Eulerian and Hamiltonian Cycles

Alexander S. Kulikov

Steklov Mathematical Institute at St. Petersburg, Russian Academy of Sciences
and
University of California, San Diego

Toy Genome Assembly Problem

Find a string whose all substrings of length 3 are

AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC.

Toy Genome Assembly Problem

Find a string whose all substrings of length 3 are

AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC.

How is it related to cycles in graphs?..

Outline

Eulerian Cycles

Hamiltonian Cycles

Genome Assembly

Eulerian Cycle

An Eulerian cycle (or path) visits every edge exactly once.

Eulerian Cycle

An Eulerian cycle (or path) visits every edge exactly once.

- The definition works for both directed and undirected graphs

Eulerian Cycle

An Eulerian cycle (or path) visits every edge exactly once.

- The definition works for both directed and undirected graphs
- A cycle must have the same starting and ending nodes

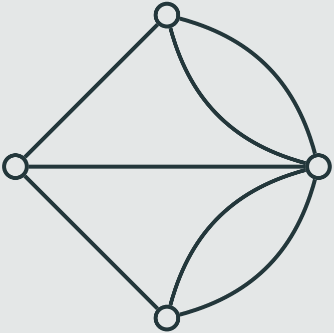
Eulerian Cycle

An Eulerian cycle (or path) visits every edge exactly once.

- The definition works for both directed and undirected graphs
- A cycle must have the same starting and ending nodes
- While in a path the starting and ending node should not necessarily be equal

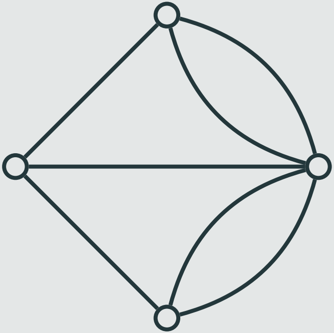
Example

Non-Eulerian graph

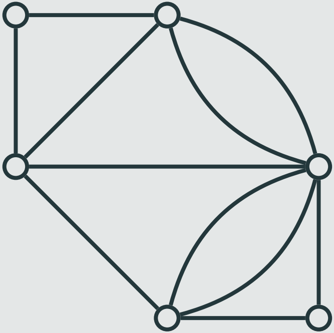


Example

Non-Eulerian graph

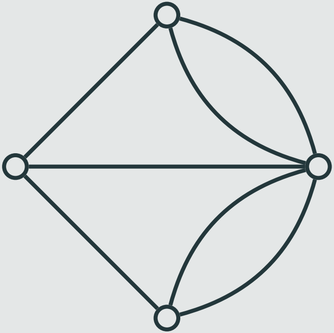


Eulerian graph

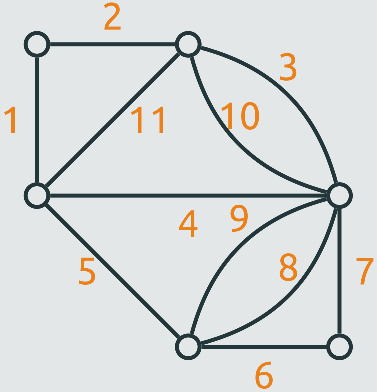


Example

Non-Eulerian graph



Eulerian graph



Criteria

Theorem

A connected *undirected* graph contains an Eulerian cycle, if and only if the degree of every node is even.

Criteria

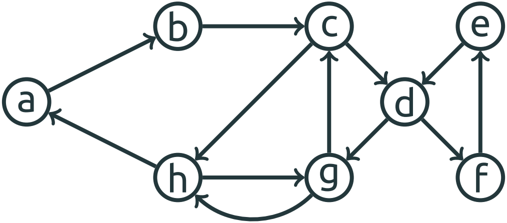
Theorem

A connected *undirected* graph contains an Eulerian cycle, if and only if the degree of every node is even.

Theorem

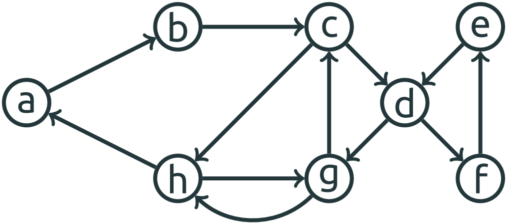
A strongly connected *directed* graph contains an Eulerian cycle, if and only if, for every node, its in-degree is equal to its out-degree.

Proof (Directed Case)



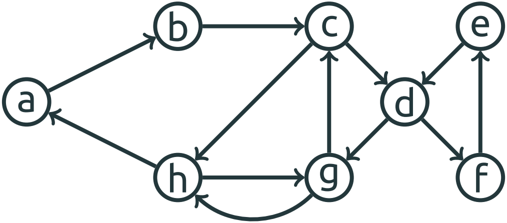
if some node is imbalanced,
there is clearly no Eulerian cycle

Proof (Directed Case)



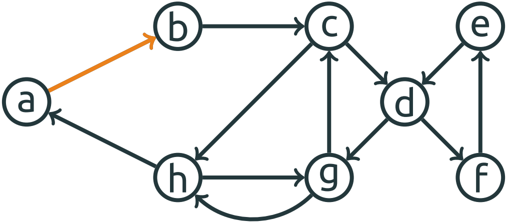
thus, assume that the graph is balanced

Proof (Directed Case)

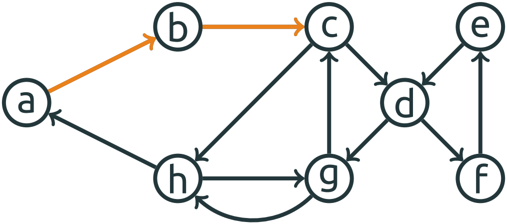


start walking from some node

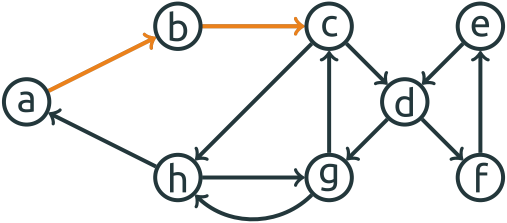
Proof (Directed Case)



Proof (Directed Case)

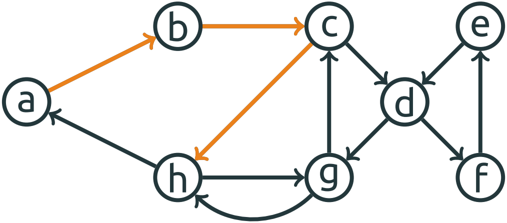


Proof (Directed Case)

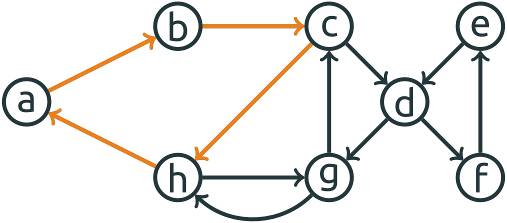


since the graph is balanced, at some point we'll return back to the starting node

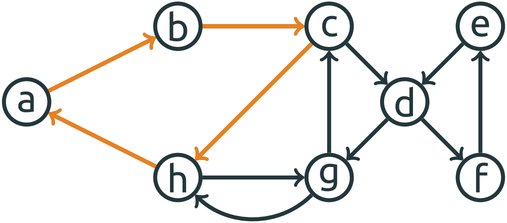
Proof (Directed Case)



Proof (Directed Case)

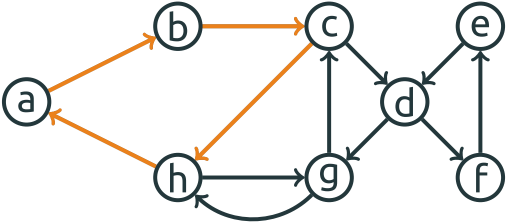


Proof (Directed Case)



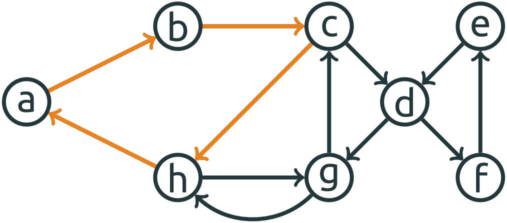
OK, what’s next? we haven’t traversed all the edges and now we are stuck

Proof (Directed Case)



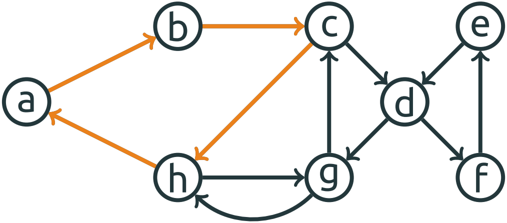
since the graph is strongly connected, there must be untraversed edges going out of a node from the current cycle

Proof (Directed Case)



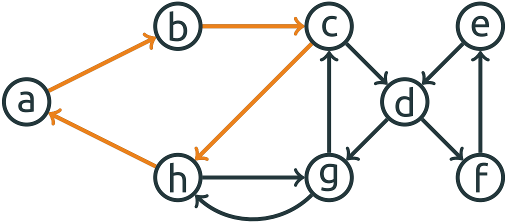
the current cycle has neither beginning nor end

Proof (Directed Case)



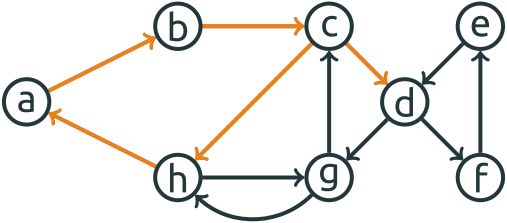
so, we may assume that its starting and ending point is the node c

Proof (Directed Case)

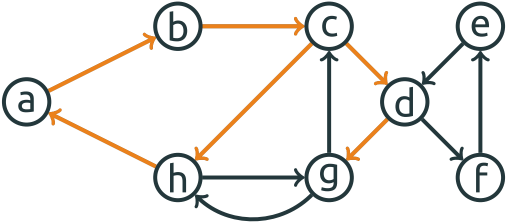


we then continue exploring the graph from the node c

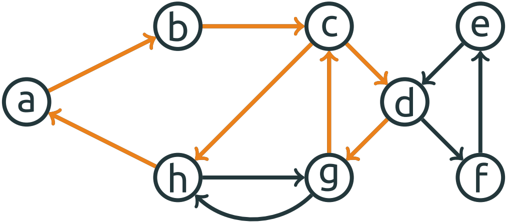
Proof (Directed Case)



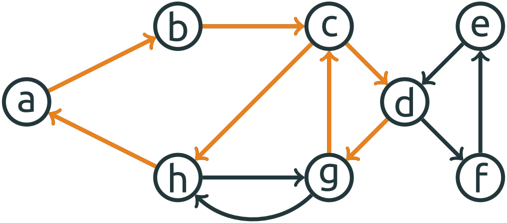
Proof (Directed Case)



Proof (Directed Case)

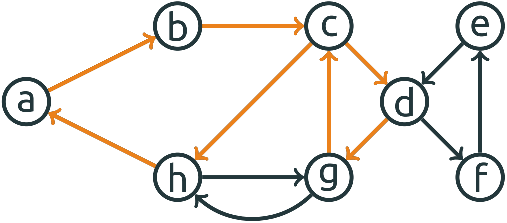


Proof (Directed Case)



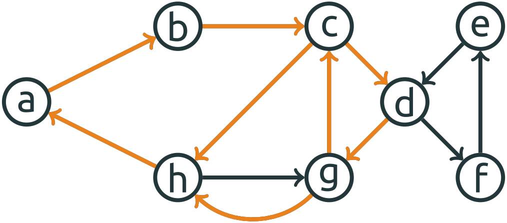
we got larger cycle

Proof (Directed Case)

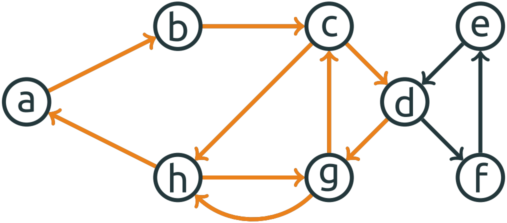


continue from g

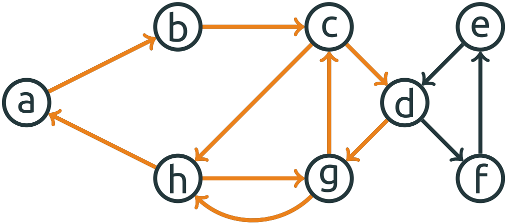
Proof (Directed Case)



Proof (Directed Case)

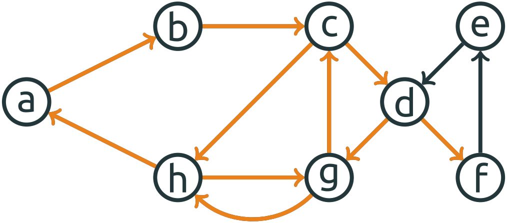


Proof (Directed Case)

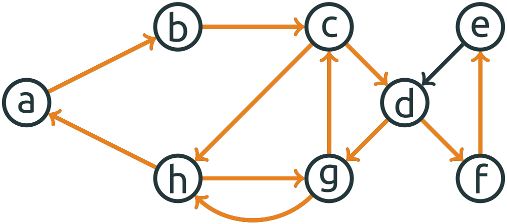


continue from d

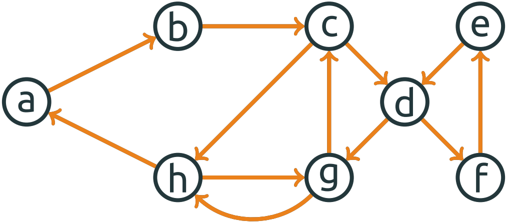
Proof (Directed Case)



Proof (Directed Case)



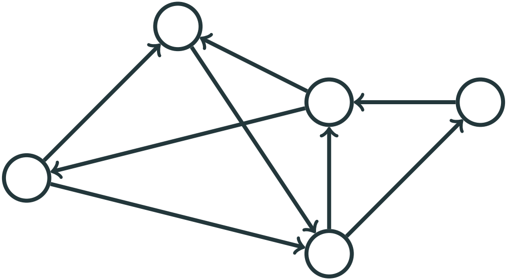
Proof (Directed Case)



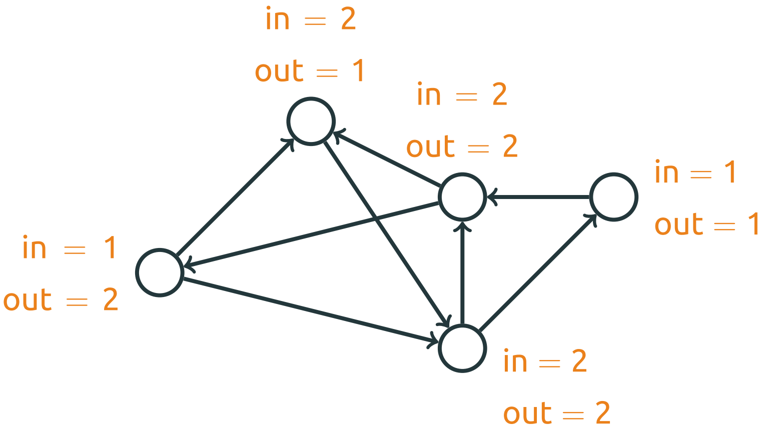
Path Instead of Cycle

- The criteria for a path is similar
- A graph is allowed to contain two imbalanced nodes: one for the starting point and one for the ending point of a path
- By adding an edge between these two nodes, one gets a graph with an Eulerian cycle

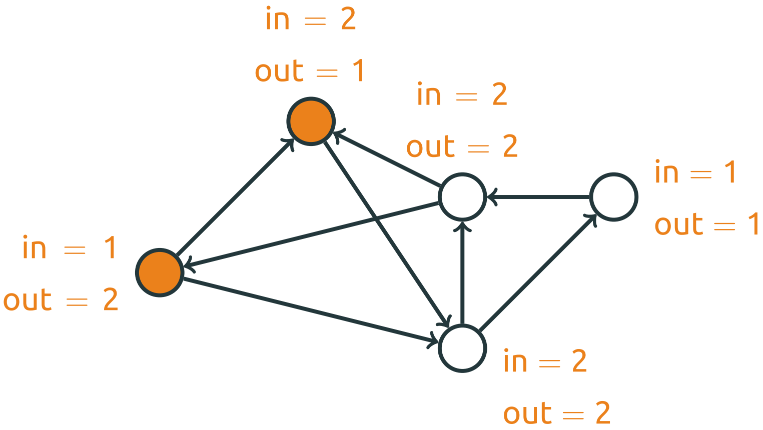
Path to Cycle



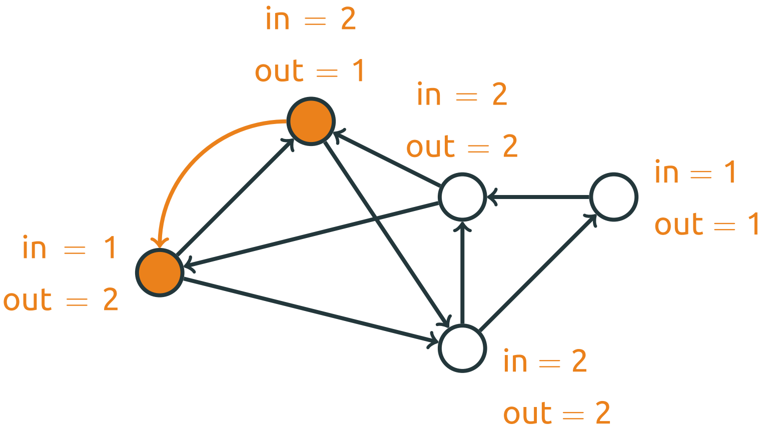
Path to Cycle



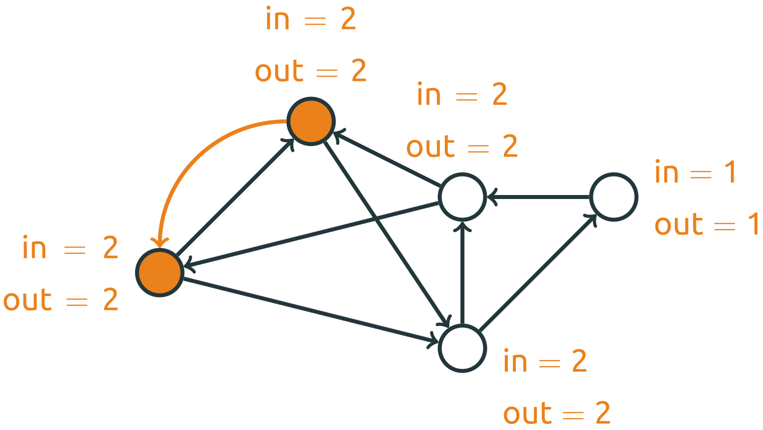
Path to Cycle



Path to Cycle



Path to Cycle



Efficient Algorithms

The proof of existence of an Eulerian cycle can be transformed into an efficient algorithm for constructing an Eulerian cycle

Outline

Eulerian Cycles

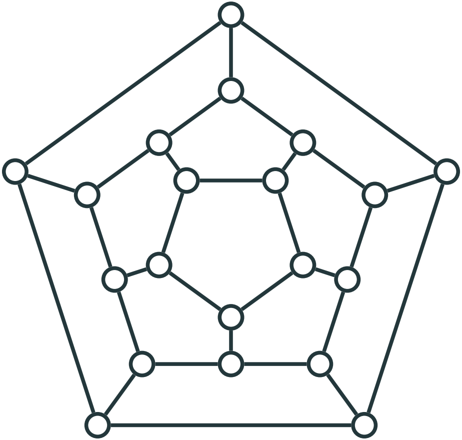
Hamiltonian Cycles

Genome Assembly

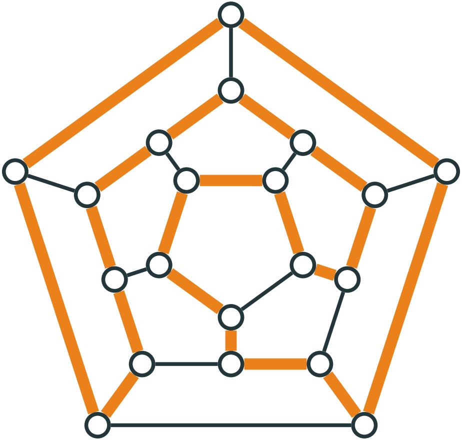
Hamiltonian Cycle

A Hamiltonian cycle visits every node of a graph exactly once.

Example



Example



Simple Criteria?

- No simple criteria is known for the Hamiltonian cycle problem

Simple Criteria?

- No simple criteria is known for the Hamiltonian cycle problem
- No polynomial time algorithm known

Simple Criteria?

- No simple criteria is known for the Hamiltonian cycle problem
- No polynomial time algorithm known
- The question whether there is a polynomial time algorithm for the Hamiltonian cycle problem is the P versus NP problem, the most important open problem in Computer Science, with a prize of $1M from the Clay Mathematics Institute (`http://www.claymath.org/millennium-problems`)

Outline

Eulerian Cycles

Hamiltonian Cycles

Genome Assembly

Toy Genome Assembly Problem

Find a string whose all substrings of length 3 are

AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC.

Toy Genome Assembly Problem

Find a string whose all substrings of length 3 are

AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC.

How is it related to cycles in graphs?..

All Substrings of Length 3

```
DISCRETE
DIS
 ISC
  SCR
   CRE
    RET
     ETE
```

All Substrings of Length 3

```
DISCRETE
DIS
 ISC
  SCR
   CRE
    RET
     ETE
```

Every two neighbor 3-substrings have
a common part, called an overlap, of length 2

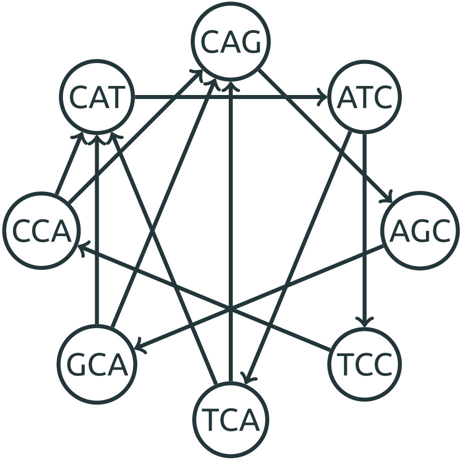
Finding a Permutation

- Goal: Find a string whose all substrings of length 3 are AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC

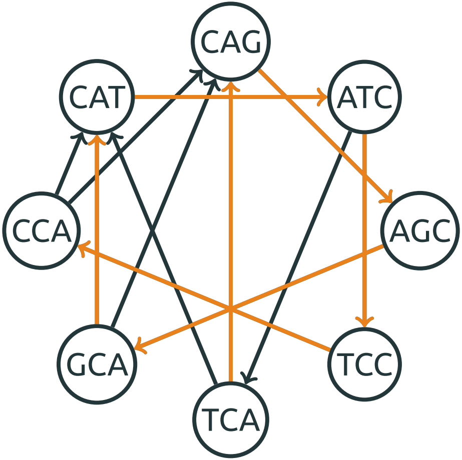
Finding a Permutation

- Goal: Find a string whose all substrings of length 3 are AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC
- Hence, we need to find an order of these 3-strings such that the overlap between any two consecutive strings is equal to 2

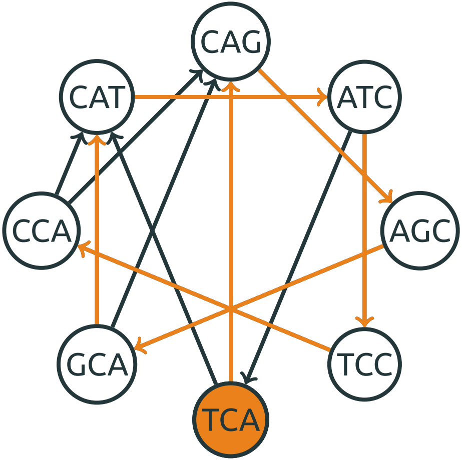
Overlap Graph



Overlap Graph

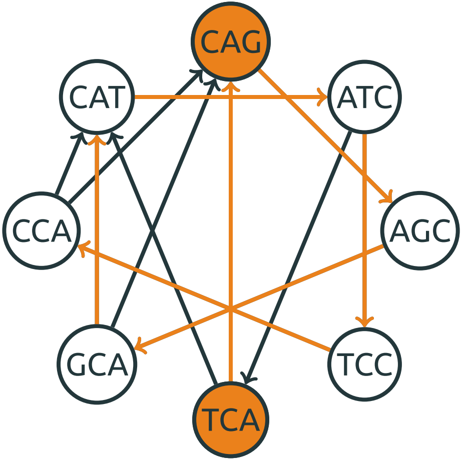


Overlap Graph



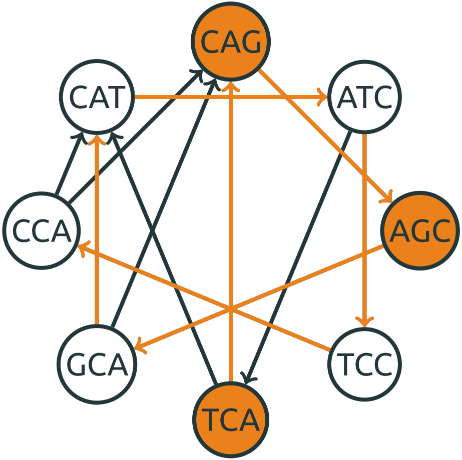
TCA

Overlap Graph



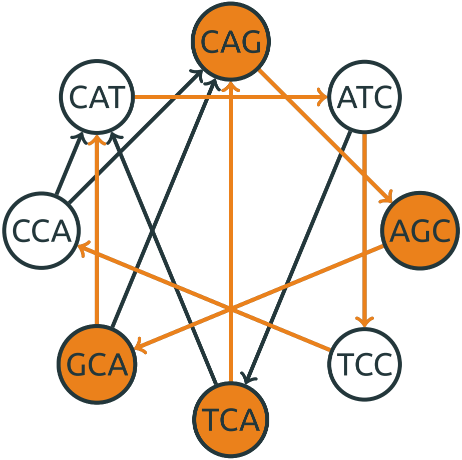
TCAG

Overlap Graph



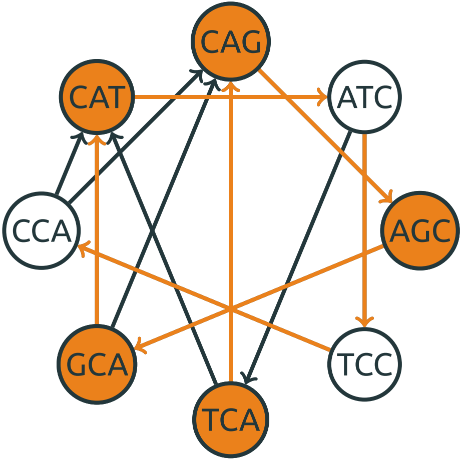
TCAGC

Overlap Graph



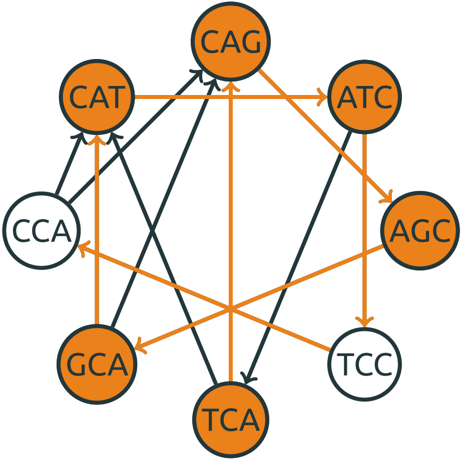
TCAGCA

Overlap Graph



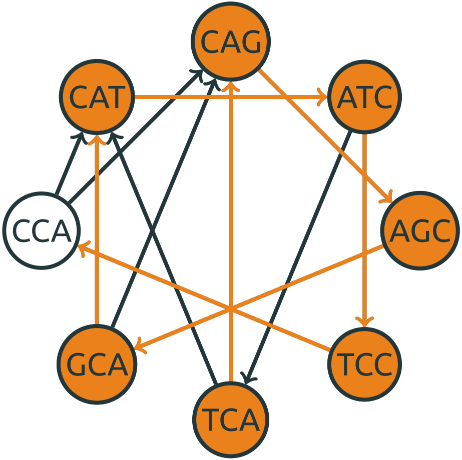
TCAGCAT

Overlap Graph



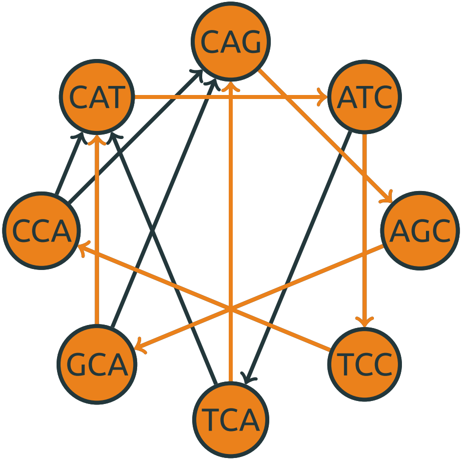
TCAGCATC

Overlap Graph



TCAGCATCC

Overlap Graph



TCAGCATCCA

Are We Done?

- OK, we’ve reduced the problem of genome assembly to the Hamiltonian cycle problem

Are We Done?

- OK, we’ve reduced the problem of genome assembly to the Hamiltonian cycle problem
- But we don’t have efficient algorithms for the Hamiltonian cycle problem!

Are We Done?

- OK, we’ve reduced the problem of genome assembly to the Hamiltonian cycle problem
- But we don’t have efficient algorithms for the Hamiltonian cycle problem!
- The approach is useless for the case when there are thousands or millions of input strings

Speculating

- In the overlap graph, each input string is represented by a node

Speculating

- In the overlap graph, each input string is represented by a node
- Let’s instead try to represent each string by an edge (De Bruijn; Pevzner, Tang, Waterman)

Speculating

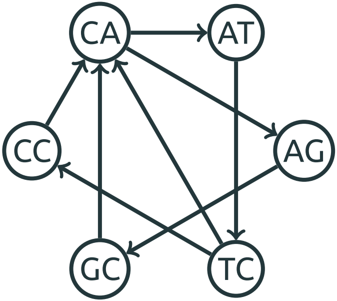
- In the overlap graph, each input string is represented by a node
- Let's instead try to represent each string by an edge (De Bruijn; Pevzner, Tang, Waterman)
- E.g., represent the string CAT as an edge CA $\rightarrow$ AT

Speculating

- In the overlap graph, each input string is represented by a node
- Let’s instead try to represent each string by an edge (De Bruijn; Pevzner, Tang, Waterman)
- E.g., represent the string CAT as an edge CA $\rightarrow$ AT
- Used in state-of-the-art genome assemblers

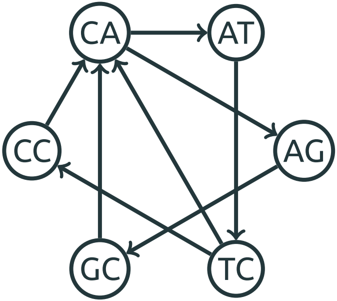
De Bruijn Graph

AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



De Bruijn Graph

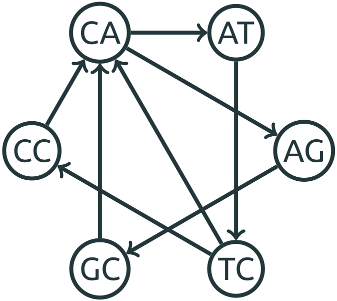
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



now, we need to find an order of edges

De Bruijn Graph

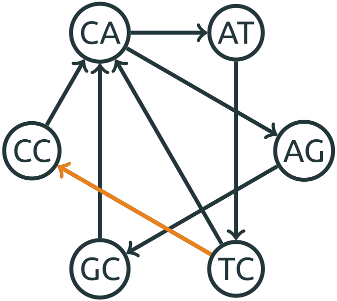
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



that is, an Eulerian cycle

De Bruijn Graph

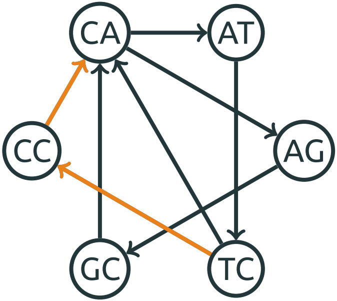
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCC

De Bruijn Graph

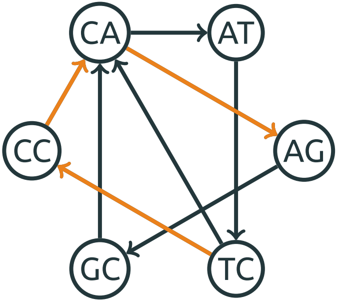
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCA

De Bruijn Graph

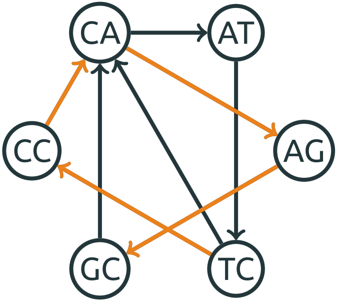
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCAG

De Bruijn Graph

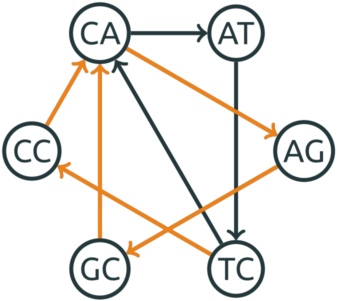
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCAGC

De Bruijn Graph

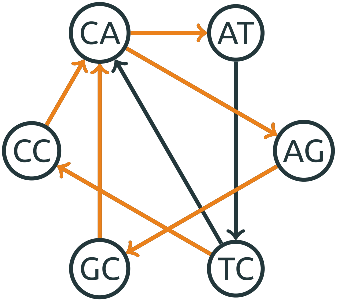
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCAGCA

De Bruijn Graph

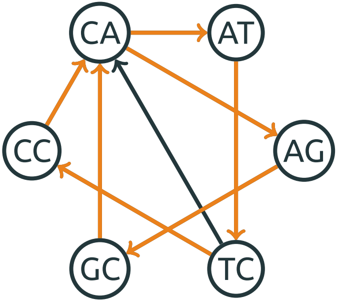
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCAGCAT

De Bruijn Graph

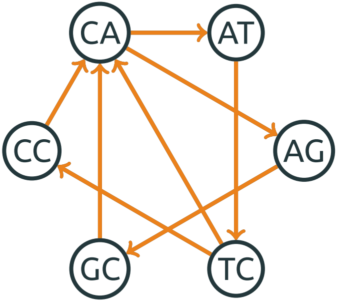
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCAGCATC

De Bruijn Graph

AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCAGCATCA

Summary

- Eulerian cycle visits every edge exactly once

Summary

- Eulerian cycle visits every edge exactly once
- Hamiltonian cycle visits every node exactly once

Summary

- Eulerian cycle visits every edge exactly once
- Hamiltonian cycle visits every node exactly once
- Look similar to each other, but differ drastically from the computational point of view

Summary

- Eulerian cycle visits every edge exactly once
- Hamiltonian cycle visits every node exactly once
- Look similar to each other, but differ drastically from the computational point of view
- Genome assembly: the right problem formulation makes all the difference!