Coloring perfect graphs by contraction

Benjamin Lévêque

Combinatorial Optimisation team

G-SCOP Laboratory, Grenoble, France

Coloring of the vertices of a graph

Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors

Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors
- Minimum number of colors : $\chi(G)$

Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors
- Minimum number of colors : $\chi(G)$ NP-complete !

Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors
- Minimum number of colors : $\chi(G)$ NP-complete !
- Maximum clique : $\omega(G)$

Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors
- Minimum number of colors : $\chi(G)$ NP-complete !
- Maximum clique : $\omega(G)$

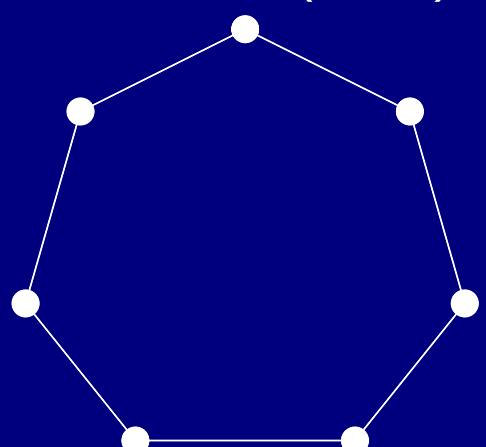
$$\boxed{\chi(G) \geq \omega(G)}$$

Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors
- Minimum number of colors : $\chi(G)$ NP-complete !
- Maximum clique : $\omega(G)$

$$\boxed{\chi(G) \geq \omega(G)}$$

Odd hole (2k+1)

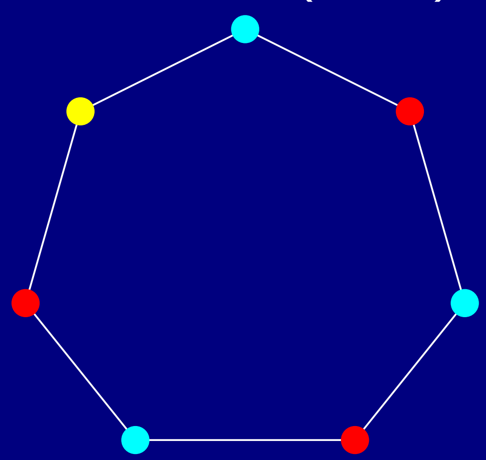


Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors
- Minimum number of colors : $\chi(G)$ NP-complete !
- Maximum clique : $\omega(G)$

$$\boxed{\chi(G) \geq \omega(G)}$$

Odd hole (2k+1)



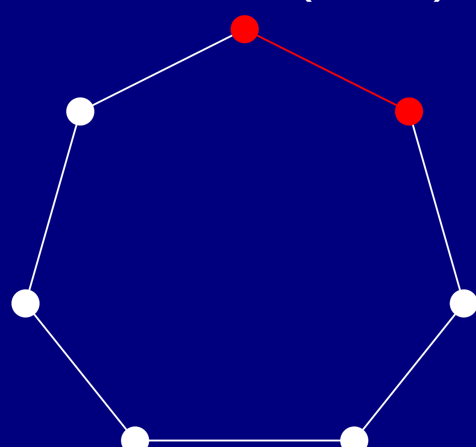
$\chi = 3$

Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors
- Minimum number of colors : $\chi(G)$ NP-complete !
- Maximum clique : $\omega(G)$

$$\boxed{\chi(G) \geq \omega(G)}$$

Odd hole (2k+1)



$\chi = 3$

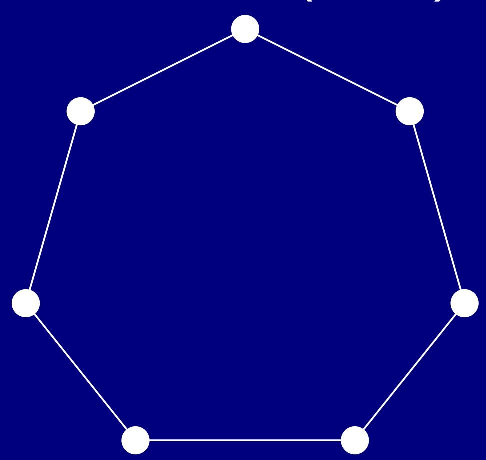
$\omega = 2$

Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors
- Minimum number of colors : $\chi(G)$ NP-complete !
- Maximum clique : $\omega(G)$

$$\boxed{\chi(G) \geq \omega(G)}$$

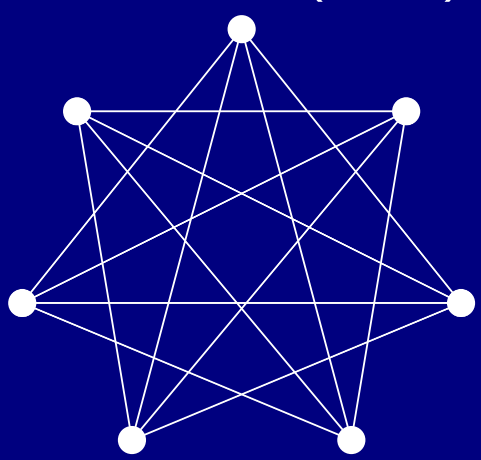
Odd hole (2k+1)



$\chi = 3$

$\omega = 2$

Odd antihole (2k+1)

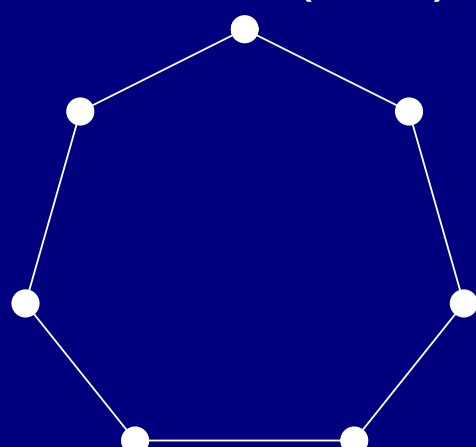


Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors
- Minimum number of colors : $\chi(G)$ NP-complete !
- Maximum clique : $\omega(G)$

$$\boxed{\chi(G) \geq \omega(G)}$$

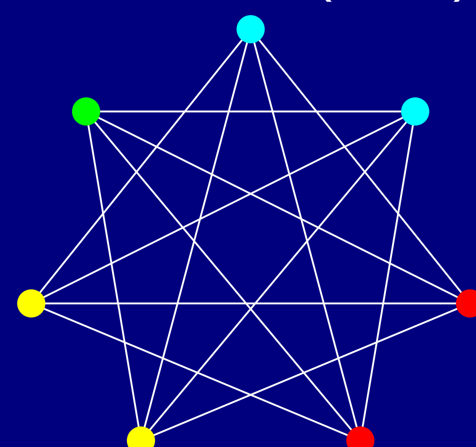
Odd hole (2k+1)



$\chi = 3$

$\omega = 2$

Odd antihole (2k+1)



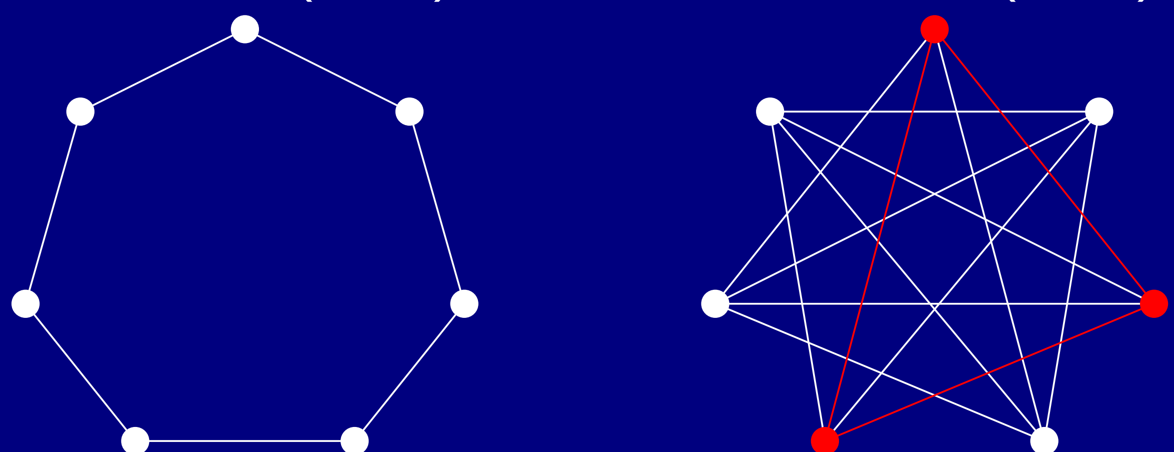
$\chi = k + 1$

Coloring of the vertices of a graph

- two adjacent vertices receive two distinct colors
- Minimum number of colors : $\chi(G)$ NP-complete !
- Maximum clique : $\omega(G)$

$$\boxed{\chi(G) \geq \omega(G)}$$

Odd hole (2k+1) Odd antihole (2k+1)



$\chi = 3$ $\chi = k+1$

$\omega = 2$ $\omega = k$

Perfection

Perfection

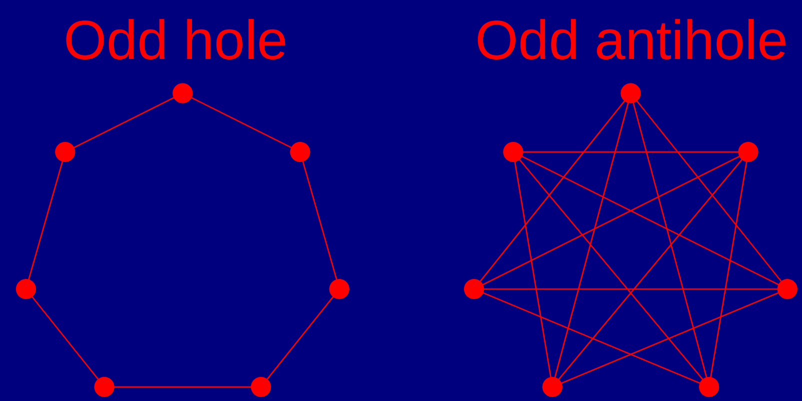
Perfect graphs (Berge - 1960)

For every induced subgraph H of G : $\chi(H) = \omega(H)$

Perfection

Perfect graphs (Berge - 1960)

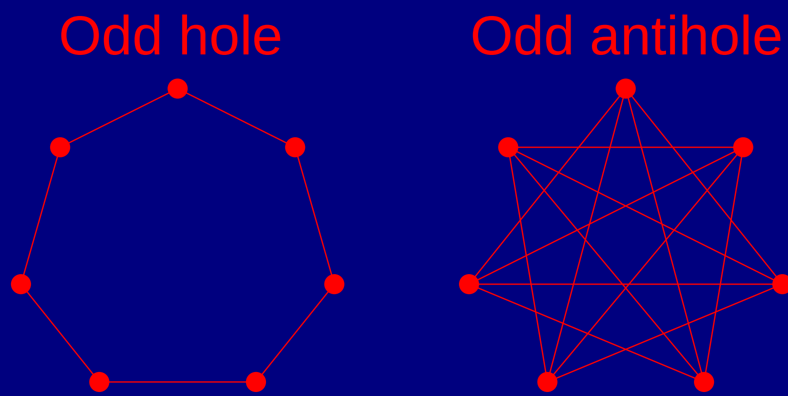
For every induced subgraph H of G : $\chi(H) = \omega(H)$



Perfection

Perfect graphs (Berge - 1960)

For every induced subgraph H of G : $\chi(H) = \omega(H)$



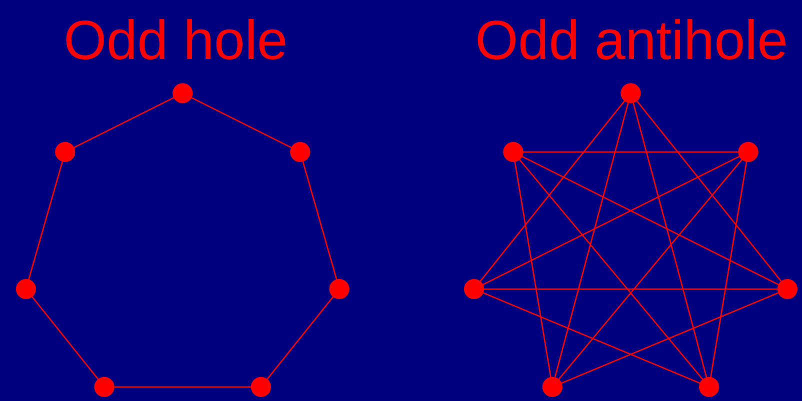
Conjecture (Berge - 1960)

G is perfect iff it contains no odd hole and no odd antihole

Perfection

Perfect graphs (Berge - 1960)

For every induced subgraph H of G : $\chi(H) = \omega(H)$



Conjecture (Berge - 1960) **Theorem** (Chud., Rob., Seym., Thom. - 2006)

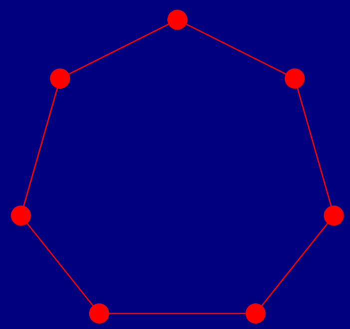
G is perfect iff it contains no odd hole and no odd antihole

Perfection

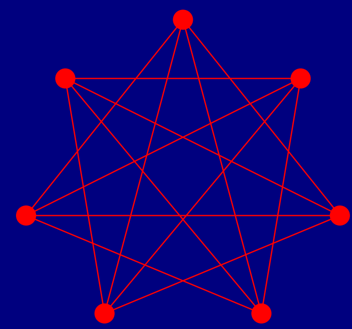
Perfect graphs (Berge - 1960)

For every induced subgraph H of G : $\chi(H) = \omega(H)$

Odd hole



Odd antihole



Conjecture (Berge - 1960) **Theorem** (Chud., Rob., Seym., Thom. - 2006)

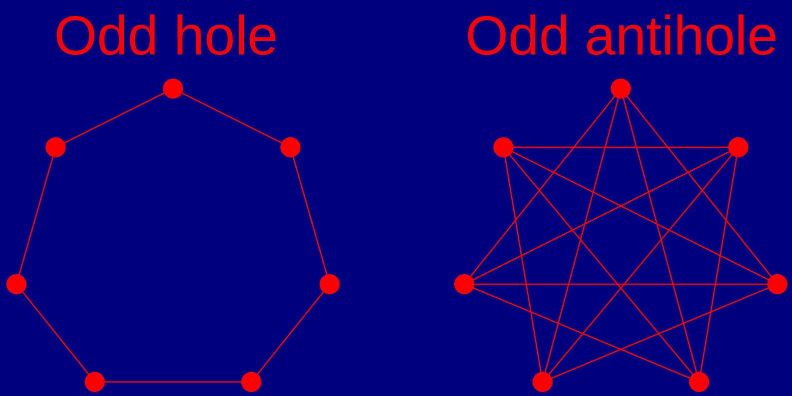
G is perfect iff it contains no odd hole and no odd antihole

Recognition algorithm

Perfection

Perfect graphs (Berge - 1960)

For every induced subgraph H of G : $\chi(H) = \omega(H)$



Conjecture (Berge - 1960) **Theorem** (Chud., Rob., Seym., Thom. - 2006)

G is perfect iff it contains no odd hole and no odd antihole

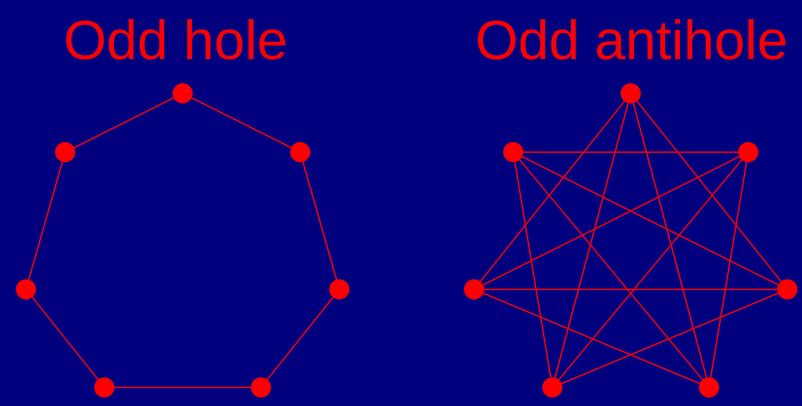
Recognition algorithm (Chudnovsky, Cornuéjols, Liu, Seymour, Vušković - 2005)

Complexity $\mathcal{O}(n^9)$

Perfection

Perfect graphs (Berge - 1960)

For every induced subgraph H of G : $\chi(H) = \omega(H)$



Conjecture (Berge - 1960) **Theorem** (Chud., Rob., Seym., Thom. - 2006)

G is perfect iff it contains no odd hole and no odd antihole

Recognition algorithm (Chudnovsky, Cornuéjols, Liu, Seymour, Vušković - 2005)

Complexity $\mathcal{O}(n^9)$

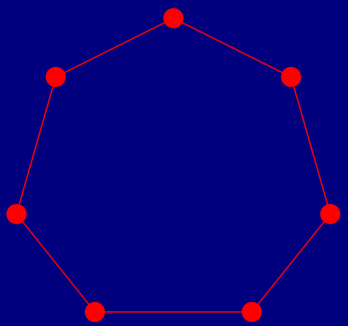
Coloring algorithm (Grötschel, Lovász, Schrijver - 1984)

Ellipsoïd method (Khachiyan - 1979)

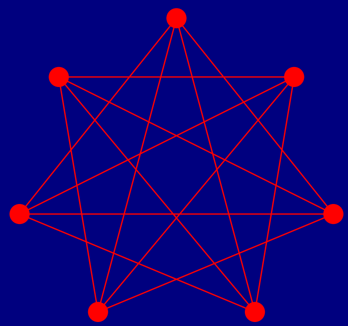
Perfection

Perfect graphs (Berge - 1960)
For every induced subgraph H of G : $\chi(H) = \omega(H)$

Odd hole



Odd antihole



Conjecture (Berge - 1960) **Theorem** (Chud., Rob., Seym., Thom. - 2006)
G is perfect iff it contains no odd hole and no odd antihole

Recognition algorithm (Chudnovsky, Cornuéjols, Liu, Seymour, Vušković - 2005)
Complexity $\mathcal{O}(n^9)$

Coloring algorithm (Grötschel, Lovász, Schrijver - 1984)
Ellipsoïd method (Khachiyan - 1979)

Purely combinatorial coloring algorithm ?

Contraction

Contraction

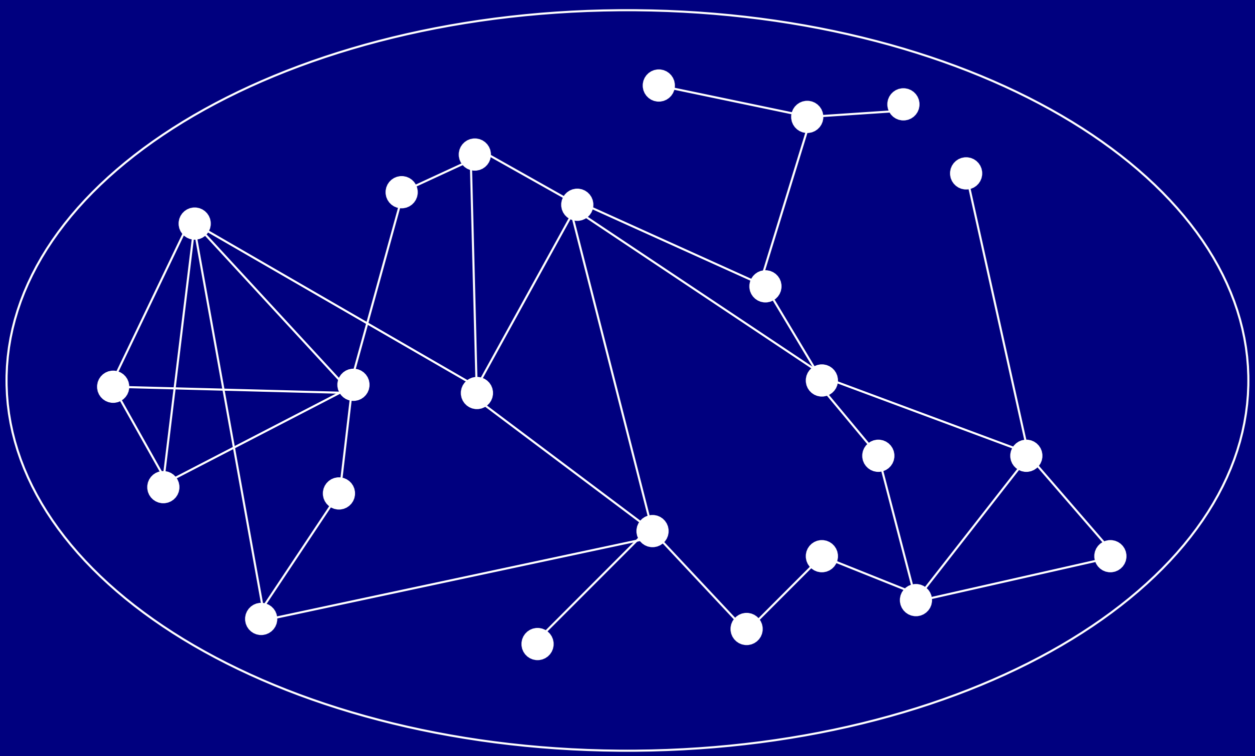
Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length

Contraction

Even pair (Meyniel - 1987)

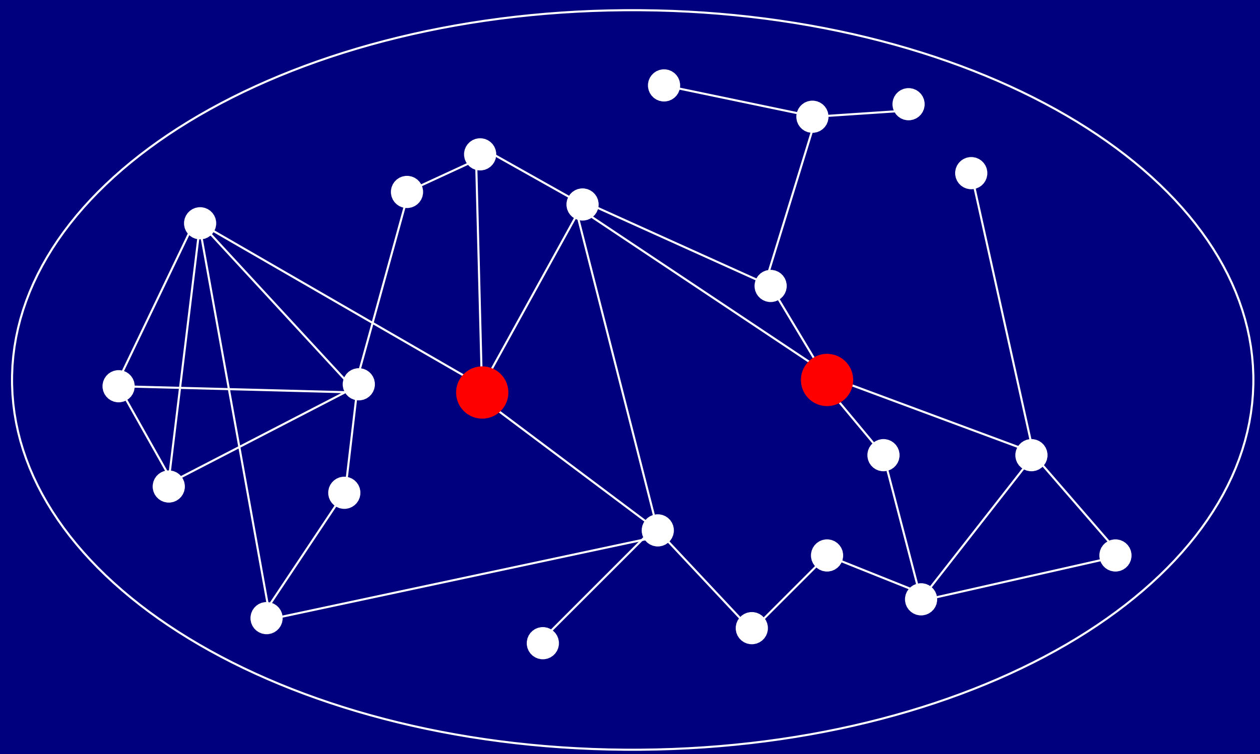
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

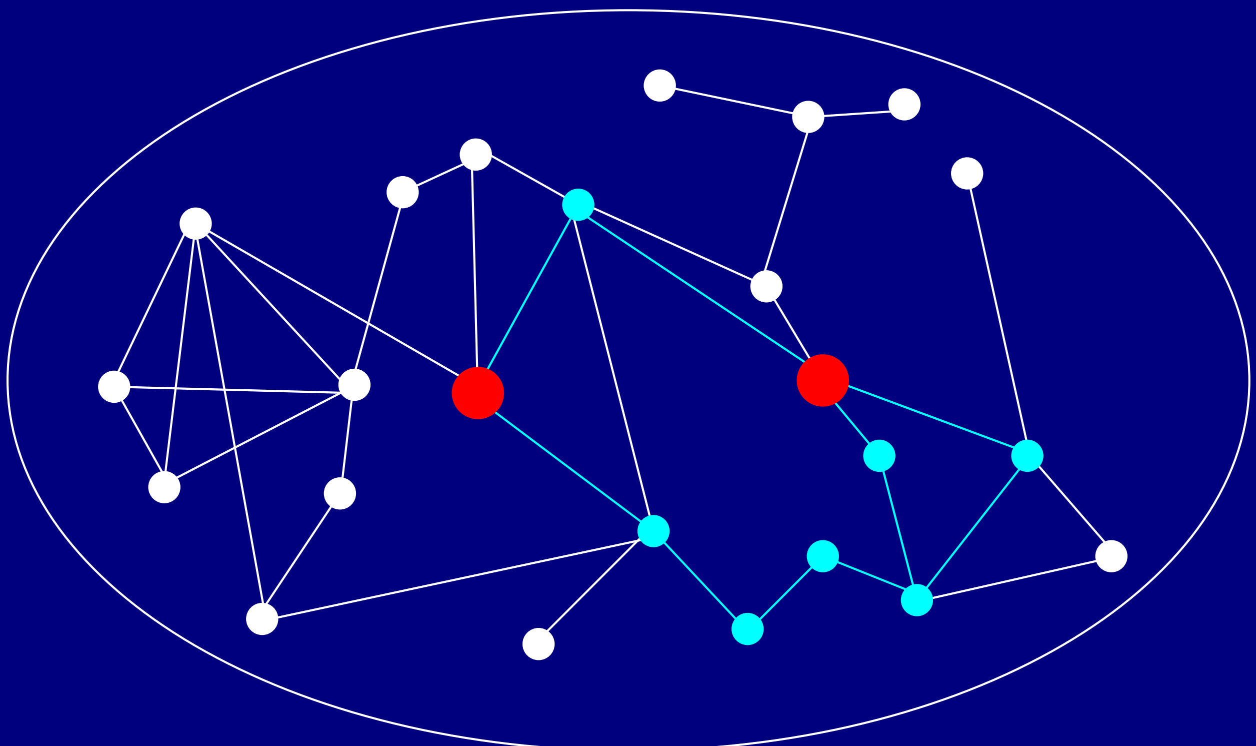
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

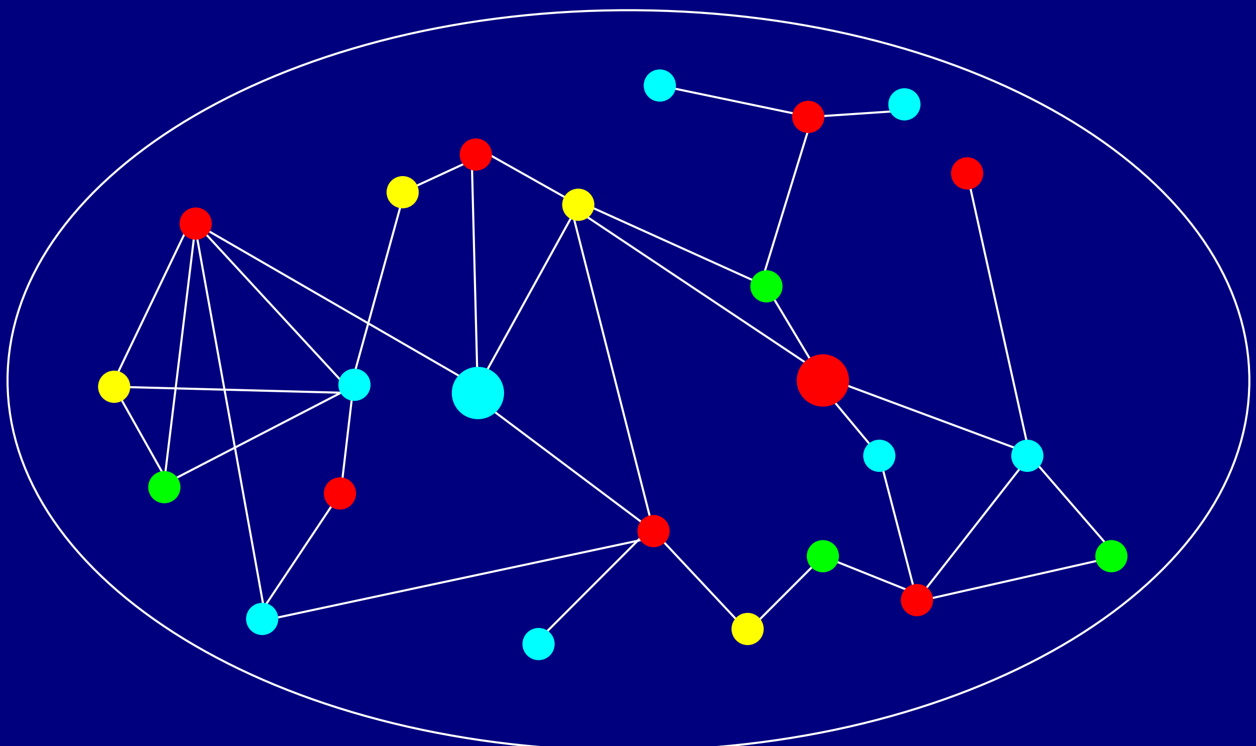
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

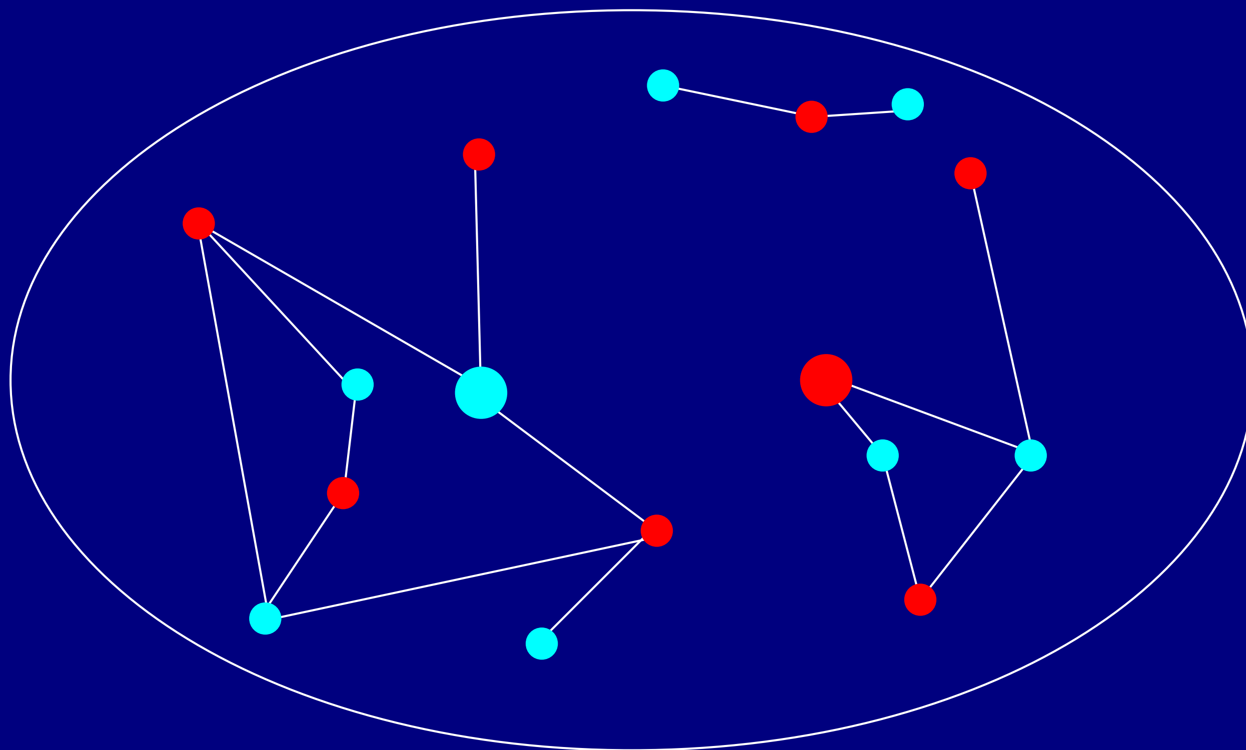
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

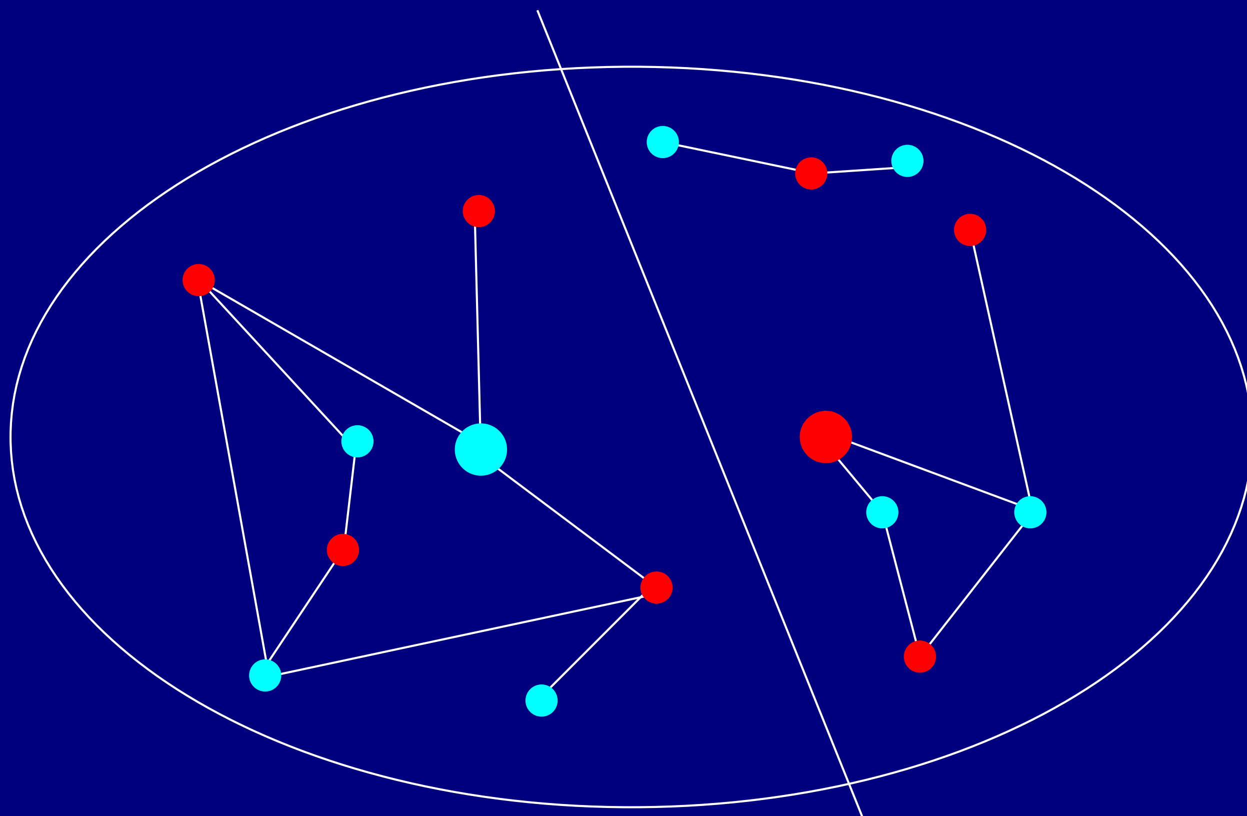
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

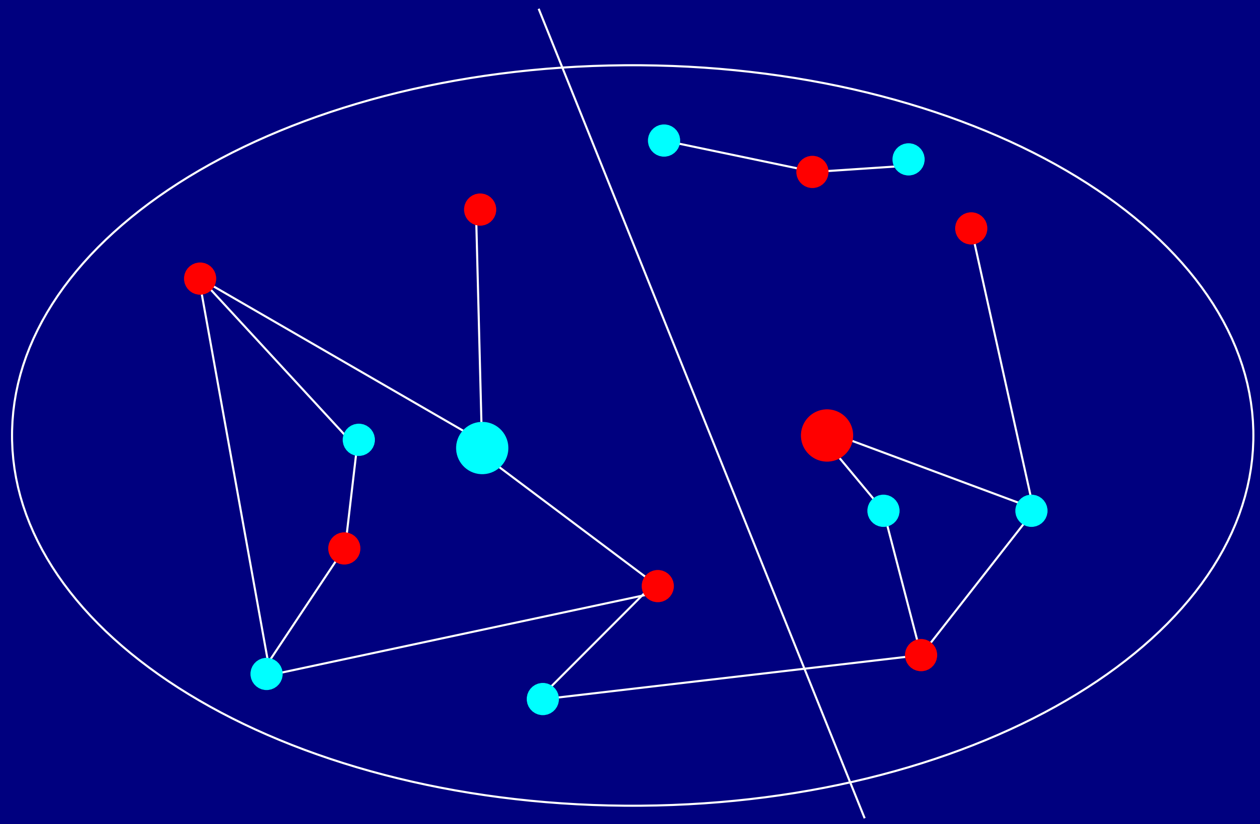
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

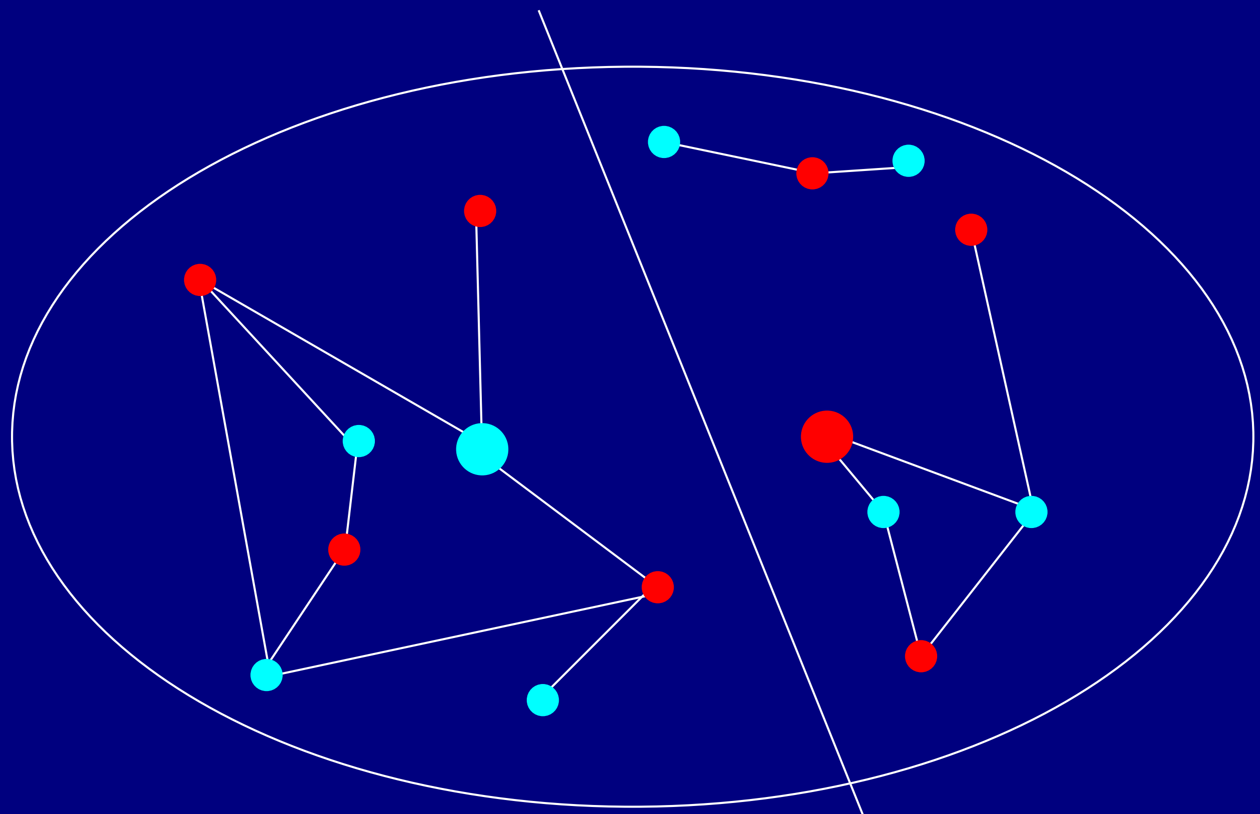
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

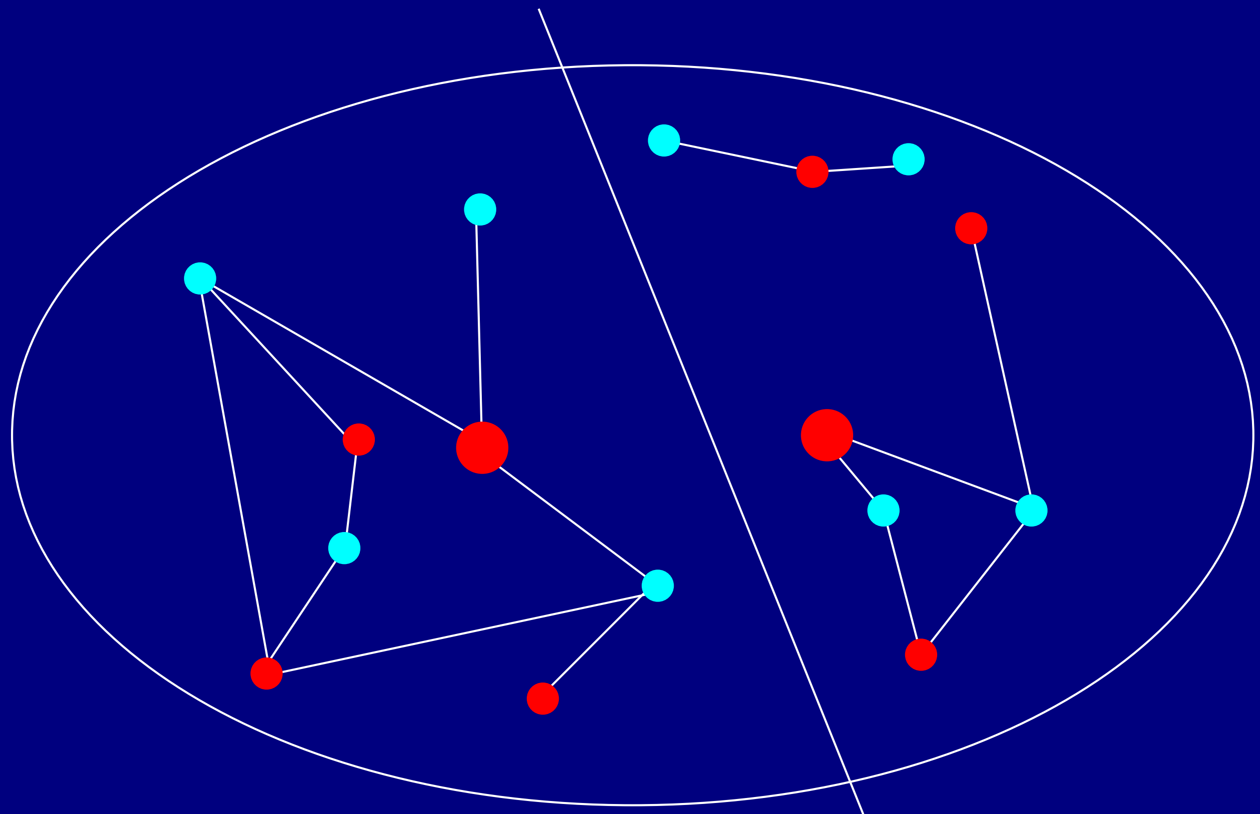
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

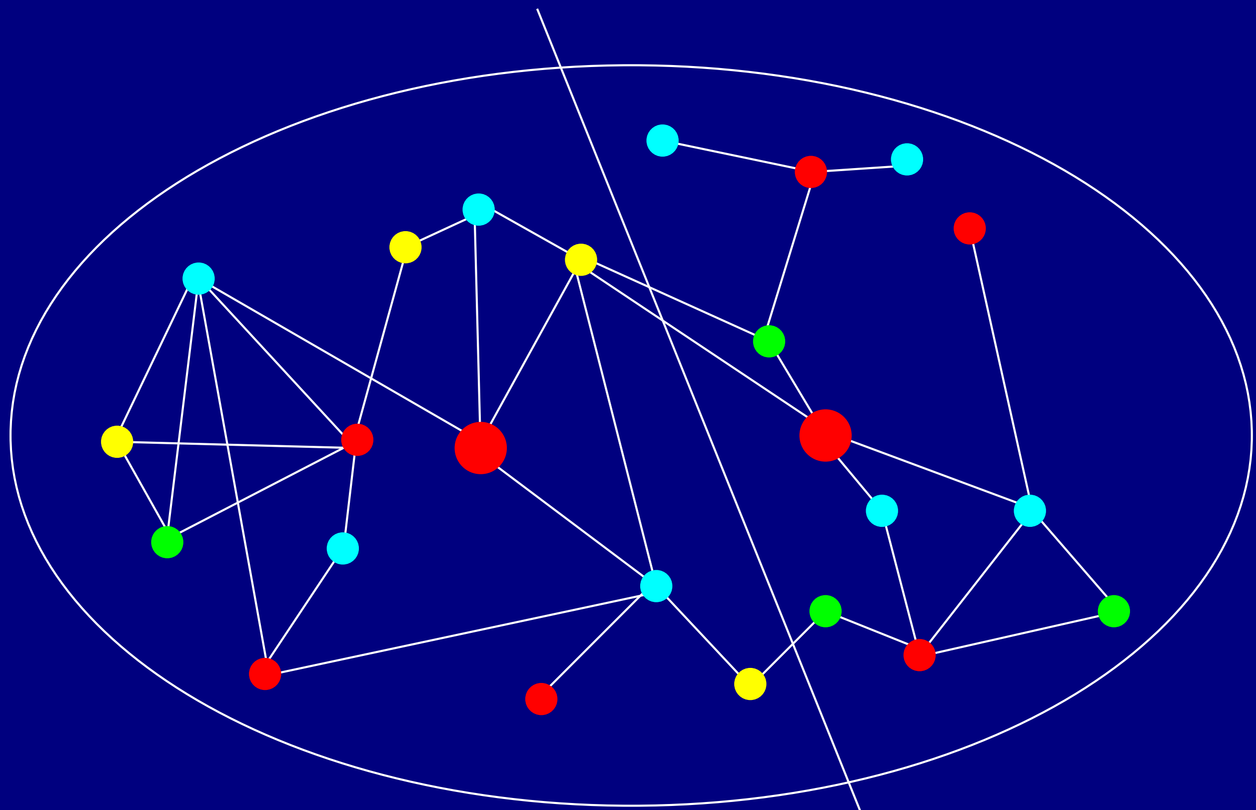
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

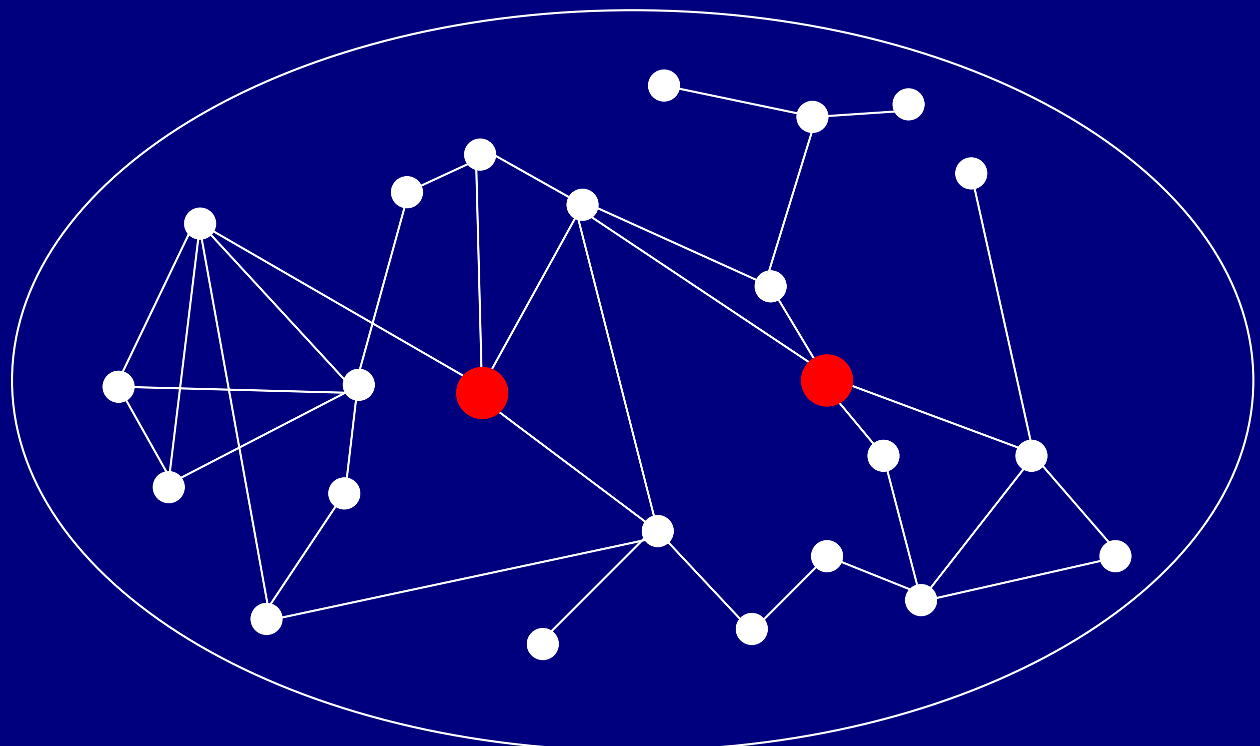
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

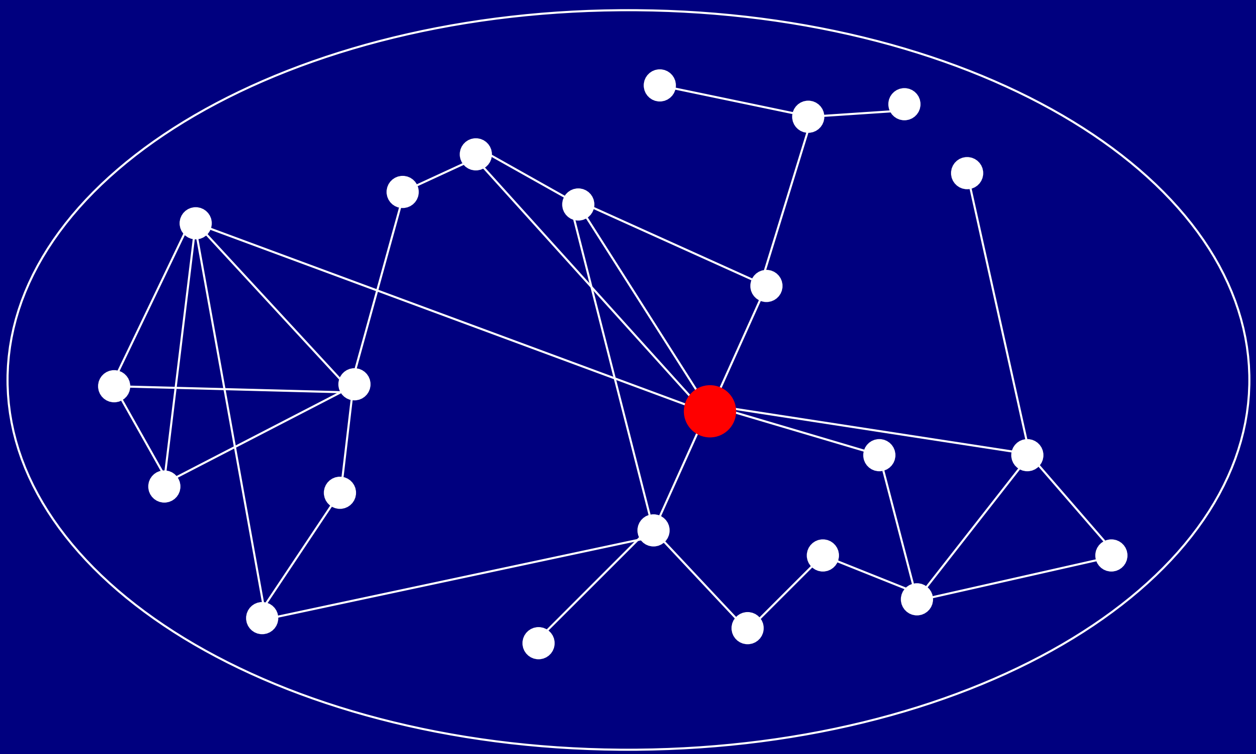
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

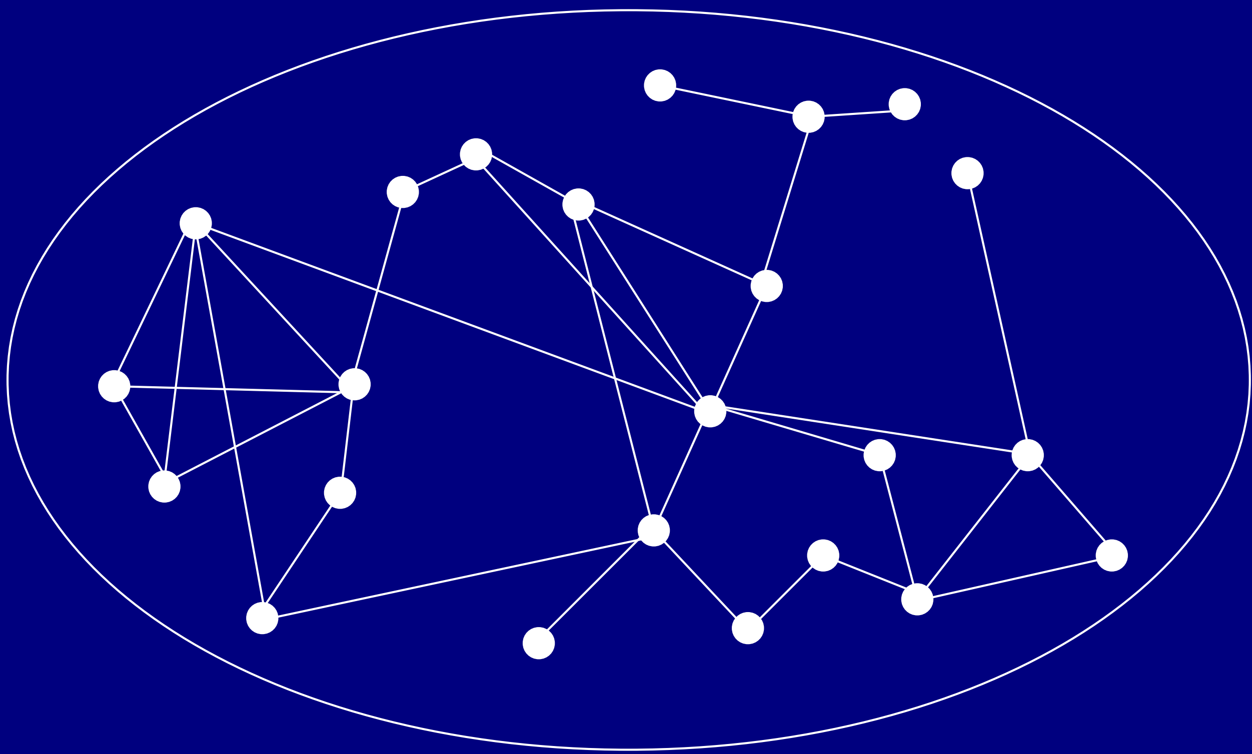
2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length



Contraction

Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length

Theorem (Fonlupt, Uhry - 1982)

Contracting an even pair preserves the value of χ

Contraction

Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length

Theorem (Fonlupt, Uhry - 1982)

Contracting an even pair preserves the value of χ and ω

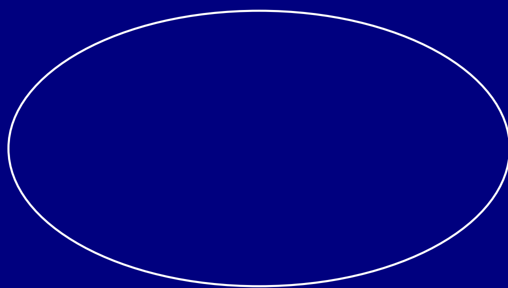
Contraction

Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length

Theorem (Fonlupt, Uhry - 1982)

Contracting an even pair preserves the value of χ and ω



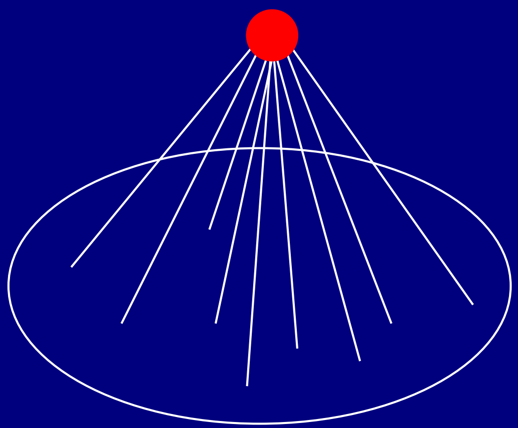
Contraction

Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length

Theorem (Fonlupt, Uhry - 1982)

Contracting an even pair preserves the value of χ and ω



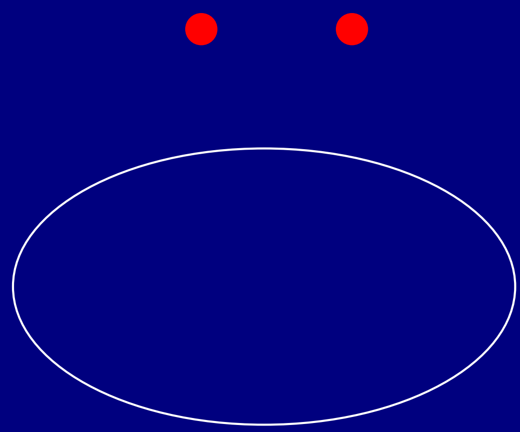
Contraction

Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length

Theorem (Fonlupt, Uhry - 1982)

Contracting an even pair preserves the value of χ and ω



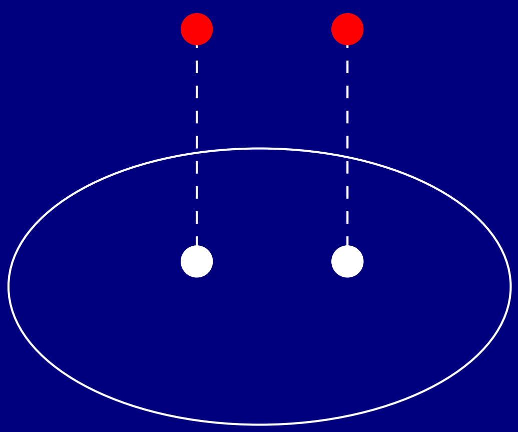
Contraction

Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length

Theorem (Fonlupt, Uhry - 1982)

Contracting an even pair preserves the value of χ and ω



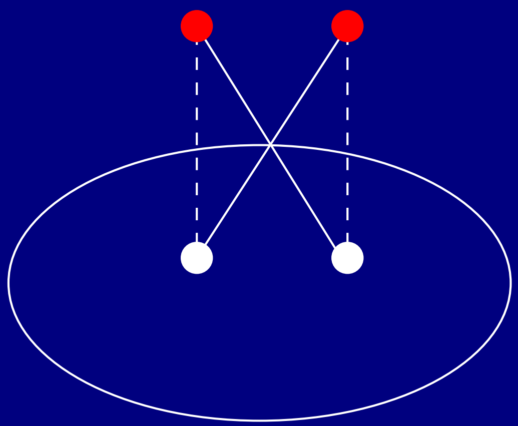
Contraction

Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length

Theorem (Fonlupt, Uhry - 1982)

Contracting an even pair preserves the value of χ and ω



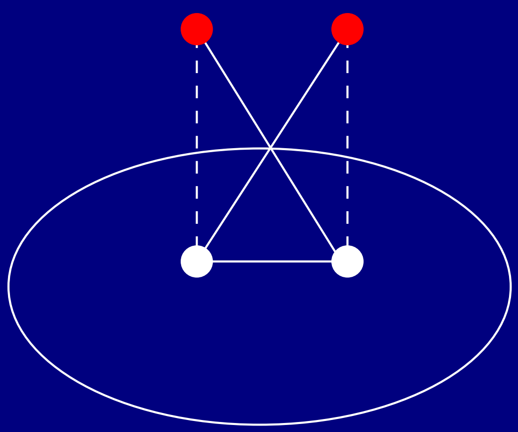
Contraction

Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length

Theorem (Fonlupt, Uhry - 1982)

Contracting an even pair preserves the value of χ and ω



Contraction

Even pair (Meyniel - 1987)

2 vertices s.t. every chordless path between them has even length

Theorem (Fonlupt, Uhry - 1982)

Contracting an even pair preserves the value of χ and ω

Even contractile (Bertschi - 1990)

Can be turned into a clique by a sequence of even pair contractions

Contraction

Even pair (Meyniel - 1987)
2 vertices s.t. every chordless path between them has even length

Theorem (Fonlupt, Uhry - 1982)
Contracting an even pair preserves the value of χ and ω

Even contractile (Bertschi - 1990)
Can be turned into a clique by a sequence of even pair contractions

Perfectly contractile (Bertschi - 1990)
Every induced subgraph is even-contractile

Contraction

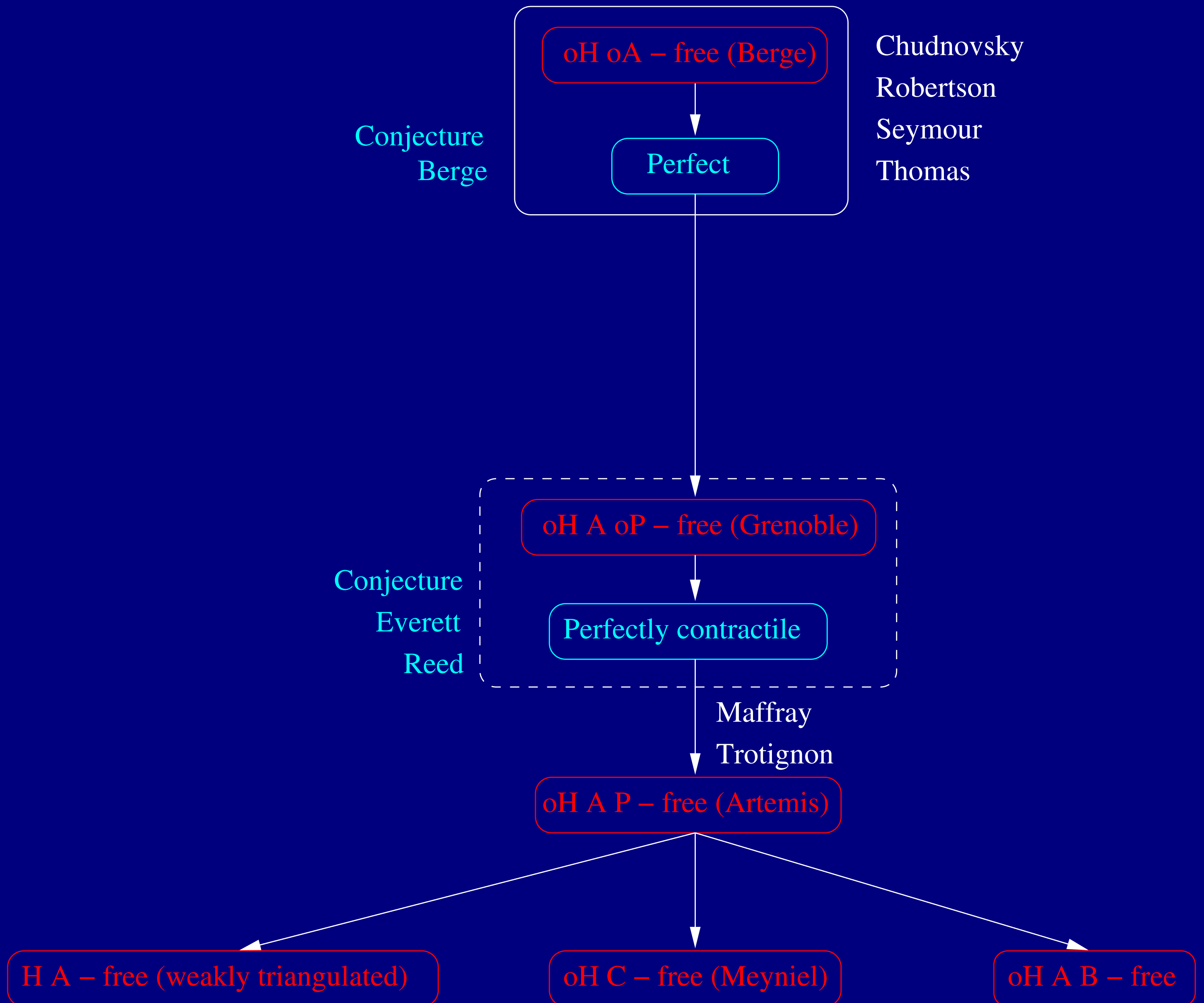
Even pair (Meyniel - 1987)
2 vertices s.t. every chordless path between them has even length

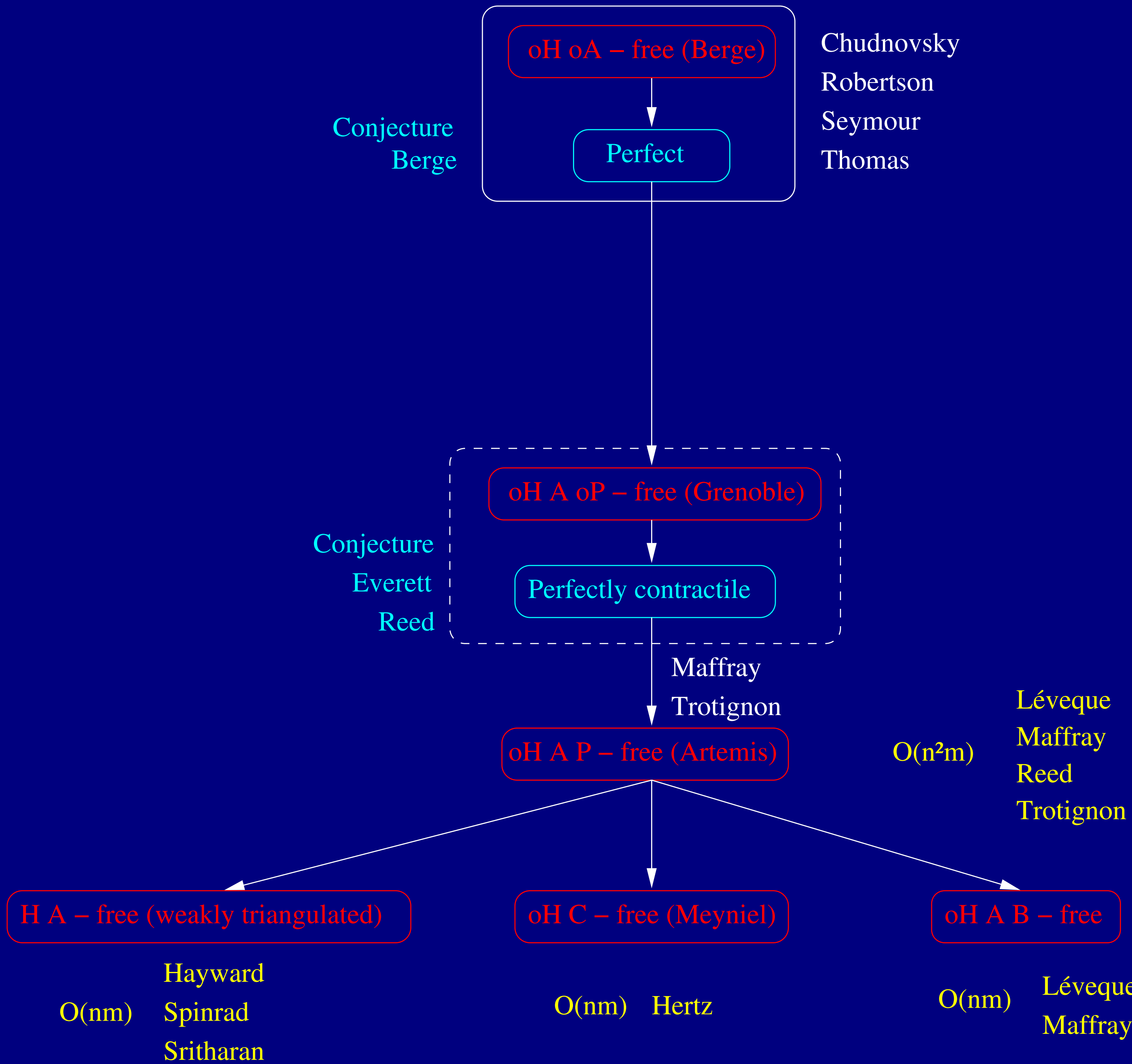
Theorem (Fonlupt, Uhry - 1982)
Contracting an even pair preserves the value of χ and ω

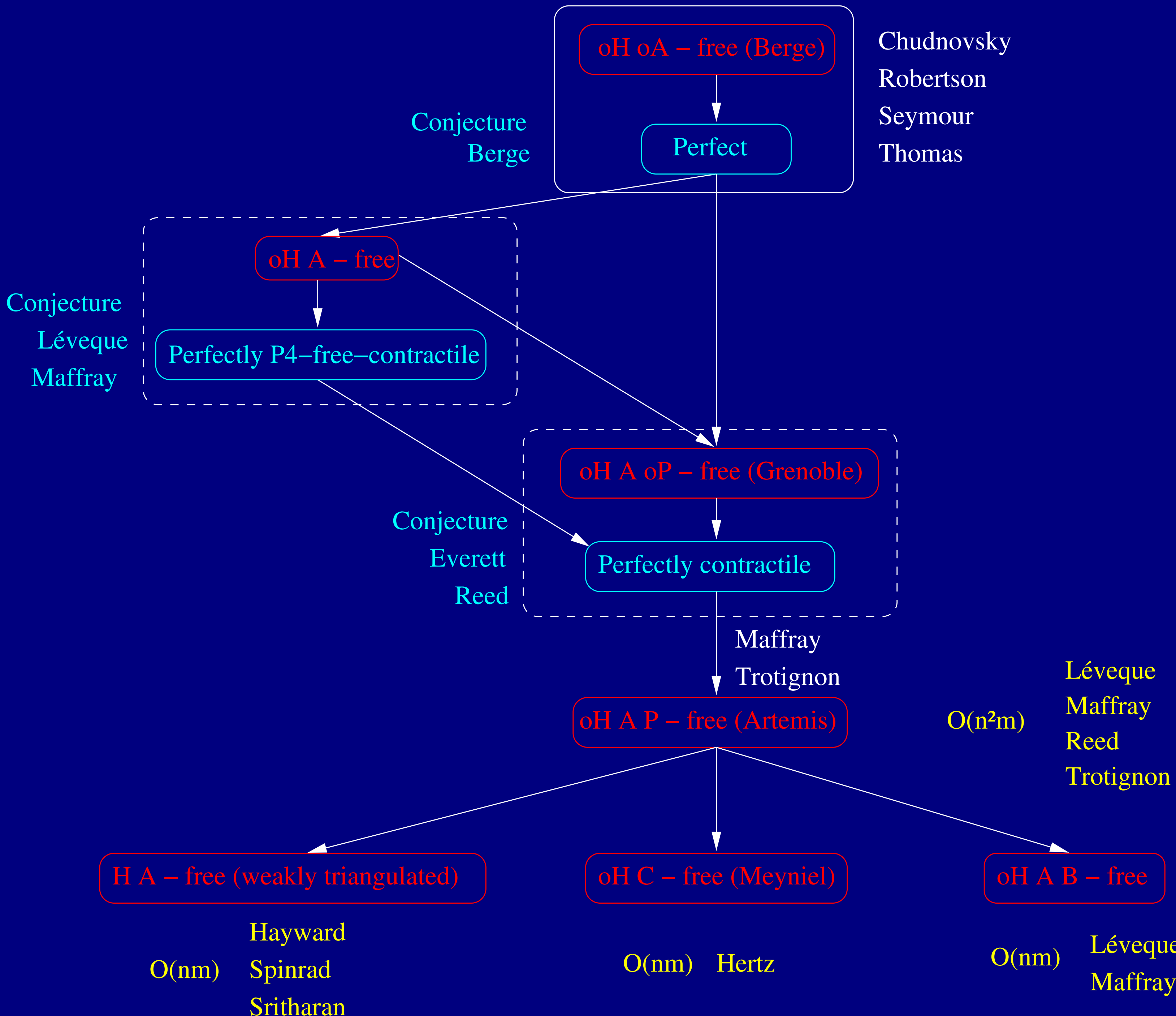
Even contractile (Bertschi - 1990)
Can be turned into a clique by a sequence of even pair contractions

Perfectly contractile (Bertschi - 1990)
Every induced subgraph is even-contractile

Conjecture (Everett, Reed - 1993)
G is perfectly contractile iff it contains no odd hole, no antihole and no odd prism

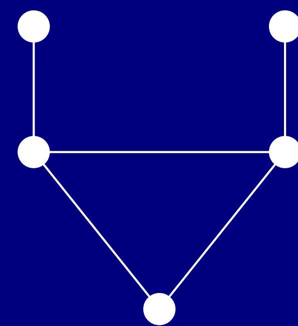

oH oA – free (Berge)
Perfect
Conjecture
Berge
Chudnovsky
Robertson
Seymour
Thomas
oH A oP – free (Grenoble)
Perfectly contractile
Conjecture
Everett
Reed
Maffray
Trotignon
oH A P – free (Artemis)
H A – free (weakly triangulated)
oH C – free (Meyniel)
oH A B – free


oH oA – free (Berge)
Perfect
Conjecture
Berge
Chudnovsky
Robertson
Seymour
Thomas
oH A oP – free (Grenoble)
Perfectly contractile
Conjecture
Everett
Reed
Maffray
Trotignon
oH A P – free (Artemis)
O(n²m)
Léveque
Maffray
Reed
Trotignon
H A – free (weakly triangulated)
oH C – free (Meyniel)
oH A B – free
O(nm)
Hayward
Spinrad
Sritharan
O(nm)
Hertz
O(nm)
Léveque
Maffray

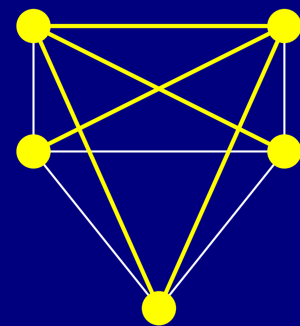

oH oA – free (Berge)
Perfect
Conjecture
Berge
Chudnovsky
Robertson
Seymour
Thomas
oH A – free
Perfectly P4–free–contractile
Conjecture
Léveque
Maffray
oH A oP – free (Grenoble)
Perfectly contractile
Conjecture
Everett
Reed
Maffray
Trotignon
oH A P – free (Artemis)
O(n²m)
Léveque
Maffray
Reed
Trotignon
H A – free (weakly triangulated)
oH C – free (Meyniel)
oH A B – free
O(nm)
Hayward
Spinrad
Sritharan
O(nm)
Hertz
O(nm)
Léveque
Maffray

Bull-free Artemis graphs

Bull-free Artemis graphs



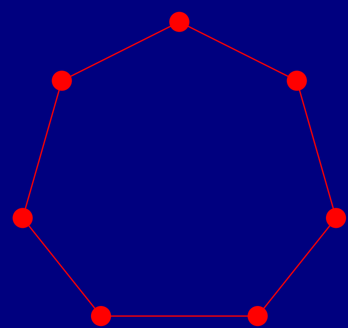
Bull-free Artemis graphs



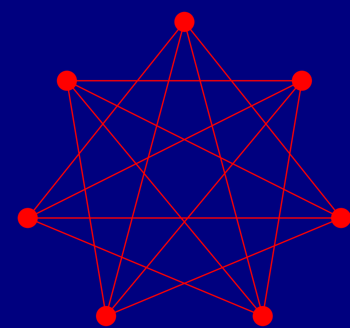
Bull-free Artemis graphs

G is bull-free Artemis iff it contains no odd hole, no antihole and no bull

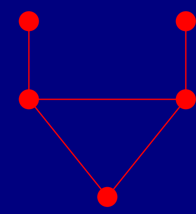
Odd hole



Antihole



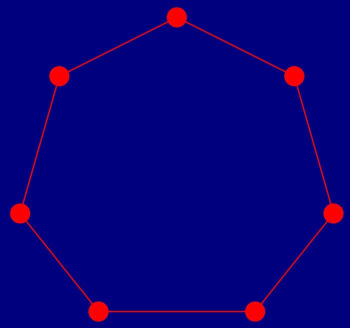
Bull



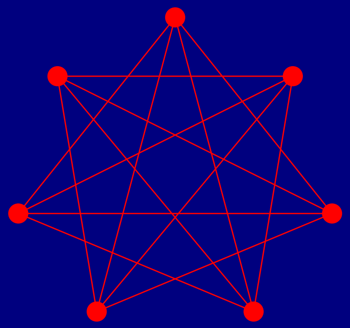
Bull-free Artemis graphs

G is bull-free Artemis iff it contains no odd hole, no antihole and no bull

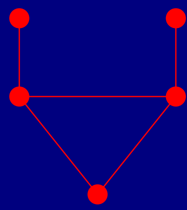
Odd hole



Antihole



Bull

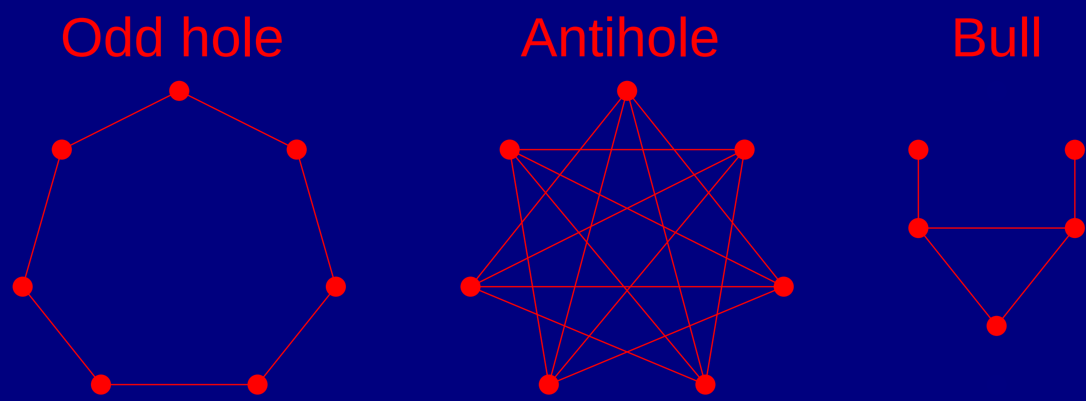


Algorithm Cosine (Hertz - 1990)

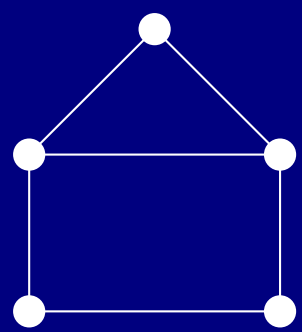
on bull-free Artemis ?

Bull-free Artemis graphs

G is bull-free Artemis iff it contains no odd hole, no antihole and no bull

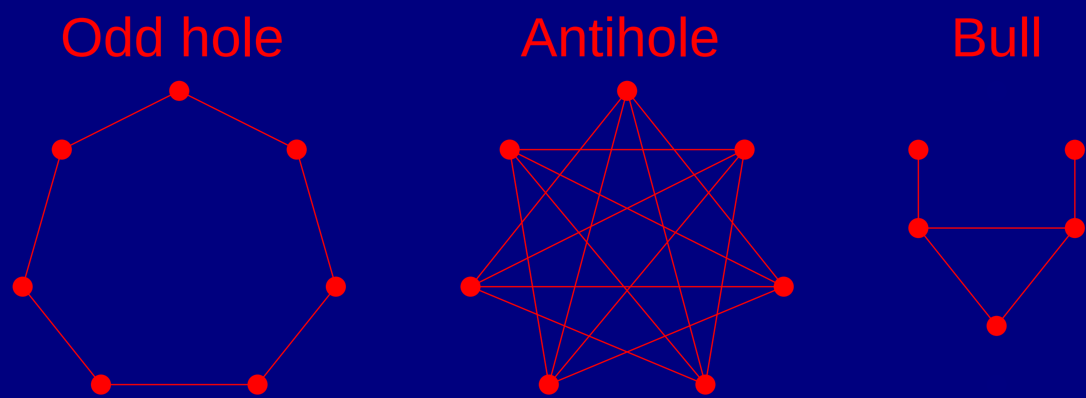


Algorithm Cosine (Hertz - 1990)
on bull-free Artemis ?



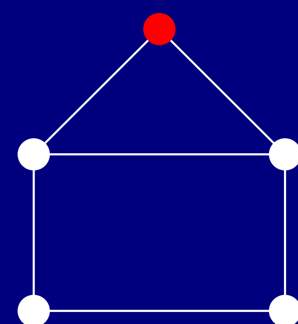
Bull-free Artemis graphs

G is bull-free Artemis iff it contains no odd hole, no antihole and no bull



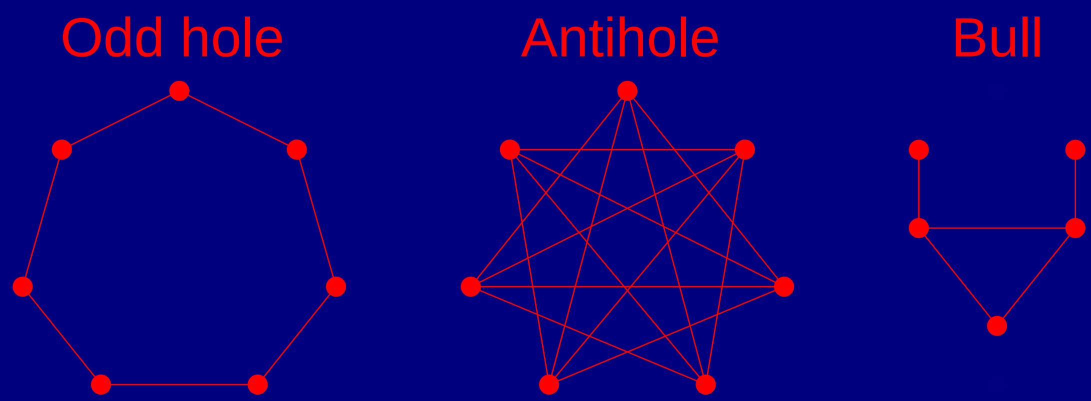
Algorithm Cosine (Hertz - 1990)

on bull-free Artemis ?

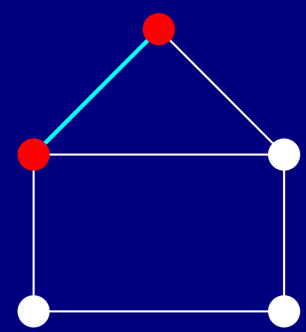


Bull-free Artemis graphs

G is bull-free Artemis iff it contains no odd hole, no antihole and no bull



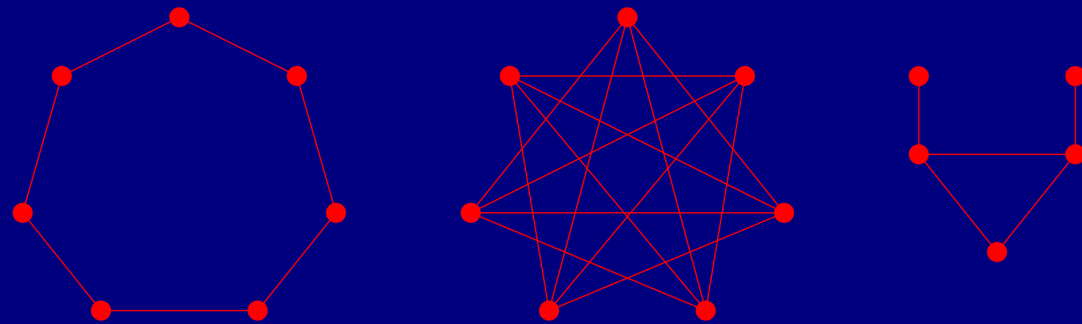
Algorithm Cosine (Hertz - 1990)
on bull-free Artemis ?



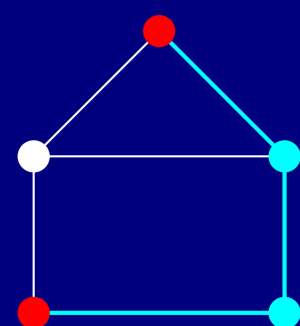
Bull-free Artemis graphs

G is bull-free Artemis iff it contains no odd hole, no antihole and no bull

Odd hole　　Antihole　　Bull



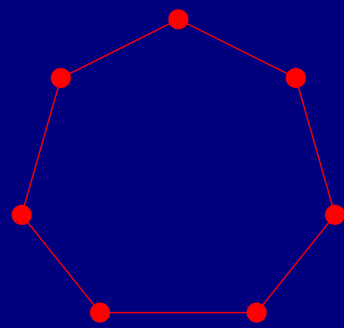
Algorithm Cosine (Hertz - 1990)
on bull-free Artemis ?



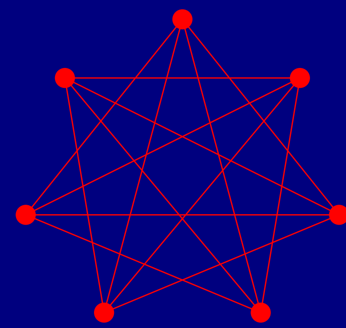
Bull-free Artemis graphs

G is bull-free Artemis iff it contains no odd hole, no antihole and no bull

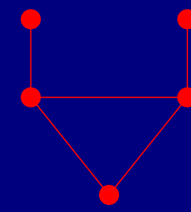
Odd hole



Antihole

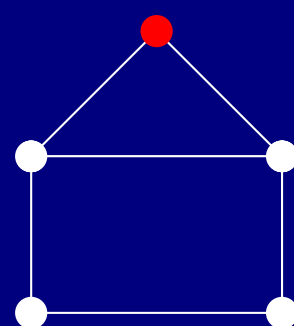


Bull



Algorithm Cosine (Hertz - 1990)

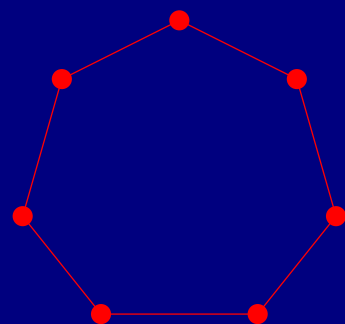
on bull-free Artemis ?



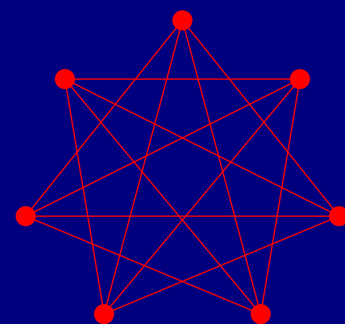
Bull-free Artemis graphs

G is bull-free Artemis iff it contains no odd hole, no antihole and no bull

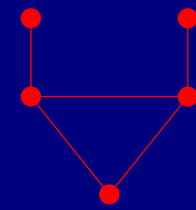
Odd hole



Antihole

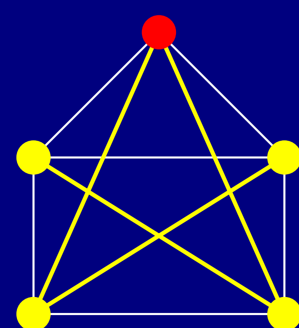


Bull



Algorithm Cosine (Hertz - 1990)

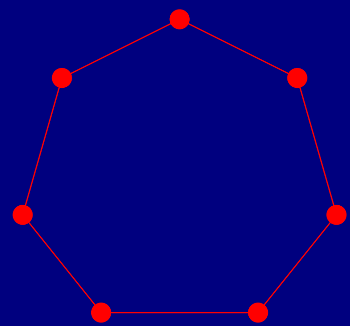
on bull-free Artemis ?



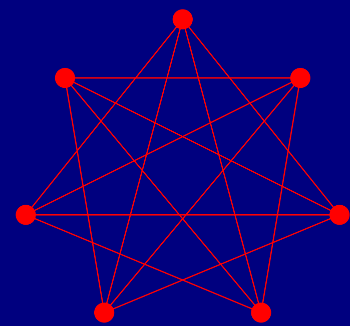
Bull-free Artemis graphs

G is bull-free Artemis iff it contains no odd hole, no antihole and no bull

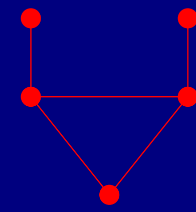
Odd hole



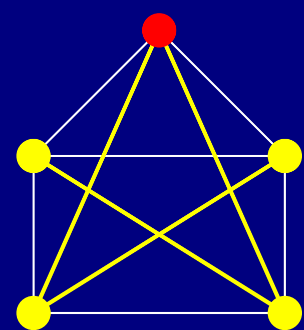
Antihole



Bull



Algorithm Cosine (Hertz - 1990)
on bull-free Artemis ?



Theorem (Lévêque, Maffray - 2007)
Every hole-free bull-free graph has a vertex that is not the middle of a P_5

Algorithm LexBFS*

ALGORITHM LEXBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

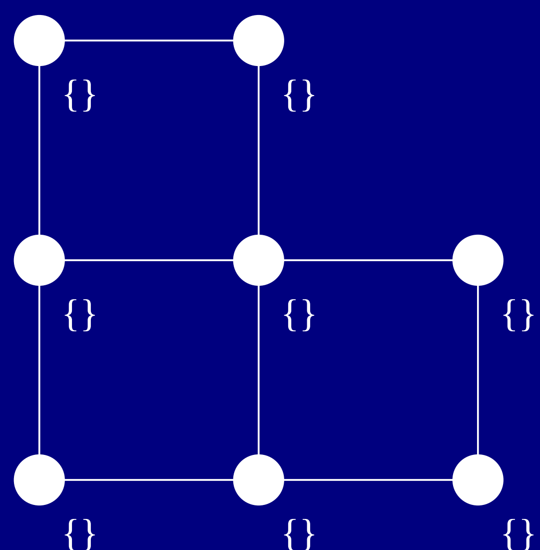
Algorithm LexBFS*

ALGORITHM LEXBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



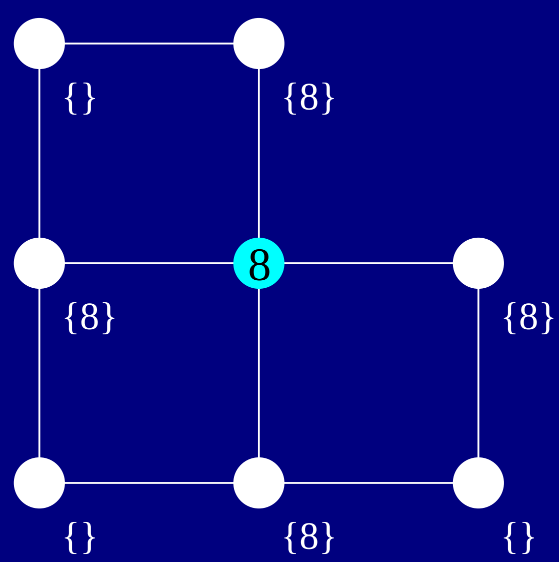
Algorithm LexBFS*

Algorithm LexBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



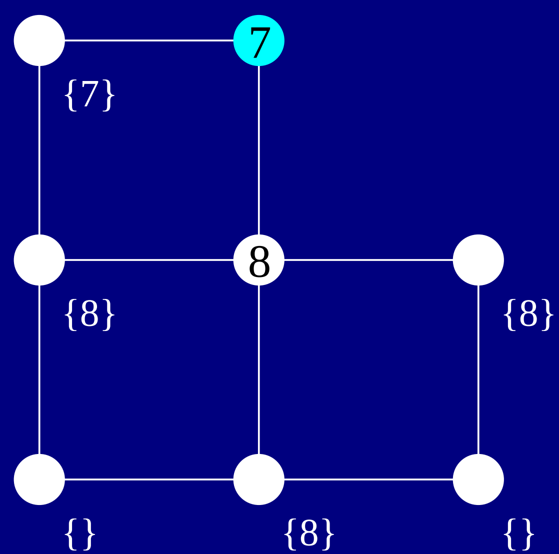
Algorithm LexBFS*

Algorithm LexBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



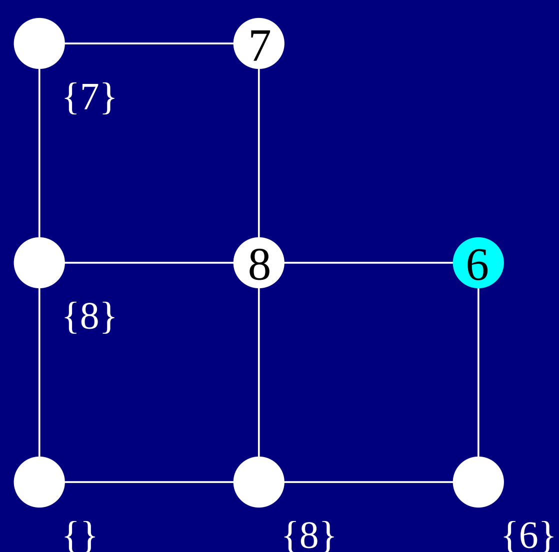
Algorithm LexBFS*

Algorithm LexBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



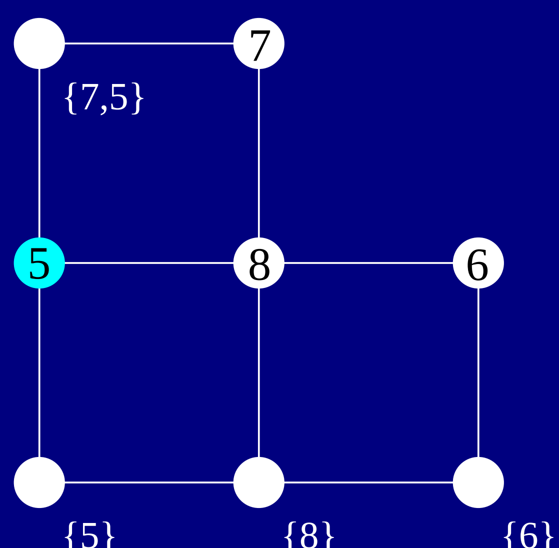
Algorithm LexBFS*

ALGORITHM LEXBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



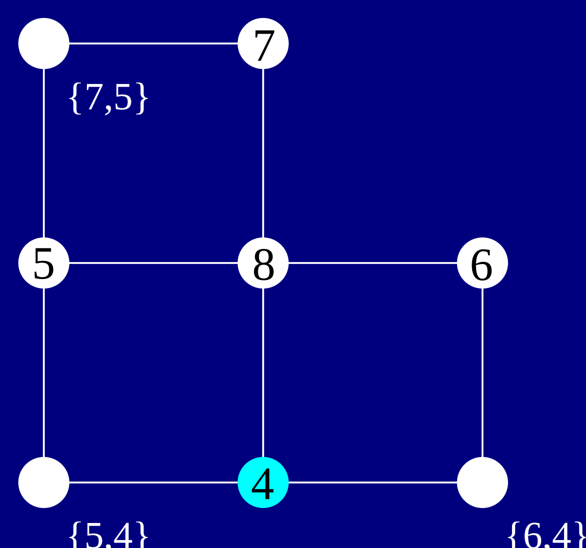
Algorithm LexBFS*

ALGORITHM LEXBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



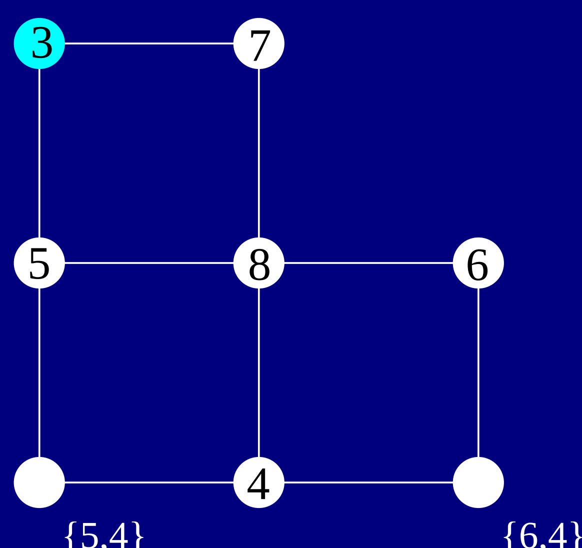
Algorithm LexBFS*

ALGORITHM LEXBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



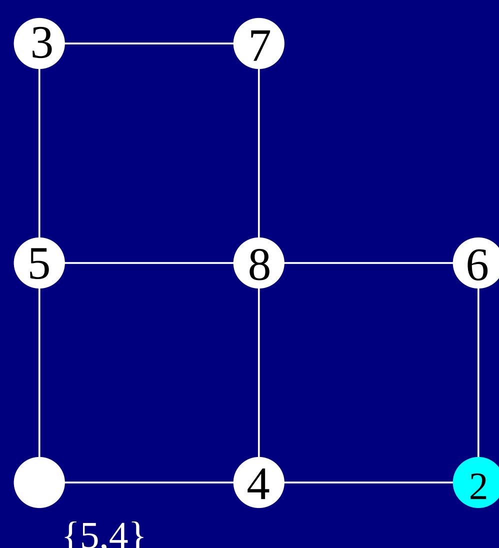
Algorithm LexBFS*

Algorithm LexBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



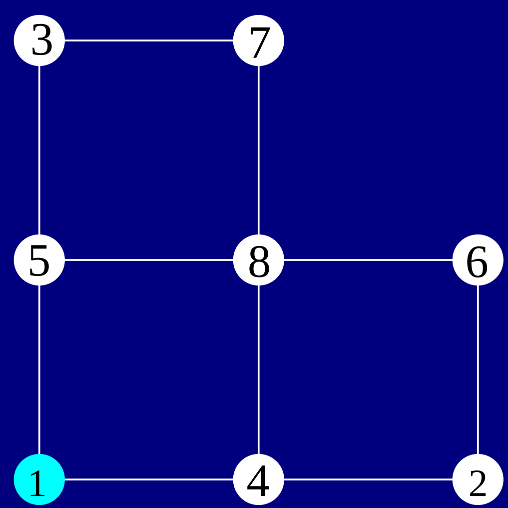
Algorithm LEXBFS*

ALGORITHM LEXBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



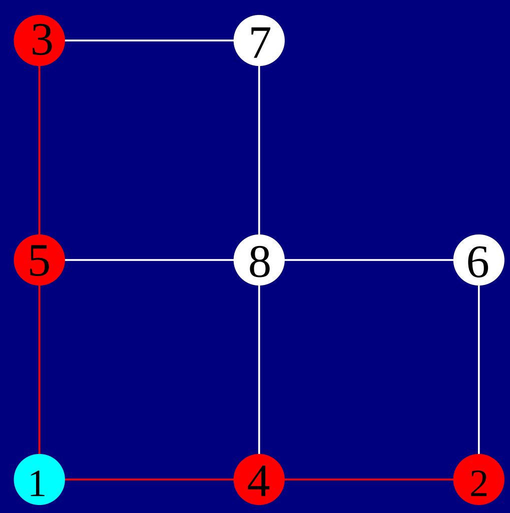
Algorithm LexBFS*

Algorithm LexBFS (Rose, Tarjan, Lueker 1976)

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.

- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

Algorithm LEXBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

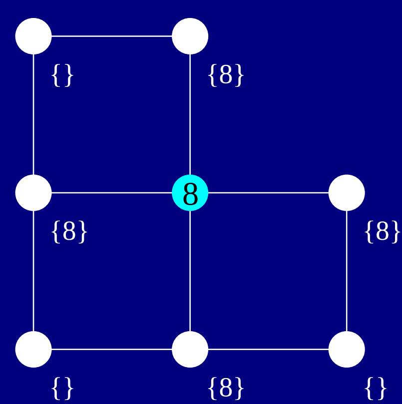
General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
  - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
  - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
  - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
  - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

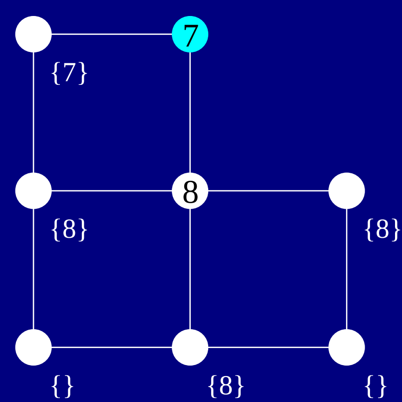


Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
  - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
  - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

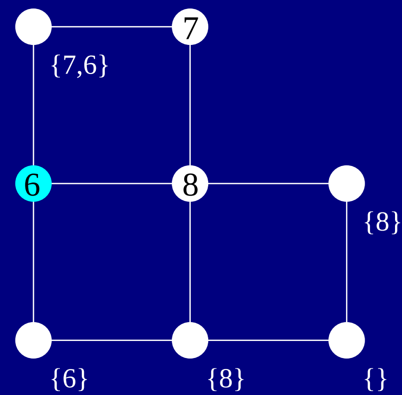


Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
  - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
  - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

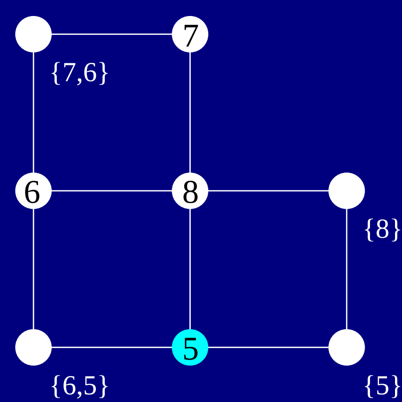


Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
  - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
  - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

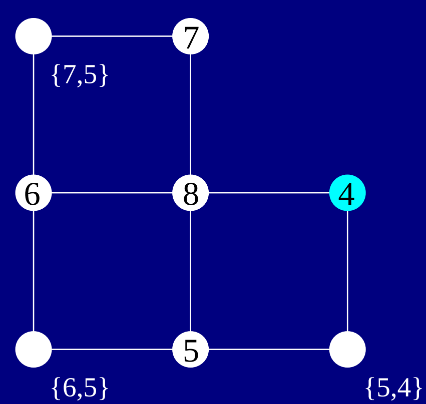


Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
  - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
  - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

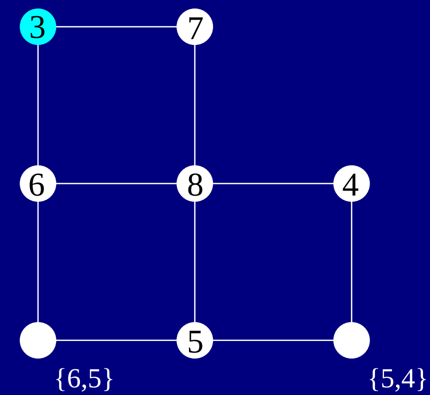


Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
  - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
  - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

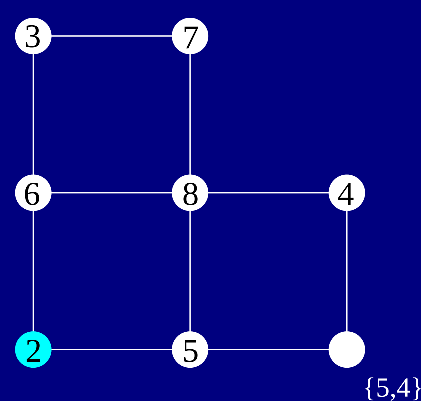


Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
    - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
    - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

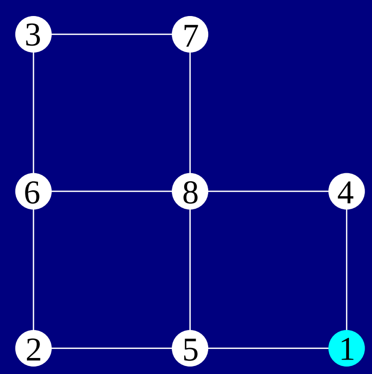


Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
  - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
  - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

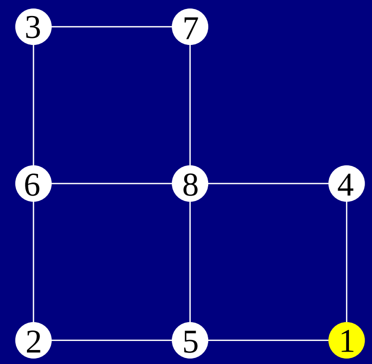


Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
  - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
  - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.



Algorithm LexBFS*

Input: A graph G with n vertices.
Output: An ordering σ on the vertices of G.

Initialization: For every vertex a of G, set $L(a) := \emptyset$;

General step: For $i = n, \ldots, 1$ do:
- Let A be the set of unnumbered vertices whose label is maximum.
- Let U be the other unnumbered vertices.
- Until $|U| = 0$ do:
  - Select a vertex $u \in U$ for which $L(u) \setminus L(A)$ is maximum.
  - Set $U := U \setminus \{u\}$. If $A \cap N(u) \neq \emptyset$, then set $A := A \cap N(u)$.
- Pick any vertex $a \in A$ and set $\sigma(a) := i$.
- For each unnumbered neighbor v of a, add i to $L(v)$.

LexBFS $\mathcal{O}(n+m) \longrightarrow$ LexBFS* $\mathcal{O}(nm)$

Algorithm COSINE*

ALGORITHM COSINE (Hertz 1990)

Input: A graph G on n vertices and an ordering σ on its vertices.
Output: A coloring of the vertices of G.

Initialization: $c = 1$;
General step: While there exist uncolored vertices do:
1. While there exist uncolored vertices that have no neighbor colored c do:
   1.1. Let A be the set of uncolored vertices that have a neighbor colored c;
   1.2. Select an uncolored vertex u that has no neighbor colored c and has the maximum number of neighbors in A;
   1.3. Color u with c;
2. $c := c + 1$.

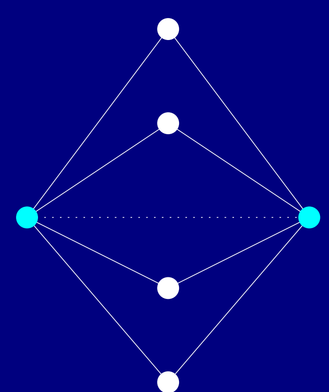
Algorithm COSINE*

ALGORITHM COSINE (Hertz 1990)

Input: A graph G on n vertices and an ordering σ on its vertices.
Output: A coloring of the vertices of G.

Initialization: $c = 1$;
General step: While there exist uncolored vertices do:

1. While there exist uncolored vertices that have no neighbor colored c do:
   1.1. Let A be the set of uncolored vertices that have a neighbor colored c;
   1.2. Select an uncolored vertex u that has no neighbor colored c and has the maximum number of neighbors in A;
   1.3. Color u with c;
2. $c := c + 1$.



Algorithm COSINE*

ALGORITHM COSINE (Hertz 1990)

Input: A graph G on n vertices and an ordering σ on its vertices.
Output: A coloring of the vertices of G.

Initialization: $c = 1$;
General step: While there exist uncolored vertices do:
1. While there exist uncolored vertices that have no neighbor colored c do:
   1.1. Let A be the set of uncolored vertices that have a neighbor colored c;
   1.2. Select an uncolored vertex u that has no neighbor colored c and has the maximum number of neighbors in A,

   1.3. Color u with c;
2. $c := c + 1$.

Algorithm COSINE*

ALGORITHM COSINE (Hertz 1990)

Input: A graph G on n vertices and an ordering σ on its vertices.
Output: A coloring of the vertices of G.

Initialization: $c = 1$;
General step: While there exist uncolored vertices do:
1. While there exist uncolored vertices that have no neighbor colored c do:
   1.1. Let A be the set of uncolored vertices that have a neighbor colored c;
   1.2. Select an uncolored vertex u that has no neighbor colored c and has the maximum number of neighbors in A, ties being broken by taking such a vertex that is minimum for σ;
   1.3. Color u with c;
2. $c := c + 1$.

Algorithm COSINE*

ALGORITHM COSINE (Hertz 1990)

Input: A graph G on n vertices and an ordering σ on its vertices.
Output: A coloring of the vertices of G.

Initialization: $c = 1$;
General step: While there exist uncolored vertices do:

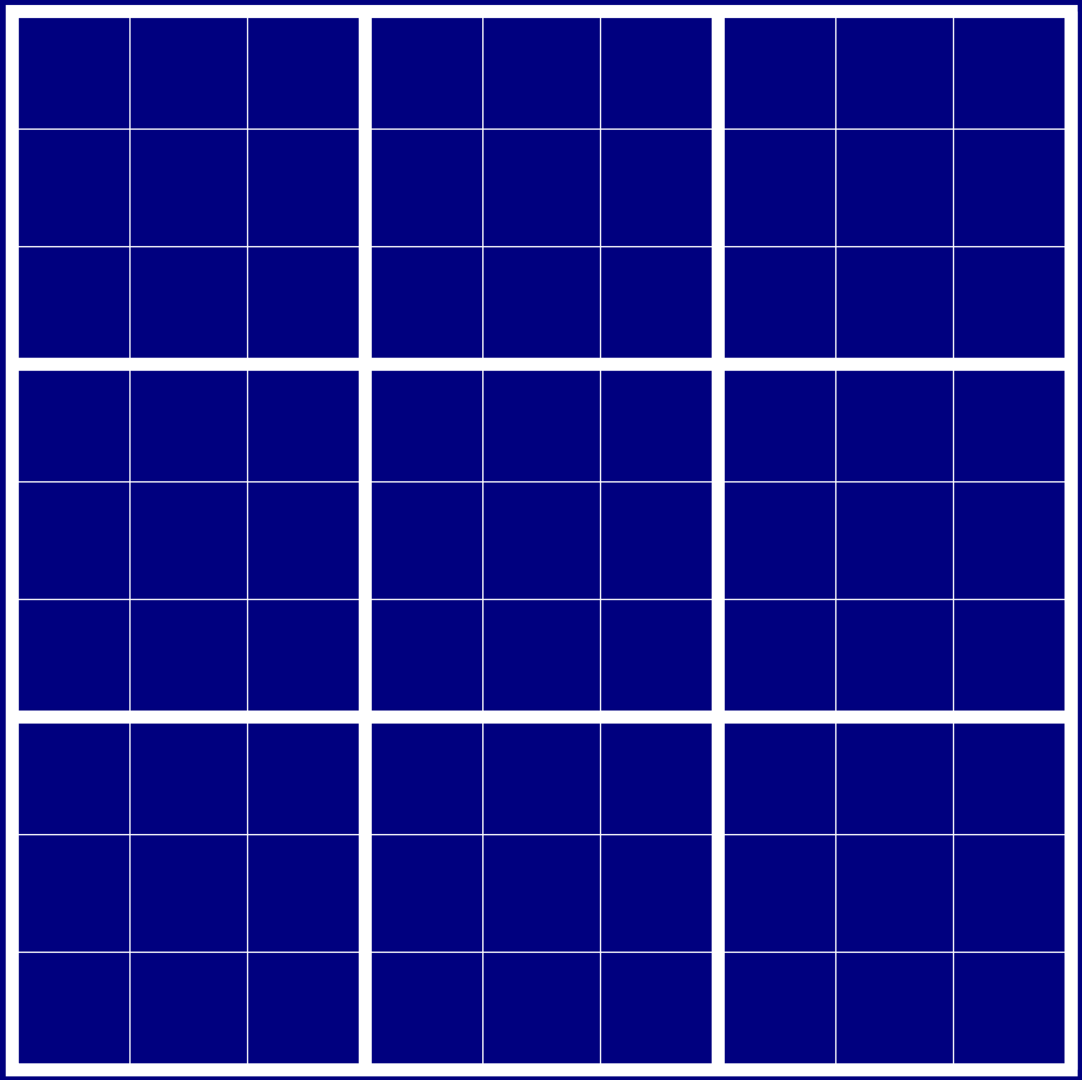
1. While there exist uncolored vertices that have no neighbor colored c do:
   1.1. Let A be the set of uncolored vertices that have a neighbor colored c;
   1.2. Select an uncolored vertex u that has no neighbor colored c and has the maximum number of neighbors in A, ties being broken by taking such a vertex that is minimum for σ;
   1.3. Color u with c;
2. $c := c + 1$.

LEXBFS* on $\overline{G}$ + COSINE* on $G \rightarrow \mathcal{O}(nm)$ coloring algorithm

Precoloring Extension

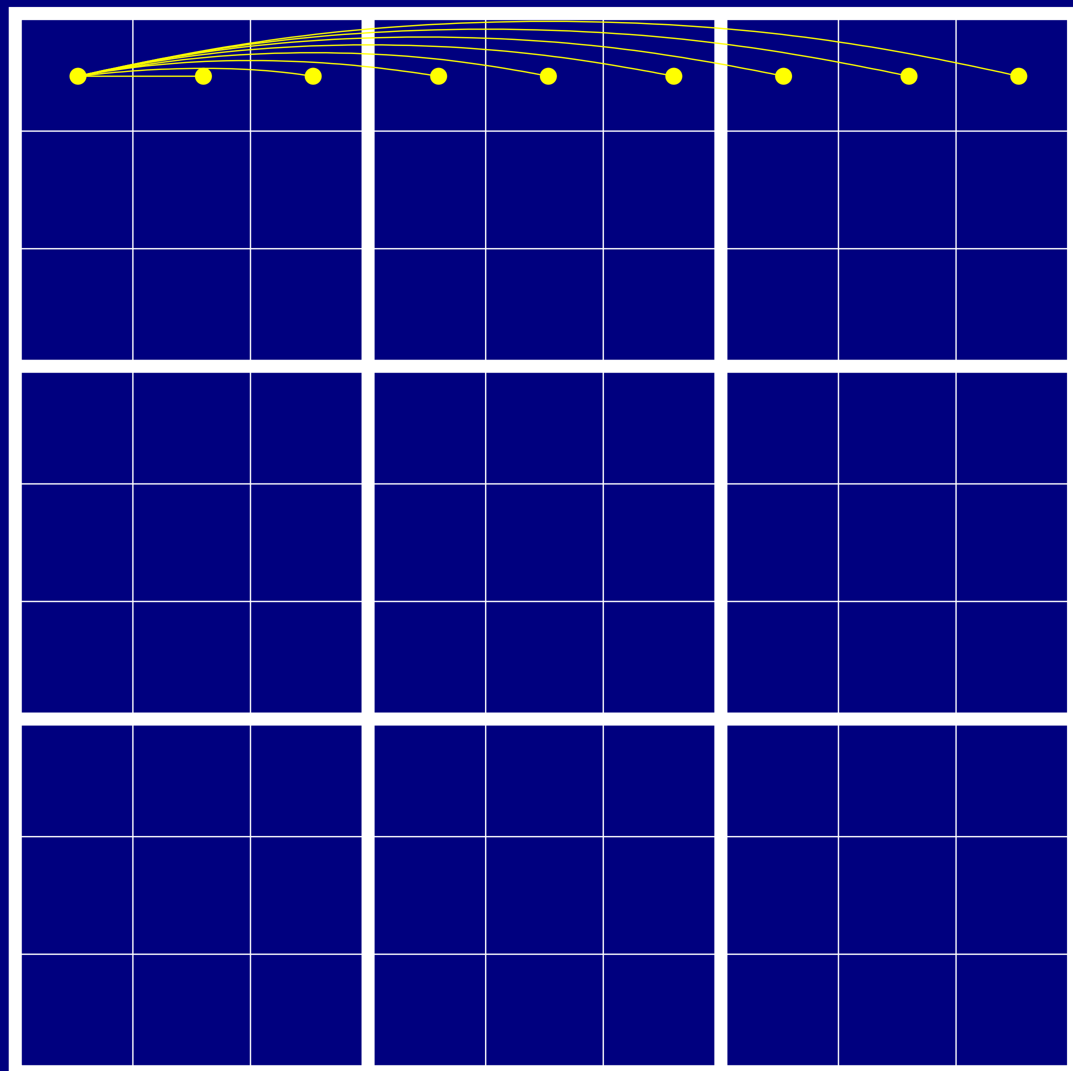
Precoloring Extension

Sudoku



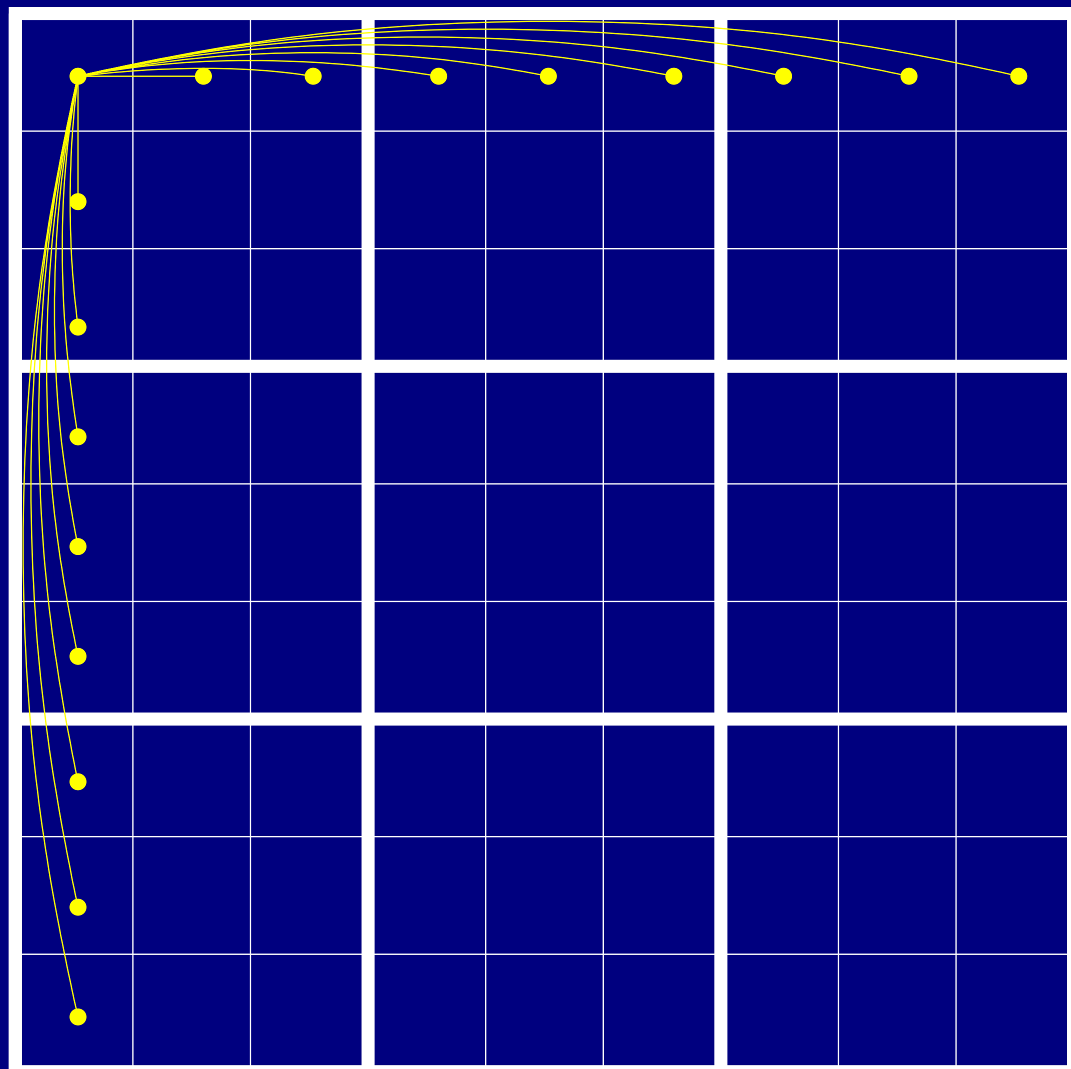
Precoloring Extension

Sudoku



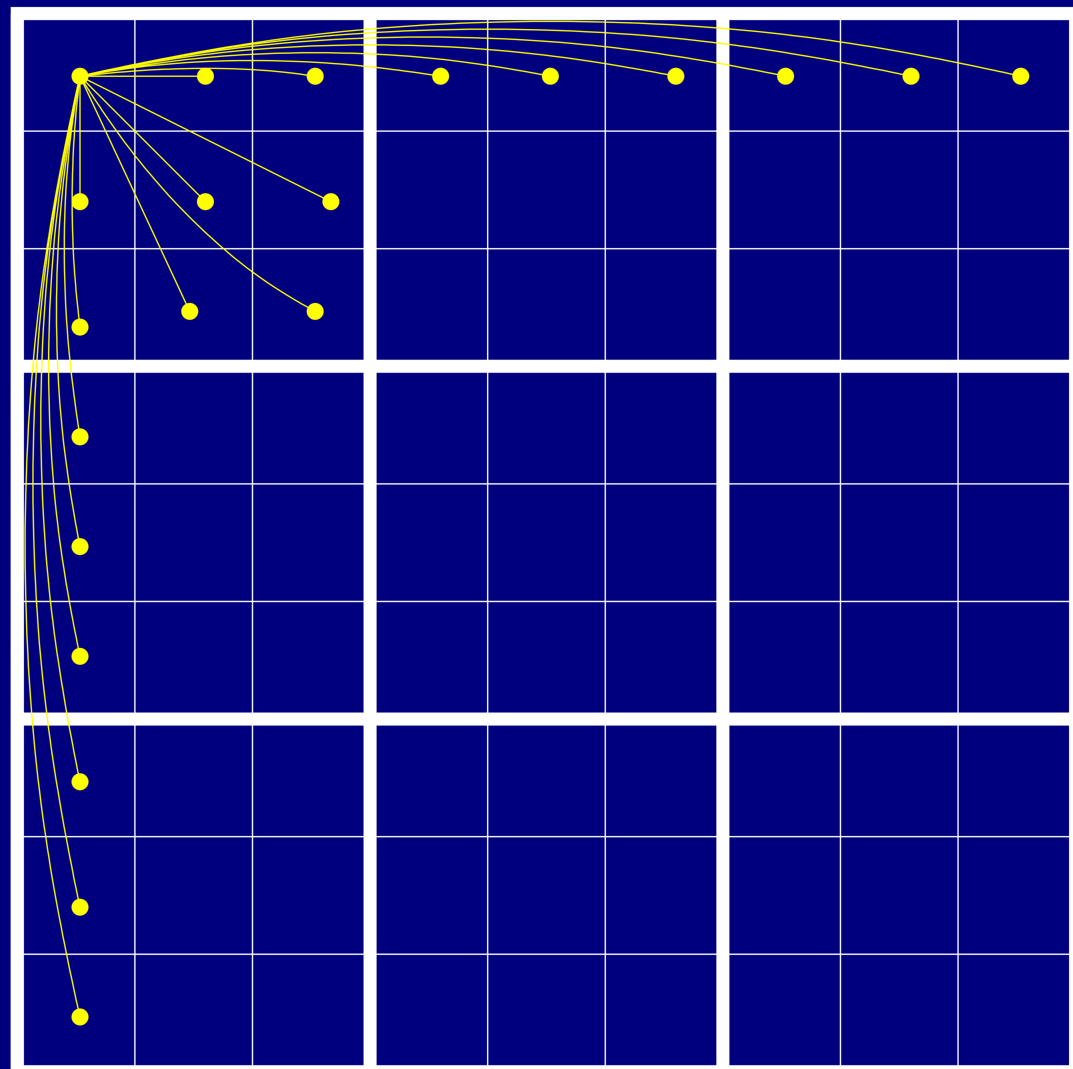
Precoloring Extension

Sudoku



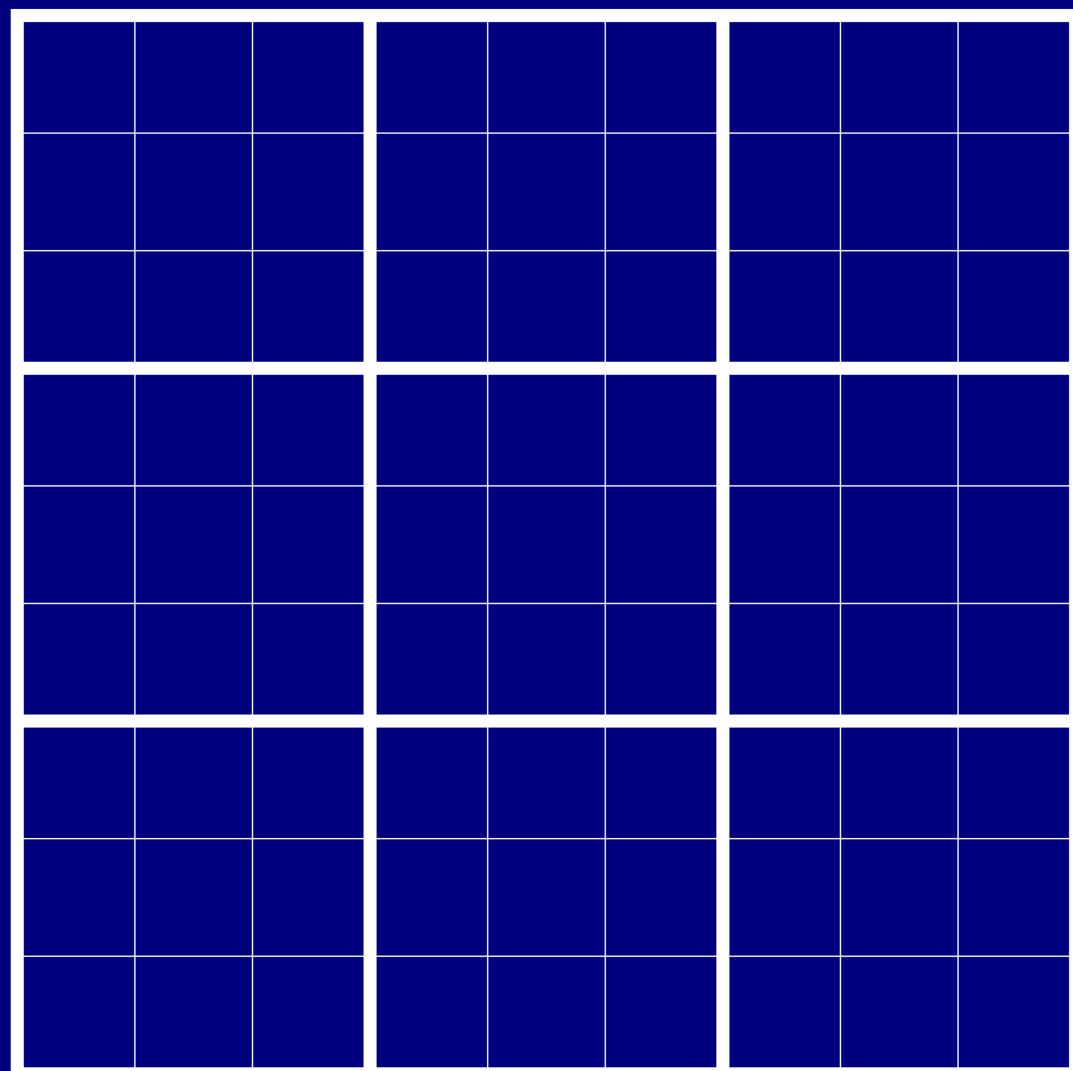
Precoloring Extension

Sudoku



Precoloring Extension

Sudoku



Precoloring Extension

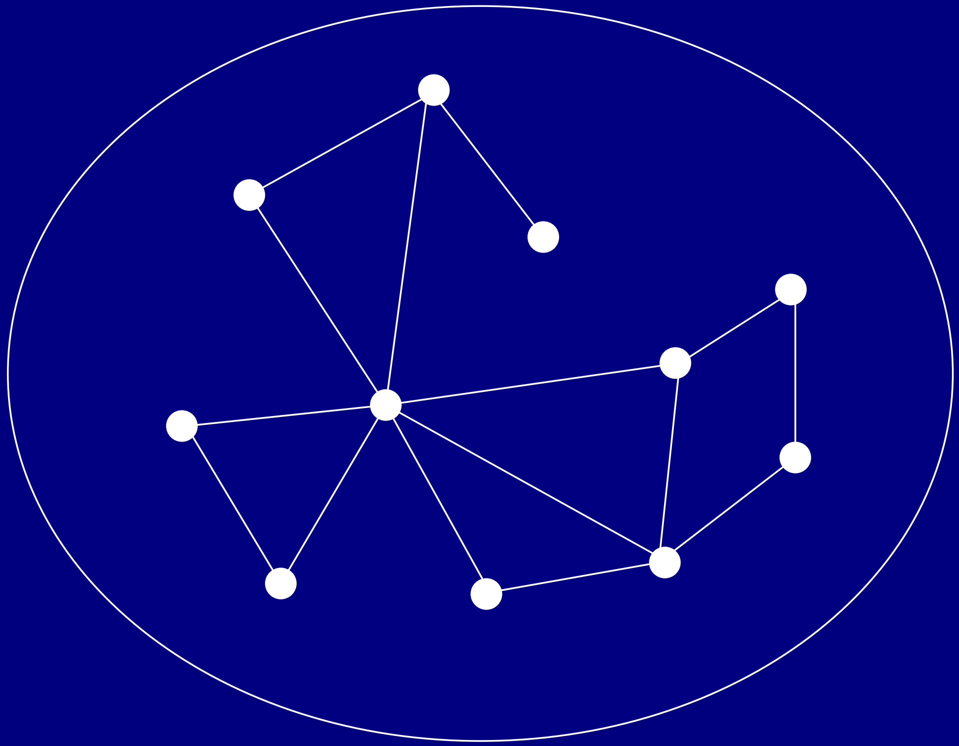
Sudoku

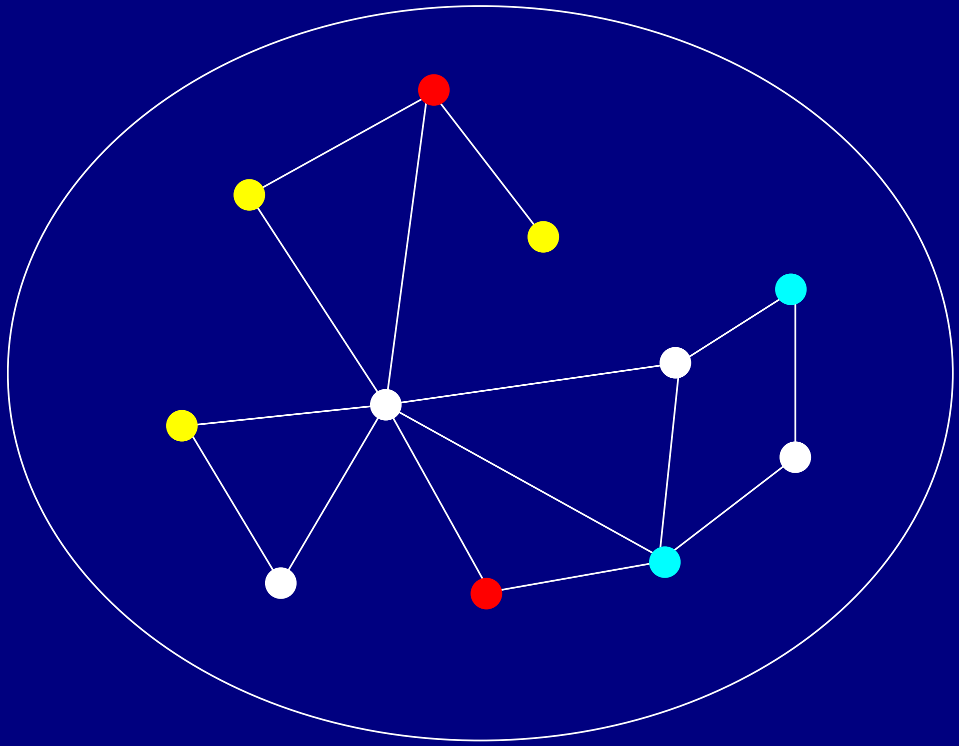
3	8		4		7		5	
				9			1	7
9	6	7		1				4
					8		2	9
6	9	2	5	7	4	3	8	1
7	3		1					
8				4		5	9	3
1	7			5				
	5		9		2		7	6

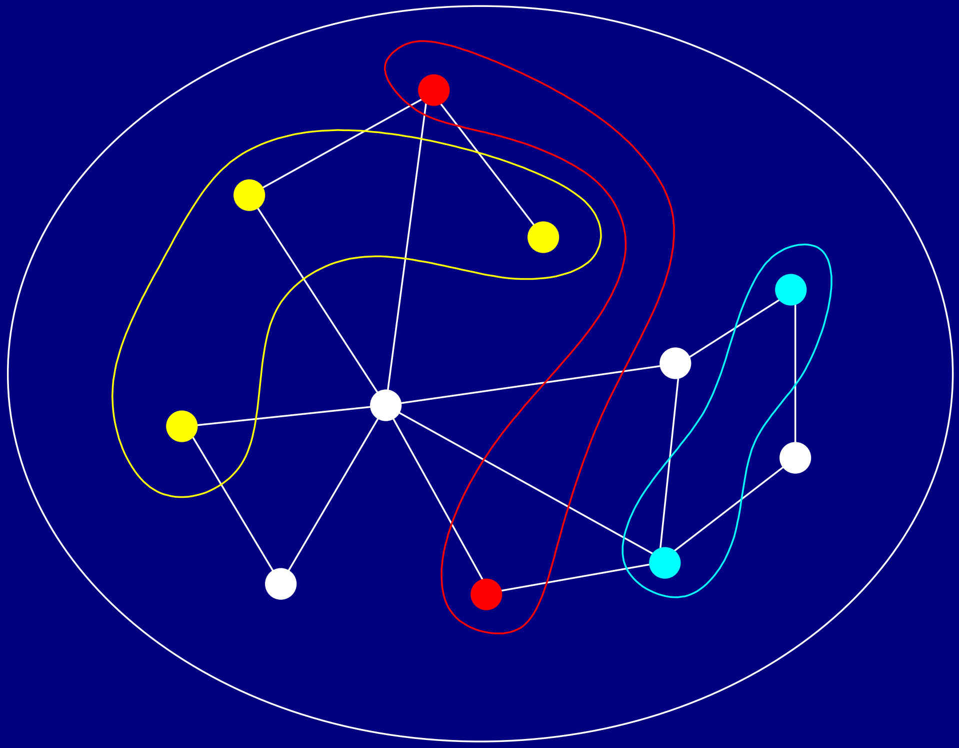
Precoloring Extension

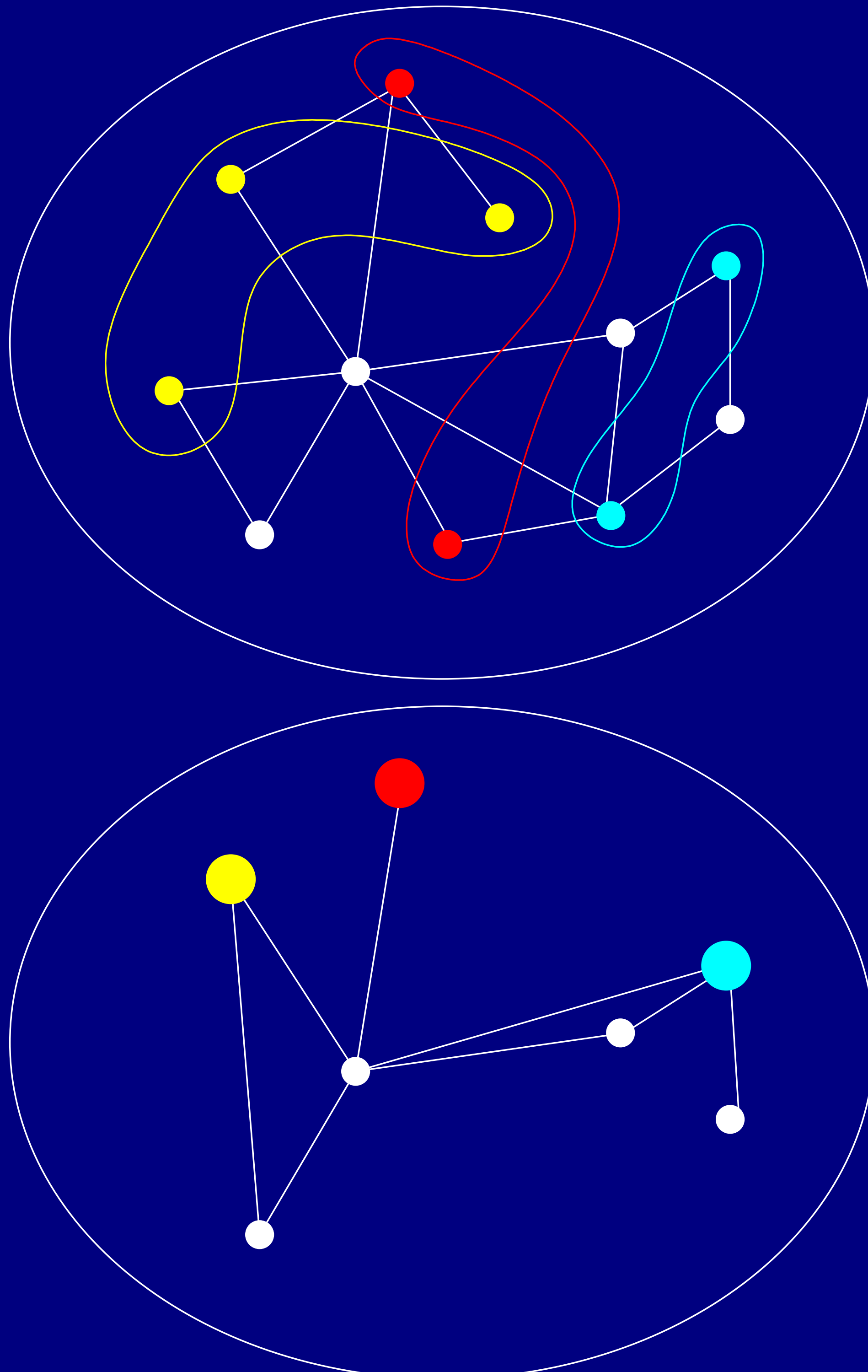
Sudoku

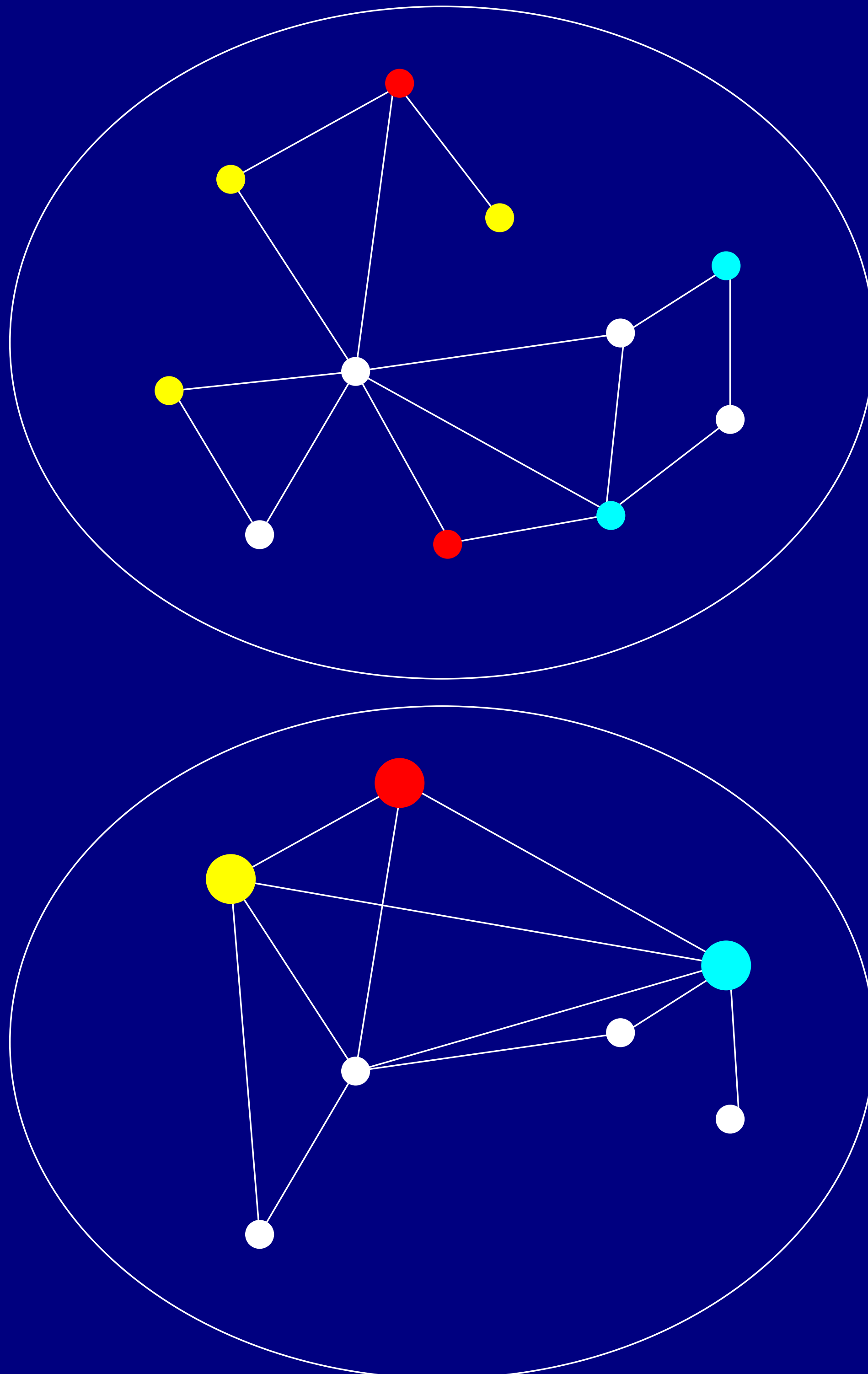
3	8	1	4	6	7	9	5	2
2	4	5	8	9	3	6	1	7
9	6	7	2	1	5	8	3	4
5	1	4	6	3	8	7	2	9
6	9	2	5	7	4	3	8	1
7	3	8	1	2	9	4	6	5
8	2	6	7	4	1	5	9	3
1	7	9	3	5	6	2	4	8
4	5	3	9	8	2	1	7	6

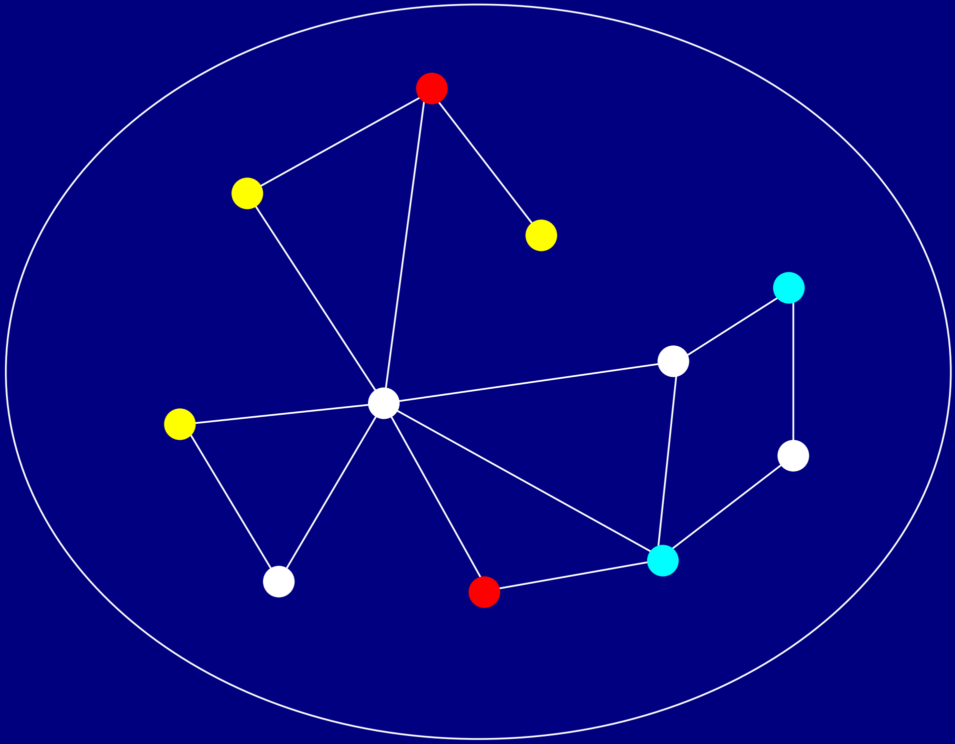


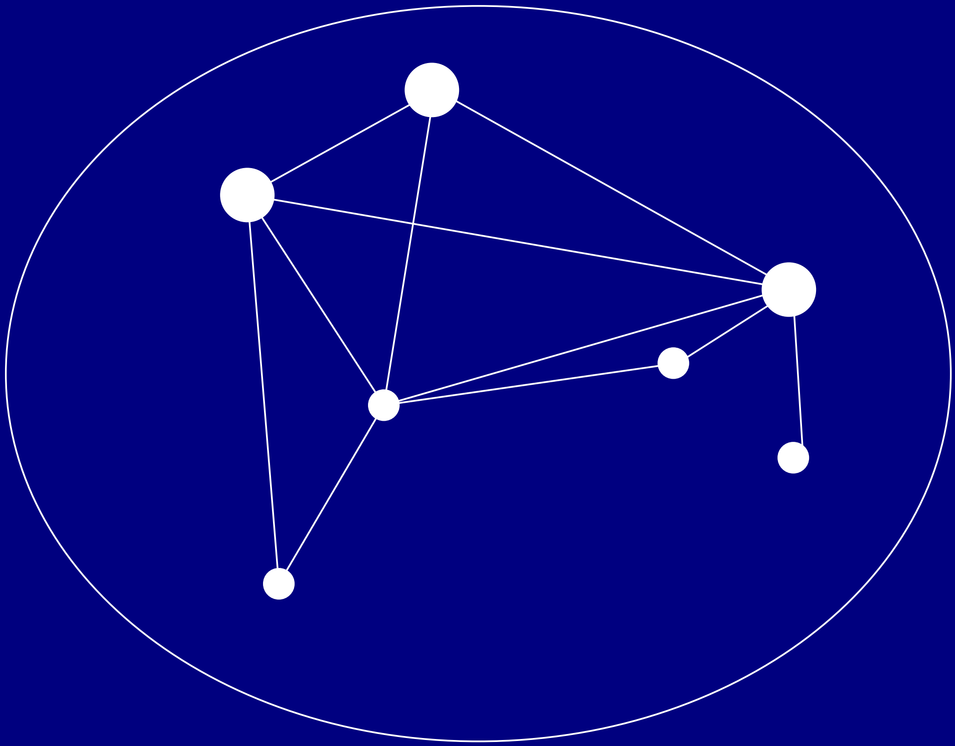












PrExt-perfection

PrExt-perfect graphs (Hujter, Tuza - 1996)

For every precoloring, the contracted graph is perfect

PrExt-perfection

PrExt-perfect graphs (Hujter, Tuza - 1996)

For every precoloring, the contracted graph is perfect

$\boxed{\text{Which are the PrExt-perfect graphs ?}}$

PrExt-perfection

PrExt-perfect graphs (Hujter, Tuza - 1996)

For every precoloring, the contracted graph is perfect

Which are the PrExt-perfect graphs ?

Theorem (Jost, Lévêque, Maffray - 2007)

PrExt-perfect = co-Meyniel

PrExt-perfection

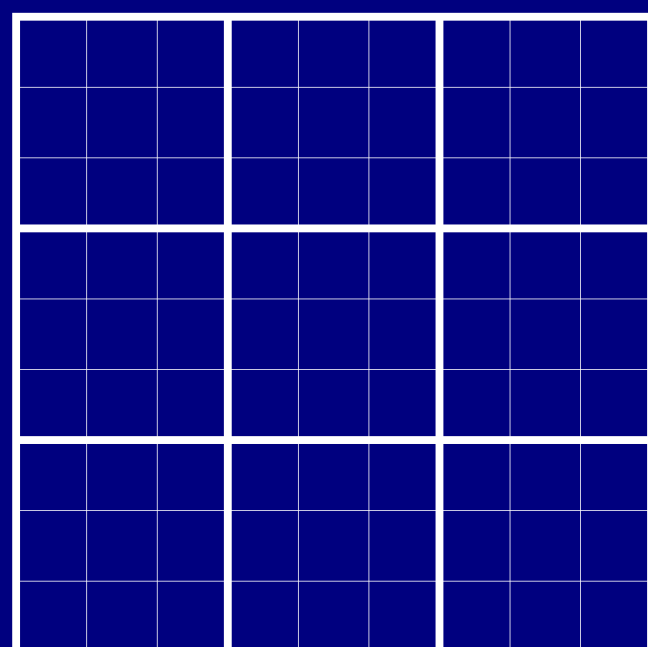
PrExt-perfect graphs (Hujter, Tuza - 1996)

For every precoloring, the contracted graph is perfect

Which are the PrExt-perfect graphs ?

Theorem (Jost, Lévêque, Maffray - 2007)

PrExt-perfect = co-Meyniel



PrExt-perfection

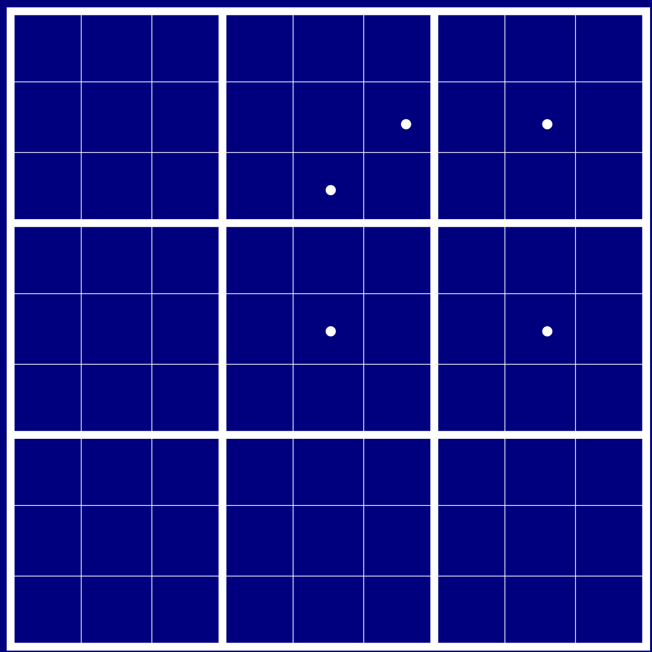
PrExt-perfect graphs (Hujter, Tuza - 1996)

For every precoloring, the contracted graph is perfect

Which are the PrExt-perfect graphs ?

Theorem (Jost, Lévêque, Maffray - 2007)

PrExt-perfect = co-Meyniel



PrExt-perfection

PrExt-perfect graphs (Hujter, Tuza - 1996)

For every precoloring, the contracted graph is perfect

Which are the PrExt-perfect graphs ?

Theorem (Jost, Lévêque, Maffray - 2007)

PrExt-perfect = co-Meyniel