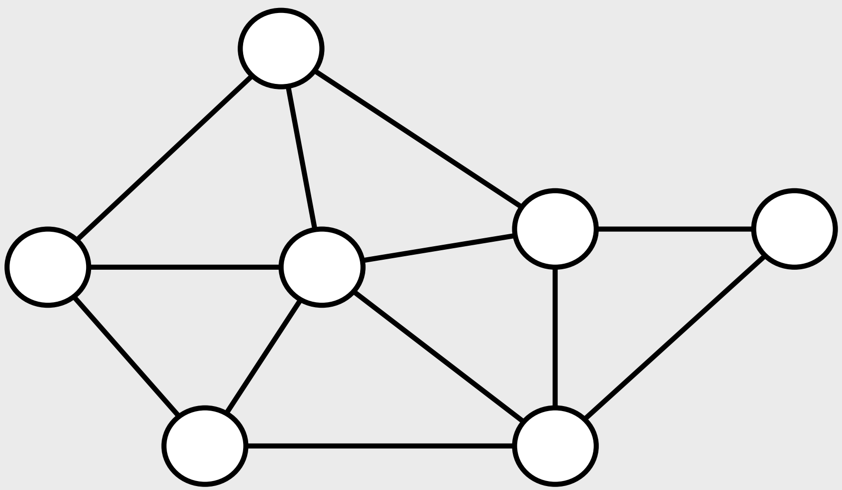
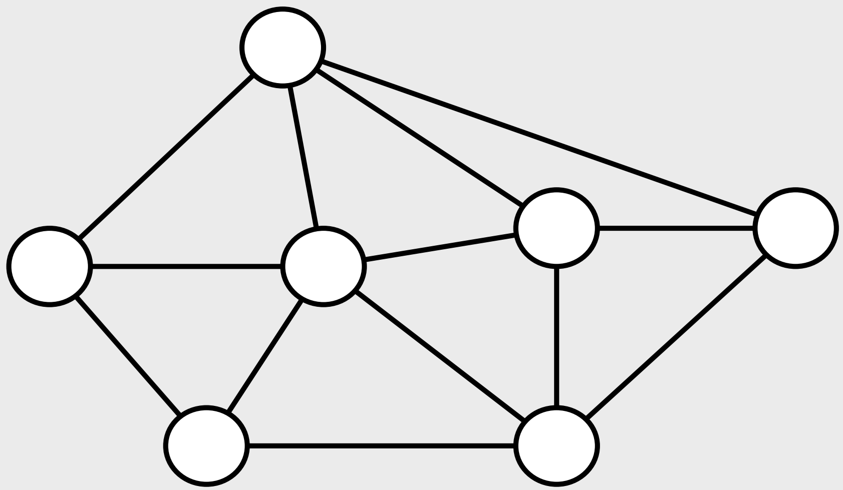
Construction Sequences and Certifying 3-Connectedness

Jens M. Schmidt

3-Connectedness

Let *G=(V,E)* be a finite graph without self-loops, *n*=|*V*|, *m*=|*E*|.

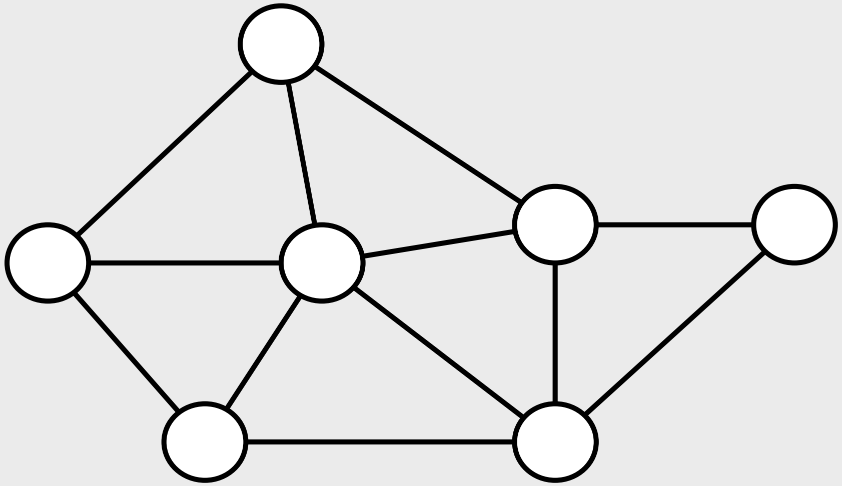


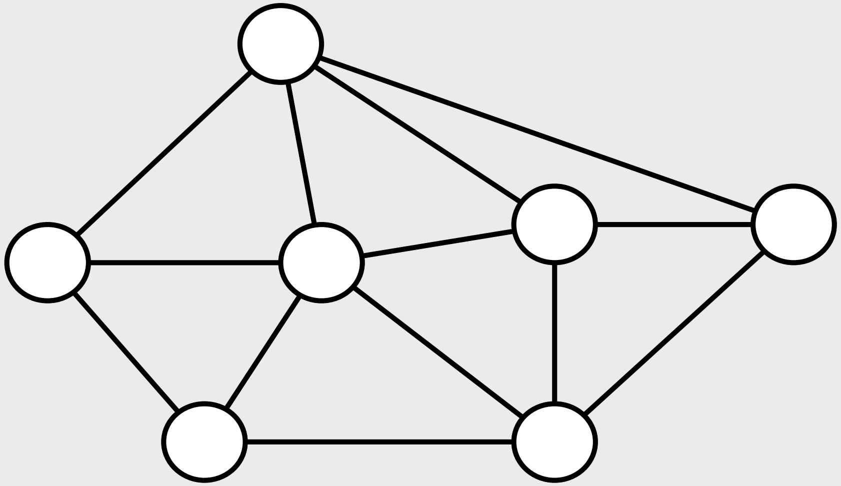


3-Connectedness

Let *G=(V,E)* be a finite graph without self-loops, *n*=|*V*|, *m*=|*E*|.

G is *3-connected* ⇔
    n > *3* and deleting *2* nodes does not result in a disconnected graph

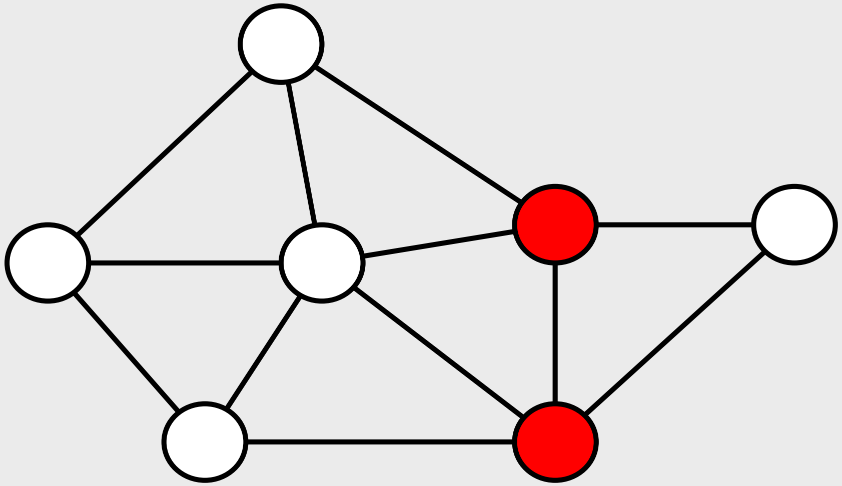


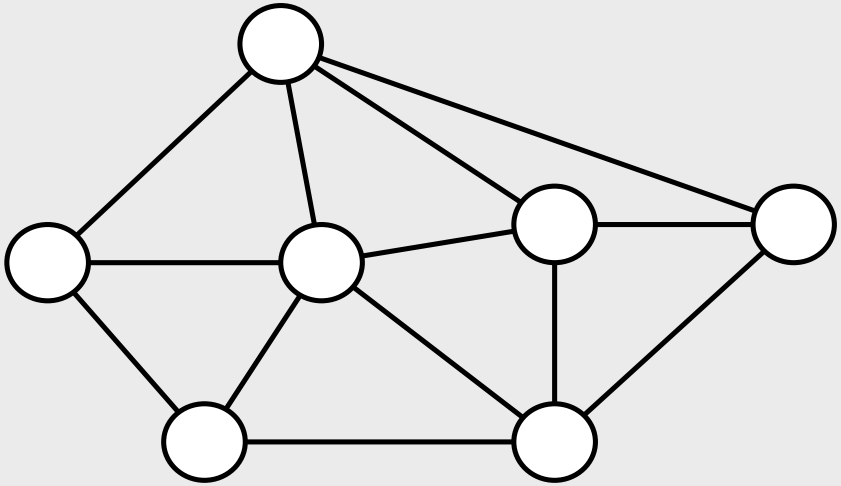


3-Connectedness

Let *G=(V,E)* be a finite graph without self-loops, *n*=|*V*|, *m*=|*E*|.

G is *3-connected* ⇔
n > 3 and deleting *2* nodes does not result in a disconnected graph

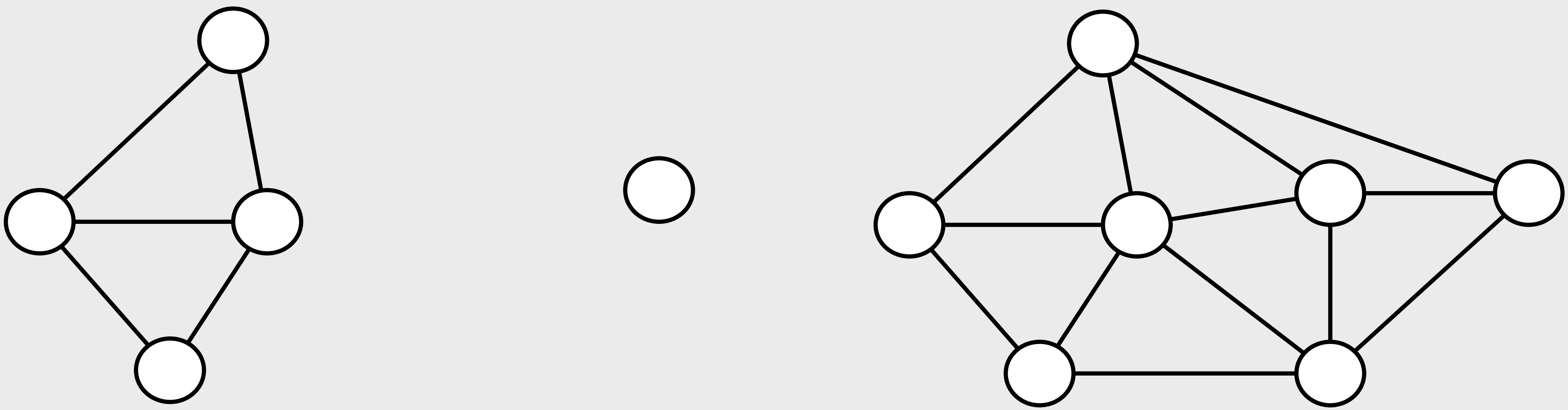




3-Connectedness

Let *G=(V,E)* be a finite graph without self-loops, *n*=|*V*|, *m*=|*E*|.

G is *3-connected* ⇔
n > *3* and deleting *2* nodes does not result in a disconnected graph

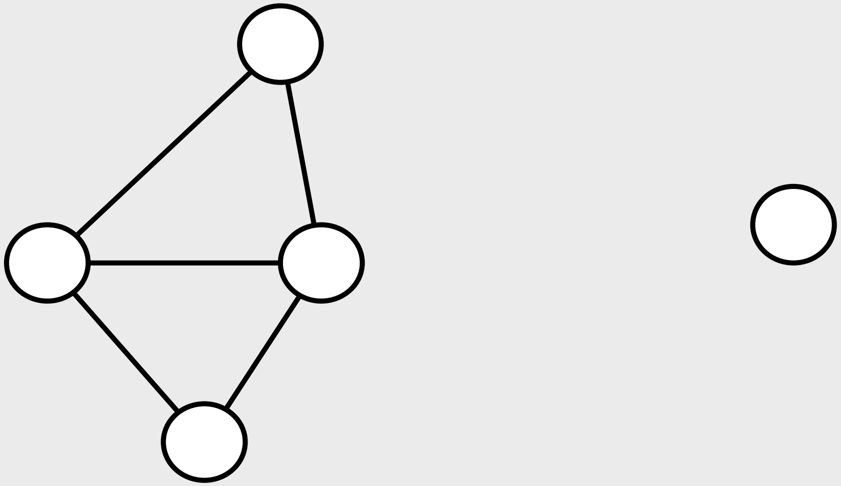


not 3-connected

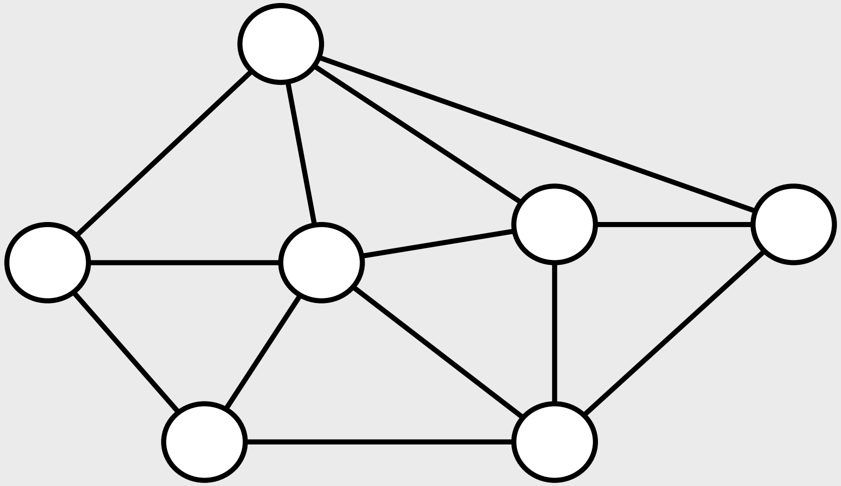
3-Connectedness

Let $G=(V,E)$ be a finite graph without self-loops, $n=|V|$, $m=|E|$.

G is *3-connected* $\Leftrightarrow$
$n > 3$ and deleting 2 nodes does not result in a disconnected graph



not 3-connected

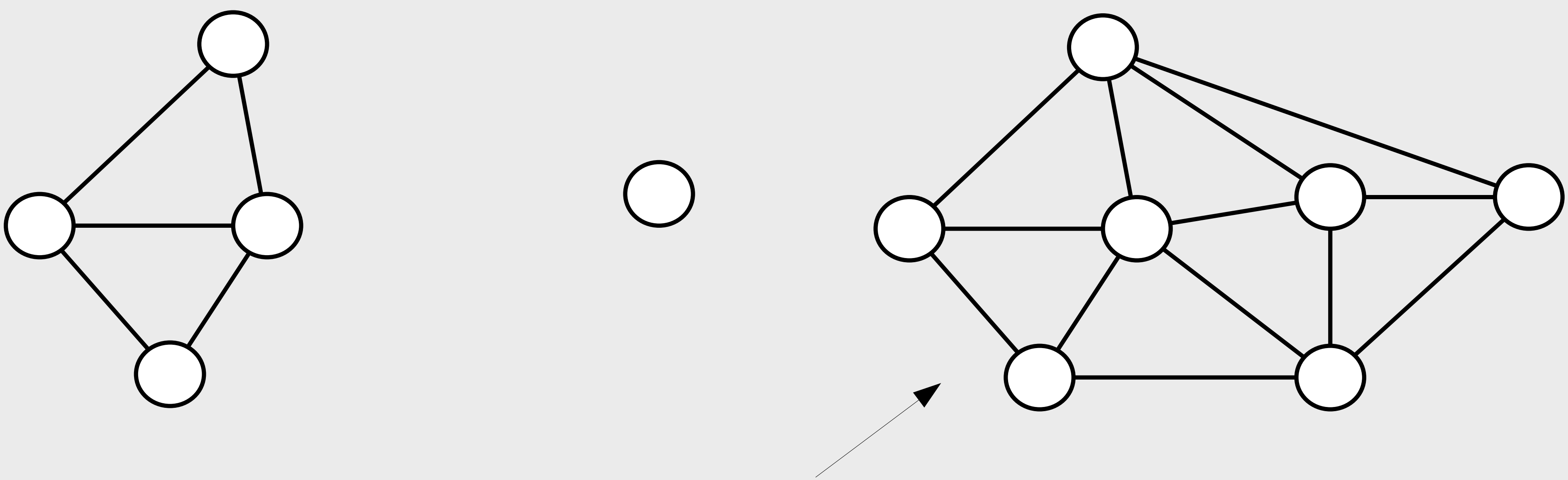


3-connected

3-Connectedness

Let $G=(V,E)$ be a finite graph without self-loops, $n=|V|$, $m=|E|$.

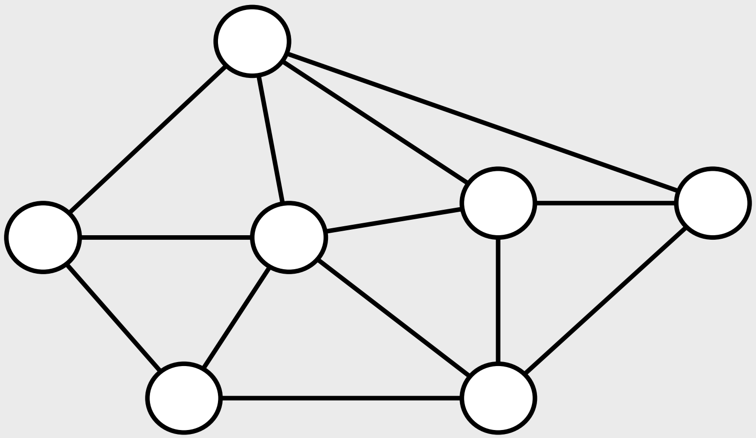
G is *3-connected* ⇔
$n > 3$ and deleting 2 nodes does not result in a disconnected graph



Problem 1: Is there a nice certificate for 3-connectedness?

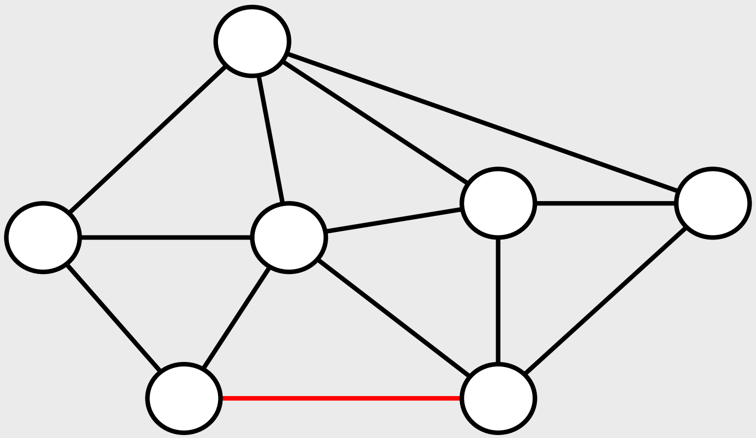
Contractible+Removable edges

An edge is *contractible* if its contraction obtains a 3-connected graph.



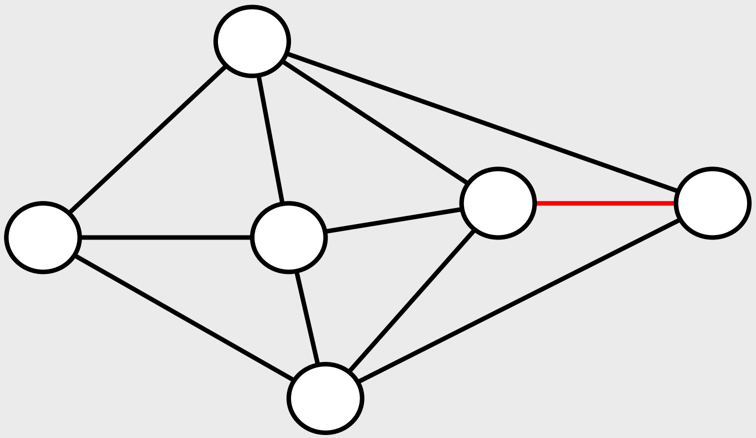
Contractible+Removable edges

An edge is *contractible* if its contraction obtains a 3-connected graph.



Contractible+Removable edges

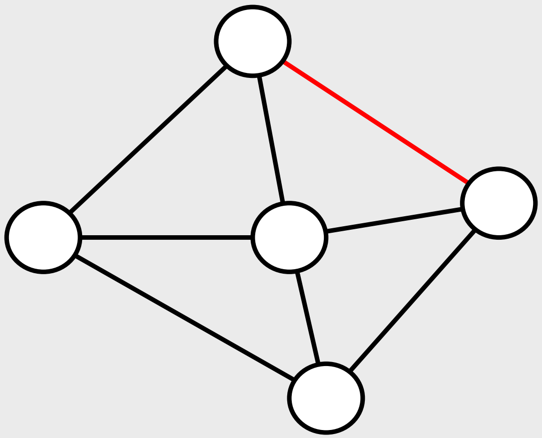
An edge is *contractible* if its contraction obtains a 3-connected graph.



(parallel edges may occur)

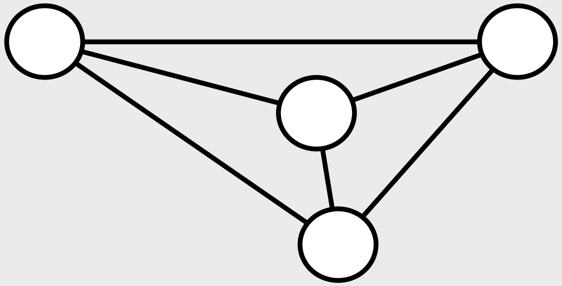
Contractible+Removable edges

An edge is *contractible* if its contraction obtains a 3-connected graph.



Contractible+Removable edges

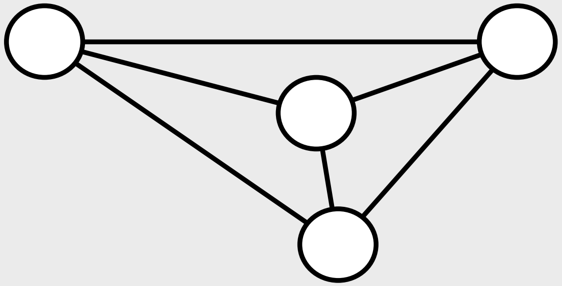
An edge is *contractible* if its contraction obtains a 3-connected graph.



...we end up with the K_4

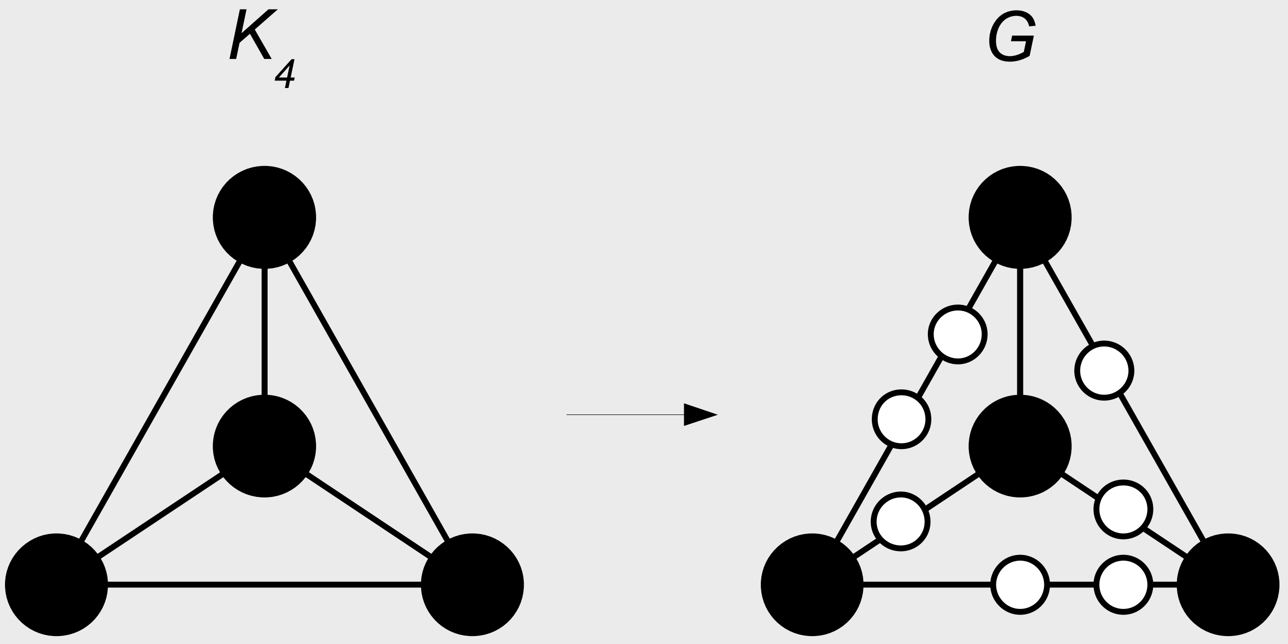
Contractible+Removable edges

An edge is *contractible* if its contraction obtains a 3-connected graph.



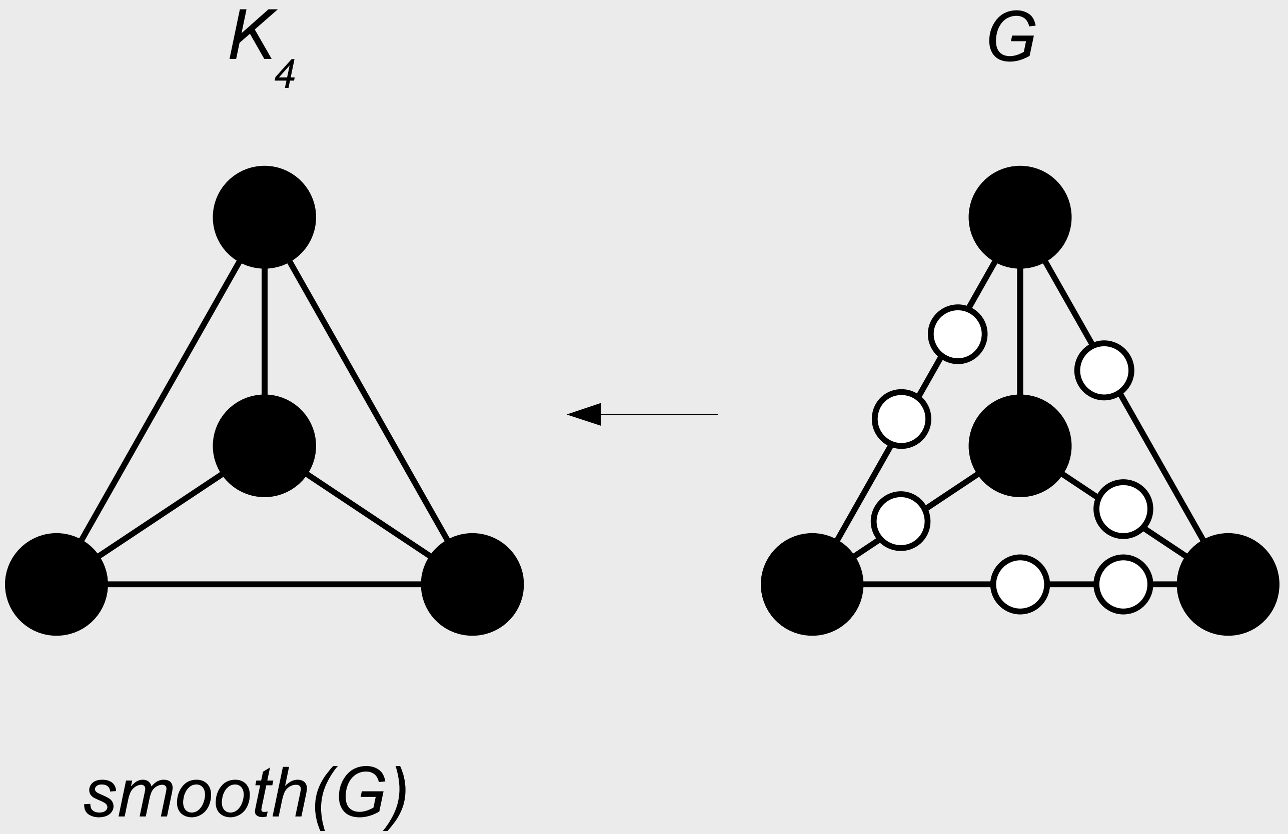
Thm (Tutte '61):
A 3-connected graph $\neq K_4$ contains a *contractible* edge.

Subdivisions



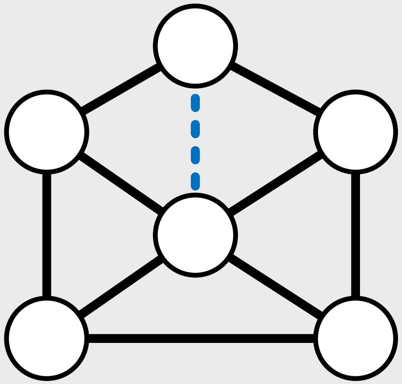
Subdivision of K_4

Subdivisions

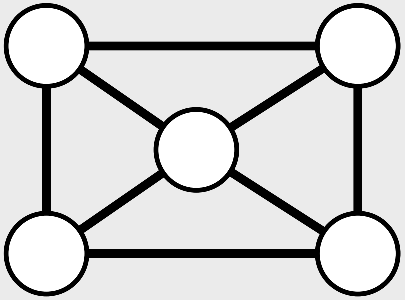


Contractible+Removable edges

An edge is *removable* if *smooth(G\e)* is 3-connected.



$\Longrightarrow$



Contractible+Removable edges

Thm (Barnette, Grünbaum '69):

A 3-connected graph $\neq K_4$ contains a *removable* edge.

Contractible+Removable edges

<u>Thm (Barnette, Grünbaum '69):</u>

A 3-connected graph $\neq K_4$ contains a *removable* edge.

Problem 2: How fast can a sequence of contractions / removals from G to the K_4 be computed?

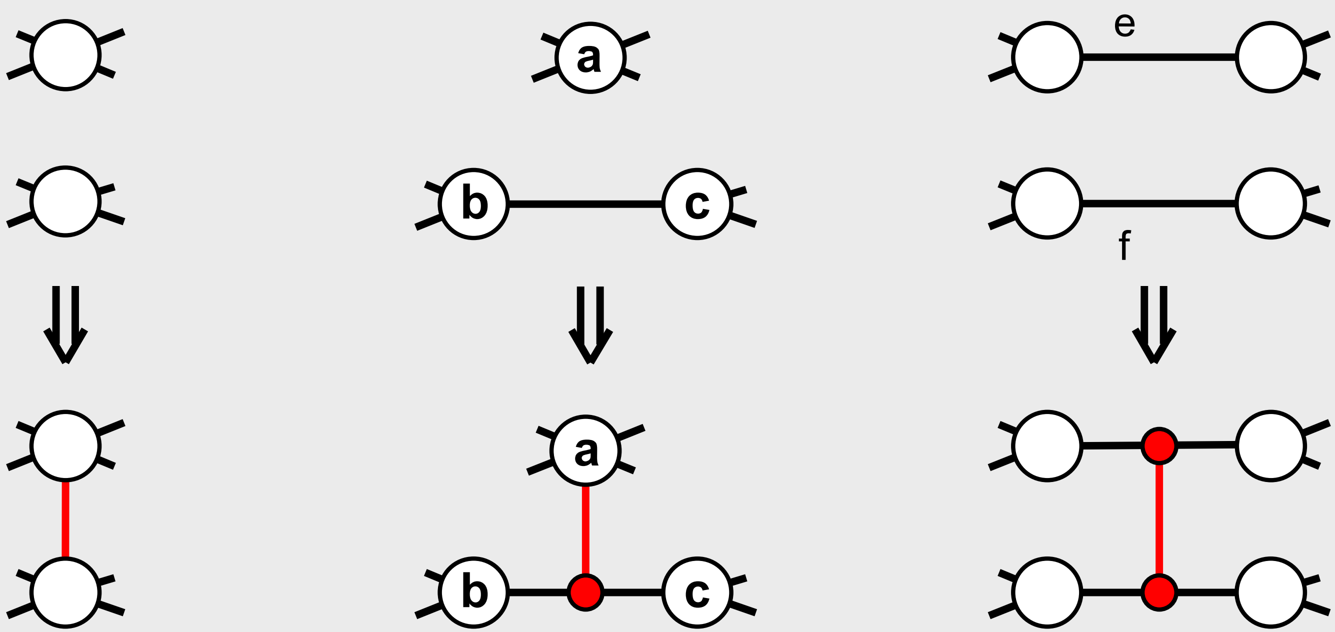
Construction Sequences

Thm (Barnette-Grünbaum '69):

G is 3-connected $\Leftrightarrow$
G can be constructed from the K_4 with BG-operations

Barnette-Grünbaum Operations

In a 3-connected graph:



parallel edges allowed

a≠b, a≠c

e≠f,
e and f not parallel

Each operation preserves 3-connectedness

Construction Sequences

A construction sequence (of BG-operations) would give

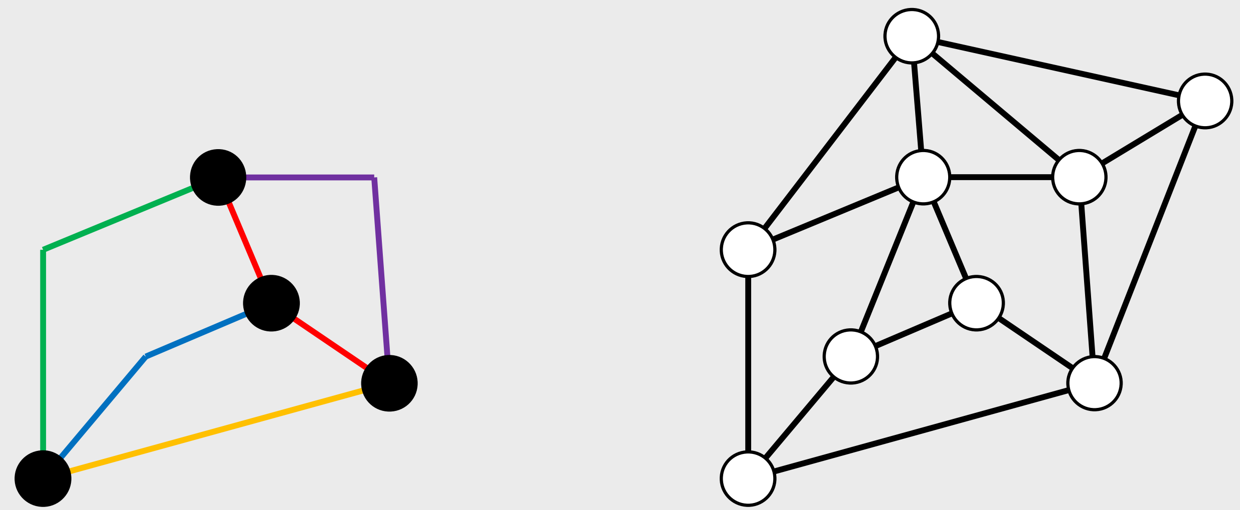
- the sequence of removals and
- a certificate for 3-connectedness.

But what about the sequence of contractions?

Thm: A sequence of BG-operations from the K_4 to G can be transformed to a contraction sequence in linear time.

Construction Sequences

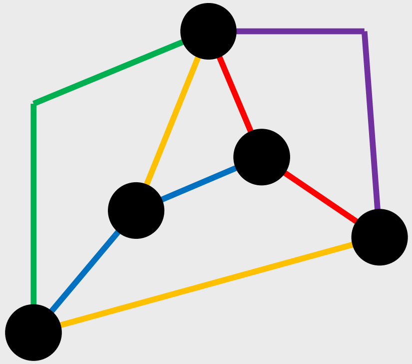
How can we compute a construction sequence?

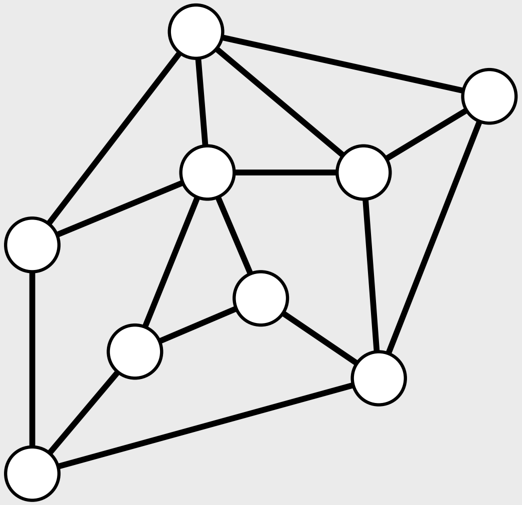


K_4 $\Longrightarrow$ G

Construction Sequences

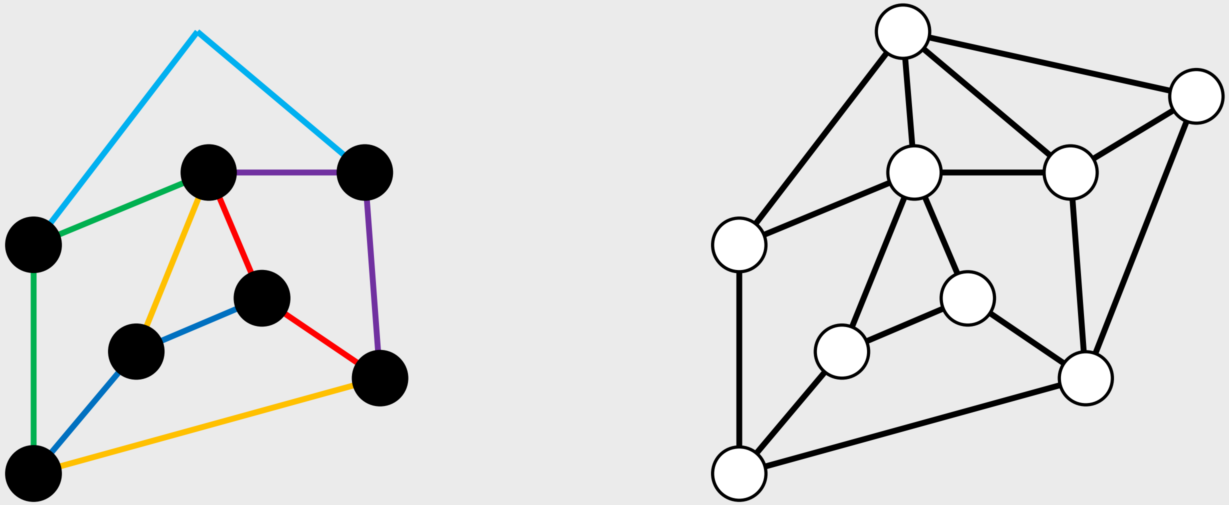
How can we compute a construction sequence?





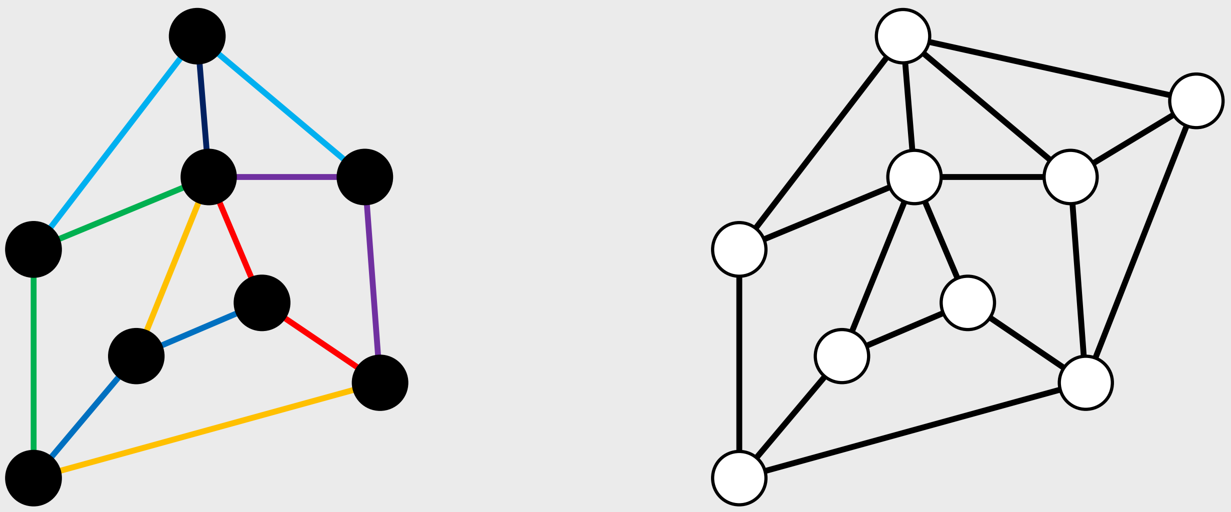
Construction Sequences

How can we compute a construction sequence?



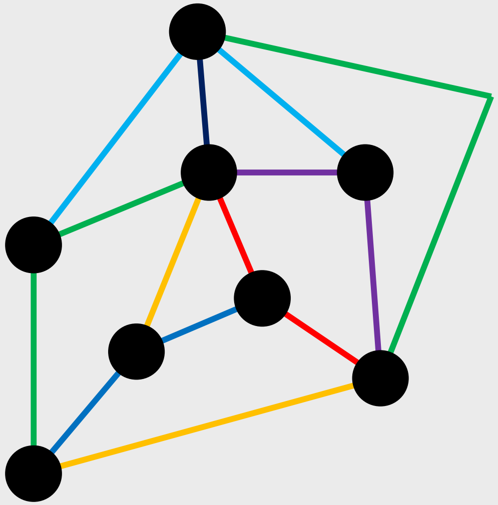
Construction Sequences

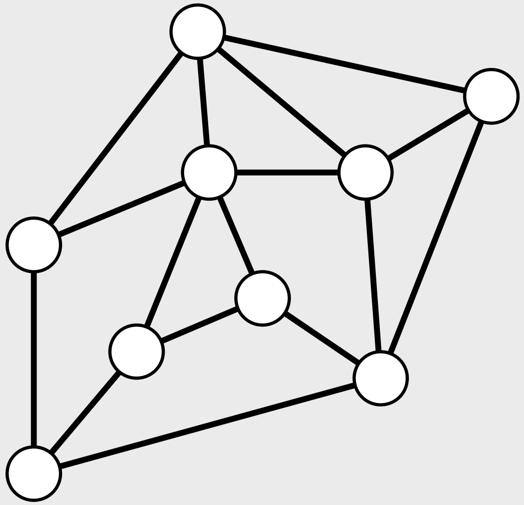
How can we compute a construction sequence?



Construction Sequences

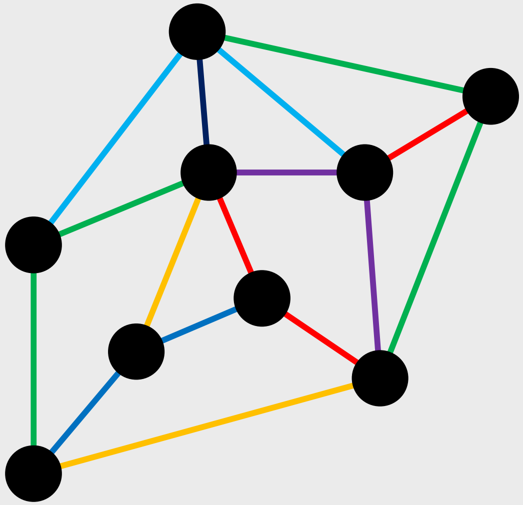
How can we compute a construction sequence?

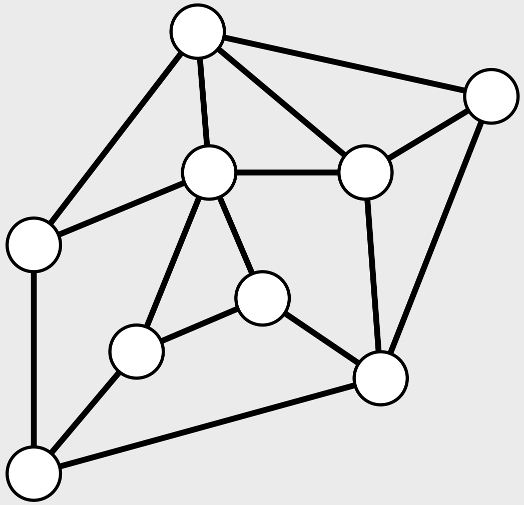




Construction Sequences

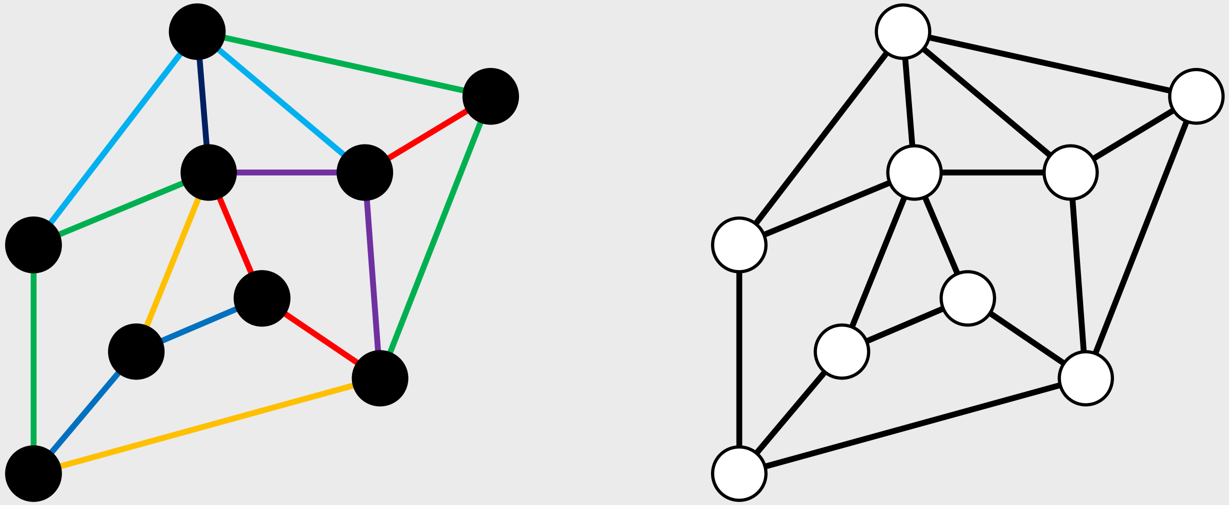
How can we compute a construction sequence?





Construction Sequences

How can we compute a construction sequence?

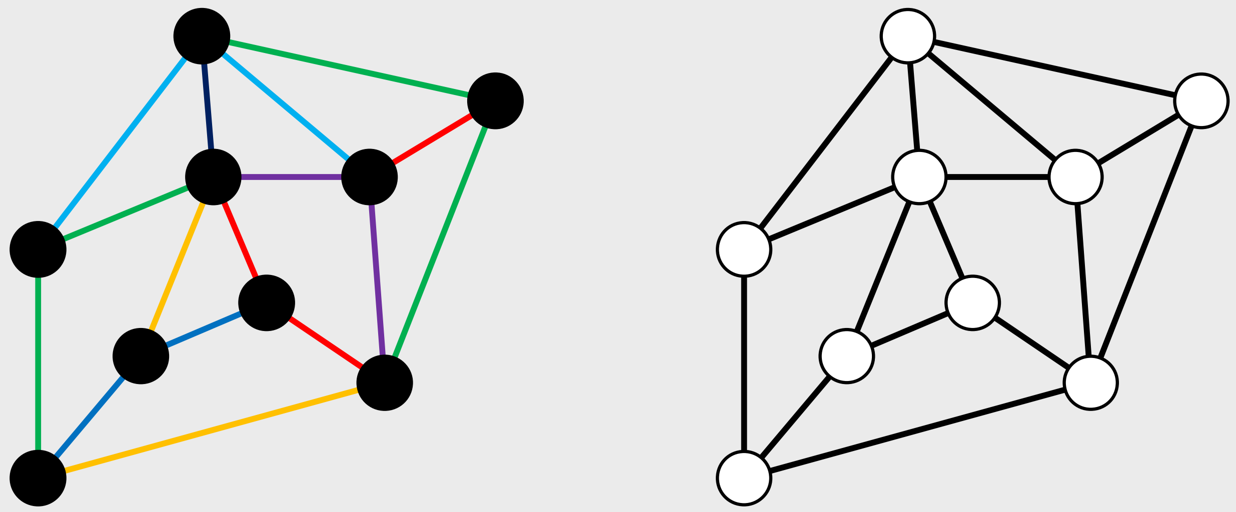


A performed BG-operation is **basic**, if it does not create parallel edges

Construction Sequences

Thm (Barnette-Grünbaum '69):

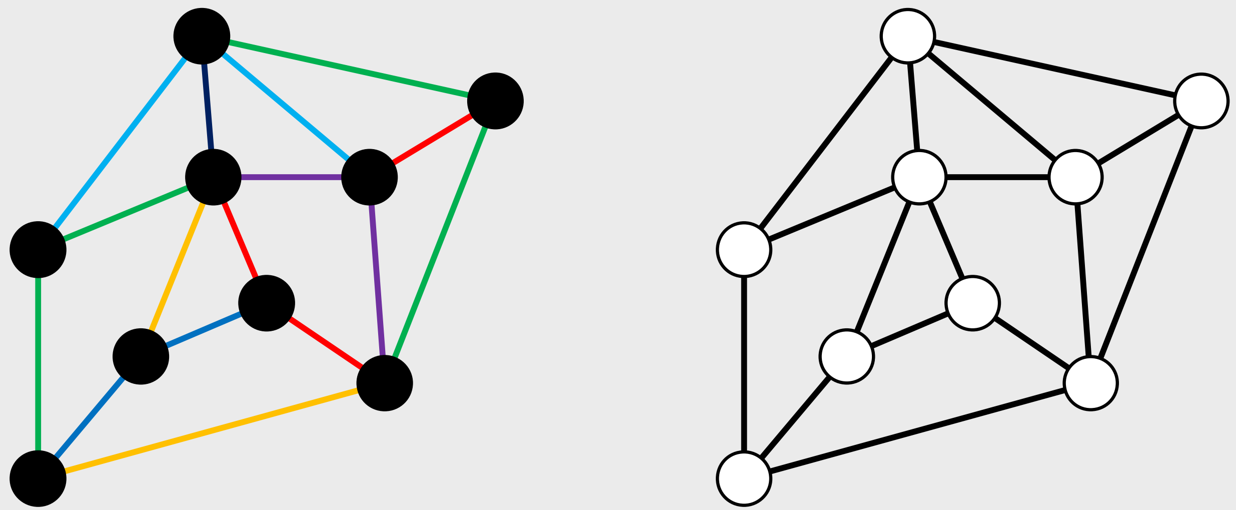
G is simple and 3-connected $\Leftrightarrow$
G can be constructed from the K_4 with basic BG-operations



From now on *G* is simple.

Construction Sequences

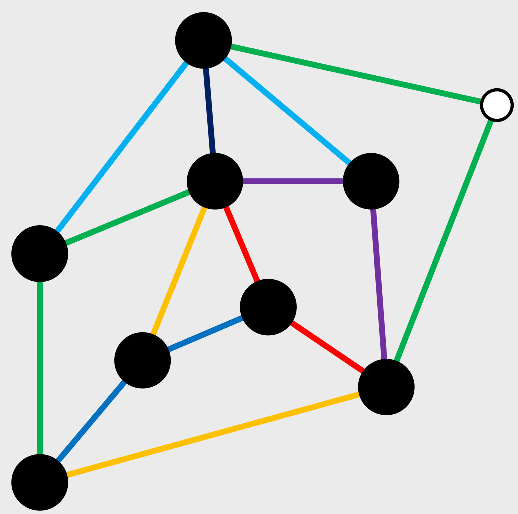
The inverse construction sequence applied to *G* yields a subgraph in *G* that is a K_4-subdivision.

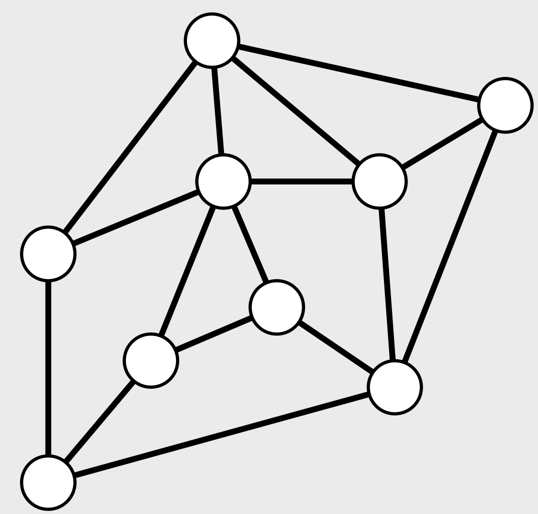


K_4 $\Longleftarrow$ G

Construction Sequences

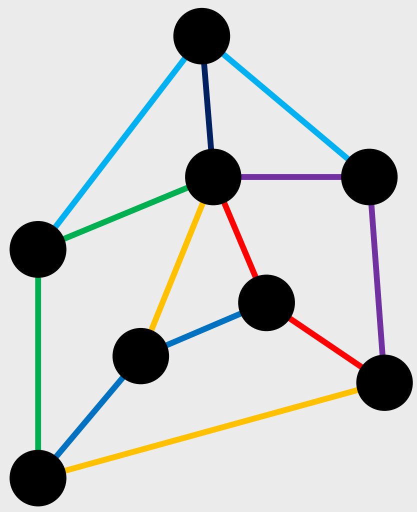
The inverse construction sequence applied to *G* yields a subgraph in *G* that is a K_4-subdivision.

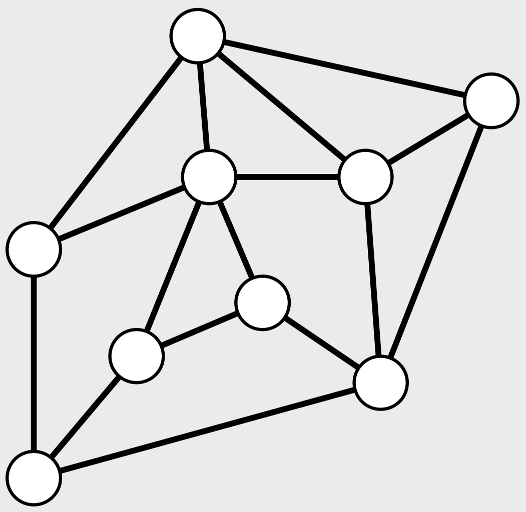




Construction Sequences

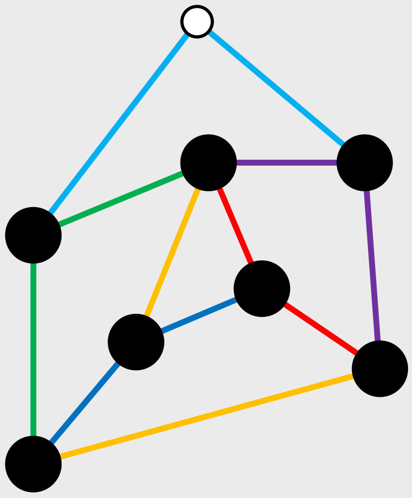
The inverse construction sequence applied to *G* yields a subgraph in *G* that is a K_4-subdivision.

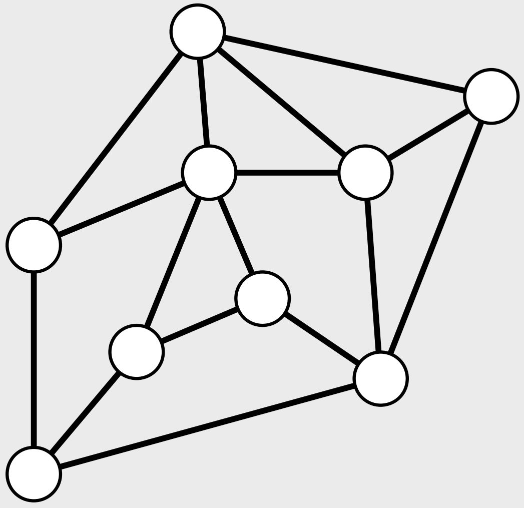




Construction Sequences

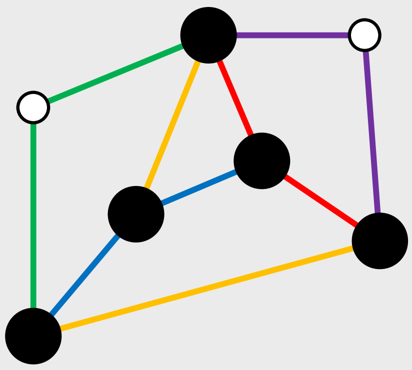
The inverse construction sequence applied to G yields a subgraph in G that is a K_4-subdivision.

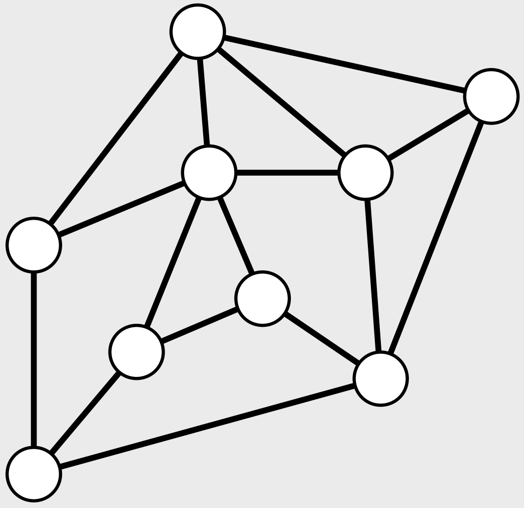




Construction Sequences

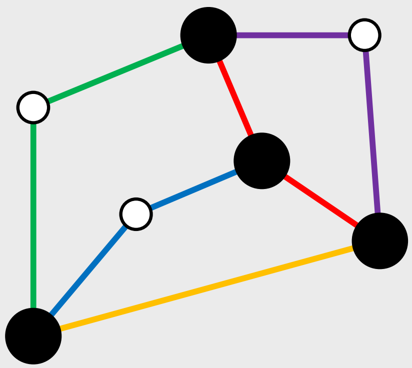
The inverse construction sequence applied to *G* yields a subgraph in *G* that is a K_4-subdivision.

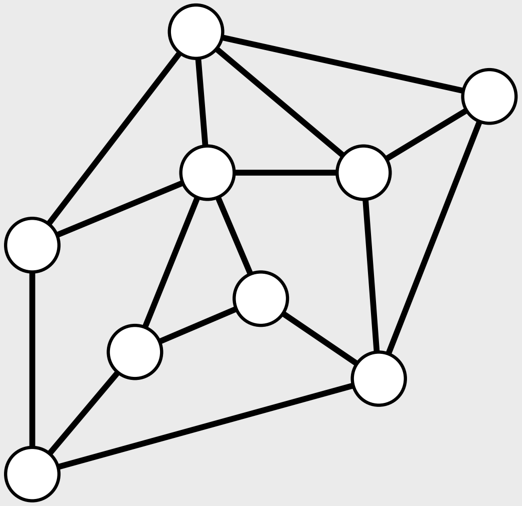




Construction Sequences

The inverse construction sequence applied to *G* yields a subgraph in *G* that is a K_4-subdivision.

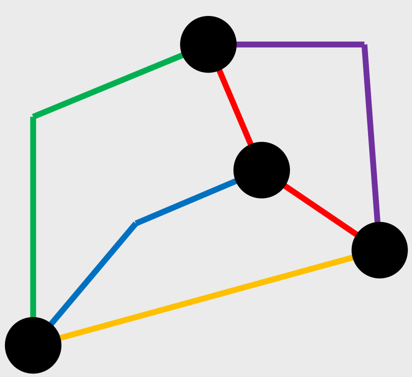




Construction Sequences

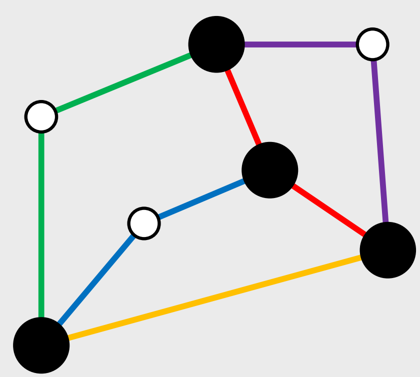
In a construction sequence

start with K_4	$\Leftrightarrow$	start with a subdivision of K_4 in G
3-conn. graphs	$\Leftrightarrow$	subdivisions of 3-conn. graphs in G
add BG-edge	$\Leftrightarrow$	add subdivided BG-edge (*BG-path*)
nodes	$\Leftrightarrow$	nodes of degree ≥ 3 (*real* nodes)



smooth(*H*)

$\Leftrightarrow$

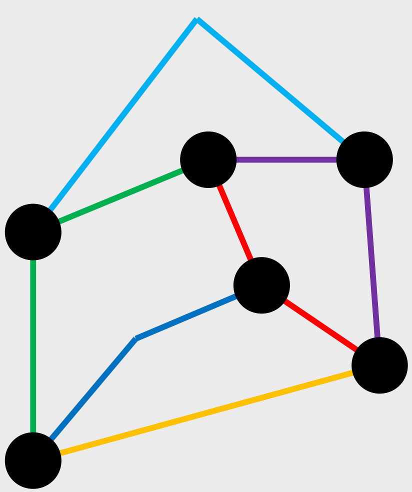


H

Construction Sequences

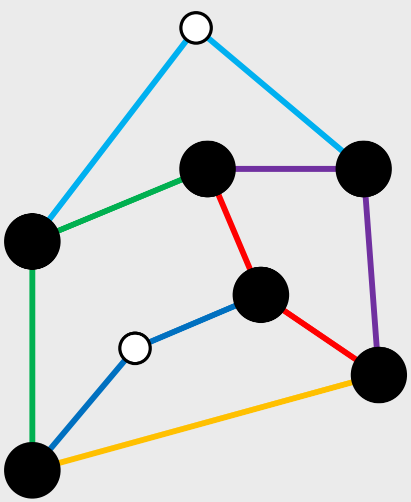
In a construction sequence

start with K_4	$\Leftrightarrow$	start with a subdivision of K_4 in G
3-conn. graphs	$\Leftrightarrow$	subdivisions of 3-conn. graphs in G
add BG-edge	$\Leftrightarrow$	add subdivided BG-edge (*BG-path*)
nodes	$\Leftrightarrow$	nodes of degree ≥ 3 (*real* nodes)



smooth(*H*)

$\Leftrightarrow$



H

Outline

1. Definitions

2. Existence Results

3. Algorithm

4. Testing 3-Connectedness

Idea

Idea: We construct the sequence bottom-up!

Idea

Barnette-Grünbaum choose a special K_4-subdivision in G.

What if we prescribe a K_4-subdivision?

Is there still a basic construction sequence starting from that subdivision?

Idea

Barnette-Grünbaum choose a special K_4-subdivision in G.

What if we prescribe a K_4-subdivision?

Is there still a basic construction sequence starting from that subdivision?

NO

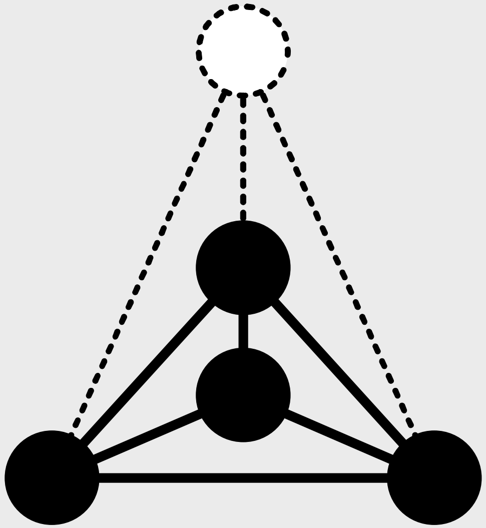
Idea

Barnette-Grünbaum choose a special K_4-subdivision in G.

What if we prescribe a K_4-subdivision?

Is there still a basic construction sequence starting from that subdivision?

NO



Idea

We drop the condition that sequences are basic.

Is there a (possibly non-basic) construction sequence starting from that K_4-subdivision?

Idea

We drop the condition that sequences are basic.

Is there a (possibly non-basic) construction sequence starting from that K_4-subdivision?

YES

Idea

We drop the condition that sequences are basic.

Is there a (possibly non-basic) construction sequence starting from that K_4-subdivision?

YES

Is there even a (possibly non-basic) construction sequence when starting from a prescribed subgraph H in G with *smooth(H)* being 3-connected?

Idea

We drop the condition that sequences are basic.

Is there a (possibly non-basic) construction sequence starting from that K_4-subdivision?

YES

Is there even a (possibly non-basic) construction sequence when starting from a prescribed subgraph H in G with $smooth(H)$ being 3-connected?

YES

Existence Result

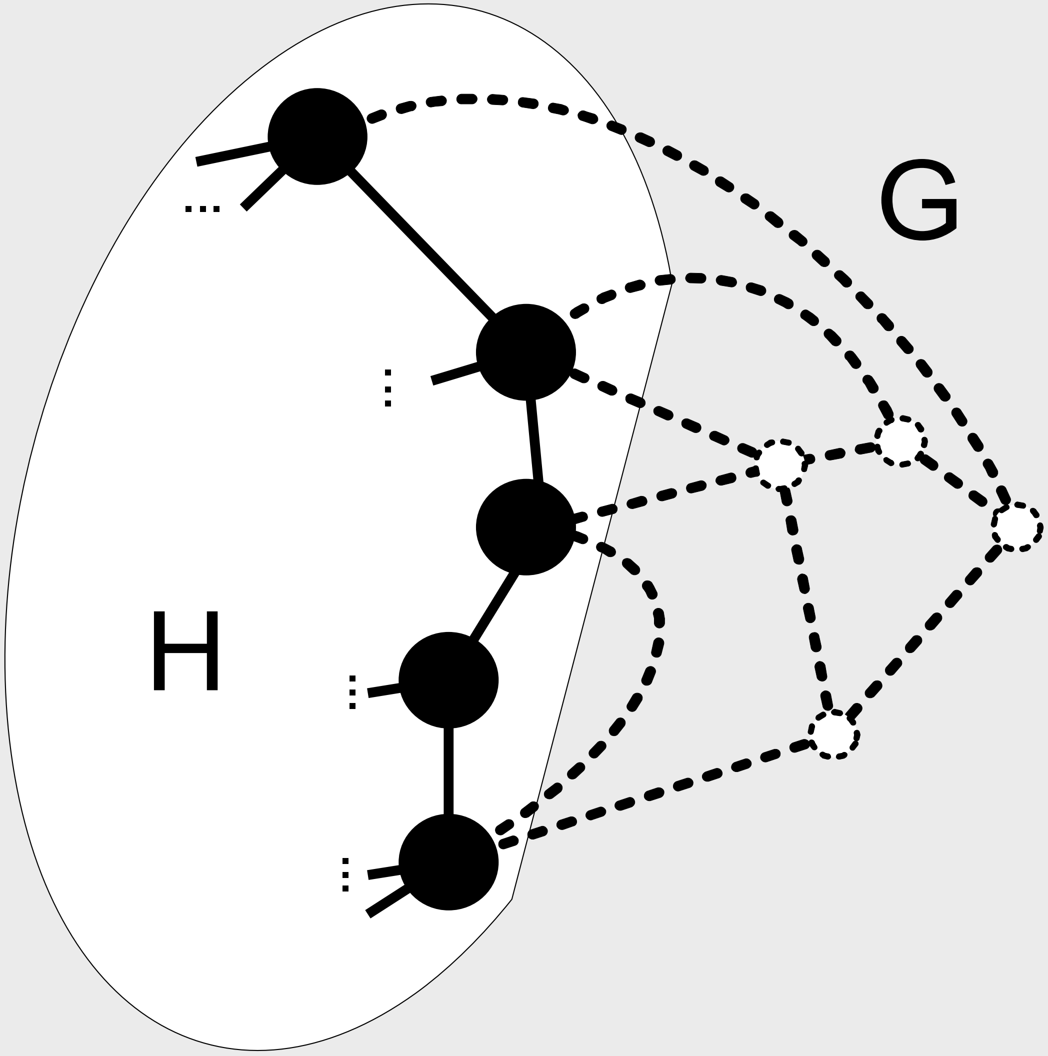
Thm. Let $H \subset G$ with G and *smooth*(H) being 3-connected. Then there is a BG-path in G that can be added to H.

Proof:

- $H = smooth(H)$
- $H \neq smooth(H)$

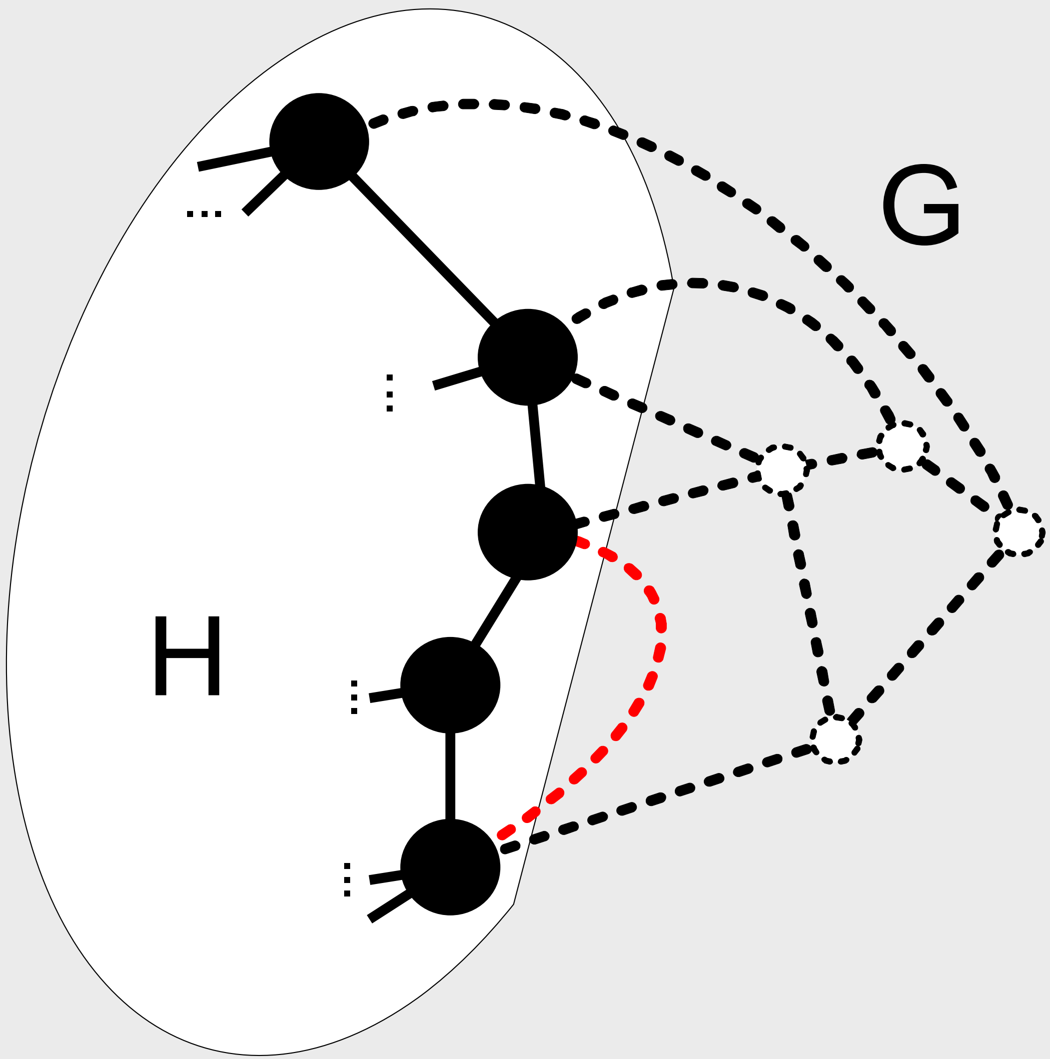
Existence Result

- *H = smooth(H)*
  Only real nodes in *H.*



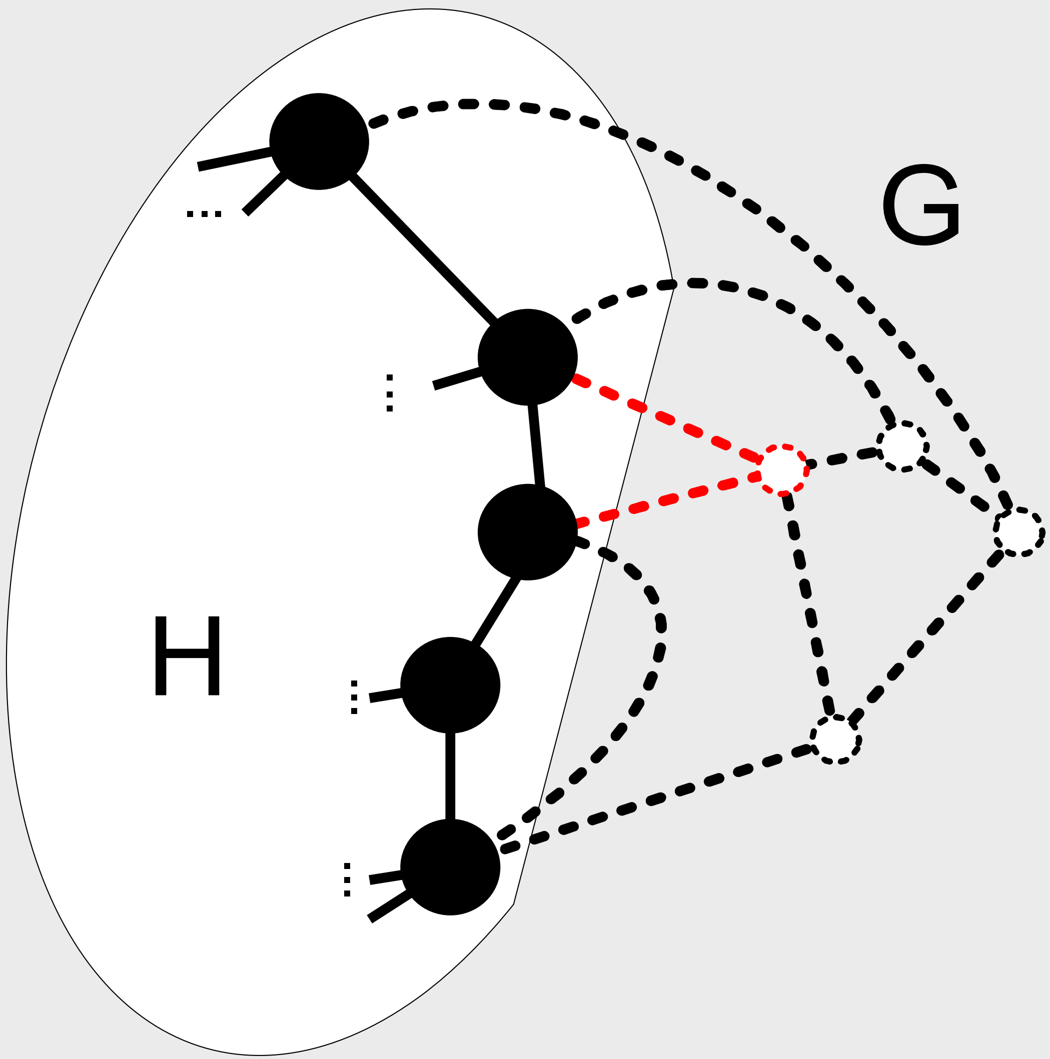
Existence Result

- *H = smooth(H)*
  Only real nodes in *H.*



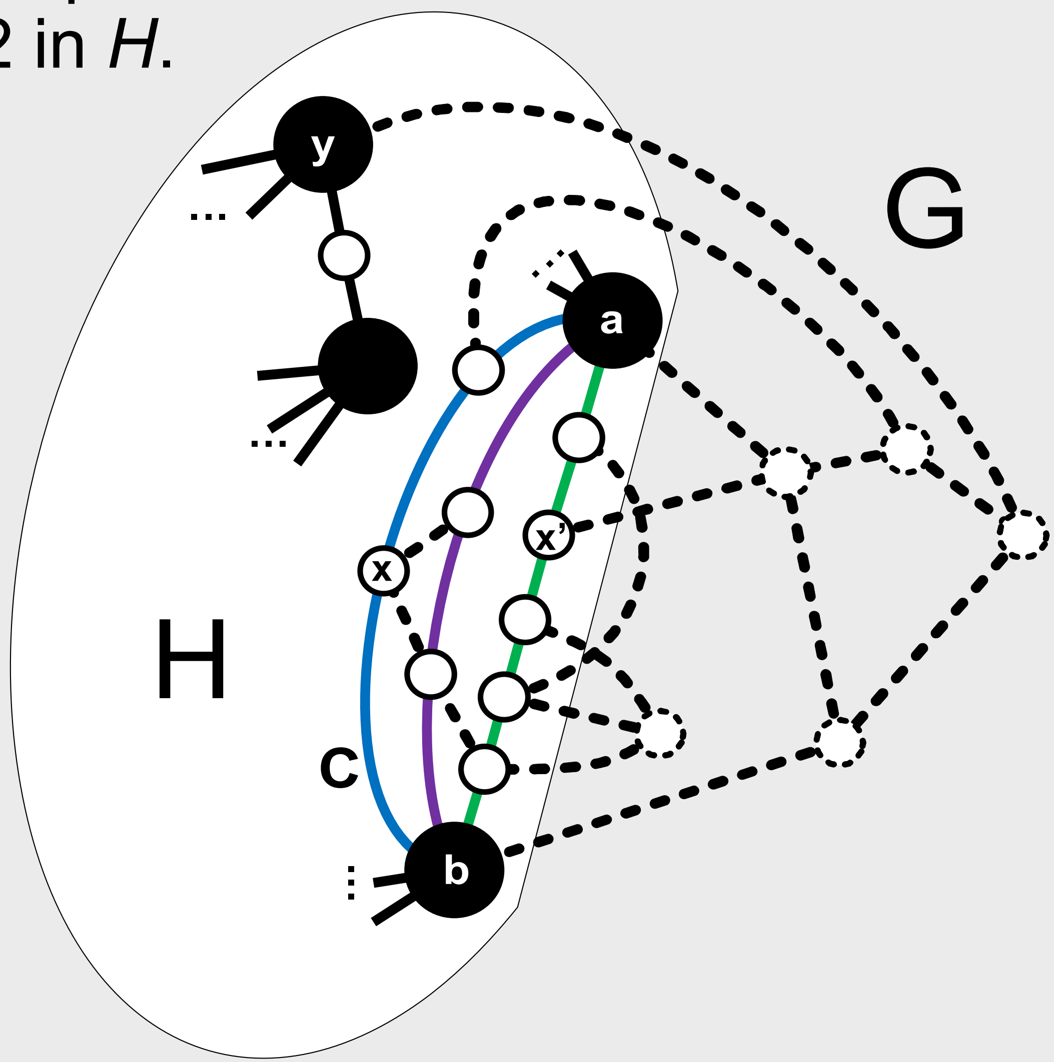
Existence Result

- *H = smooth(H)*
  Only real nodes in *H.*



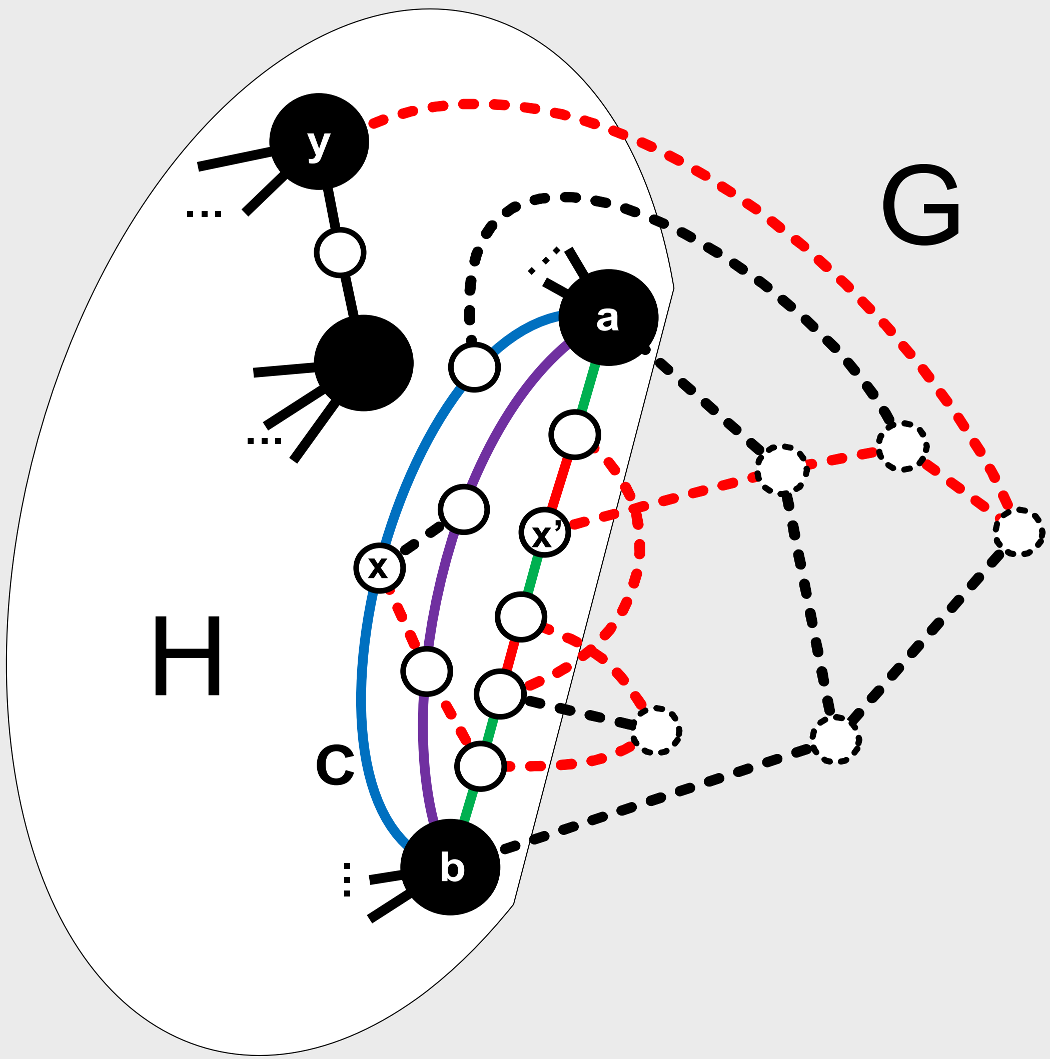
Existence Result

- *H ≠ smooth(H)*
  Some BG-path C=a→b in *H* contains a node *x* having degree 2 in *H*.



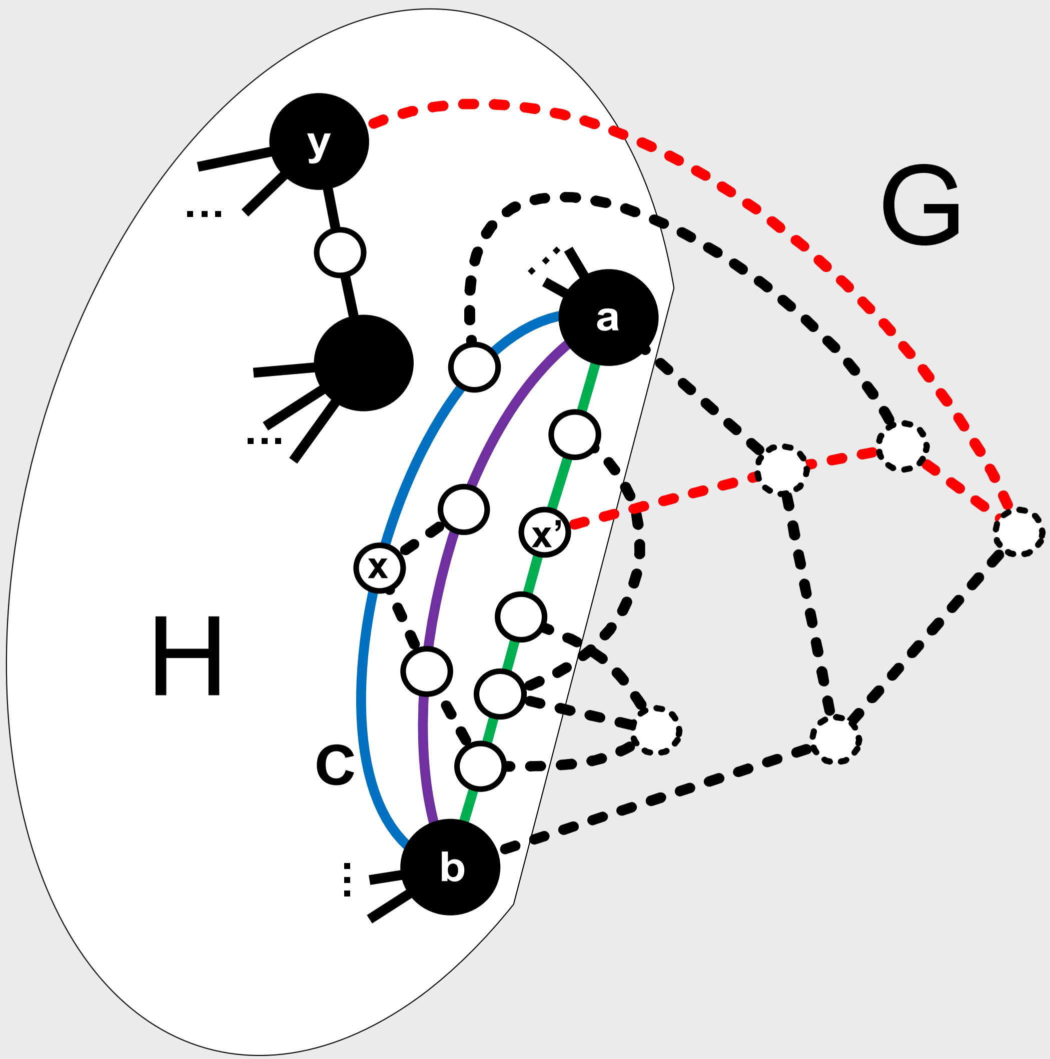
Existence Result

- Then there is a path to a node that is neither in C nor in a parallel BG-path.



Existence Result

- Take x' as the last node being in C or a parallel BG-path.



Existence Result

<u>Corollary</u>
Let $H \subseteq G$ with *smooth*(H) being 3-connected. Then

G is 3-connected
$\Leftrightarrow \exists$ construction sequence from *smooth(H)* to G

Existence Result

<u>Corollary</u>

Let $H \subseteq G$ with *smooth*(H) being 3-connected. Then

might be non-basic

G is 3-connected
$\Leftrightarrow \exists$ construction sequence from *smooth(H)* to G

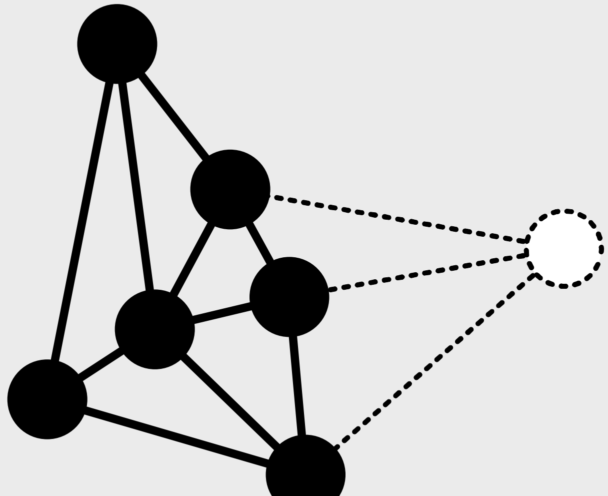
Existence Result

<u>Corollary</u>

Let $H \subseteq G$ with *smooth*(H) being 3-connected. Then

might be non-basic

G is 3-connected
$\Leftrightarrow \exists$ construction sequence from *smooth(H)* to G
$\Leftrightarrow \exists$ basic construction sequence from *smooth(H)* to G using the additional operation *Expand*



Expand
(attach degree-3 node, preserves 3-connectedness with Menger)

Outline

1. Definitions

2. Existence Results

3. Algorithm

4. Testing 3-Connectedness

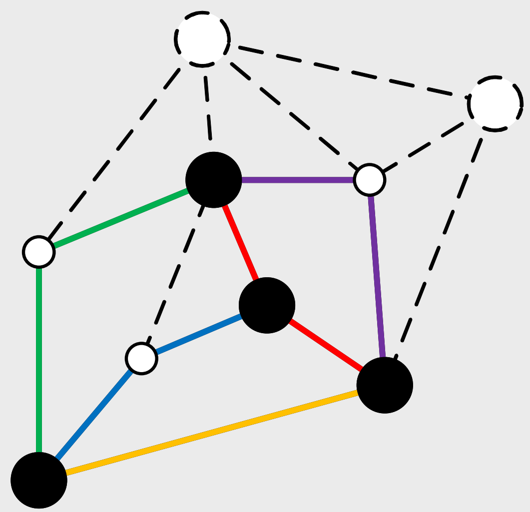
Computing construction sequences

Let *H* be given. How to compute the (possibly non-basic) sequence?

$O(m^3)$ by trying to remove every edge not in *H* and checking the graph on 3-connectedness

$O(n^3)$ by preprocessing that reduces the graph to one with $O(n)$ edges (Nagamochi, Ibaraki '92)

$O(n^2)$ here



$O(n+m)$? (open even for H being a K_4-subdivision)

Outline

1. Definitions

2. Existence Results

3. Algorithm

4. Testing 3-Connectedness

Testing 3-connectedness

Hopcroft & Tarjan '73:

Test on 3-connectedness in O(n+m)

- difficult to understand / implement
- *G* not 3-connected: returns separation pair (easy to check)
- *G* 3-connected: returns no certificate

Testing 3-connectedness

Hopcroft & Tarjan '73:

Test on 3-connectedness in O(n+m)

- difficult to understand / implement
- *G* not 3-connected: returns separation pair (easy to check)
- *G* 3-connected: returns no certificate

Alternative approach with construction sequences:

- Find any K_4-subdivision in *G* in O(n)
- Find construction sequence

Testing 3-connectedness

Hopcroft & Tarjan '73:

Test on 3-connectedness in O(n+m)

- difficult to understand / implement
- *G* not 3-connected: returns separation pair (easy to check)
- *G* 3-connected: returns no certificate

Alternative approach with construction sequences:

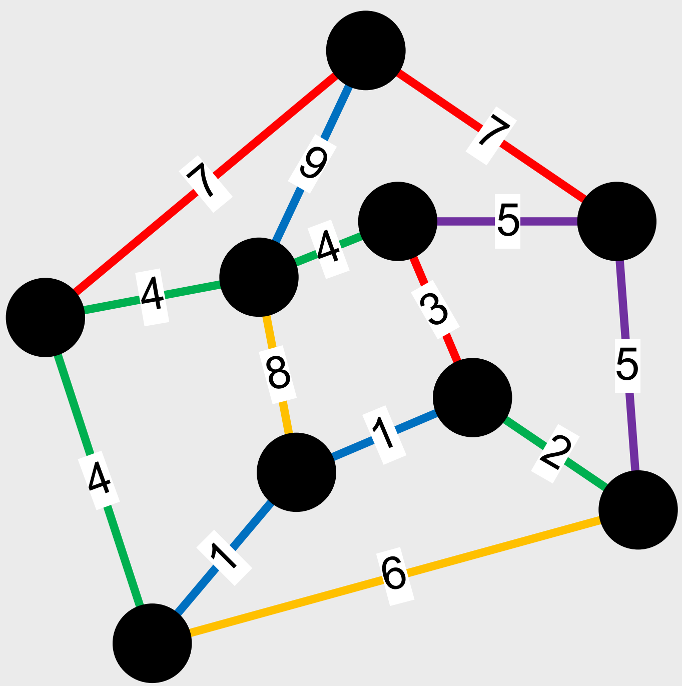
- Find any K_4-subdivision in *G* in O(n)
- Find construction sequence

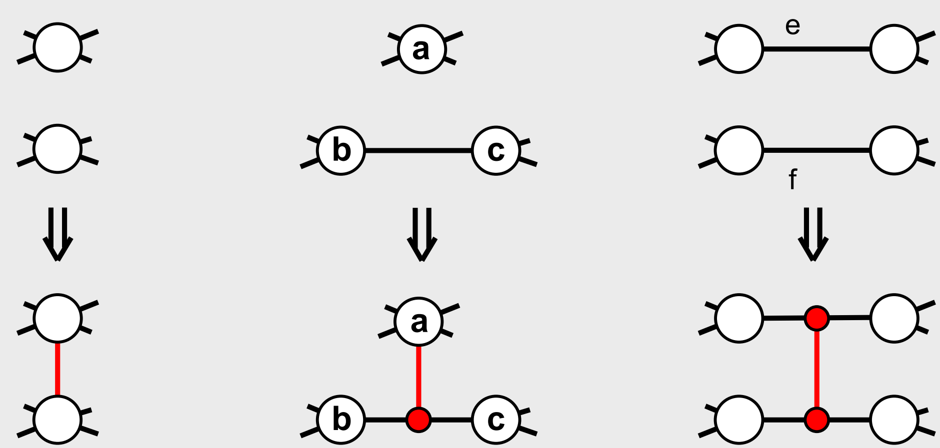
Test on 3-connectedness in the same time as finding sequence

- here $O(n^2)$, but simple
- *G* not 3-connected: returns separation pair (easy to check)
- *G* 3-connected: returns construction sequence (easy to check)

Certificate for 3-connectivity

How to validate a construction sequence in linear time





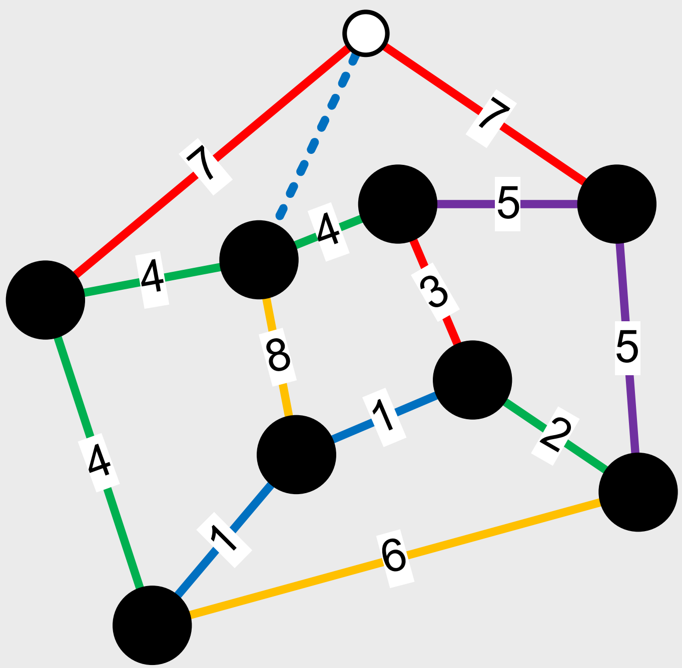
parallel edges allowed

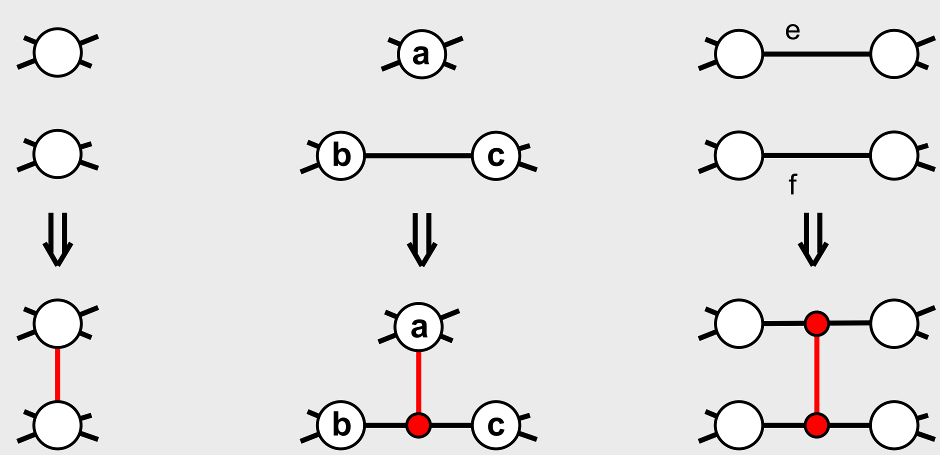
a≠b, a≠c

e≠f,
e and f not parallel

Certificate for 3-connectivity

How to validate a construction sequence in linear time





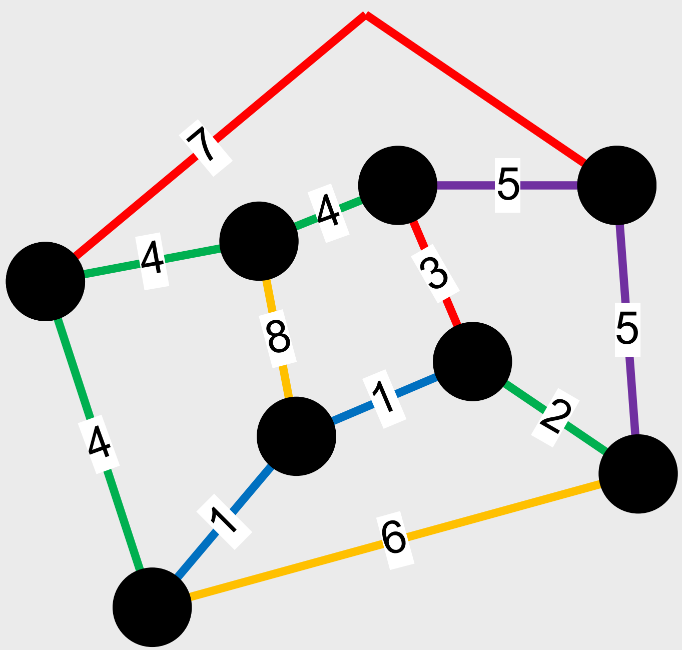
parallel edges allowed

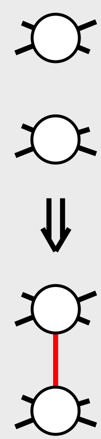
a≠b, a≠c

e≠f,
e and f not parallel

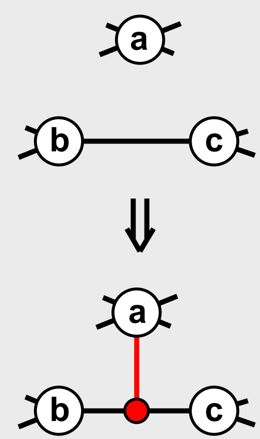
Certificate for 3-connectivity

How to validate a construction sequence in linear time

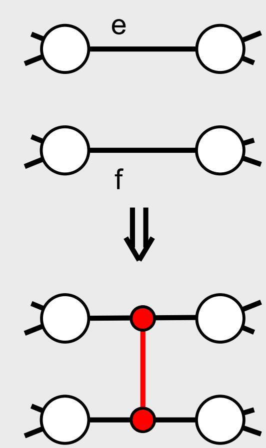




parallel edges allowed



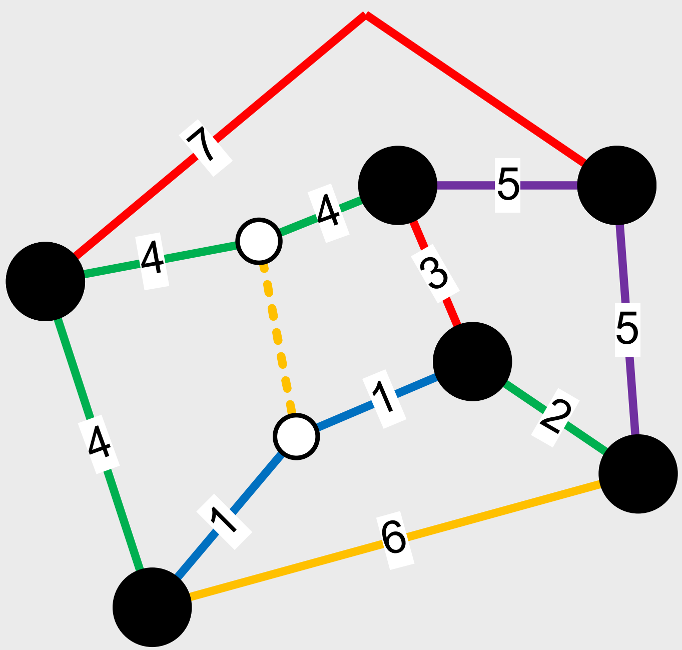
a≠b, a≠c

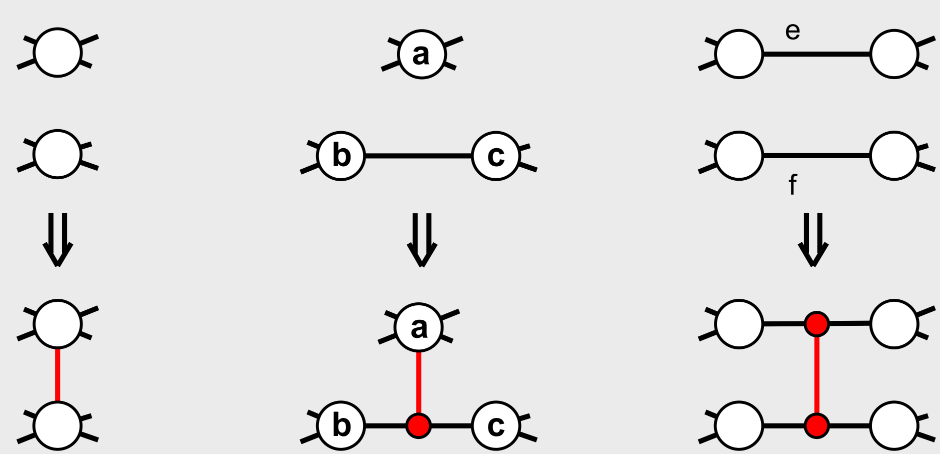


e≠f,
e and f not parallel

Certificate for 3-connectivity

How to validate a construction sequence in linear time





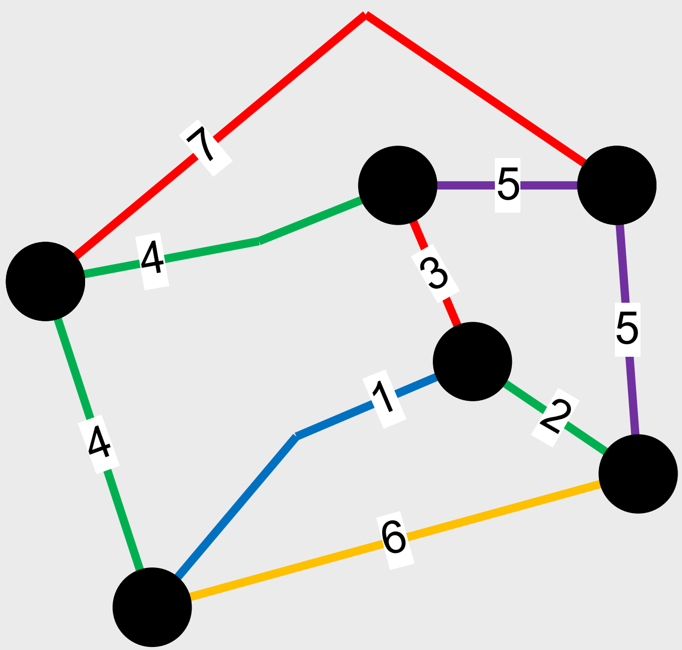
parallel edges allowed

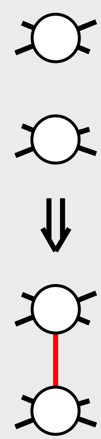
a≠b, a≠c

e≠f,
e and f not parallel

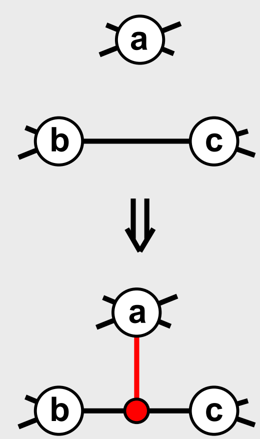
Certificate for 3-connectivity

How to validate a construction sequence in linear time

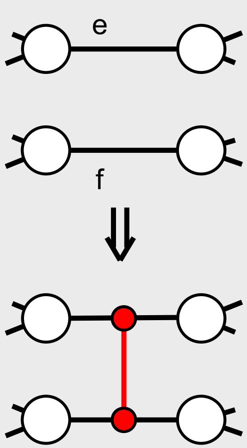




parallel edges allowed



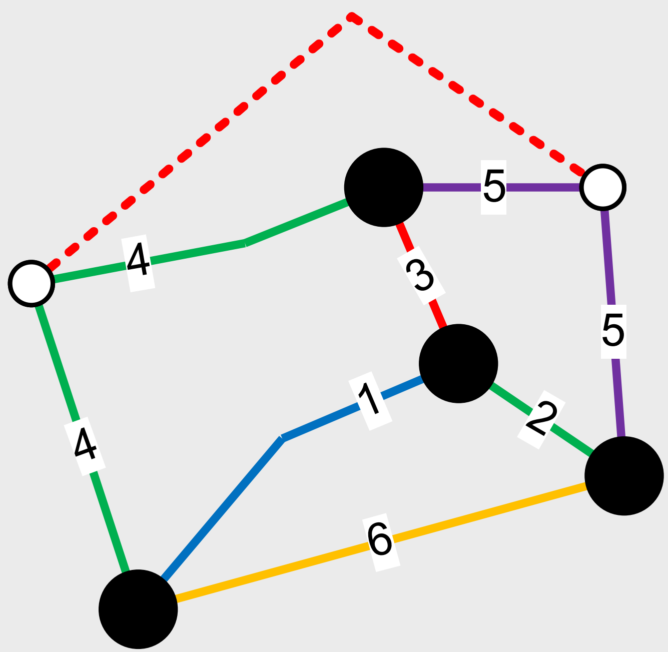
a≠b, a≠c

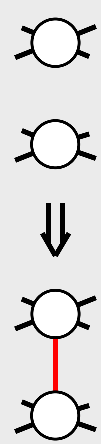


e≠f,
e and f not parallel

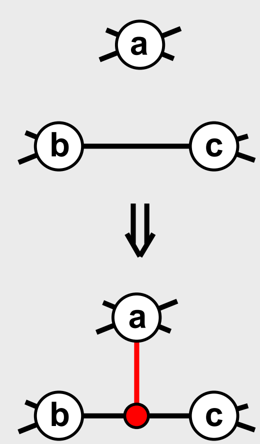
Certificate for 3-connectivity

How to validate a construction sequence in linear time

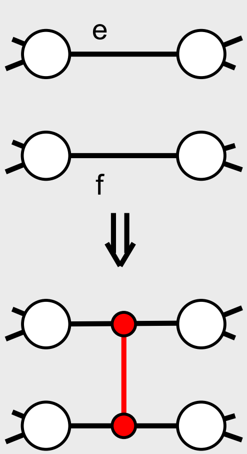




parallel edges allowed



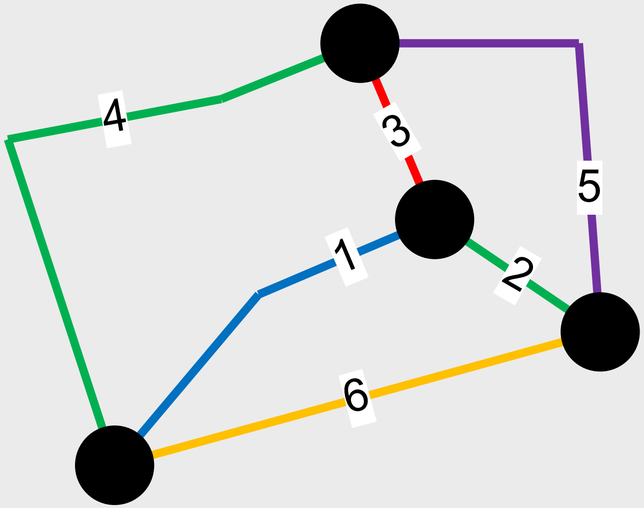
a≠b, a≠c



e≠f,
e and f not parallel

Certificate for 3-connectivity

How to validate a construction sequence in linear time



K_4 easy to check