Algorithms on Graphs

Umang Bhaskar & Juhi Chaudhary

Day 2, session 1: Walks in graphs

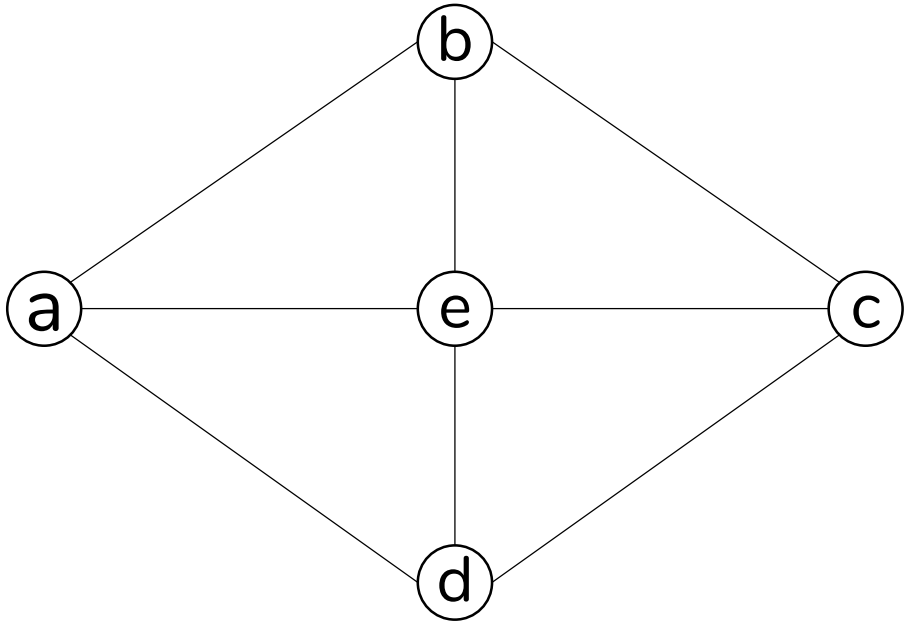
STCS Vigyan Vidushi 2024

Basic Algorithms

Paths vs walks:

Path: each vertex appears at most once

Walk: a vertex can appear multiple times

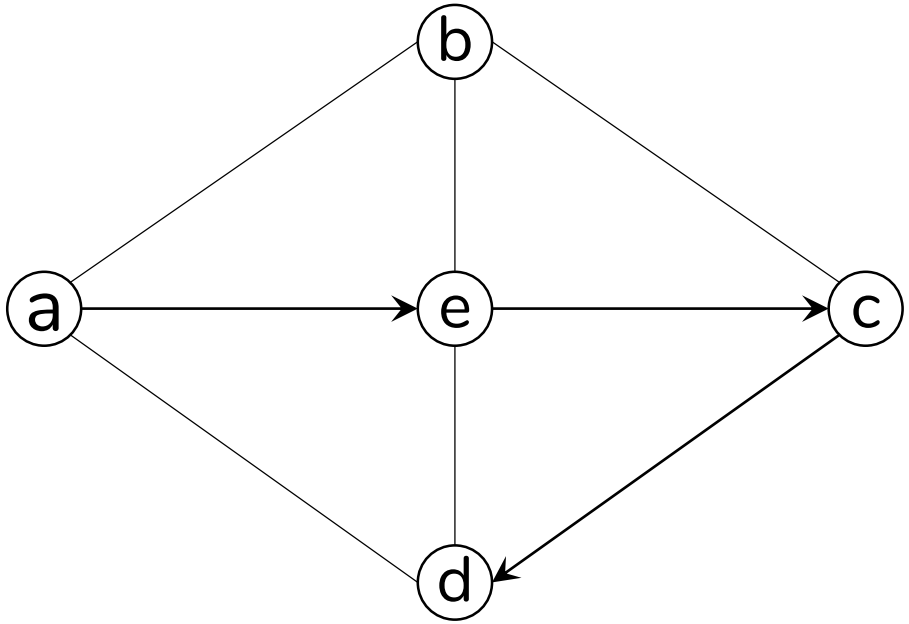


Basic Algorithms

Paths vs walks:

Path: each vertex appears at most once

Walk: a vertex can appear multiple times

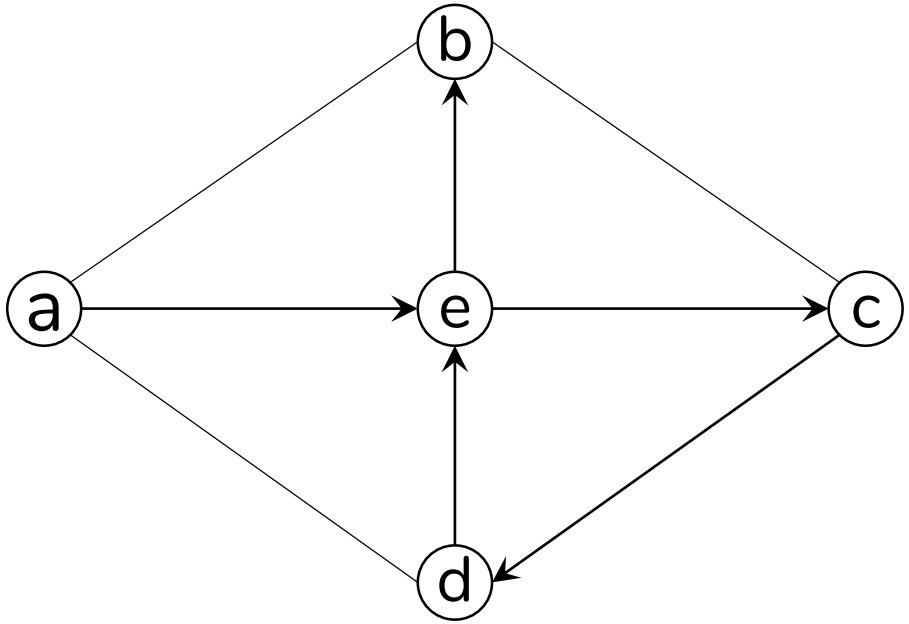


Basic Algorithms

Paths vs walks:

Path: each vertex appears at most once

Walk: a vertex can appear multiple times

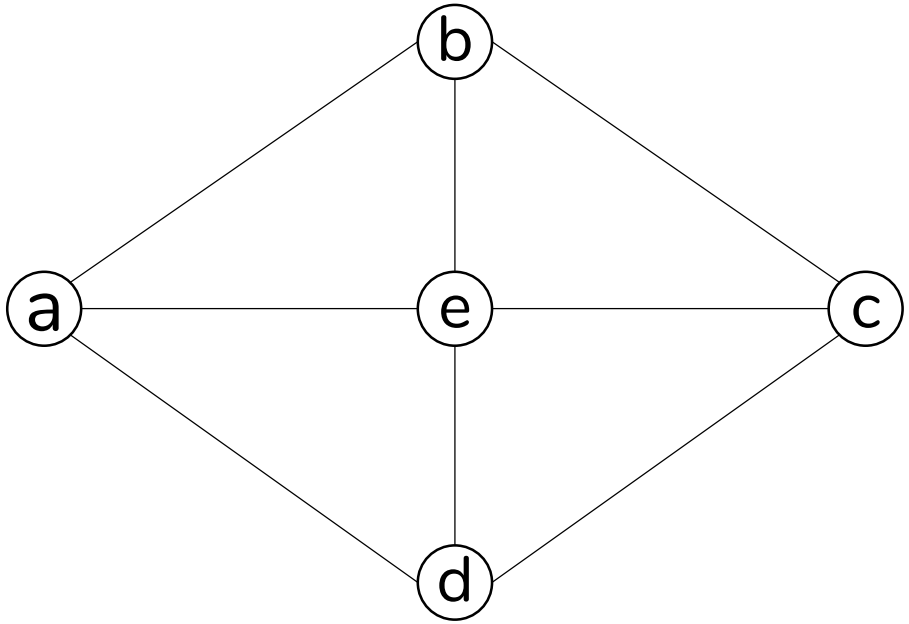


Basic Algorithms

Cycles vs circuits:

Cycle: each vertex appears at most once (except for first and last)

Circuit: a vertex can appear multiple times

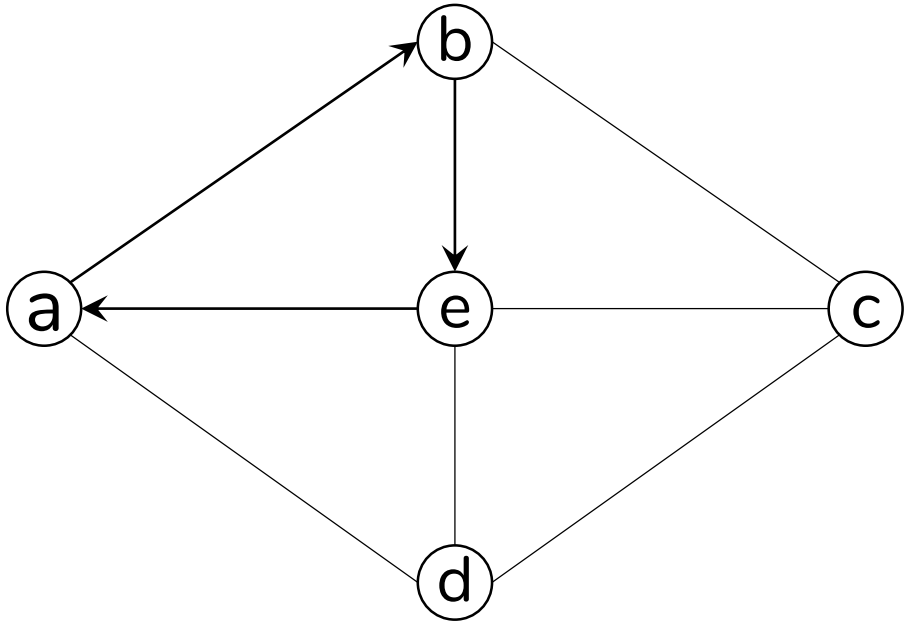


Basic Algorithms

Cycles vs circuits:

Cycle: each vertex appears at most once (except for first and last)

Circuit: a vertex can appear multiple times

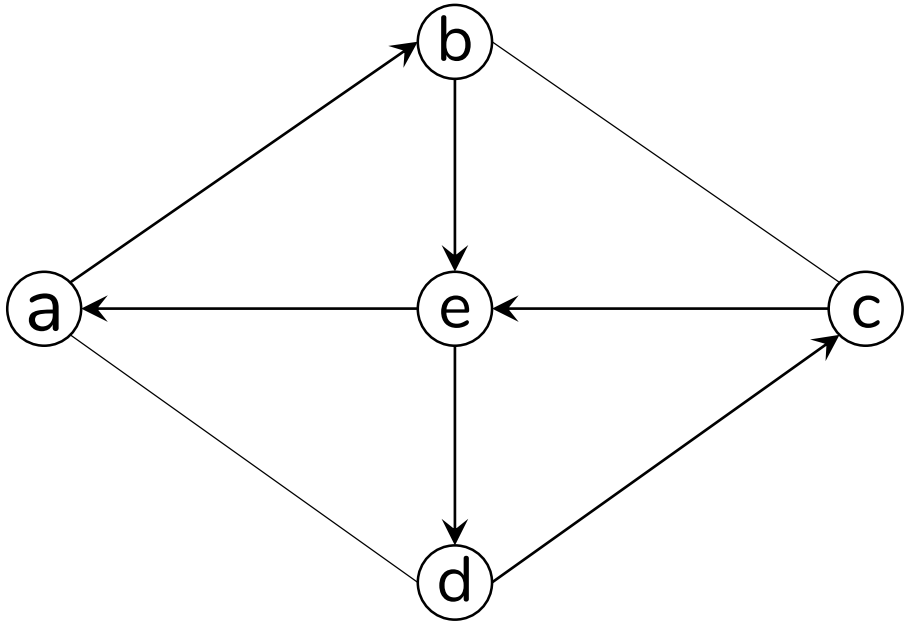


Basic Algorithms

Cycles vs circuits:

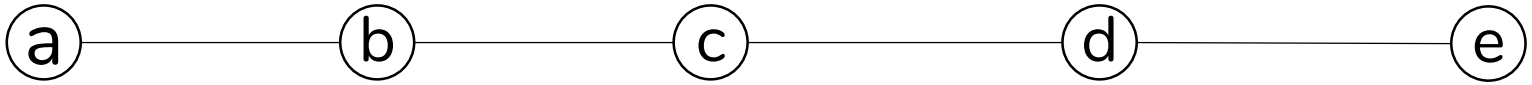
Cycle: each vertex appears at most once (except for first and last)

Circuit: a vertex can appear multiple times



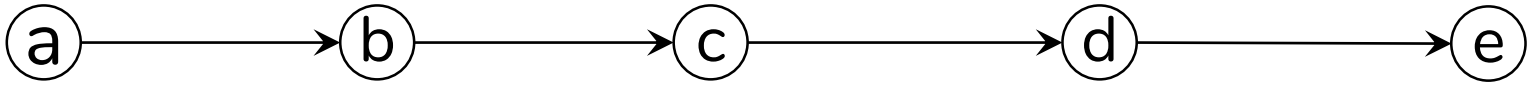
4. find a walk that uses each edge exactly once

Eulerian walk: sequence of adjacent edges,
such that each edge appears exactly once.



4. find a walk that uses each edge exactly once

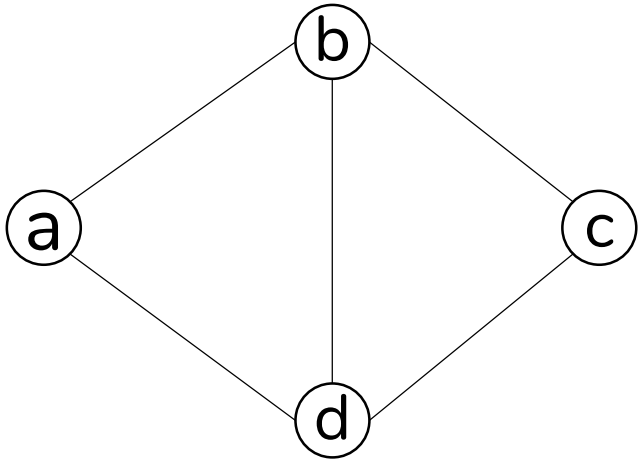
Eulerian walk: sequence of adjacent edges,
such that each edge appears exactly once.

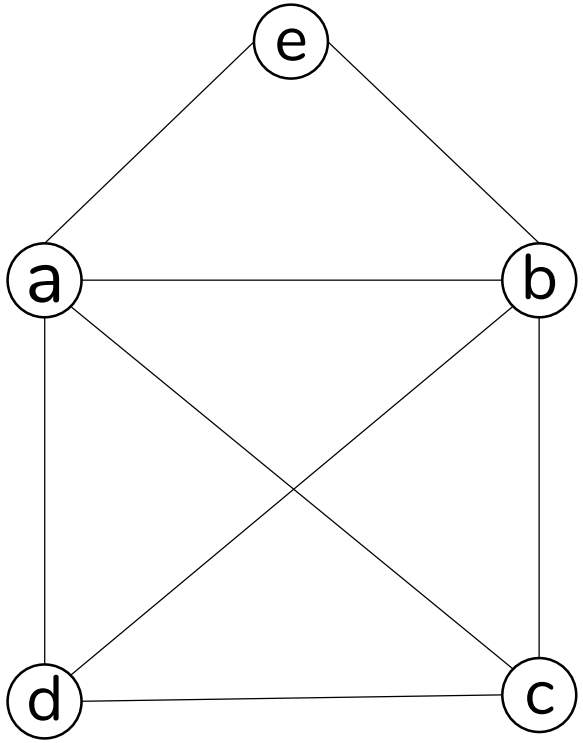


Basic Algorithms

4. find a walk that uses each edge exactly once

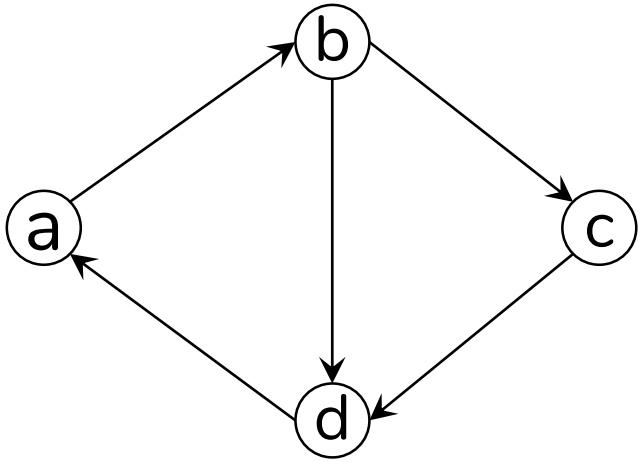
Eulerian walk: sequence of adjacent edges, such that each edge appears exactly once.



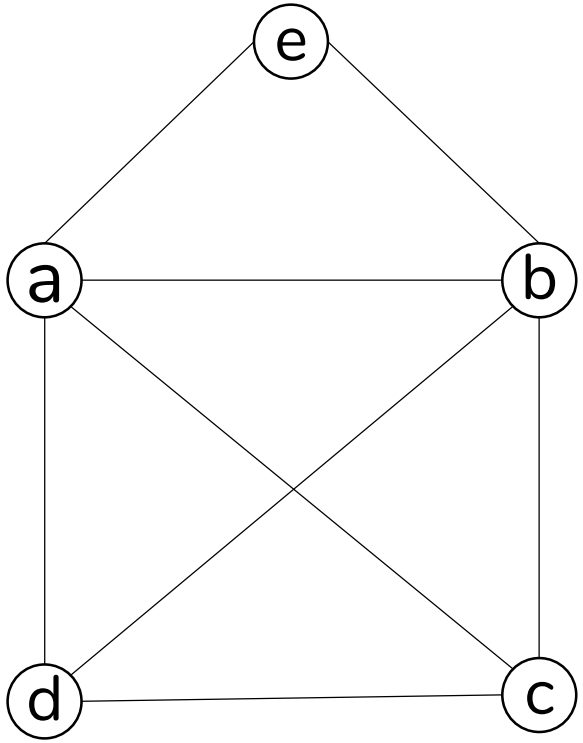


4. find a walk that uses each edge exactly once

Eulerian walk: sequence of adjacent edges, such that each edge appears exactly once.

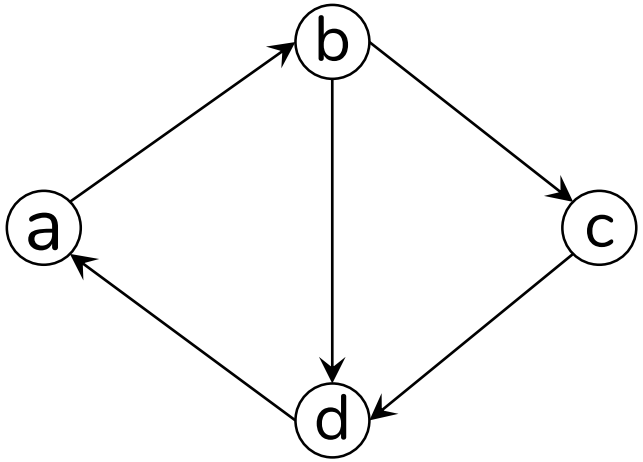


Start from b,
end at d

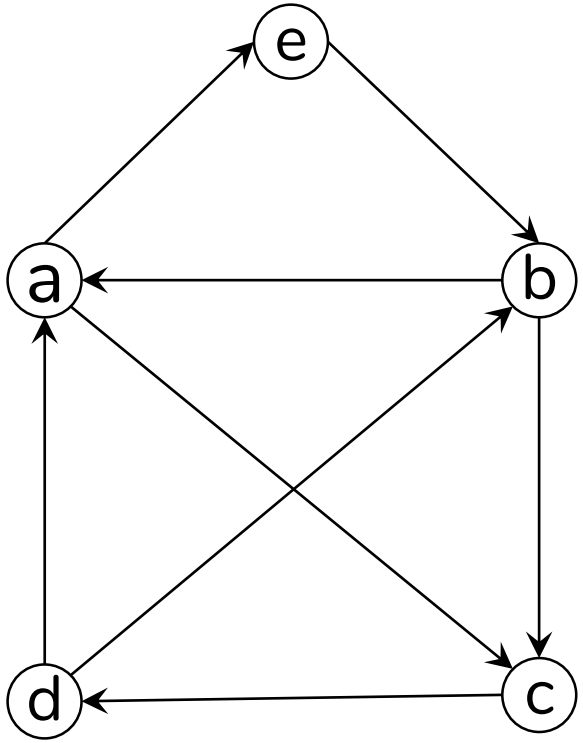


4. find a walk that uses each edge exactly once

Eulerian walk: sequence of adjacent edges, such that each edge appears exactly once.



Start from b,
end at d

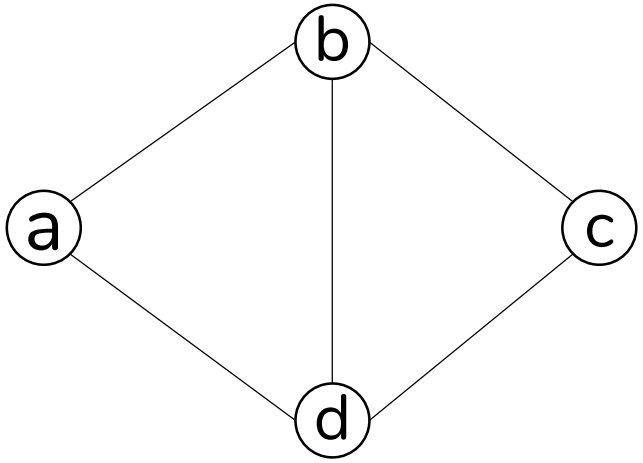


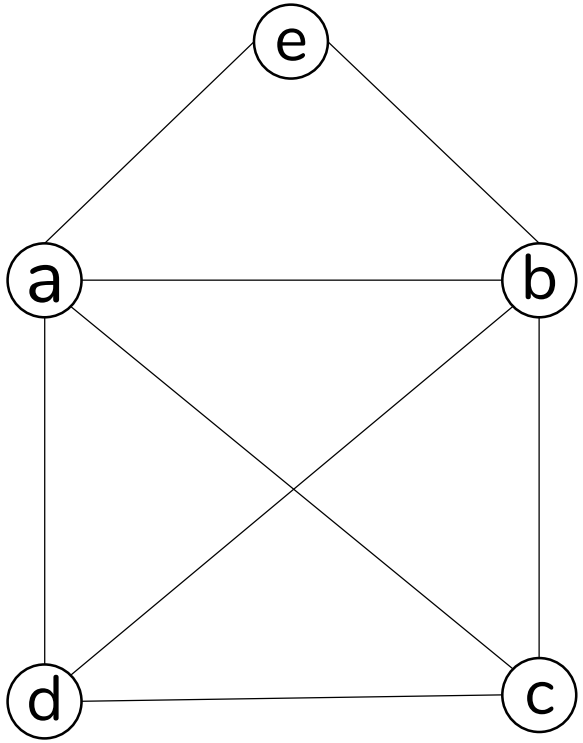
Start from d,
end at c

Basic Algorithms

4. find a circuit that uses each edge exactly once

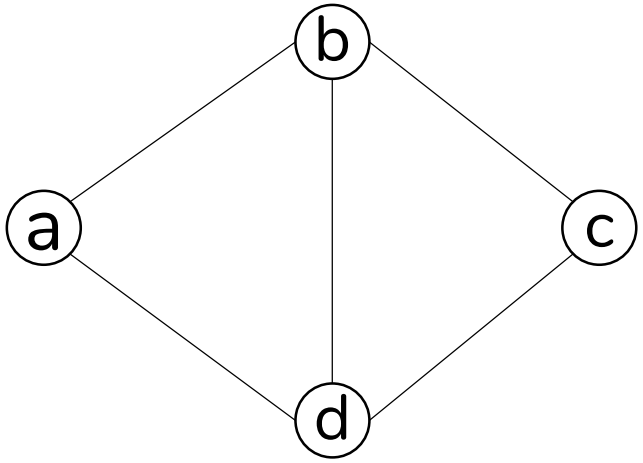
Eulerian circuit: sequence of adjacent edges, such that each edge appears exactly once, and begins and ends at the same vtx.

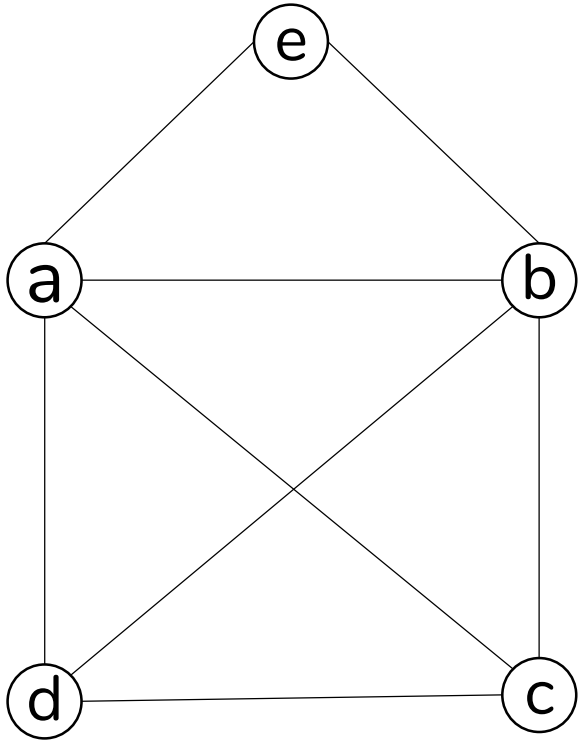




4. find a circuit that uses each edge exactly once

Eulerian circuit: sequence of adjacent edges, such that each edge appears exactly once, and begins and ends at the same vtx.



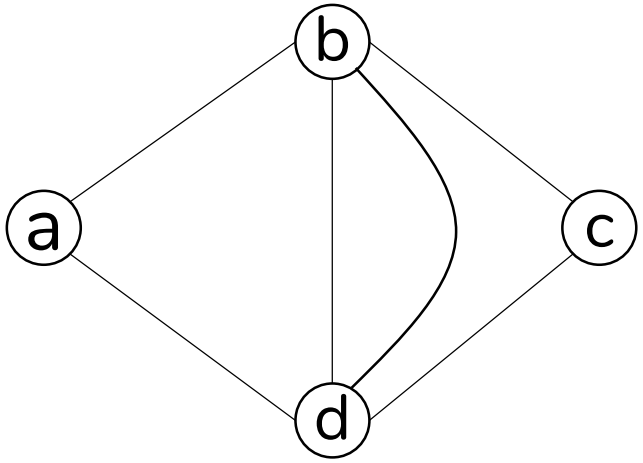


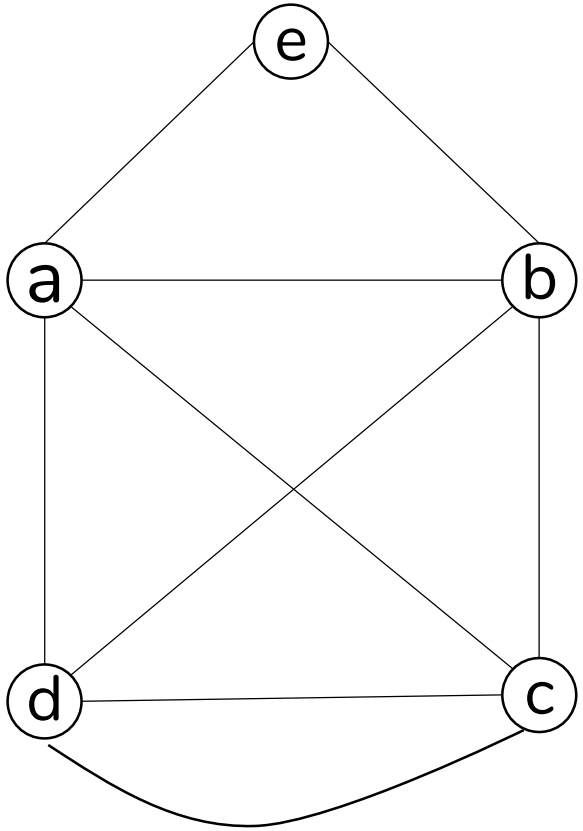
no Eulerian circuit

Basic Algorithms

4. find a circuit that uses each edge exactly once

Eulerian circuit: sequence of adjacent edges, such that each edge appears exactly once, and begins and ends at the same vtx.

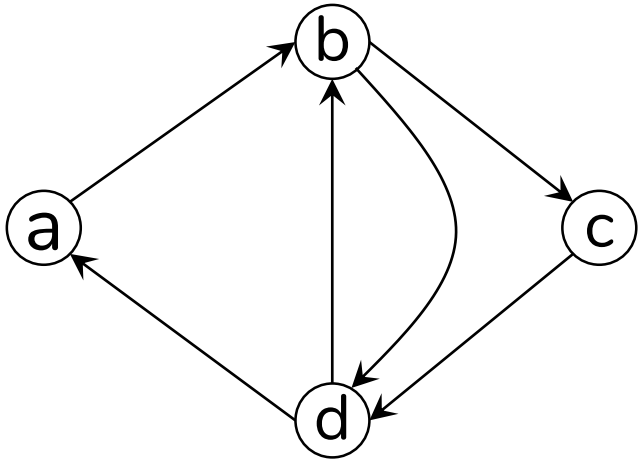


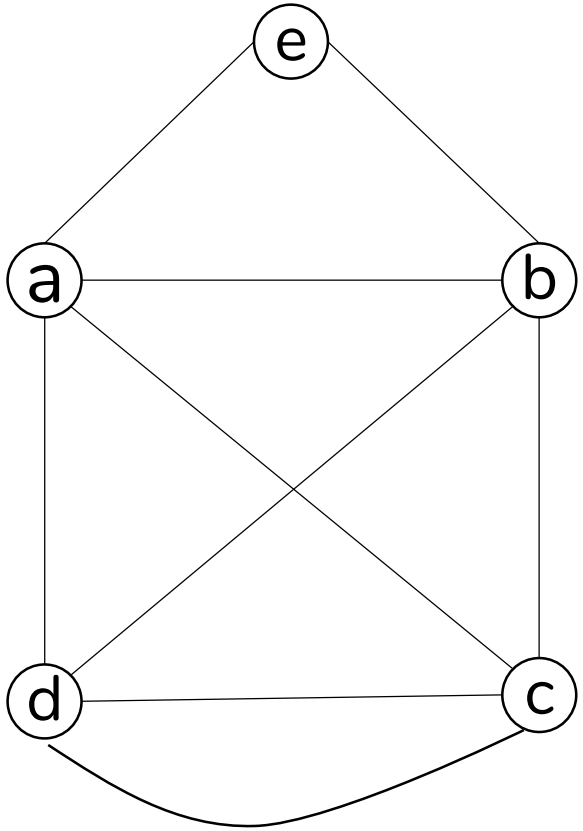


Basic Algorithms

4. find a circuit that uses each edge exactly once

Eulerian circuit: sequence of adjacent edges, such that each edge appears exactly once, and begins and ends at the same vtx.

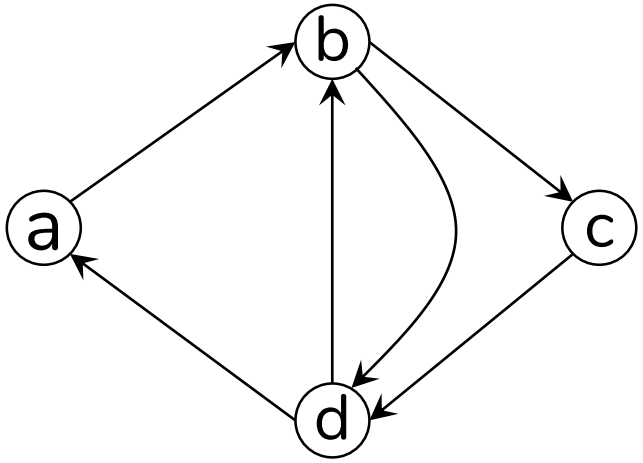


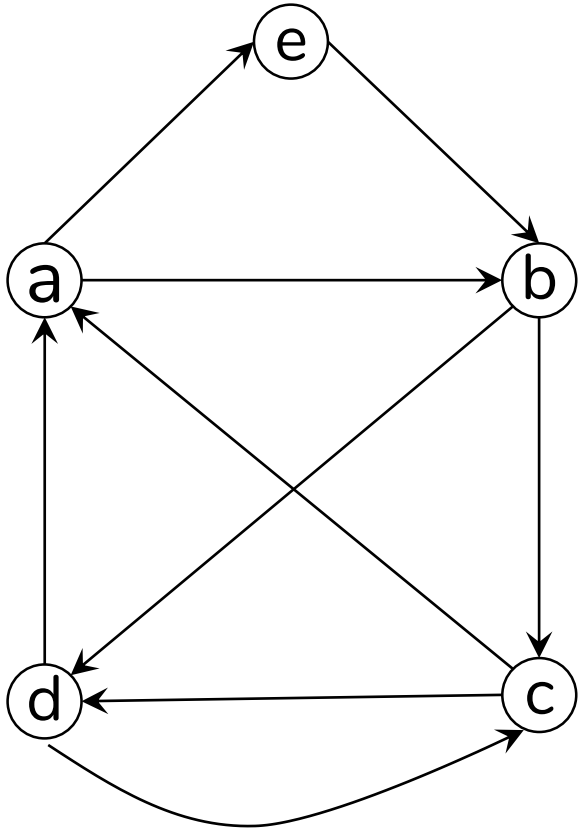


Basic Algorithms

4. find a circuit that uses each edge exactly once

Eulerian circuit: sequence of adjacent edges, such that each edge appears exactly once, and begins and ends at the same vtx.



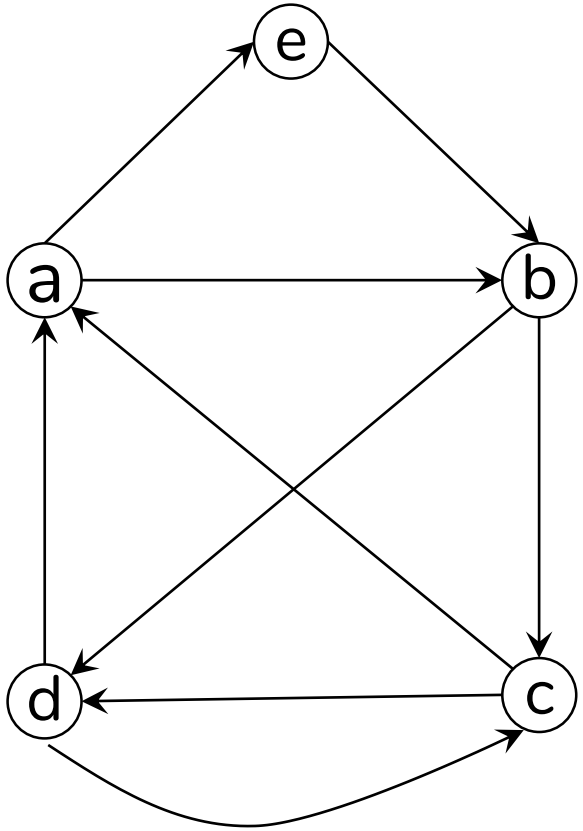


Basic Algorithms

4. find a circuit that uses each edge exactly once

Eulerian circuit: sequence of adjacent edges, such that each edge appears exactly once, and begins and ends at the same vtx.

Theorem: Graph G = (V,E) has an Eulerian ckt iff every vertex has even degree



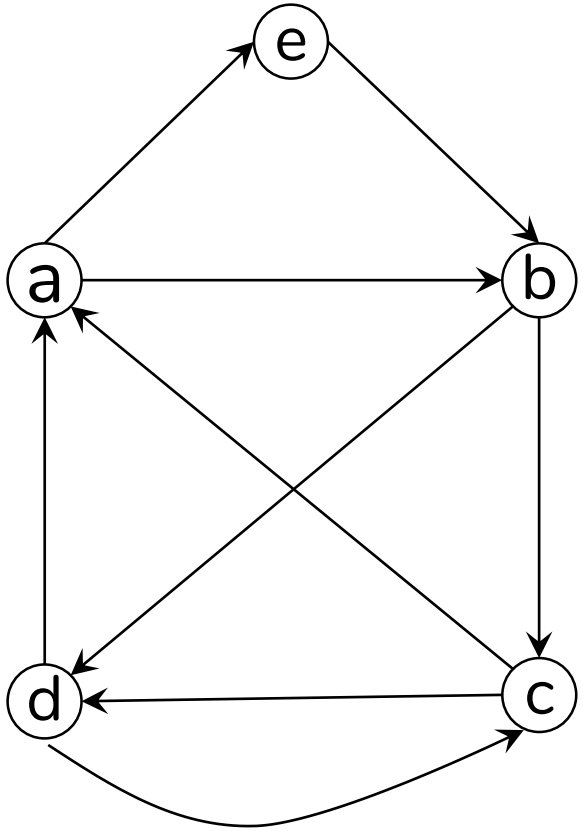
Basic Algorithms

4. find a circuit that uses each edge exactly once

Eulerian circuit: sequence of adjacent edges, such that each edge appears exactly once, and begins and ends at the same vtx.

Theorem: Graph G = (V,E) has an Eulerian ckt iff every vertex has even degree

Eulerian ckt ⇒ even degree: easy

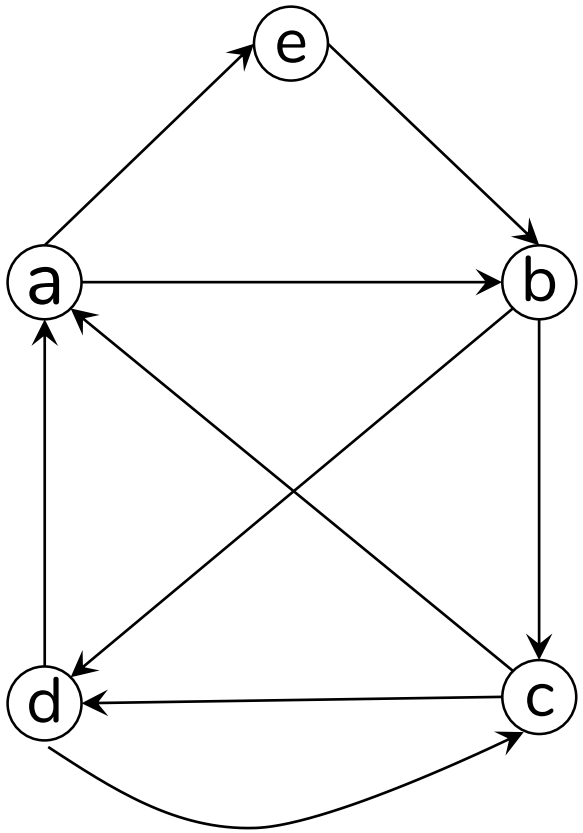


Basic Algorithms

4. find a circuit that uses each edge exactly once

Eulerian circuit: sequence of adjacent edges, such that each edge appears exactly once, and begins and ends at the same vtx.

Theorem: Graph G = (V,E) has an Eulerian ckt iff every vertex has even degree



Eulerian ckt ⇒ even degree: easy

Even degree ⇒ Eulerian ckt: via algorithms

Basic Algorithms

4. find a circuit that uses each edge exactly once

Hierholzer's Algorithm:

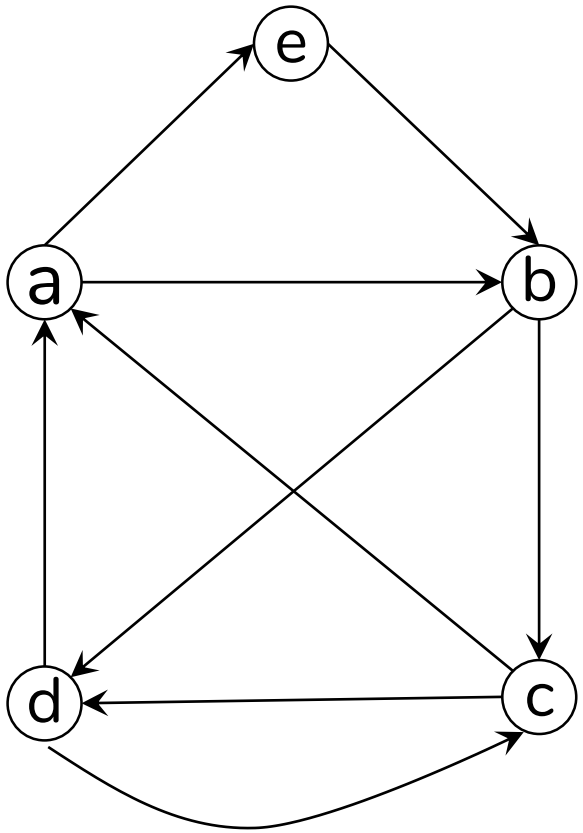
from any vertex (say e),

keep following unused edges,
until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices

e



Basic Algorithms

4. find a circuit that uses each edge exactly once

Hierholzer's Algorithm:

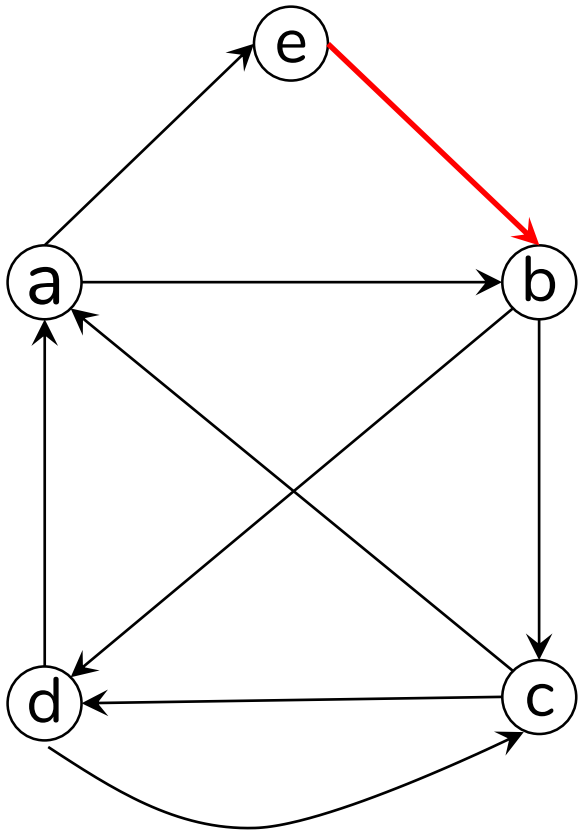
from any vertex (say e),

keep following unused edges,
until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices

e



Basic Algorithms

4. find a circuit that uses each edge exactly once

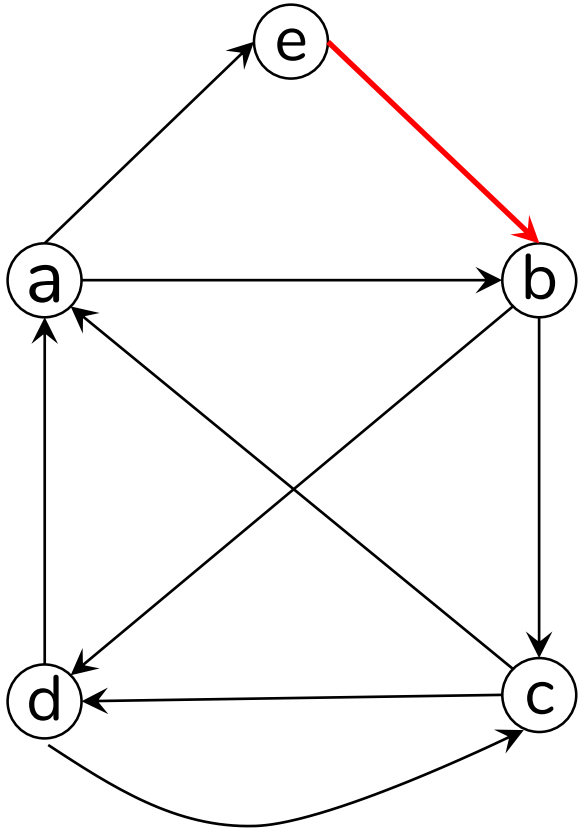
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges,
until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b

Basic Algorithms

4. find a circuit that uses each edge exactly once

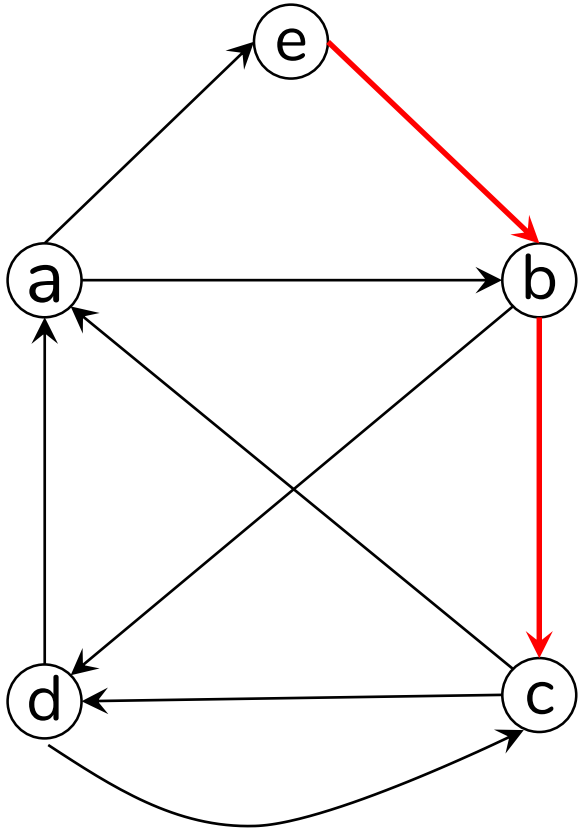
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges,
until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b

Basic Algorithms

4. find a circuit that uses each edge exactly once

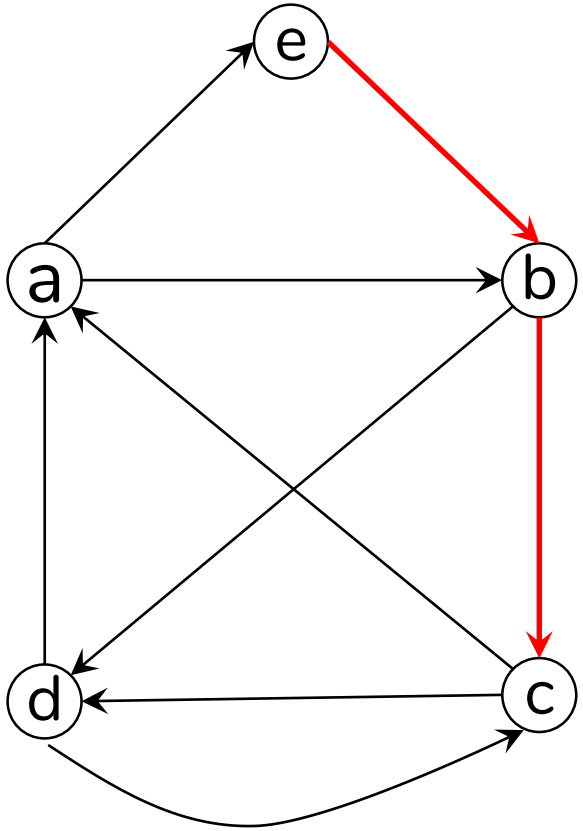
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges,
until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b
c

Basic Algorithms

4. find a circuit that uses each edge exactly once

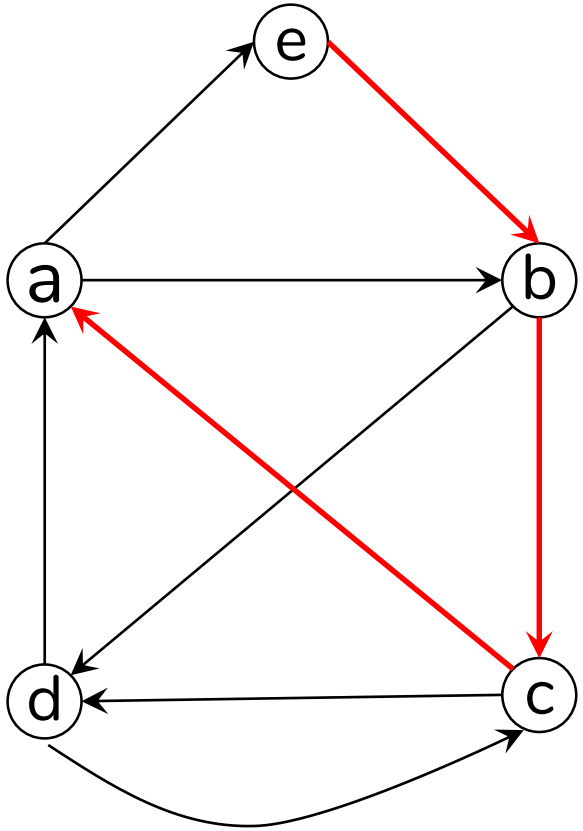
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges,
until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b
c

Basic Algorithms

4. find a circuit that uses each edge exactly once

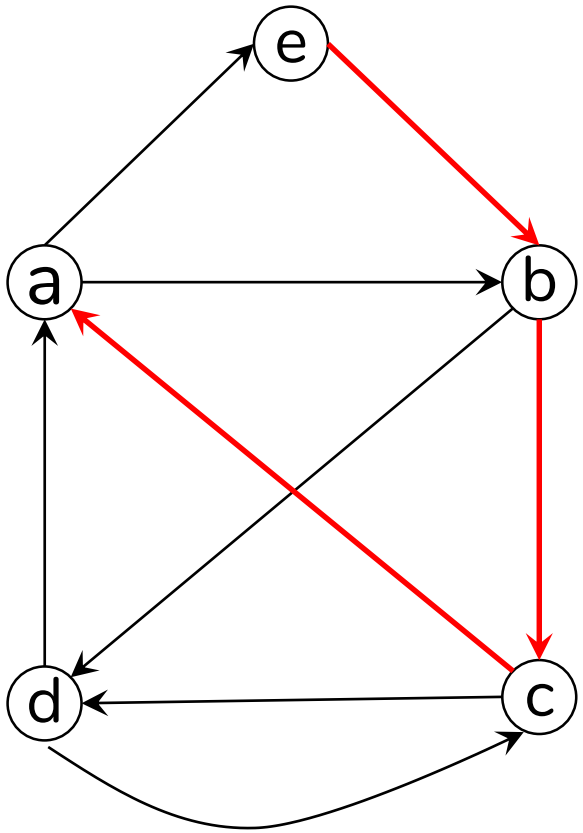
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges,
until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b
c
a

Basic Algorithms

4. find a circuit that uses each edge exactly once

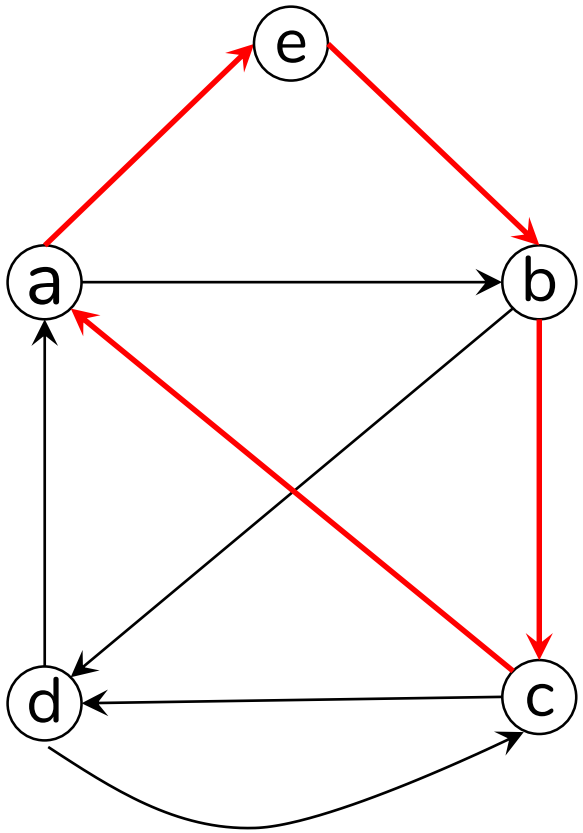
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges, until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b
c
a

Basic Algorithms

4. find a circuit that uses each edge exactly once

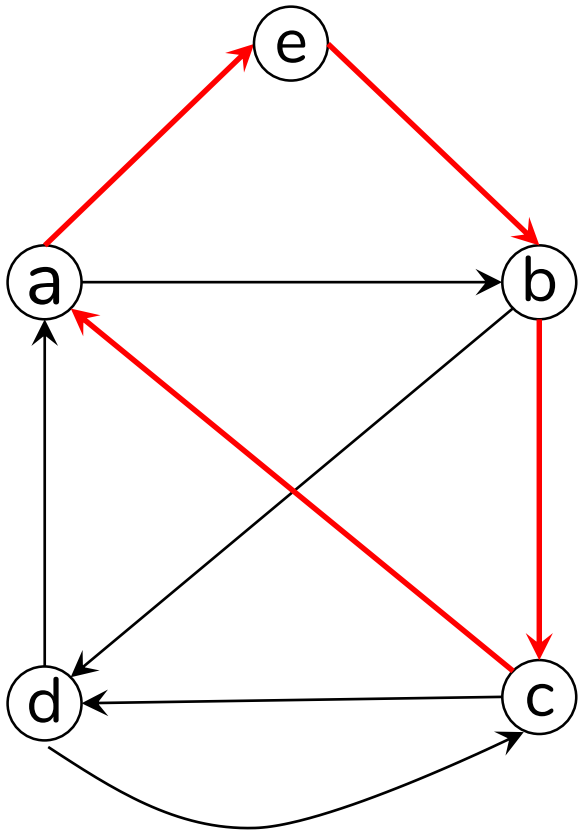
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges, until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b
c
a
e

Basic Algorithms

4. find a circuit that uses each edge exactly once

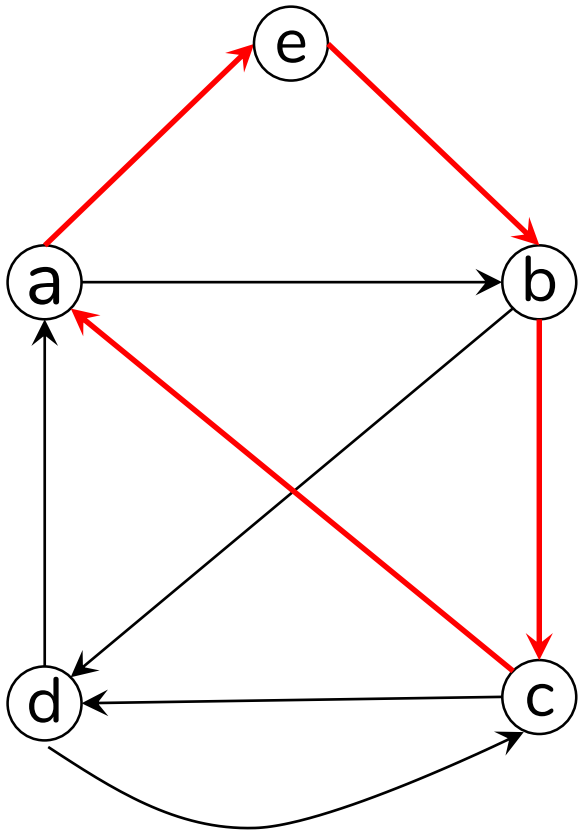
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges, until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b
c
a
e

Basic Algorithms

4. find a circuit that uses each edge exactly once

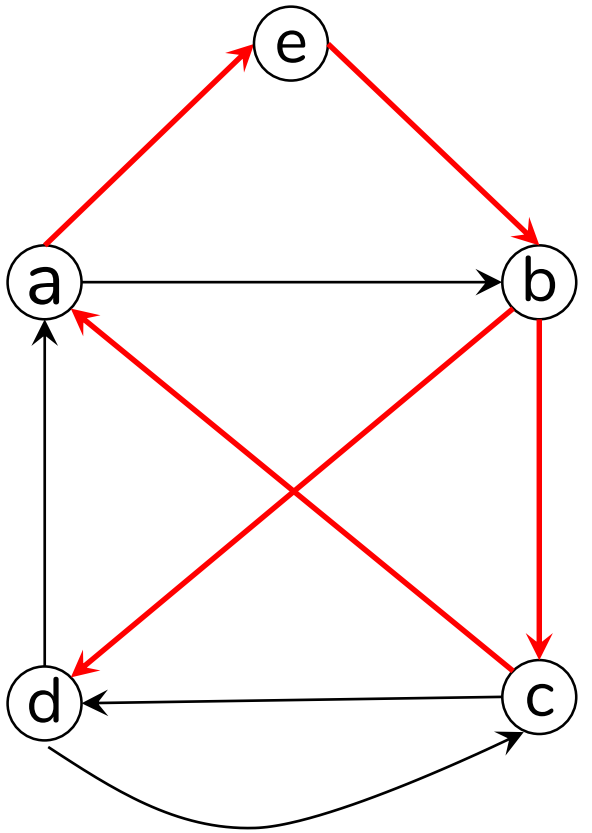
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges, until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b
d
c
a
e

Basic Algorithms

4. find a circuit that uses each edge exactly once

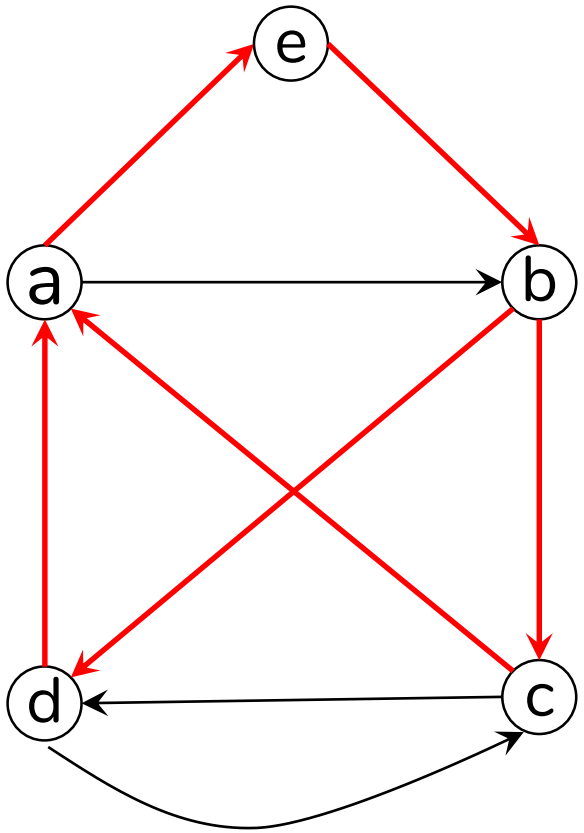
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges, until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b
d
a
c
a
d
e

Basic Algorithms

4. find a circuit that uses each edge exactly once

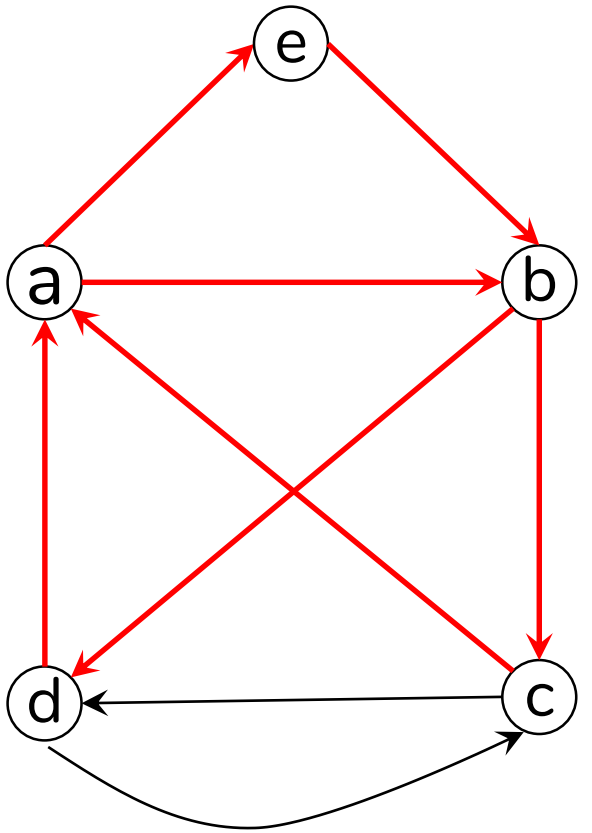
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges, until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b
d
a
b
c
a
d
e

Basic Algorithms

4. find a circuit that uses each edge exactly once

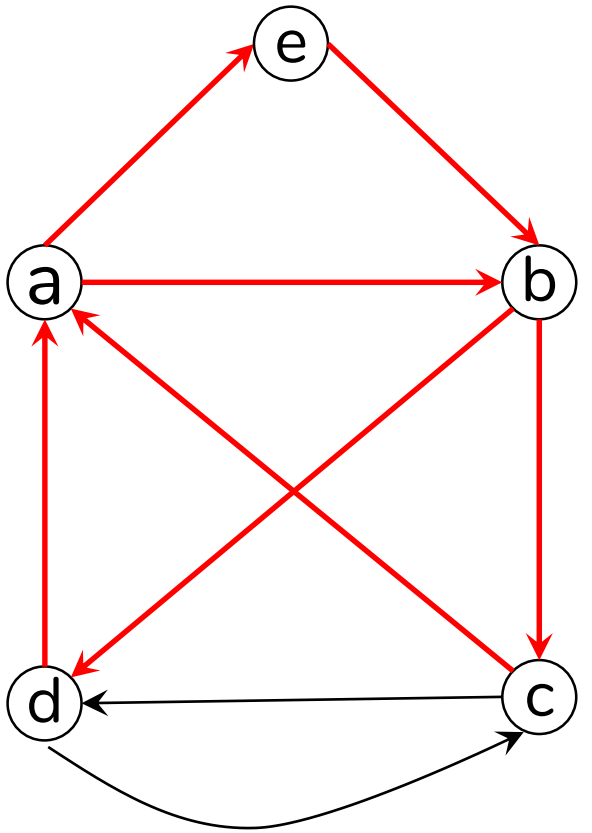
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges,
until back to e

store vertices on this trail that have
other edges

repeat procedure for these vertices



e
b
d
a
b
c
a
d
e

Basic Algorithms

4. find a circuit that uses each edge exactly once

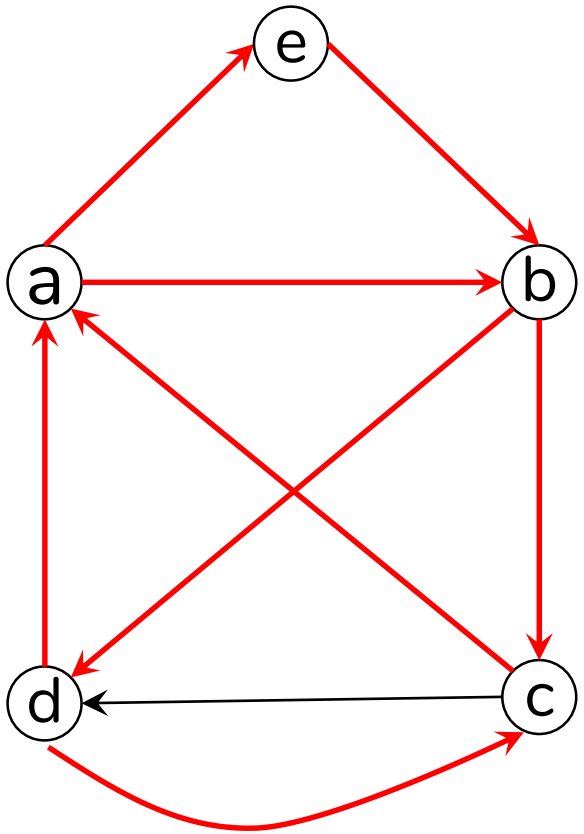
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges, until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



e
b
d
c
a
b
c
a
d
e

Basic Algorithms

4. find a circuit that uses each edge exactly once

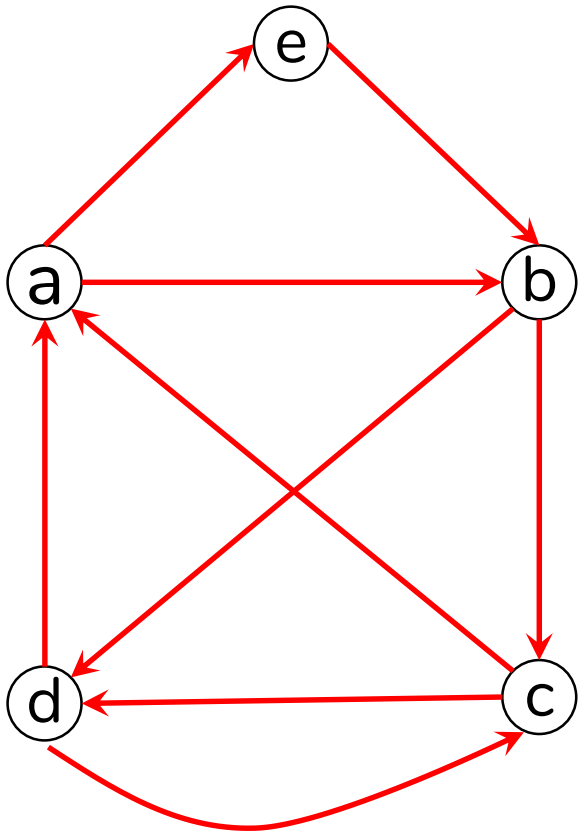
Hierholzer's Algorithm:

from any vertex (say e),

keep following unused edges, until back to e

store vertices on this trail that have other edges

repeat procedure for these vertices



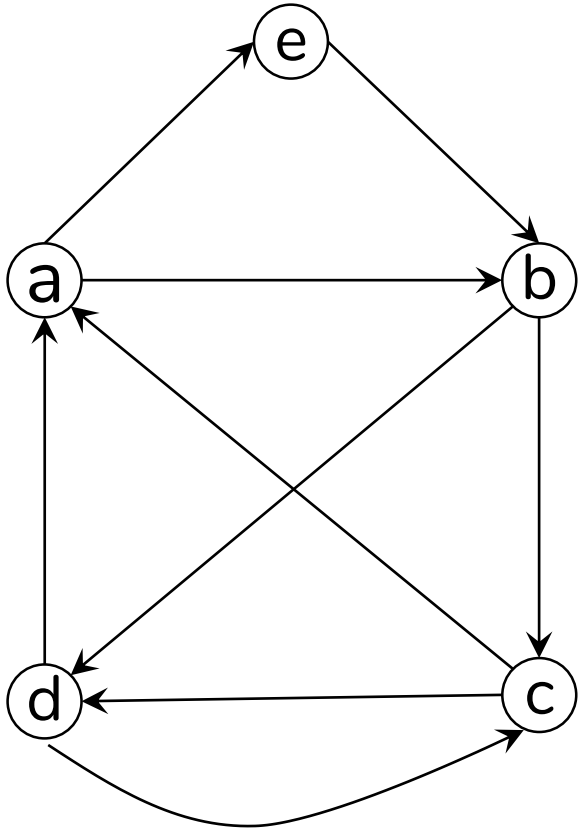
e
b
d
c
d
a
b
c
a
d
e

Basic Algorithms

4. find a circuit that uses each edge exactly once

Eulerian circuit: sequence of adjacent edges, such that each edge appears exactly once, and begins and ends at the same vtx.

Theorem: Graph G = (V,E) has an Eulerian ckt iff every vertex has even degree



The Card Trick

The Card Trick

16 cards were in a fixed sequence:

1♠, 4♣, 8♣, 2♠, 3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥

The Card Trick

16 cards were in a fixed sequence:

1♠, 4♣, 8♣, 2♠, 3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥

One (or more) straight cut only permutes the cards cyclically:

3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥, 1♠, 4♣, 8♣, 2♠

The Card Trick

16 cards were in a fixed sequence:

1♠, 4♣, 8♣, 2♠, 3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥

One (or more) straight cut only permutes the cards cyclically:

3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥, 1♠, 4♣, 8♣, 2♠

A sequence of four cards is picked from the top:

3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥, 1♠, 4♣, 8♣, 2♠

But how would I know which card is on top?

The Card Trick

Let’s look at the sequence again:

1♠, 4♣, 8♣, 2♠, 3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥

The Card Trick

Let's look at the sequence again:

1♠, 4♣, 8♣, 2♠, 3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥

B B B B R B B R R B R B R R R R

The Card Trick

Let's look at the sequence again:

1♠, 4♣, 8♣, 2♠, 3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥

B B B B R B B R R B R B R R R R

Each 4-sequence of Bs and Rs appears exactly once in this sequence (considered cyclically)!

($2^4 = 16$ such sequences, & 16 cards, hence exactly 16 starting points)

The Card Trick

Let's look at the sequence again:

1♠, 4♣, 8♣, 2♠, 3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥

B B B B R B B R R B R B R R R R

Each 4-sequence of Bs and Rs appears exactly once in this sequence (considered cyclically)!

When I asked all Red card holders to raise their hand, I got a sequence

R	R	B	B

The Card Trick

Let's look at the sequence again:

1♠, 4♣, 8♣, 2♠, 3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥

B B B B R B B R R B R B R R R R

Each 4-sequence of Bs and Rs appears exactly once in this sequence (considered cyclically)!

When I asked all Red card holders to raise their hand, I got a sequence

R R B B

The Card Trick

Let's look at the sequence again:

1♠, 4♣, 8♣, 2♠, 3♥, 5♣, 3♣, 9♦, 2♦, 1♣, 7♥, 7♠, 1♦, 5♦, 4♥, 2♥

B B B B R B B R R B R B R R R R

Each 4-sequence of Bs and Rs appears exactly once in this sequence (considered cyclically)!

When I asked all Red card holders to raise their hand, I got a sequence

R R B B

gives 4♥, 2♥, 1♠, 4♣

The Card Trick

So what's the graph theory connection?

The Card Trick

So what's the graph theory connection?

B B B B R B B R R B R B R R R R

Each 4-sequence of Bs and Rs appears exactly once in this sequence (considered cyclically)!

This is called a de Bruijn sequence with window k.

The Card Trick

So what's the graph theory connection?

B B B B R B B R R B R B R R R R

Each 4-sequence of Bs and Rs appears exactly once in this sequence (considered cyclically)!

This is called a de Bruijn sequence with window k.

Definition: A k-window de Bruijn sequence is a binary sequence of length 2^k , where every length k binary sequence appears exactly once.

De Bruijn Sequences

Definition: A k-window de Bruijn sequence is a binary sequence of length 2^k , where every length k binary sequence appears exactly once.

$k = 2$: 0 0 1 1

De Bruijn Sequences

Definition: A k-window de Bruijn sequence is a binary sequence of length 2^k , where every length k binary sequence appears exactly once.

$k = 2$: 0 0 1 1

$k = 3$: 0 0 0 1 0 1 1 1

De Bruijn Sequences

Definition: A k-window de Bruijn sequence is a binary sequence of length 2^k, where every length k binary sequence appears exactly once.

$k = 2$: 0 0 1 1

$k = 3$: 0 0 0 1 0 1 1 1

$k = 4$:

B	B	B	B	R	B	B	R	R	B	R	B	R	R	R	R
0	0	0	0	1	0	0	1	1	0	1	0	1	1	1	1

De Bruijn Sequences

Definition: A k-window de Bruijn sequence is a binary sequence of length 2^k, where every length k binary sequence appears exactly once.

$k = 2$: 0 0 1 1

$k = 3$: 0 0 0 1 0 1 1 1

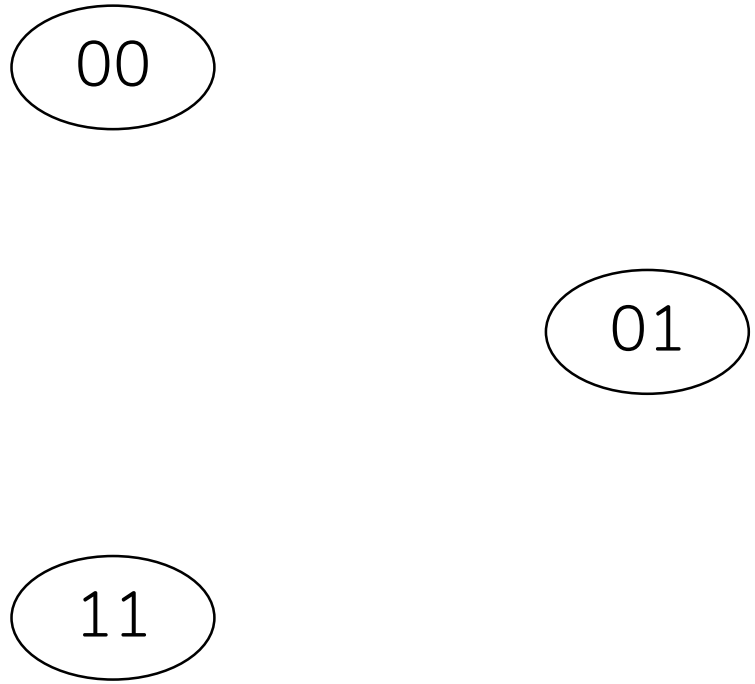
$k = 4$: **B B B B R B B R R B R B R R R R**

0 0 0 0 1 0 0 1 1 0 1 0 1 1 1 1

But do such sequences always exist? How do we find them?

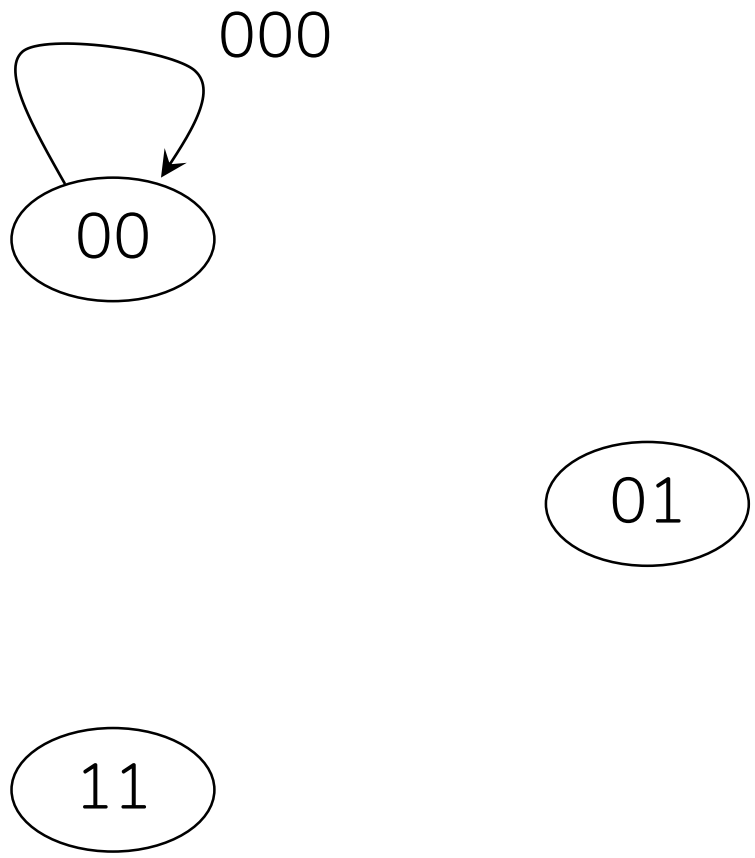
De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1



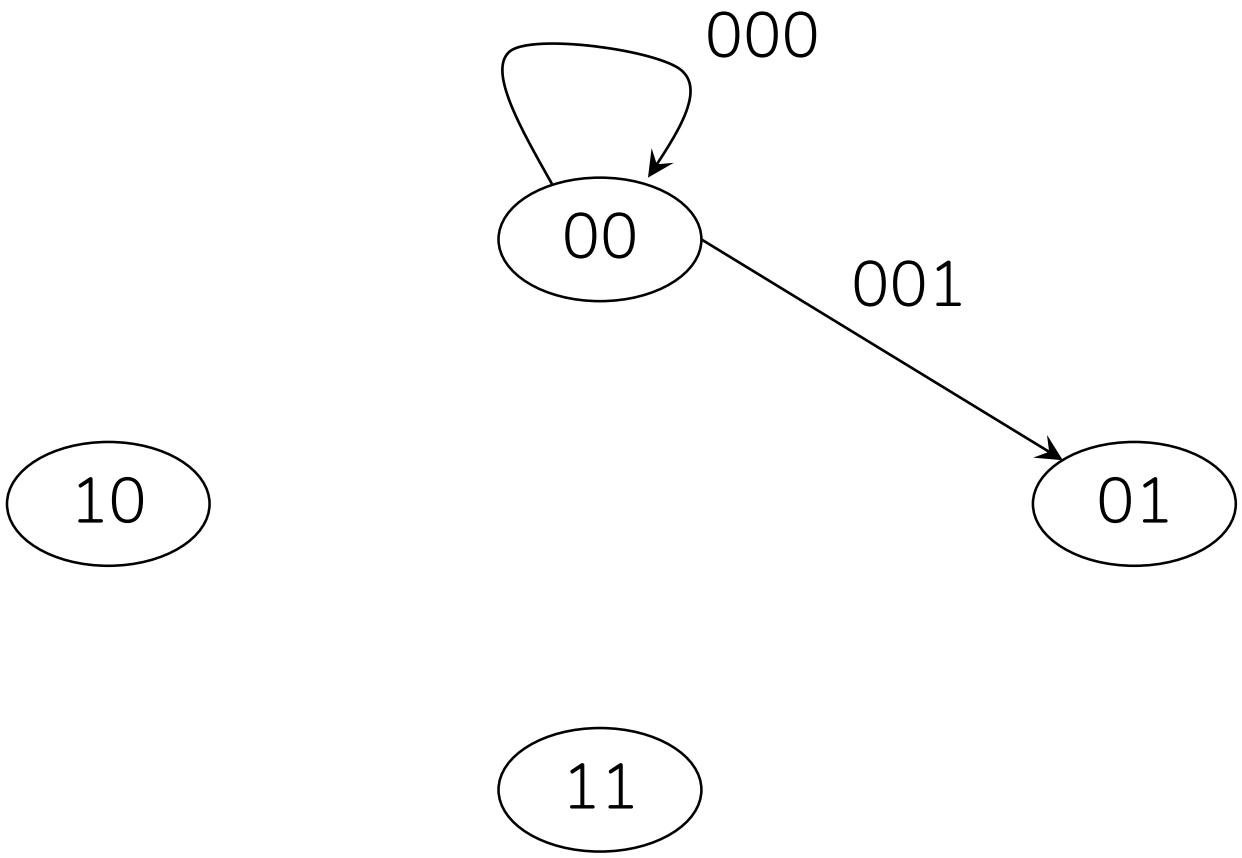
De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1



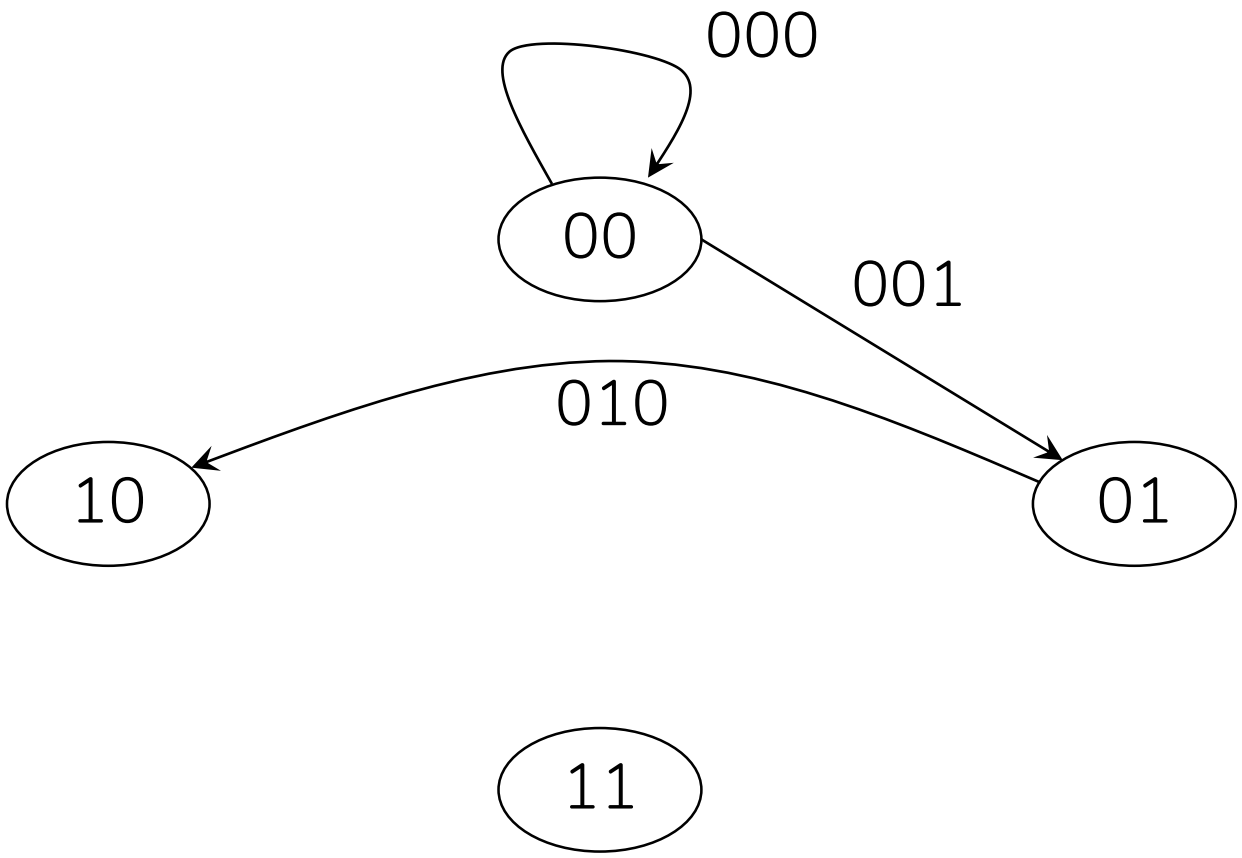
De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1



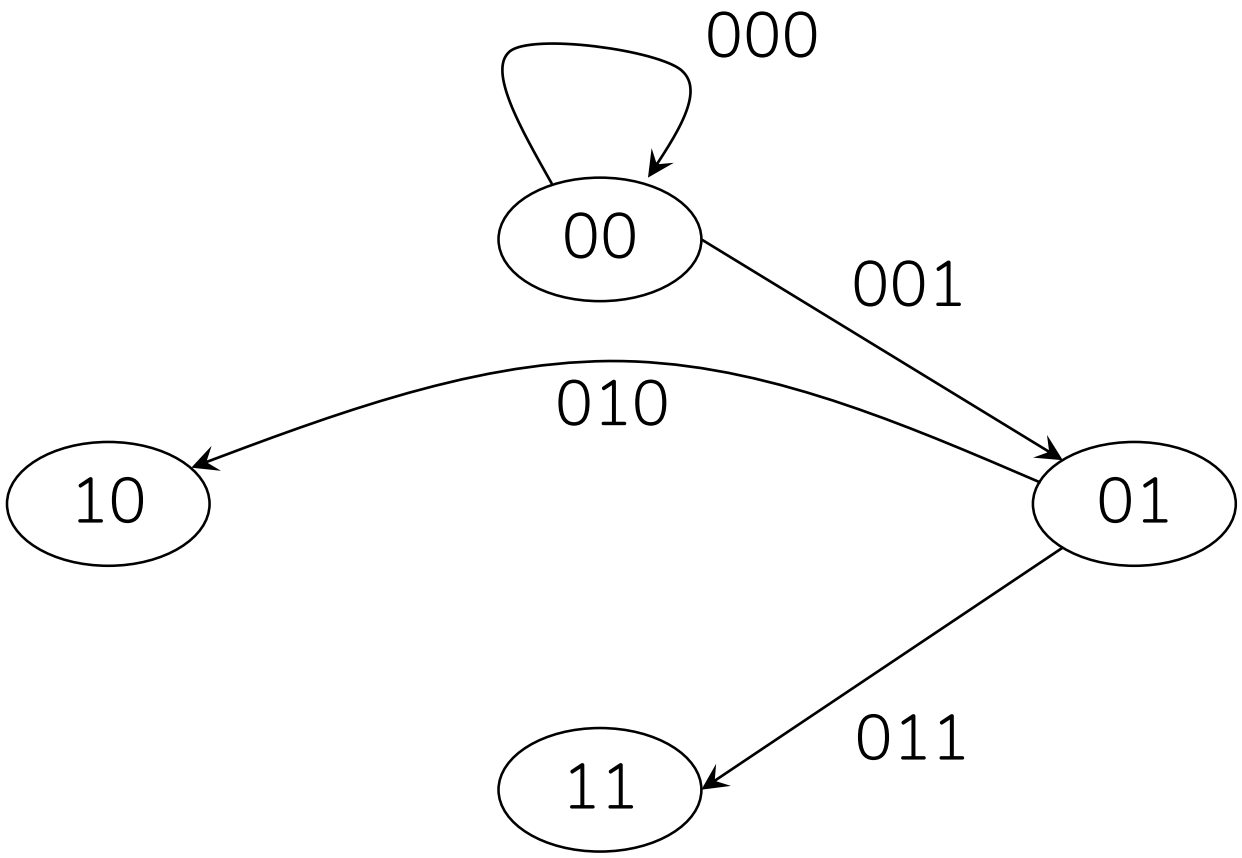
De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1



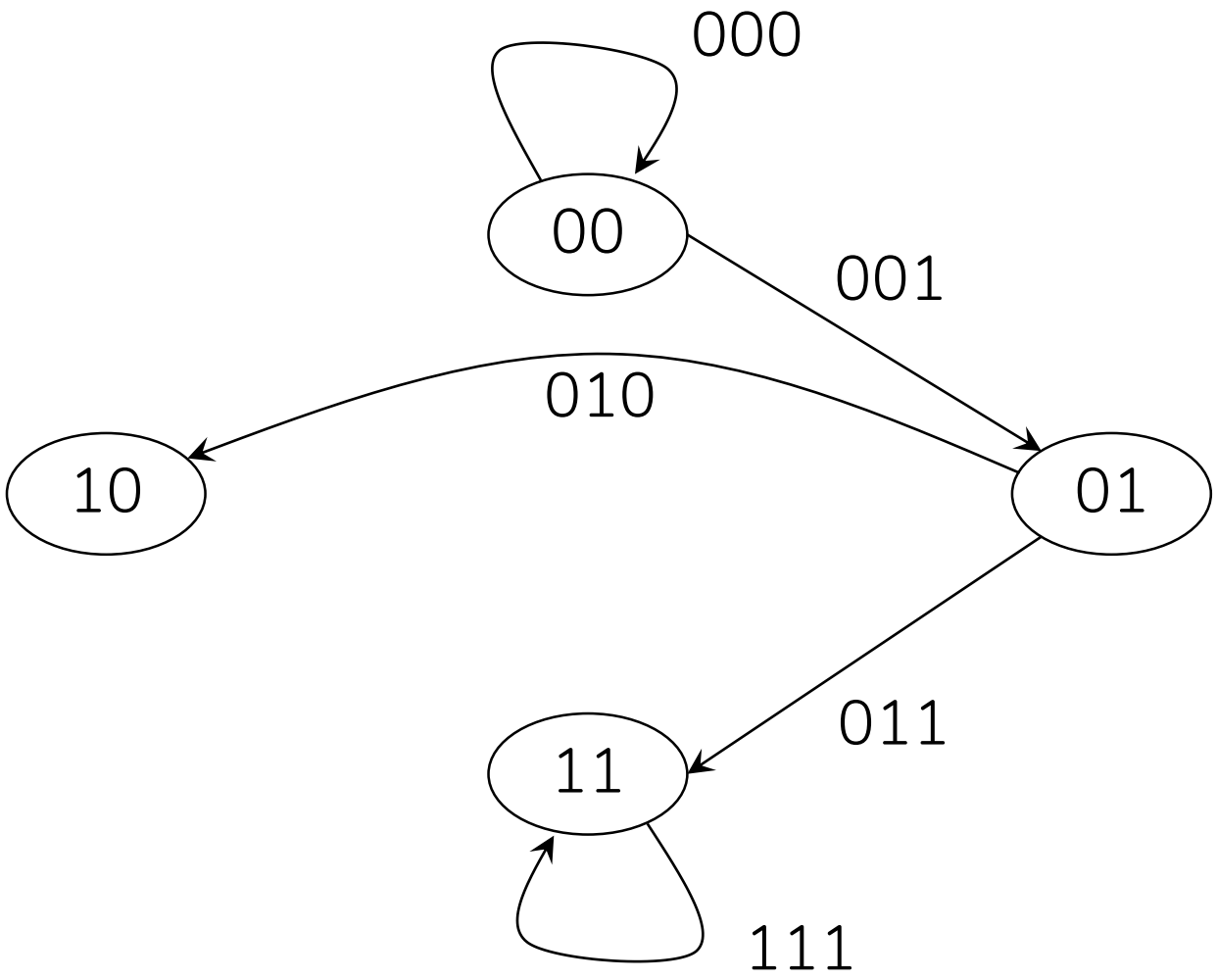
De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1



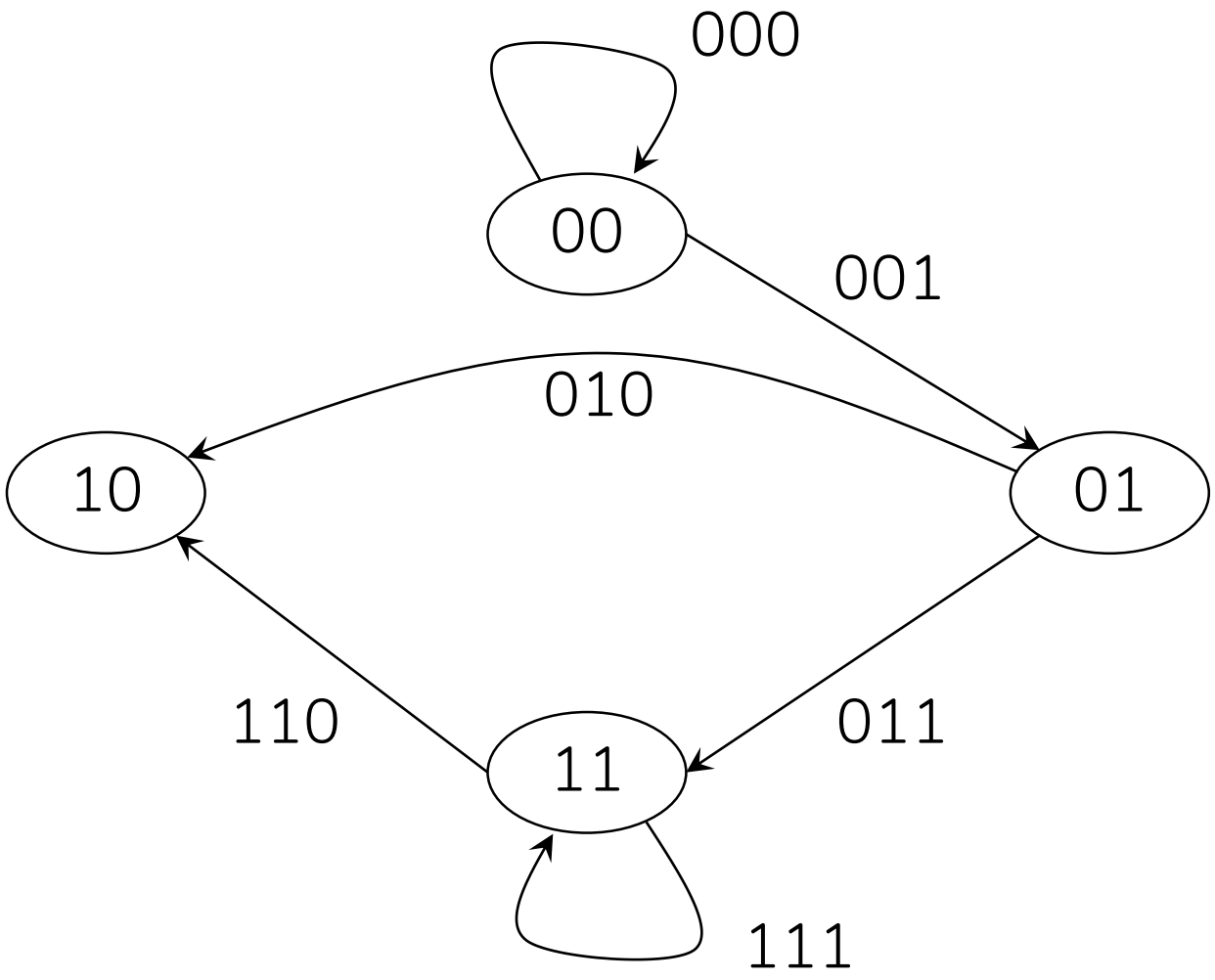
De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1



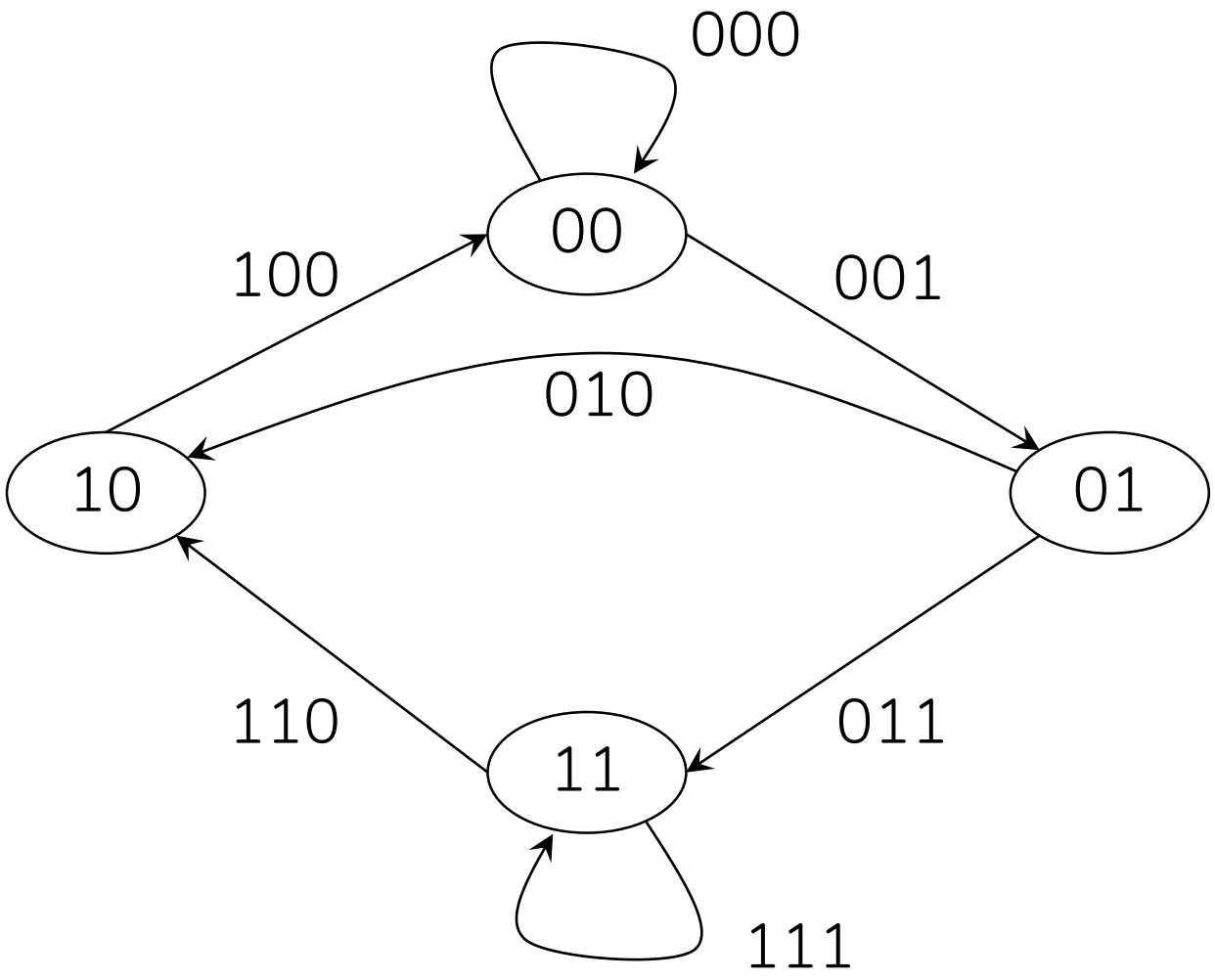
De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1



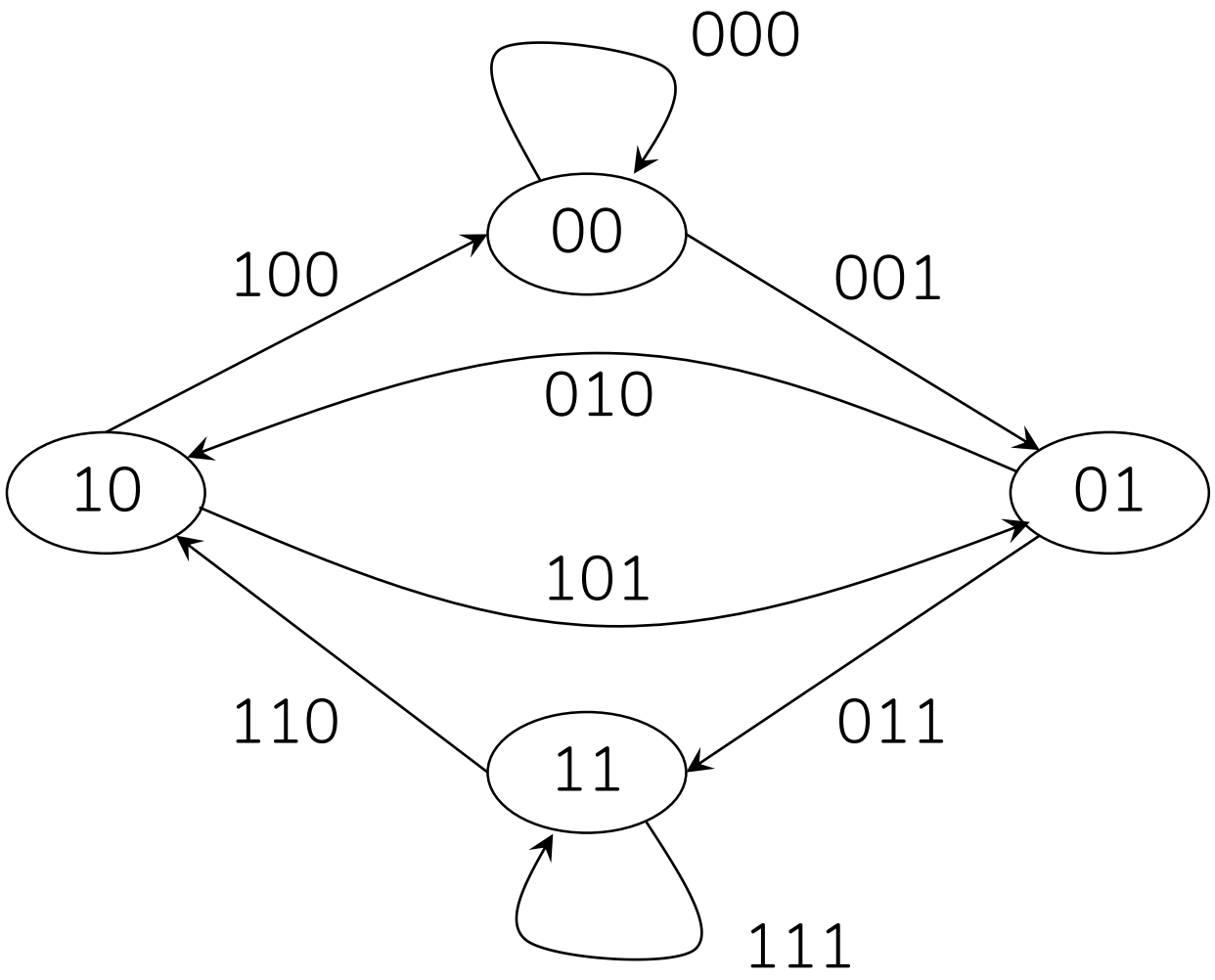
De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1



De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1

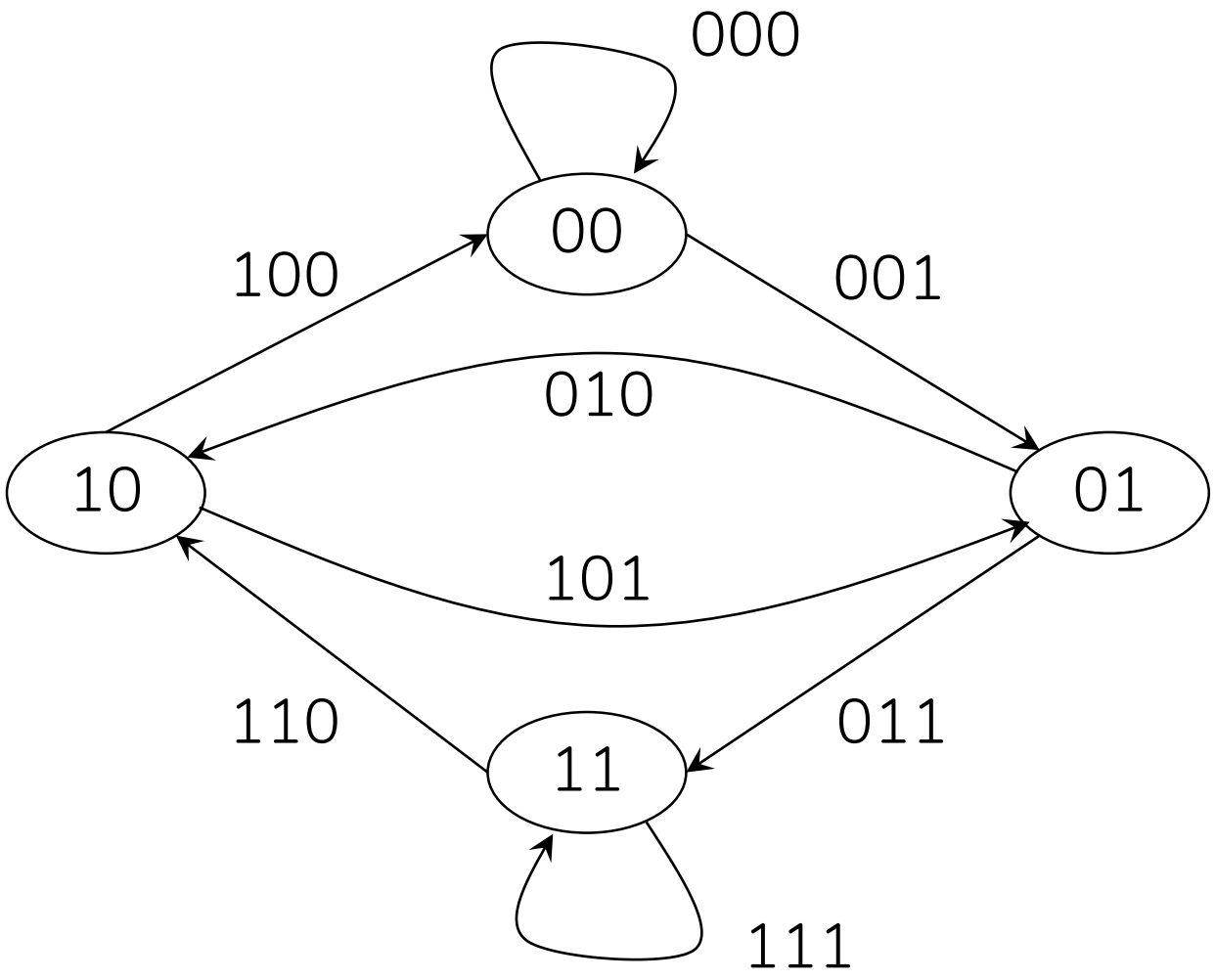


De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1

Each vtx has
in-degree = out-degree = 2

8 edges



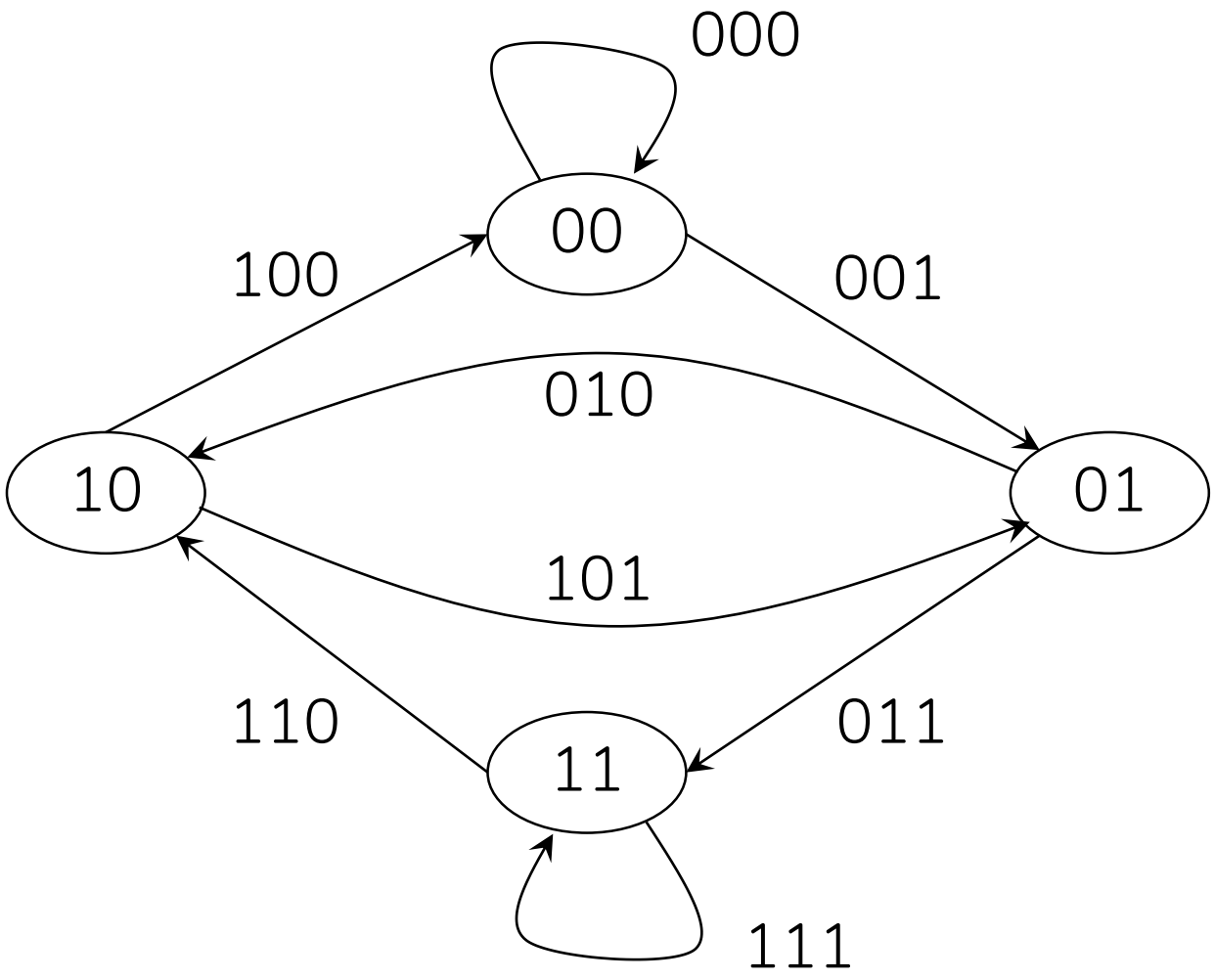
De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1

Each vtx has
in-degree = out-degree = 2

8 edges

Each 3-sequence of 1s and 0s appears exactly once



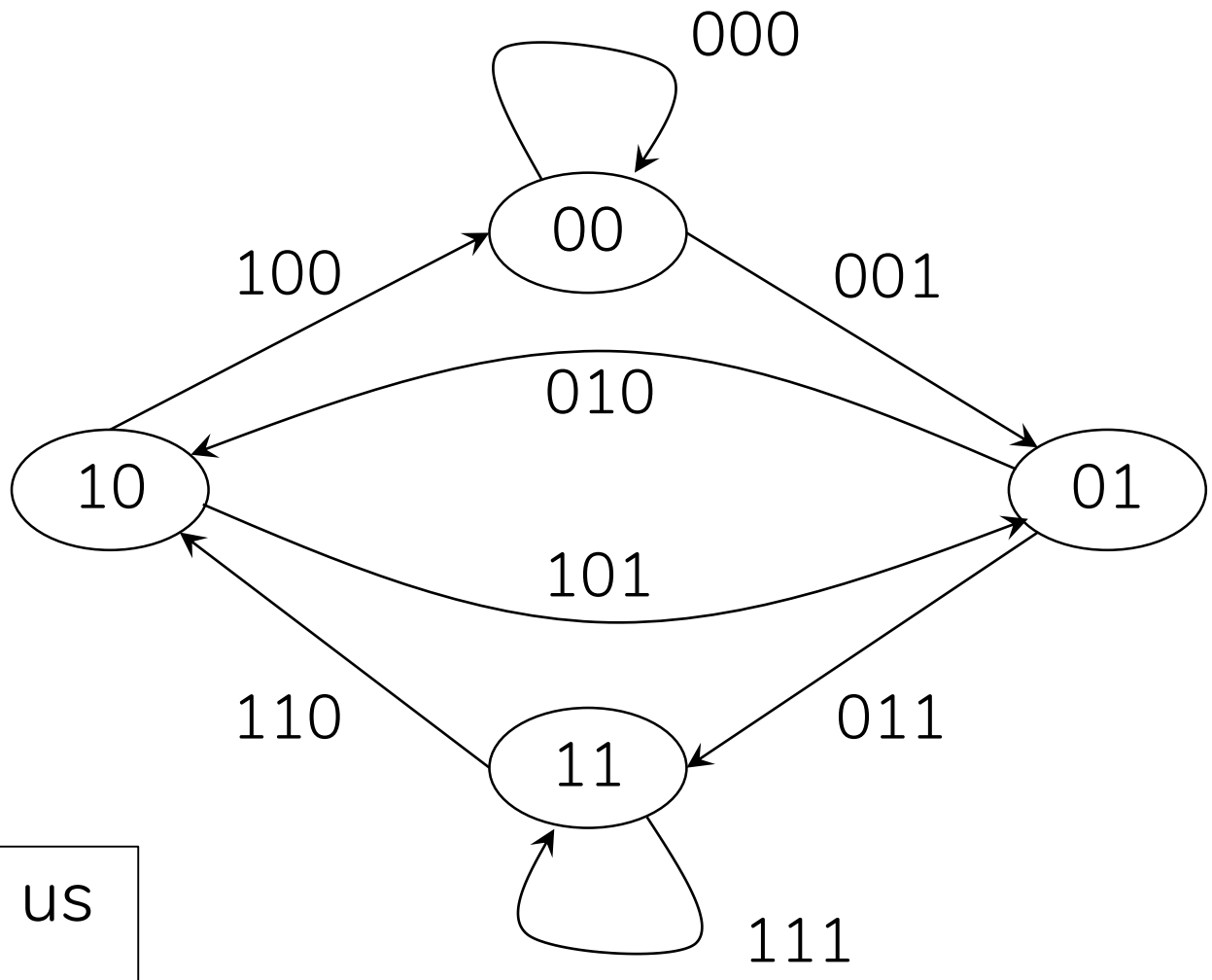
De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1

Each vtx has
in-degree = out-degree = 2

8 edges

Each 3-sequence of 1s and 0s
appears exactly once



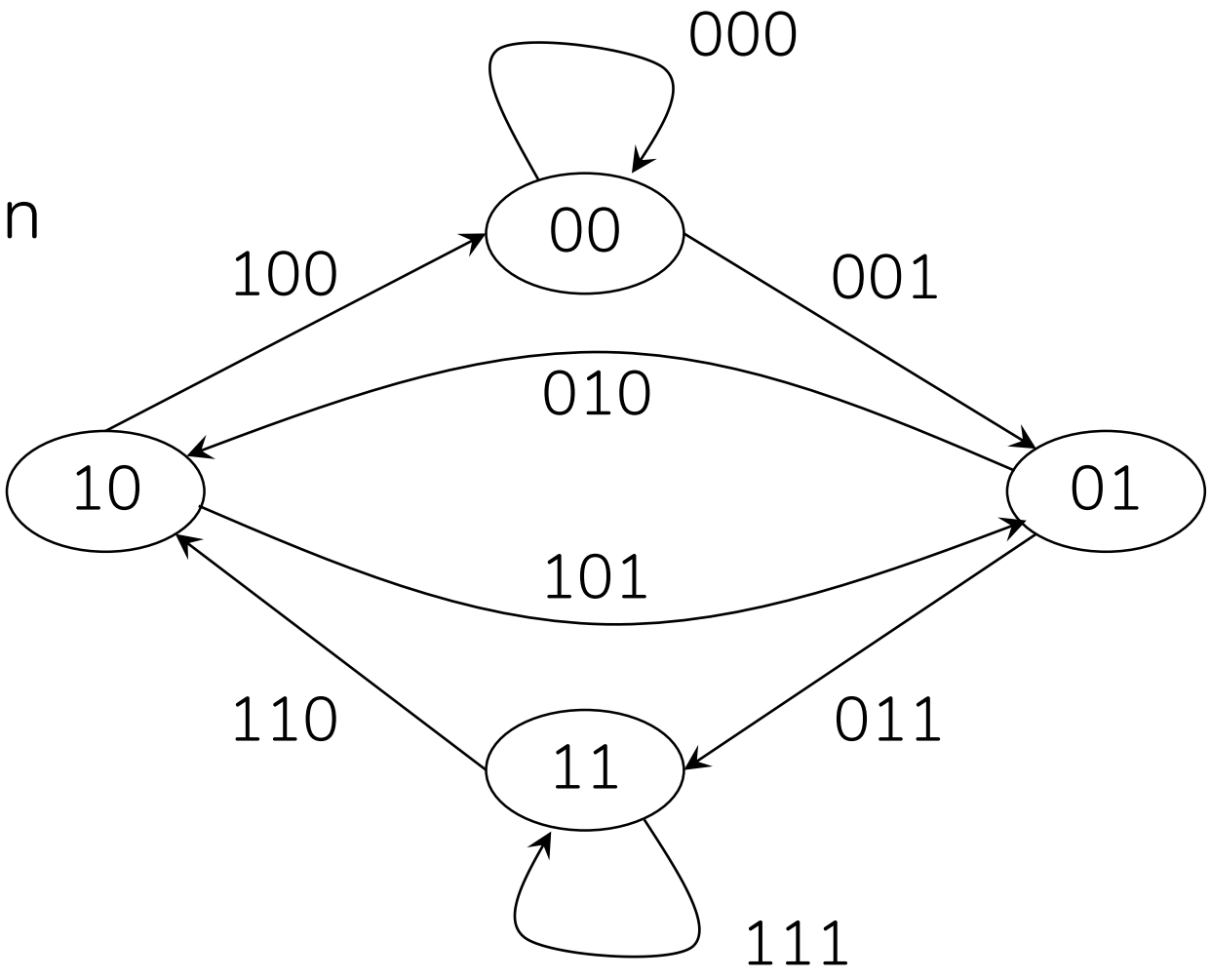
Eulerian walk on this graph gives us
a 3-window de Bruijn sequence.

De Bruijn Sequences and Eulerian Walks

$k = 3$: 0 0 0 1 0 1 1 1

To generate a k-window de Bruijn sequence:

- draw graph with 2^{k-1} nodes, each node with length $k-1$ binary string
- add edges (2 incoming and outgoing for each node), each edge gives length k string
- find an Eulerian walk



Other Uses

de Bruijn sequences useful for many applications:

- robotics, for robots to decide where they are
- encoding and decoding
- genome reconstruction
- ...

Other Uses

de Bruijn sequences useful for many applications:

- robotics, for robots to decide where they are
- encoding and decoding
- genome reconstruction
- ...

Thanks for listening!