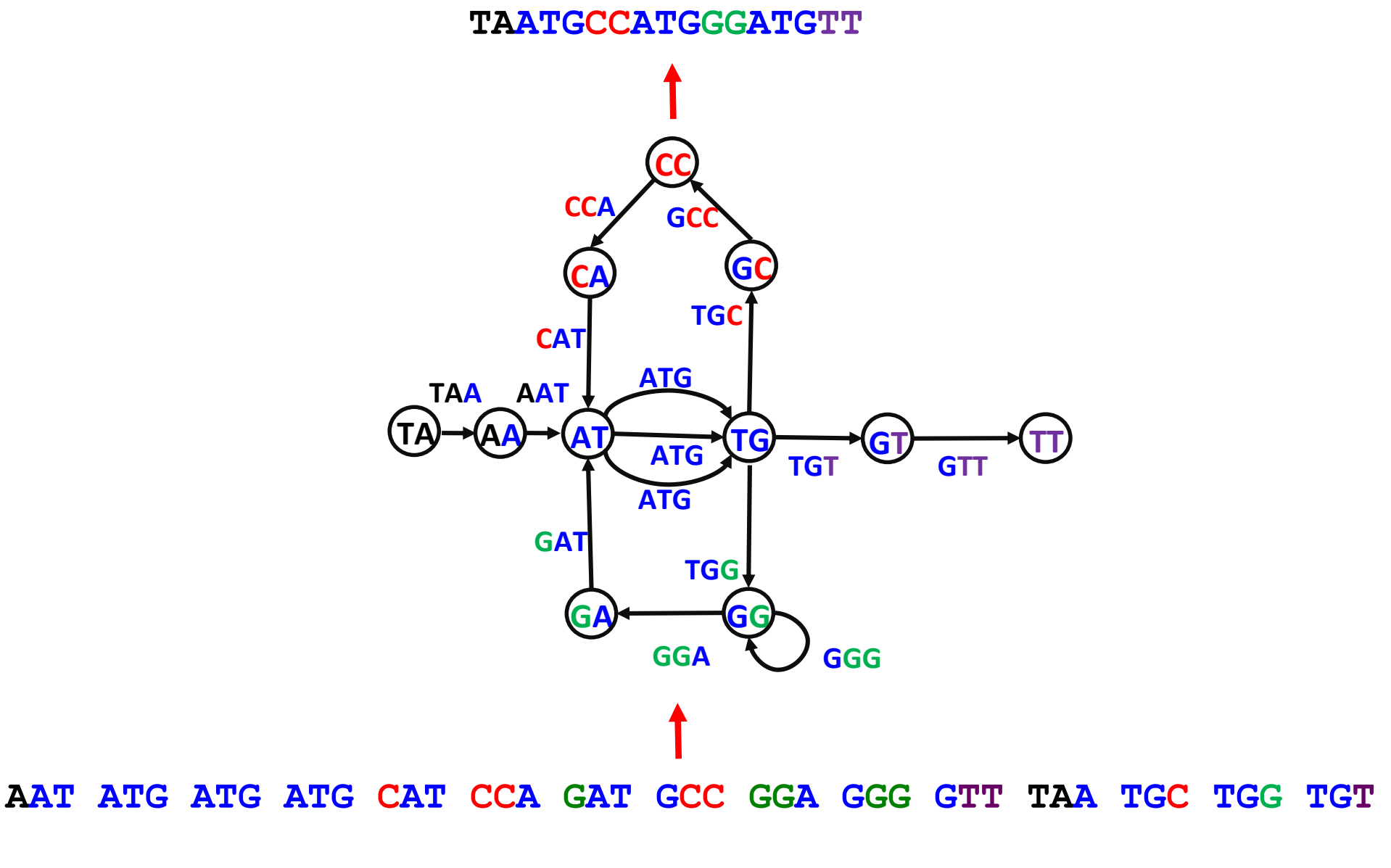
Problem: Implement linear-time algorithm for constructing Eulerian cycles


0011
001
011
0010
1011
0001
0111
0000
1111
1001
000
101
0101
010
111
0110
1010
1000
1110
0100
1101
100
110
1100

Outline

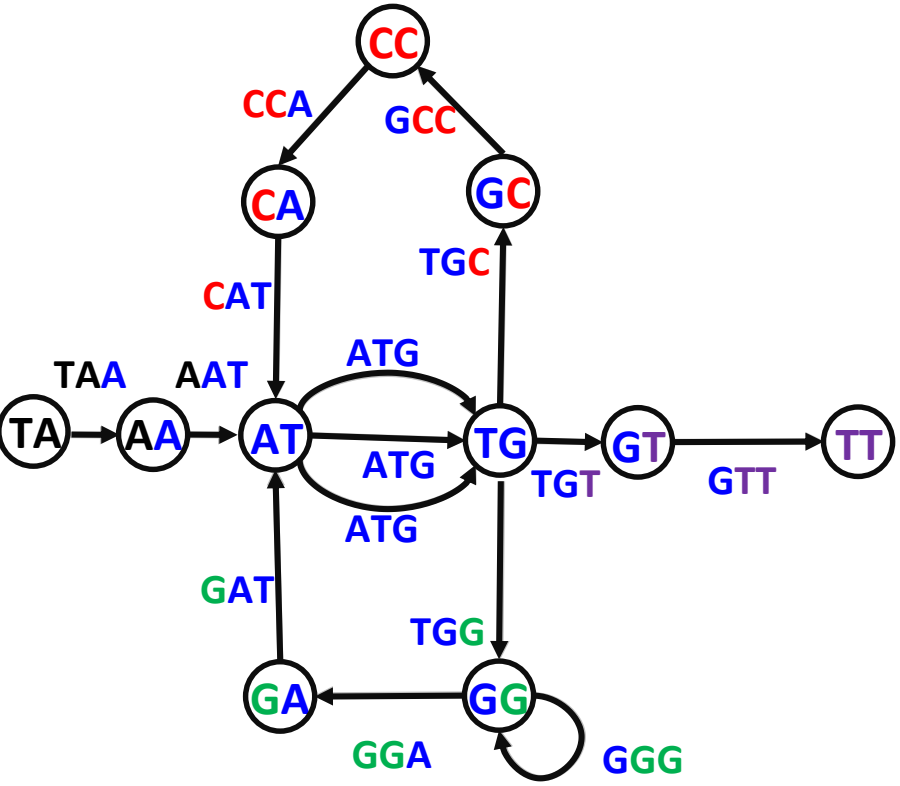
- 2011 European *E. coli* outbreak
- Assembling phage genome
- DNA arrays
- Assembling genomes from *k*-mers
- De Bruijn graphs
- Bridges of Königsberg and universal strings
- Euler theorem
- **Splitting the genome into contigs**
- From reads to read-pairs
- Genome assembly faces real sequencing data

From Reads to de Bruijn Graph to Genome

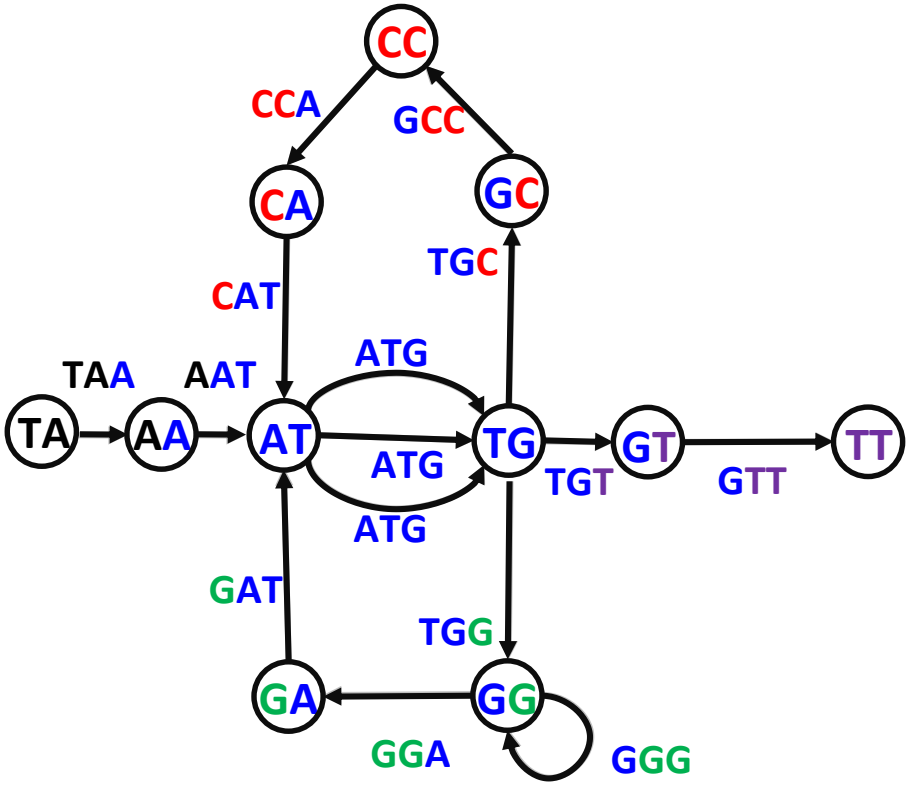


Multiple Eulerian Paths

TAATGCCATGGGATGTT

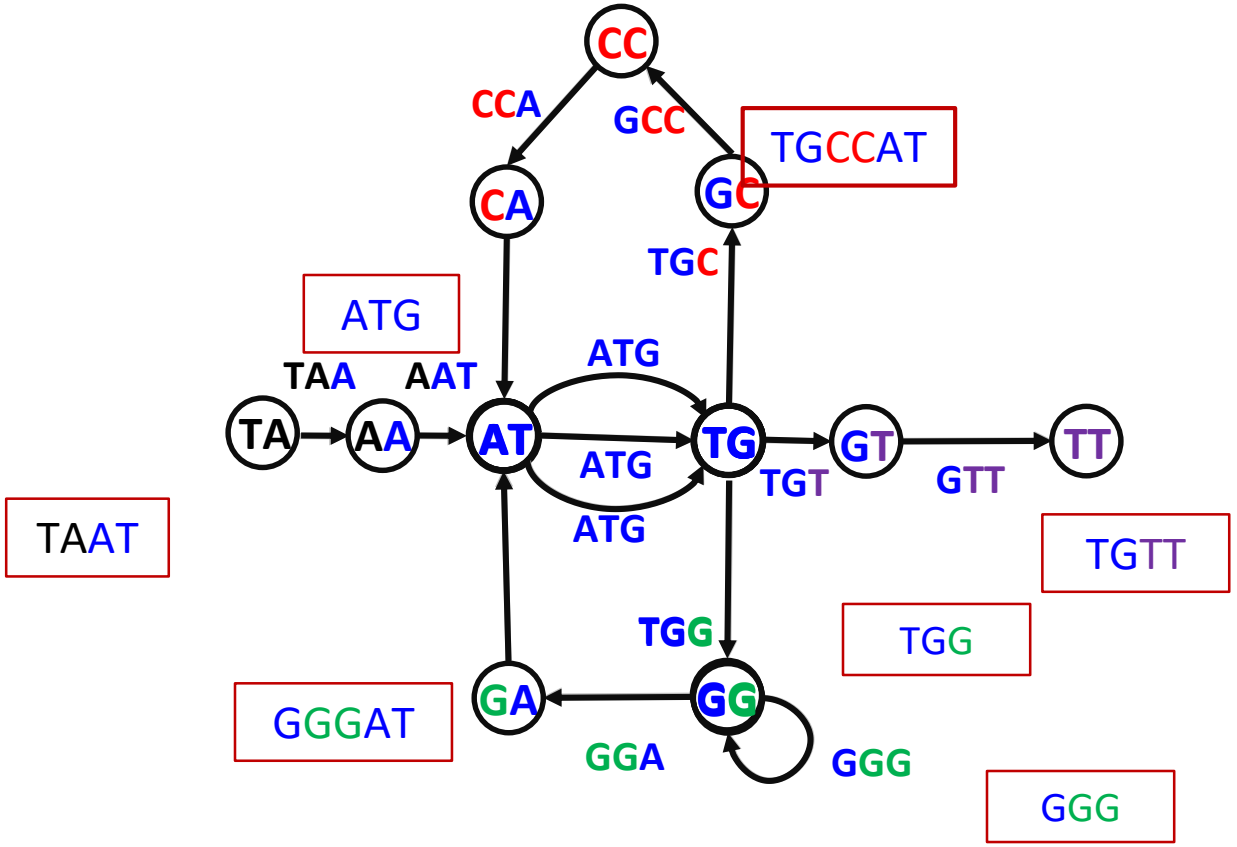


TAATGGGATGCCATGTT



Breaking Genome Into Contigs

TAATGCCATGGGATGTT



Contig Generation Problem: Given a set of *k*-mers *Patterns*, generate all contigs in **DeBRUIJN**$_k$**(***Patterns***)**

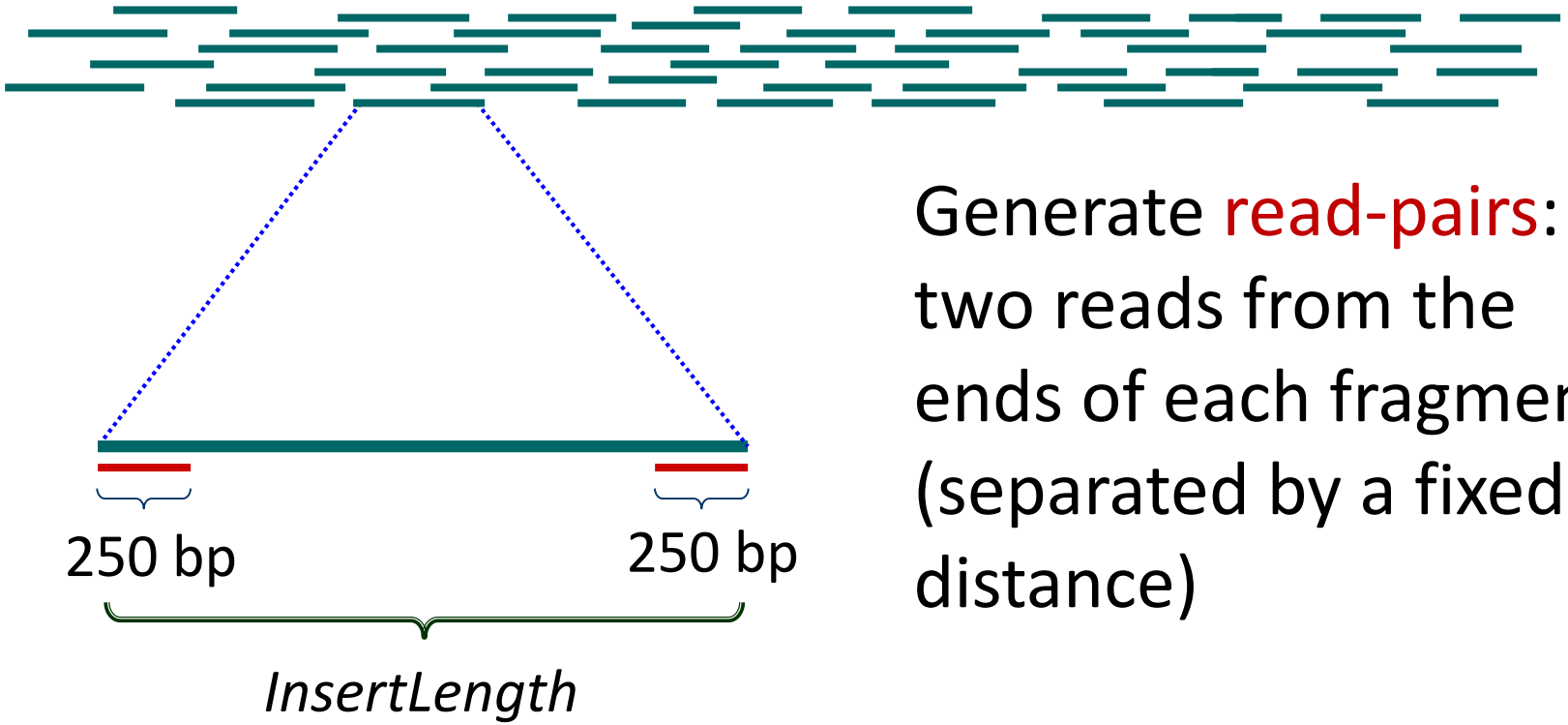
Outline

- 2011 European *E. coli* outbreak
- Assembling phage genome
- DNA arrays
- Assembling genomes from *k*-mers
- De Bruijn graphs
- Bridges of Königsberg and universal strings
- Euler theorem
- Splitting the genome into contigs
- **From reads to read-pairs**
- Genome assembly faces real sequencing data

Genome Sequencing with Read-Pairs

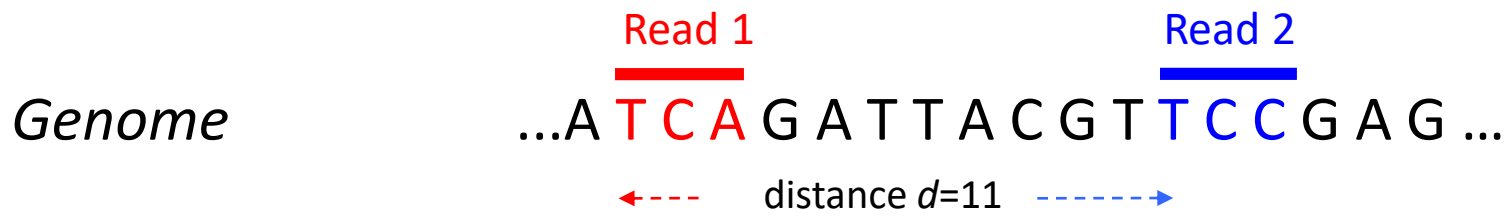
Multiple identical copies of genome

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



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

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.

Disclaimer: 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

String Reconstruction from Read-Pairs

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

- **Input.** A set of paired *k*-mers.
- **Output.** A string *Text* such that *PairedComposition*(*Text*) coincides with the set of paired *k*-mers.

How would de Bruijn assemble paired *k*-mers?

Hint:

Outline

- 2011 European *E. coli* outbreak
- Assembling phage genome
- DNA arrays
- Assembling genomes from *k*-mers
- De Bruijn graphs
- Bridges of Königsberg and universal strings
- Euler theorem
- Splitting the genome into contigs
- From reads to read-pairs
- **Genome assembly faces real sequencing data**

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.

In reality…

- **Imperfect** coverage of a genome by reads (reads do not start at each position of a genome)
- Reads are **error-prone**.
- Multiplicities of *k*-mers are **unknown.**
- Distances between reads within read-pairs are **inexact.**
- **Etc., etc., etc.**

1st unrealistic assumption: perfect coverage

```
atgccgtatggacaacgact
atgccgtatg
  gccgtatgga
     gtatggacaa
          gacaacgact
```

250-nucleotide reads generated by Illumina technology capture only a small fraction of 250-mers from the genome, thus violating the key assumption of the de Bruijn graphs.

Breaking reads into shorter *k*-mers

```
atgccgtatggacaacgact
atgccgtatg
  gccgtatgga
     gtatggacaa
          gacaacgact
```

```
atgccgtatggacaacgact
atgcc
 tgccg
  gccgt
   ccgta
    cgtat
     gtatg
      tatgg
       atgga
        tggac
         ggaca
          gacaa
           acaac
            caacg
             aacga
              acgac
               cgact
```

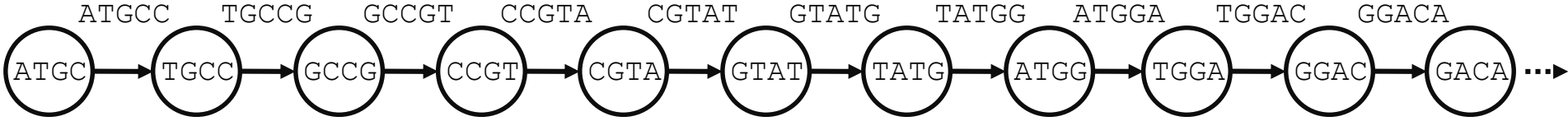
2nd unrealistic assumption: error-free reads

```
atgccgtatggacaacgact
atgccgtatg
  gccgtatgga
     gtatggacaa
          gacaacgact
    cgtaCggaca
```

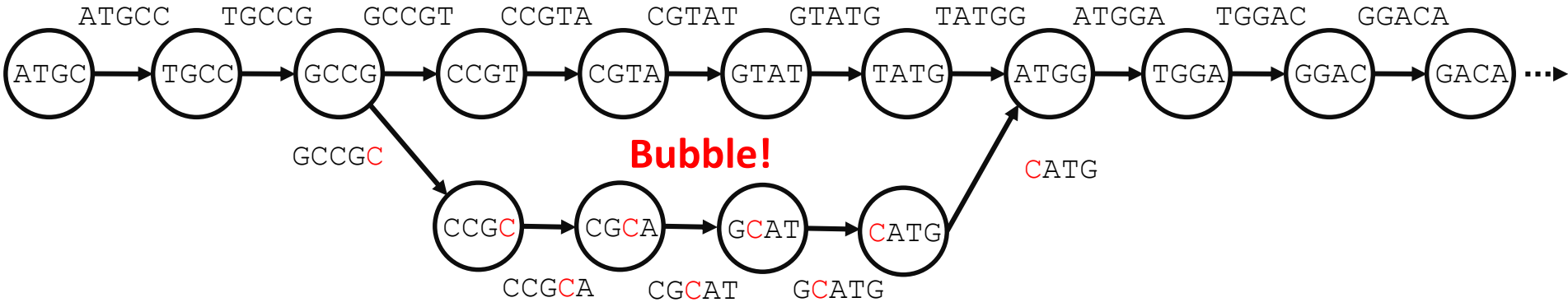
Erroneous read
(change of t into C)

```
atgccgtatggacaacgact
atgcc
 tgccg
  gccgt
   ccgta
    cgtat
     gtatg
      tatgg
       atgga
        tggac
         ggaca
          gacaa
           acaac
            caacg
             aacga
              acgac
               cgact
   cgtaC
    gtaCg
     taCgg
      aCgga
       Cggac
```

De Bruijn graph of ATGGCGTGCAATG… constructed from error-free reads

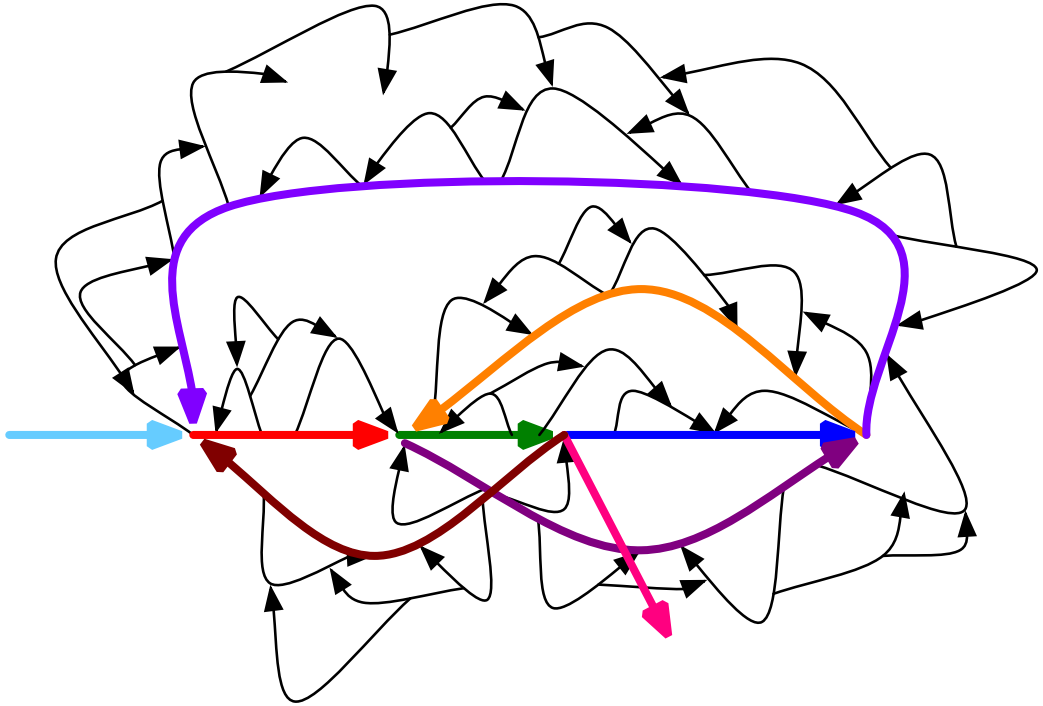


Errors in reads lead to “bubbles” in de Bruijn graphs



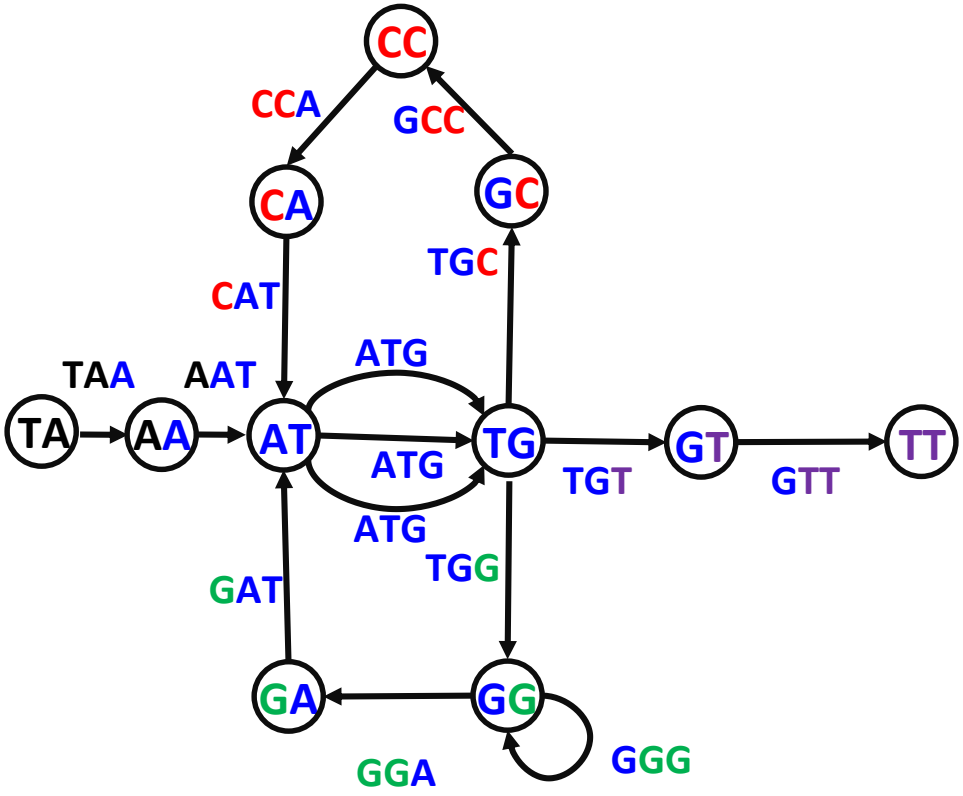
Bubble Detection Problem. Design an algorithm for detecting bubbles in a directed graph. Output the number of bubbles in the de Bruijn graph constructed from the *k*-mers occurring in 1000 error-prone reads from a mutated phi X174 genome.

Bubble explosion…
where are the correct edges of the de Bruijn graph?



Reconstructing the phi X174 genome from error-prone reads using de Bruijn graphs. Given 1000 error-prone reads from a mutated phi X174 genome, construct their de Bruijn graph and simplify it by removing bubbles and other artifacts.

3rd unrealistic assumption: multiplicities of *k*-mers are known



STOP and Think: The problem of inferring multiplicities of *k*-mers can be formulated as one of the problems you have already studied in this Specialization. Which one?

Assembling *E. coli* X genome from real reads. Given a set of real reads from *E. coli* X, reconstruct the *E. coli* X genome. How many contigs does your reconstruction have? Use the QUAST tool to generate the assembly quality report.

Assembling *E. coli* X genome from real read-pairs. Given a set of real read-pairs from *E. coli* X , reconstruct the *E. coli* X genome. How many contigs does your reconstruction have? Use the QUAST tool to generate the assembly quality report.