Enumerating all Triangulations up to Symmetry

Or: The Power of Order Rightly Used

Jörg Rambau

UNIVERSITÄT BAYREUTH

Lehrstuhl für Wirtschaftsmathematik

September 5–9, 2022
Workshop “Geometry meets Combinatorics”
Bielefeld

Agenda

Agenda

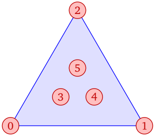
How Many Triangulations Are There?

How Many Triangulations Are There?

Given

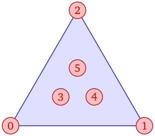
How Many Triangulations Are There?

Given A point configuration



How Many Triangulations Are There?

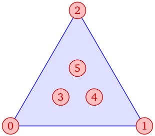
Given A point configuration



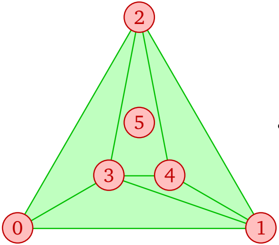
Question 1 How many triangulations does it have?

How Many Triangulations Are There?

Given A point configuration

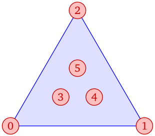


Question 1 How many triangulations does it have?



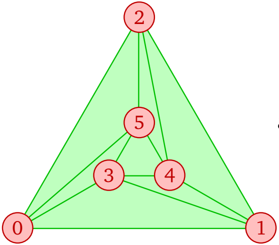
a triangulation

How Many Triangulations Are There?



Given A point configuration

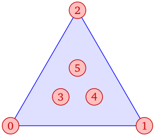
Question 1 How many triangulations does it have?



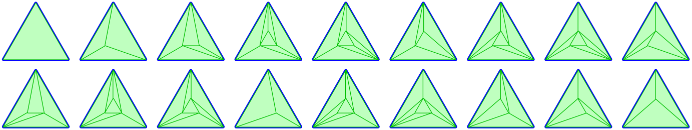
another triangulation

How Many Triangulations Are There?

Given A point configuration

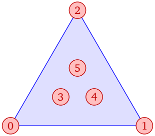


Question 1 How many triangulations does it have?

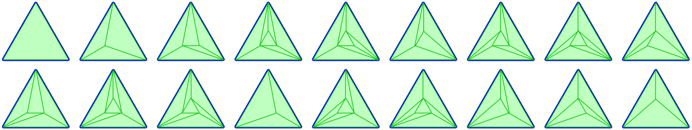


How Many Triangulations Are There?

Given A point configuration



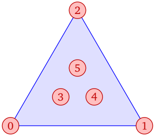
Question 1 How many triangulations does it have?



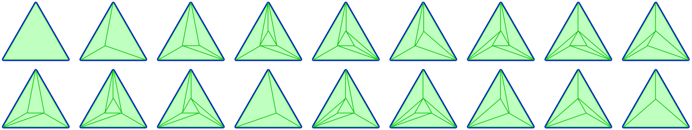
Question 2 How many triangulations does it have up to symmetry?

How Many Triangulations Are There?

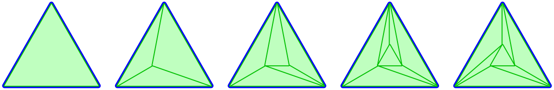
Given A point configuration



Question 1 How many triangulations does it have?

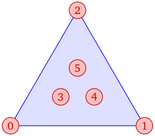


Question 2 How many triangulations does it have up to symmetry?

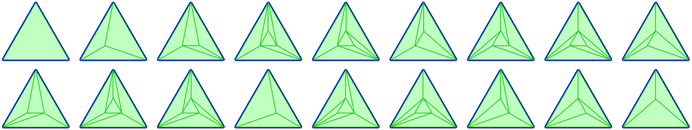


How Many Triangulations Are There?

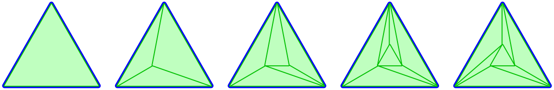
Given A point configuration



Question 1 How many triangulations does it have?



Question 2 How many triangulations does it have up to symmetry?



This Talk Enumerate them with a computer.

Selected History

Selected History

De Loera 1994 — Flip-based symmetric BFS in flip-graph component; (maple-code PUNTOS)

Selected History

De Loera 1994
Flip-based symmetric BFS in flip-graph component;
(maple-code PUNTOS)

R. 2000
Flip-based symmetric BFS in flip-graph component;
simplex-by-simplex-based DFS for all triang's;
(oriented-matroid-based C++-code TOPCOM 0.x.x).

Selected History

De Loera 1994	Flip-based symmetric BFS in flip-graph component; (maple-code PUNTOS)
R. 2000	Flip-based symmetric BFS in flip-graph component; simplex-by-simplex-based DFS for all triang's; (oriented-matroid-based C++-code TOPCOM 0.x.x).
Santos 2000	The flip-graph of triangulations can be disconnected.

Selected History

De Loera 1994	Flip-based symmetric BFS in flip-graph component; (maple-code PUNTOS)
R. 2000	Flip-based symmetric BFS in flip-graph component; simplex-by-simplex-based DFS for all triang's; (oriented-matroid-based C++-code TOPCOM 0.x.x).
Santos 2000	The flip-graph of triangulations can be disconnected.
Imai et al. 2002	Flip-based reverse search for orbits of regular triang's; Stable-set-based enumeration of all triang's.

Selected History

De Loera 1994	Flip-based symmetric BFS in flip-graph component; (maple-code PUNTOS)
R. 2000	Flip-based symmetric BFS in flip-graph component; simplex-by-simplex-based DFS for all triang's; (oriented-matroid-based C++-code TOPCOM 0.x.x).
Santos 2000	The flip-graph of triangulations can be disconnected.
Imai et al. 2002	Flip-based reverse search for orbits of regular triang's; Stable-set-based enumeration of all triang's.
Jordan et al. 2018	Parallel flip-based reverse search for orbits of sub-regular triang's; (C++-code MPTOPCOM).

Selected History

De Loera 1994	Flip-based symmetric BFS in flip-graph component; (maple-code PUNTOS)
R. 2000	Flip-based symmetric BFS in flip-graph component; simplex-by-simplex-based DFS for all triang's; (oriented-matroid-based C++-code TOPCOM 0.x.x).
Santos 2000	The flip-graph of triangulations can be disconnected.
Imai et al. 2002	Flip-based reverse search for orbits of regular triang's; Stable-set-based enumeration of all triang's.
Jordan et al. 2018	Parallel flip-based reverse search for orbits of sub-regular triang's; (C++-code MPTOPCOM).
New: R. 2022	Parallel symmetric lexicographic subset reverse search for all triang's (new C++-code TOPCOM 1.x.x)

Agenda

Reverse Search (RS)

Reverse Search (RS)

Goal Enumerate the nodes (= objects) of a graph (V, E) with

Reverse Search (RS)

Goal

- Enumerate the nodes (= objects) of a graph (V, E) with an objective function on V with unique opt v_{opt};

Reverse Search (RS)

Goal Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\text{opt}}\}$.

Reverse Search (RS)

Goal Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\text{opt}}\}$.

Method Reverse Search (RS) [Avis & Fukuda 1996]:

Reverse Search (RS)

Goal Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\text{opt}}\}$.

Method Reverse Search (RS) [Avis & Fukuda 1996]:

Reverse Search (RS)

Goal Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\text{opt}}\}$.

Method Reverse Search (RS) [Avis & Fukuda 1996]:

- generate an arbitrary object

Reverse Search (RS)

Goal

Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\mathrm{opt}}\}$.

Method

Reverse Search (RS) [Avis & Fukuda 1996]:

- generate an arbitrary object
- pivot to the optimum object

Reverse Search (RS)

Goal Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\mathrm{opt}}\}$.

Method Reverse Search (RS) [Avis & Fukuda 1996]:

- generate an arbitrary object
- pivot to the optimum object
- return ReverseSearch(optimum object)

Reverse Search (RS)

Goal

Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\text{opt}}\}$.

Method

Reverse Search (RS) [Avis & Fukuda 1996]:

- generate an arbitrary object
- pivot to the optimum object
- return ReverseSearch(optimum object)

Subroutine

ReverseSearch(object):

Reverse Search (RS)

Goal

Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\text{opt}}\}$.

Method

Reverse Search (RS) [Avis & Fukuda 1996]:

- generate an arbitrary object
- pivot to the optimum object
- return ReverseSearch(optimum object)

Subroutine

ReverseSearch(object):

- increase counter

Reverse Search (RS)

Goal
Enumerate the nodes (= objects) of a graph (V, E) with
- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\text{opt}}\}$.

Method
Reverse Search (RS) [Avis & Fukuda 1996]:
- generate an arbitrary object
- pivot to the optimum object
- return ReverseSearch(optimum object)

Subroutine
ReverseSearch(object):
- increase counter
- for all neighbors of object do

Reverse Search (RS)

Goal Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\text{opt}}\}$.

Method Reverse Search (RS) [Avis & Fukuda 1996]:

- generate an arbitrary object
- pivot to the optimum object
- return ReverseSearch(optimum object)

Subroutine ReverseSearch(object):

- increase counter
- for all neighbors of object do
  - if neighbor pivots to object

Reverse Search (RS)

Goal

Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\mathrm{opt}}\}$.

Method

Reverse Search (RS) [Avis & Fukuda 1996]:

- generate an arbitrary object
- pivot to the optimum object
- return ReverseSearch(optimum object)

Subroutine

ReverseSearch(object):

- increase counter
- for all neighbors of object do
  - if neighbor pivots to object
  - increase counter by ReverseSearch(object)

Reverse Search (RS)

Goal Enumerate the nodes (= objects) of a graph (V, E) with

- an objective function on V with unique opt v_{opt};
- a pivot function choosing a better neighbor on $V \setminus \{v_{\text{opt}}\}$.

Method Reverse Search (RS) [Avis & Fukuda 1996]:

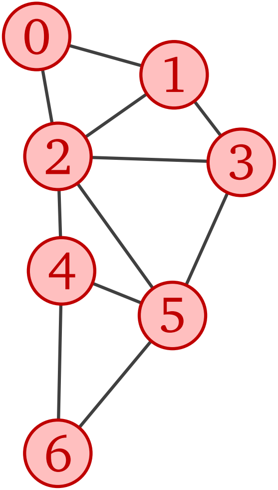
- generate an arbitrary object
- pivot to the optimum object
- return ReverseSearch(optimum object)

Subroutine ReverseSearch(object):

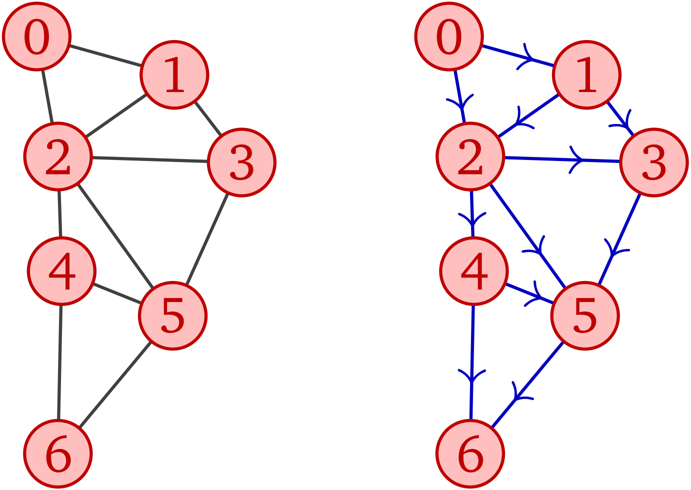
- increase counter
- for all neighbors of object do
  - if neighbor pivots to object
  - increase counter by ReverseSearch(object)
- return counter.

RS Example

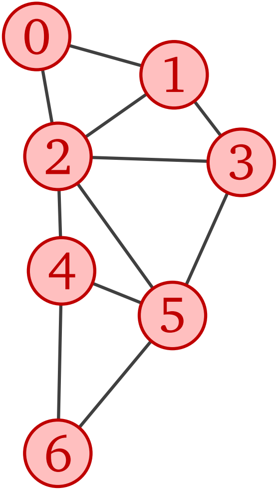
RS Example

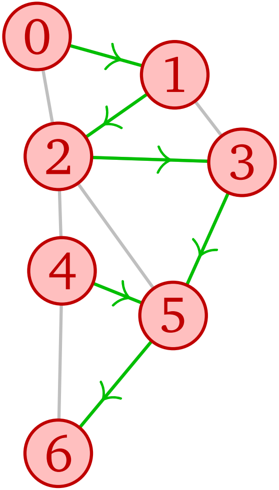


RS Example

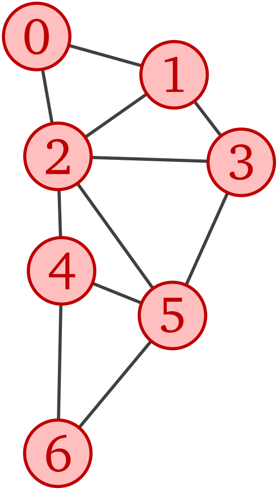


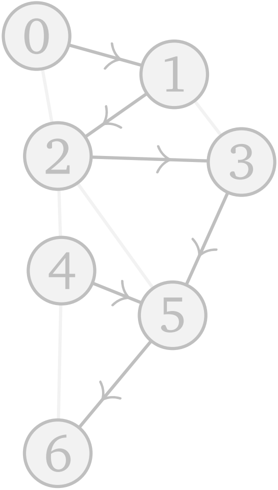
RS Example



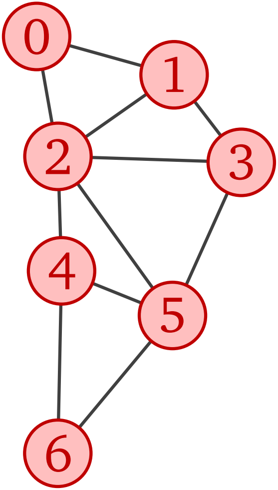


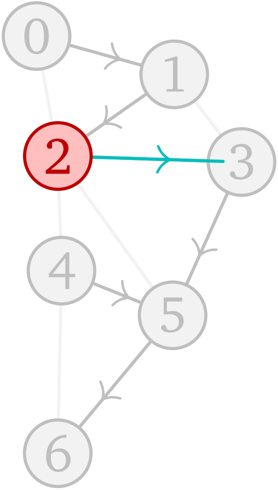
RS Example



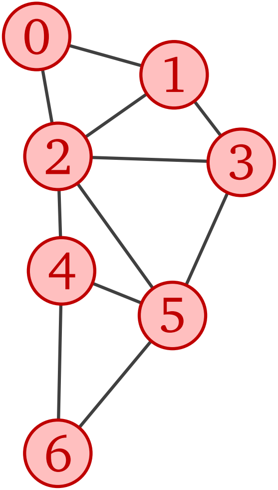


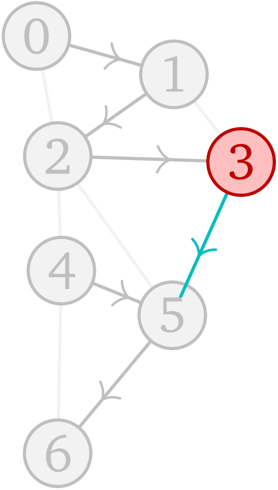
RS Example



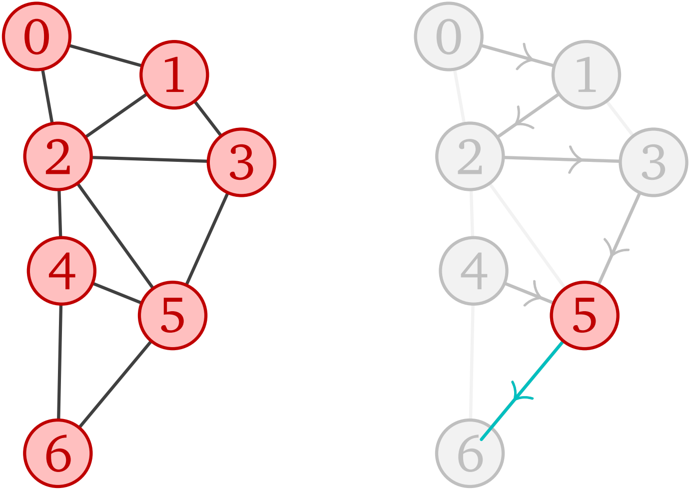


RS Example

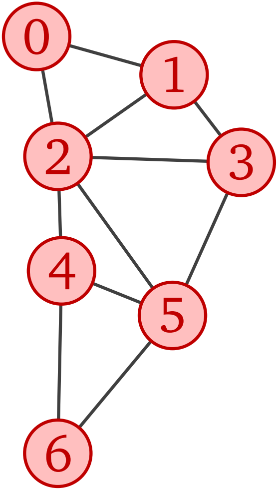


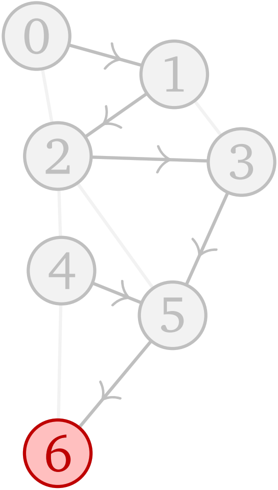


RS Example

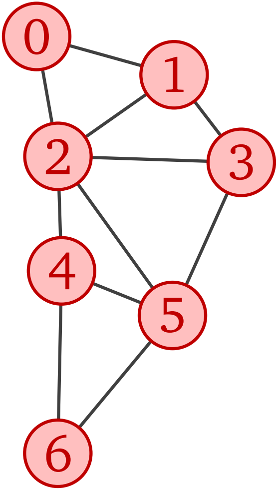


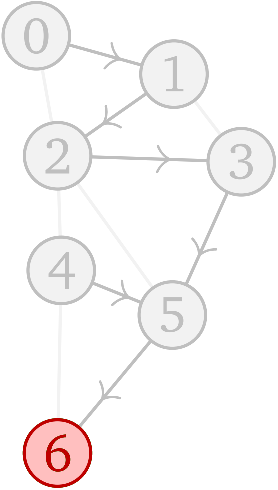
RS Example



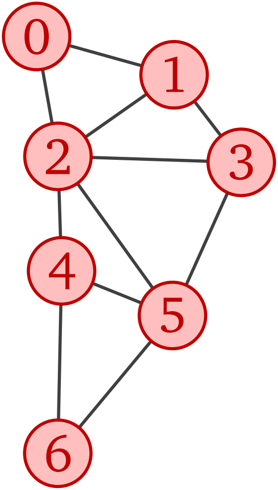


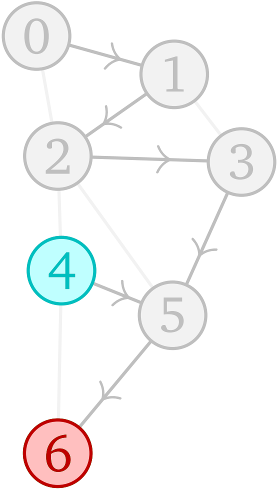
RS Example



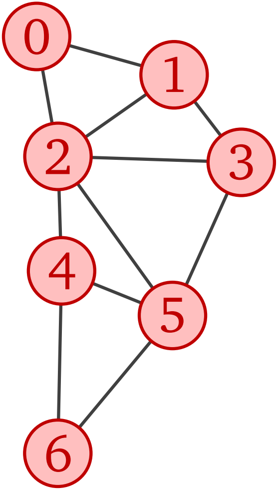


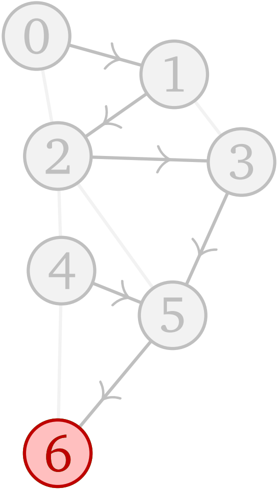
RS Example



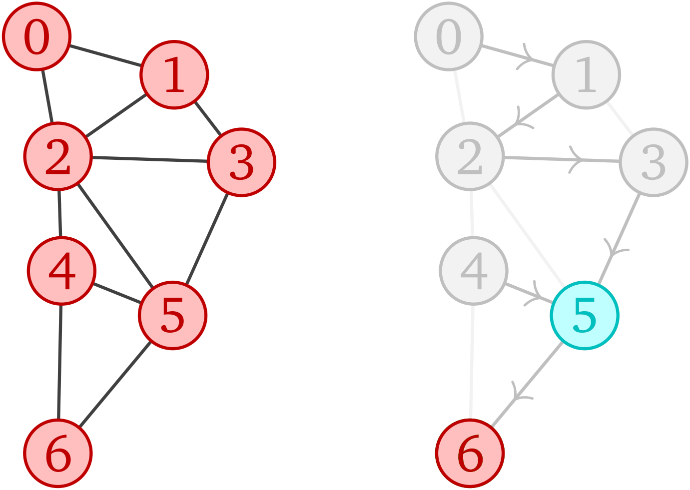


RS Example

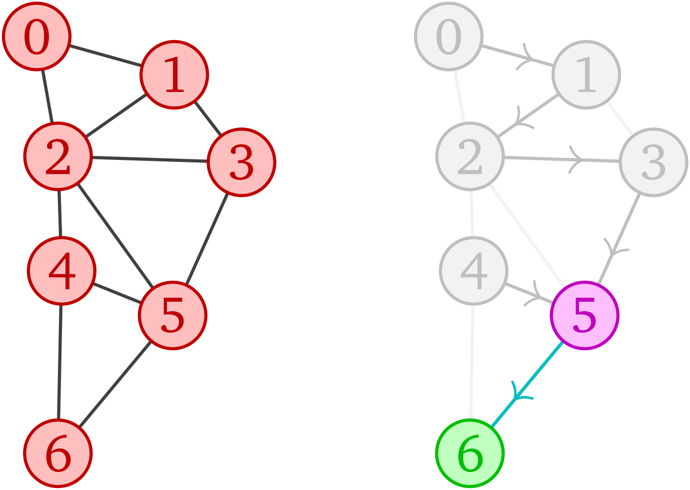




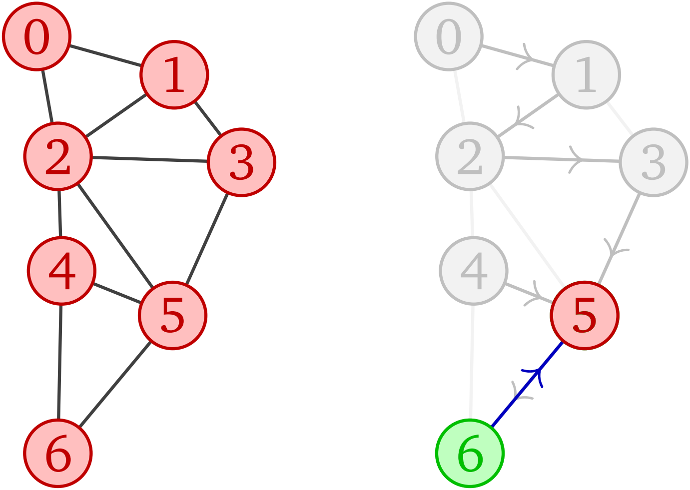
RS Example



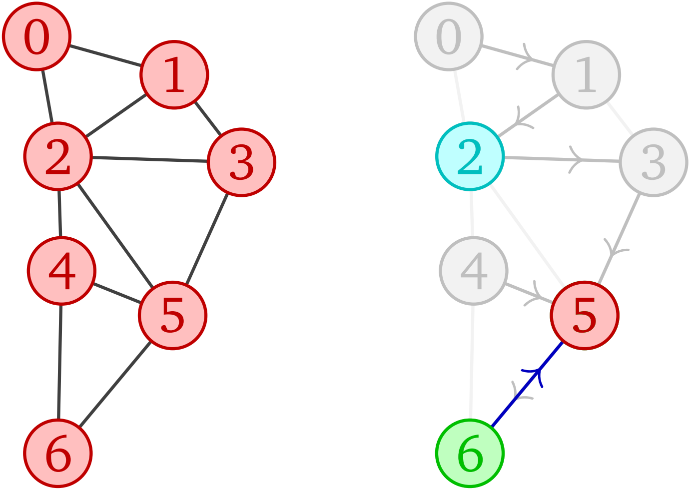
RS Example



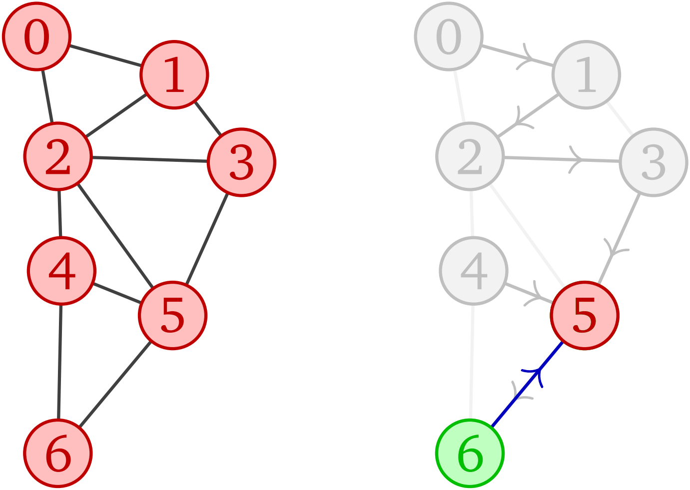
RS Example



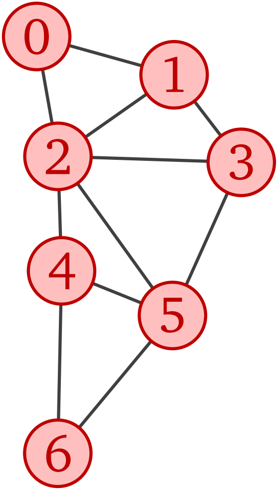
RS Example

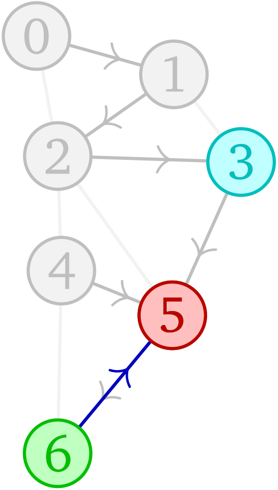


RS Example

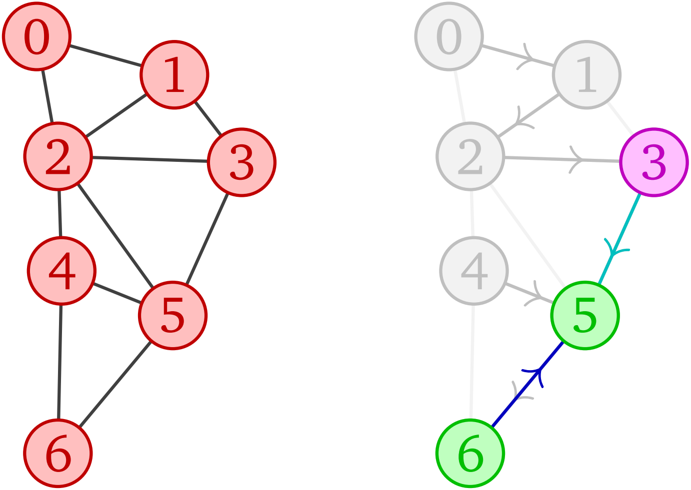


RS Example

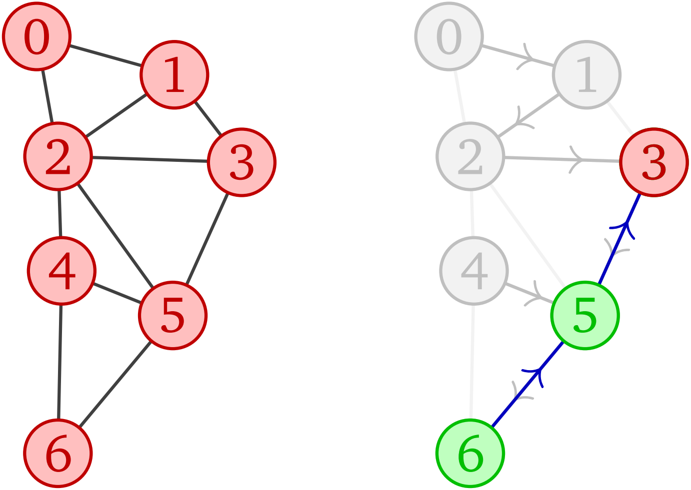




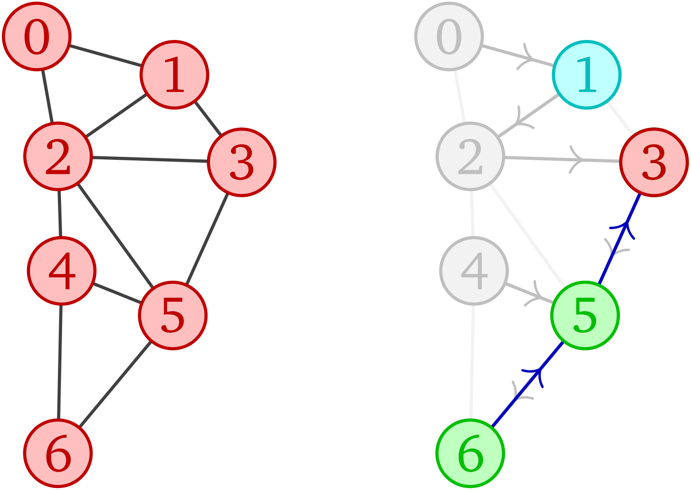
RS Example



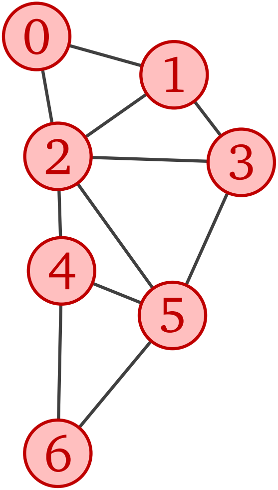
RS Example

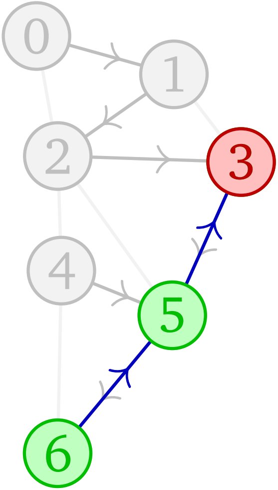


RS Example

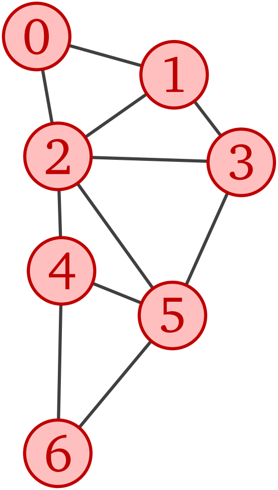


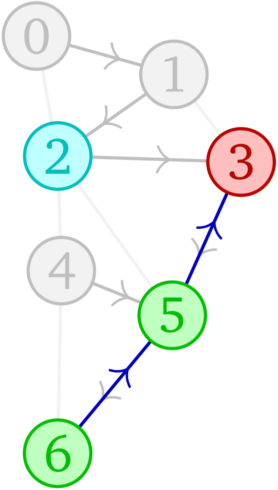
RS Example



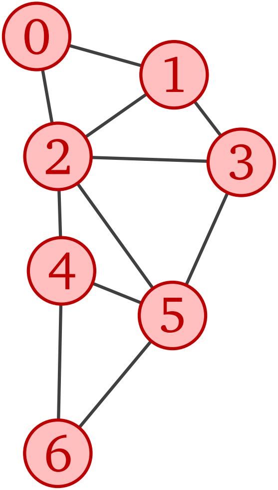


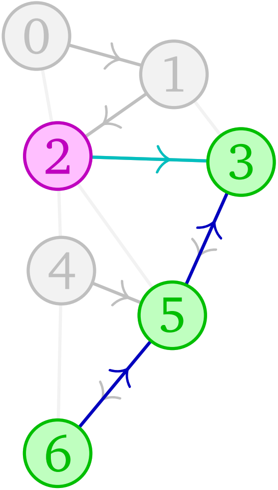
RS Example



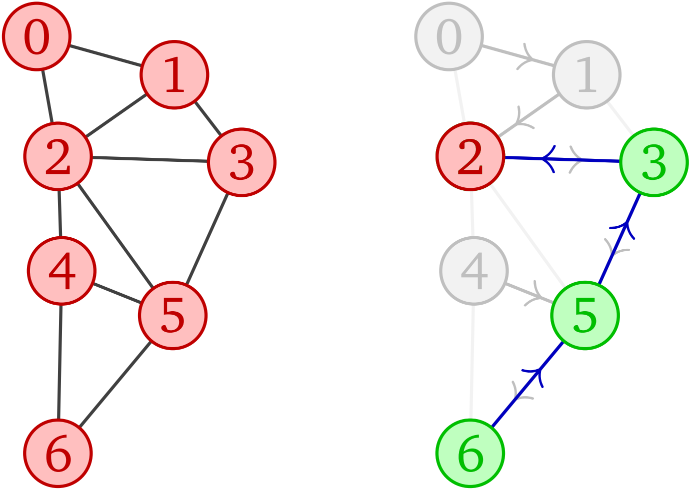


RS Example

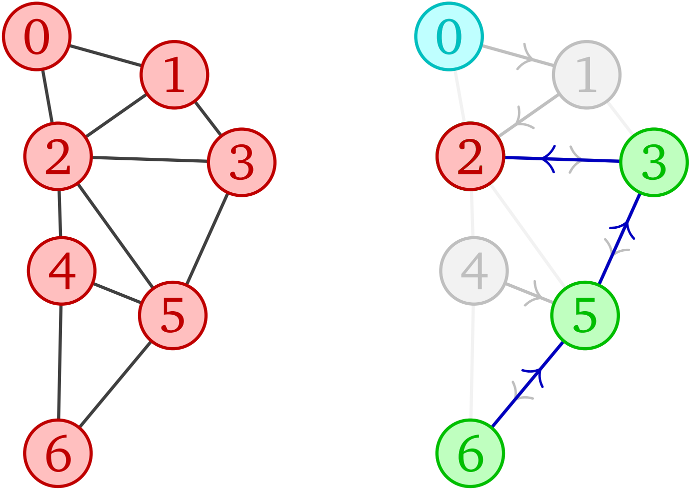




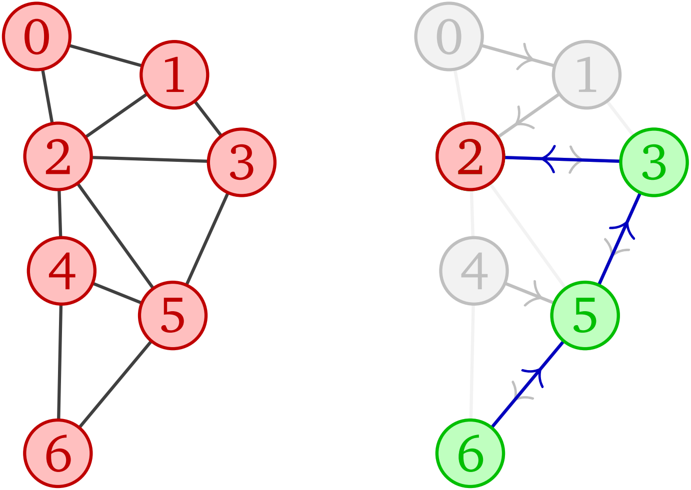
RS Example



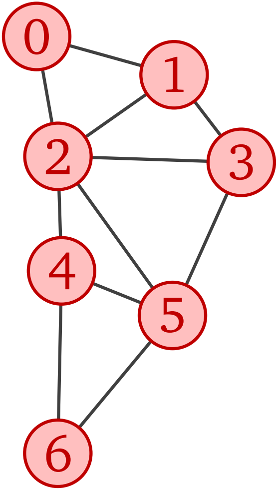
RS Example

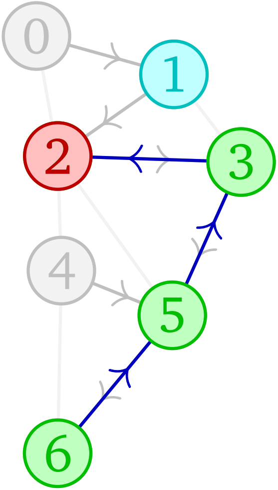


RS Example

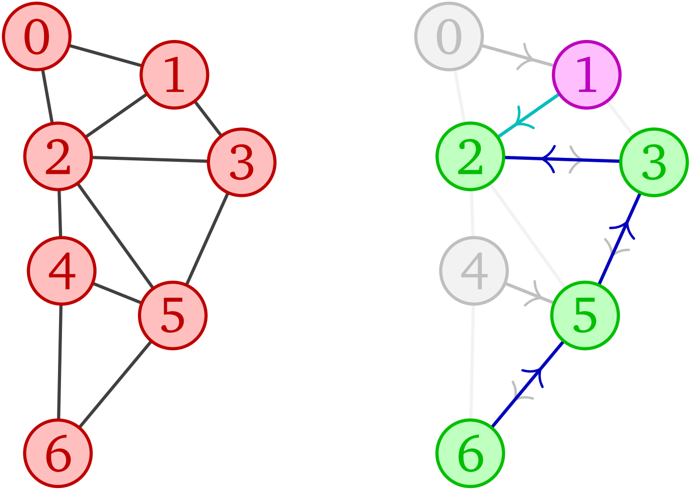


RS Example

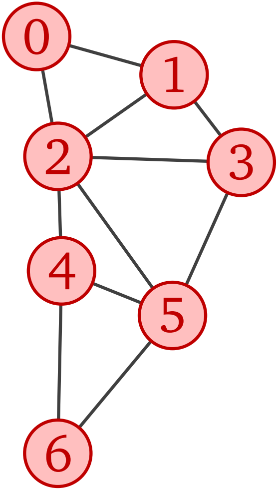


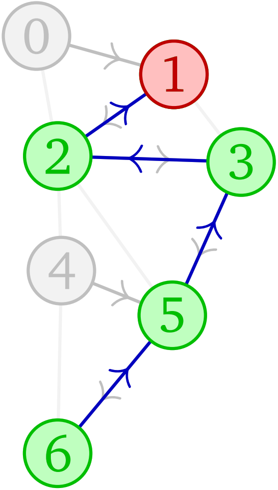


RS Example

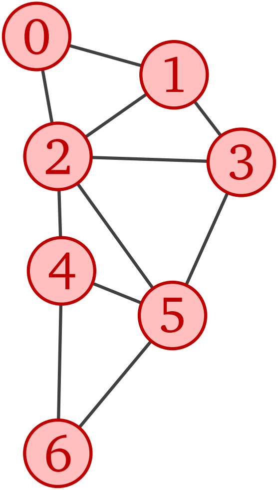


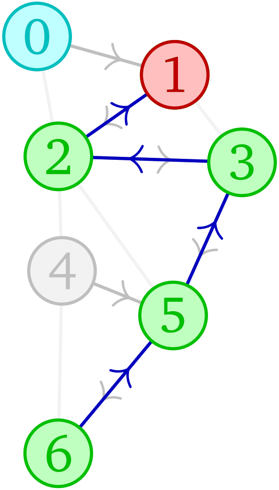
RS Example



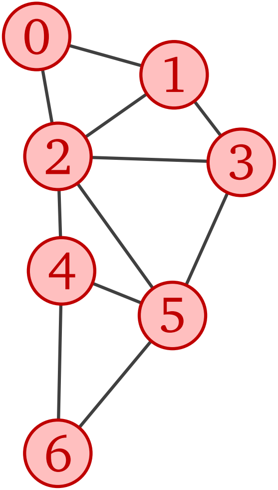


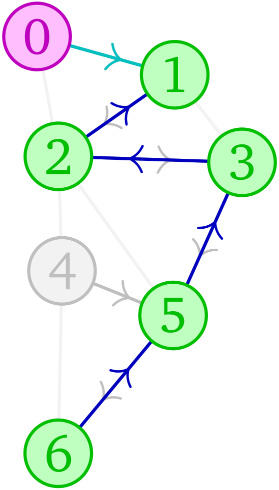
RS Example



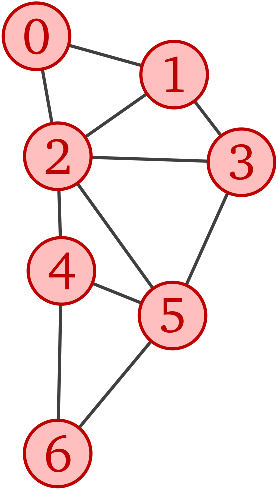


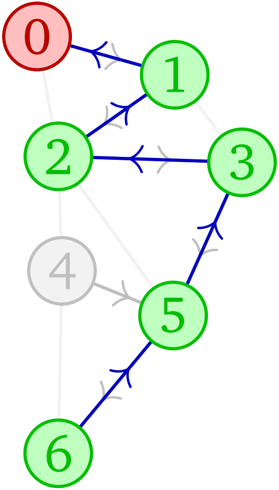
RS Example



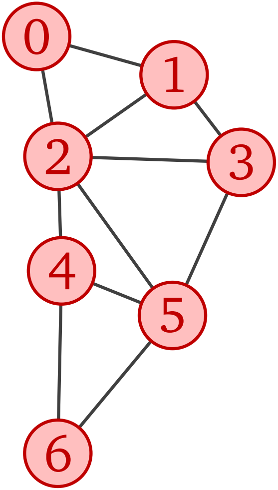


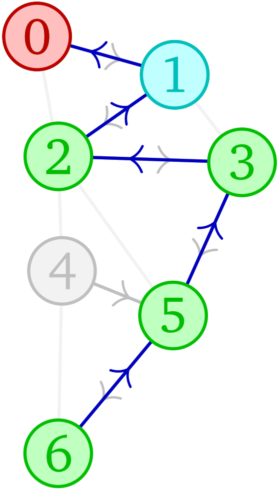
RS Example



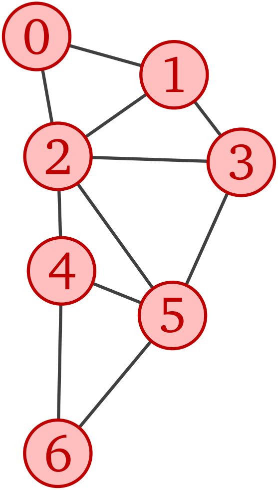


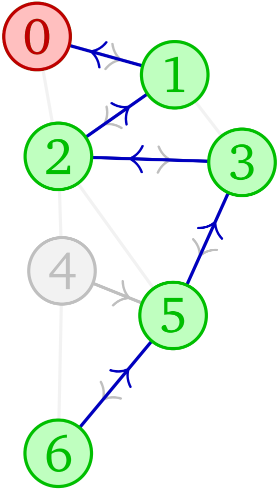
RS Example



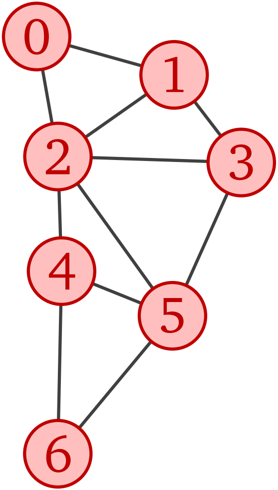


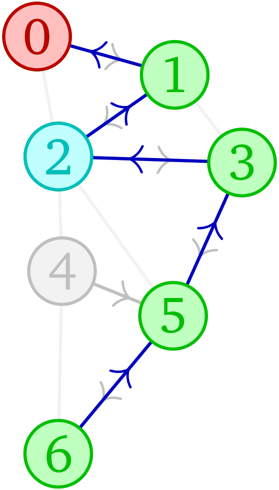
RS Example



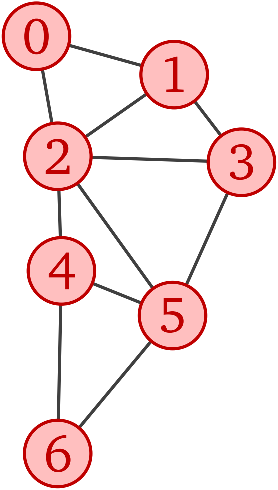


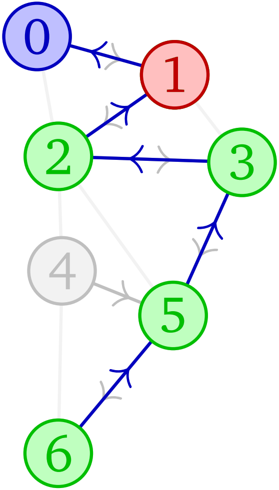
RS Example



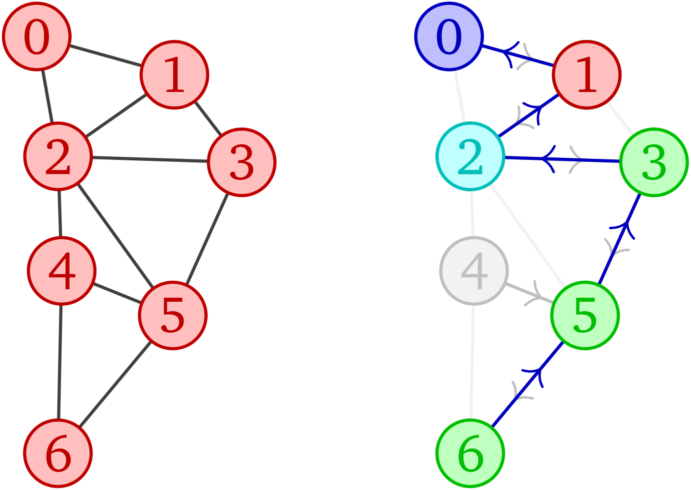


RS Example

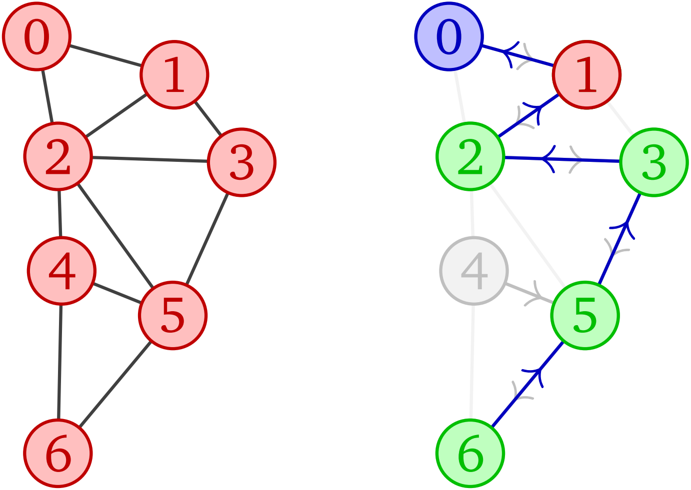




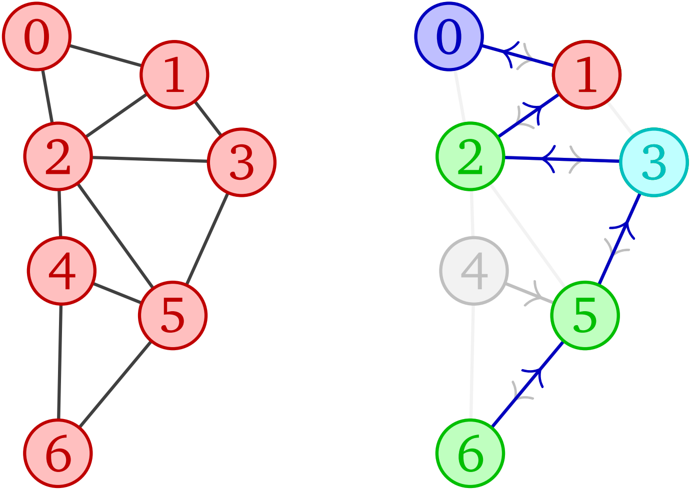
RS Example



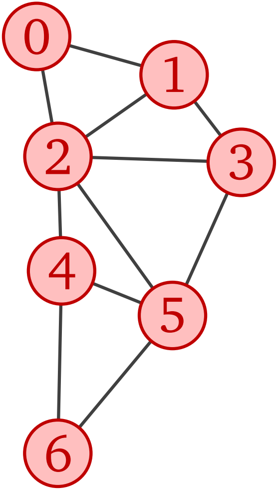
RS Example

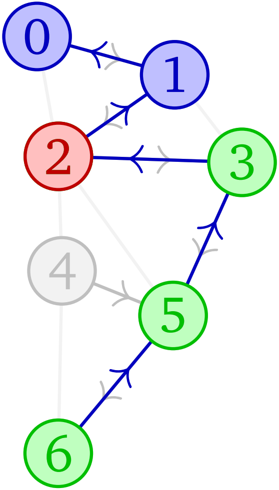


RS Example

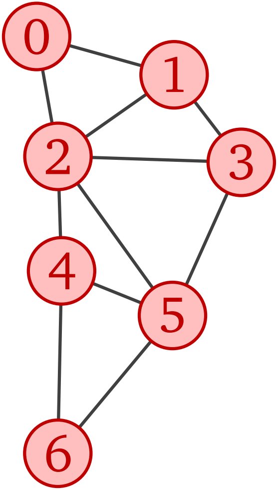


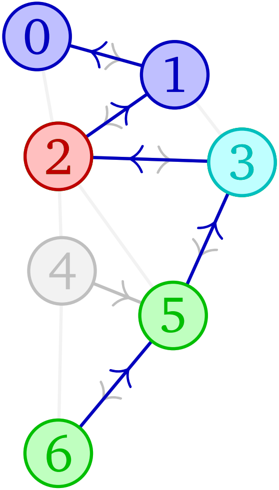
RS Example



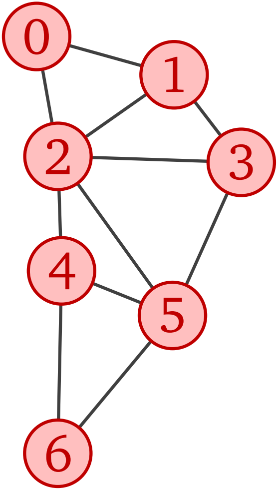


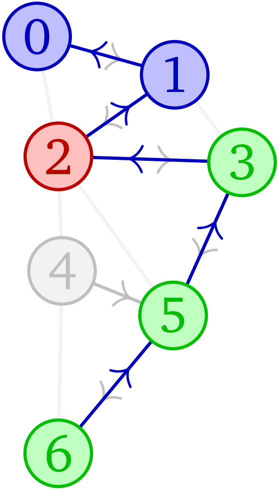
RS Example



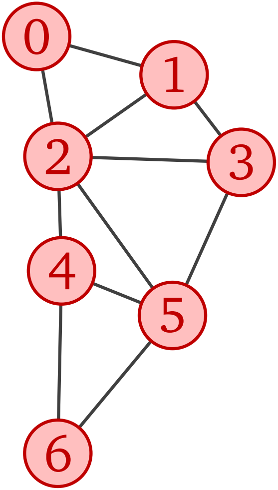


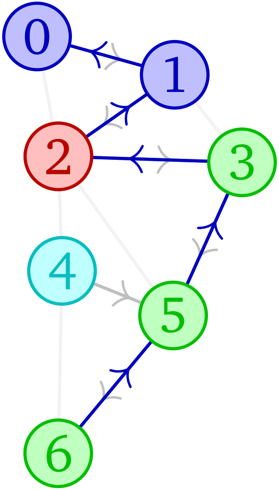
RS Example



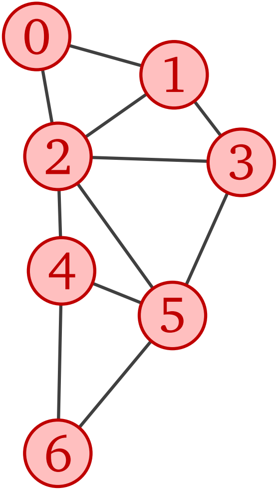


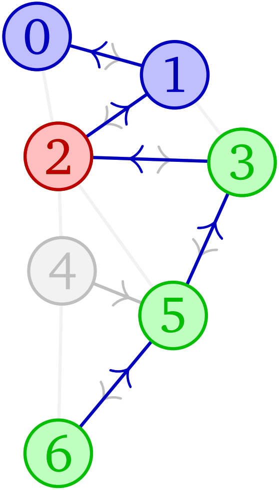
RS Example



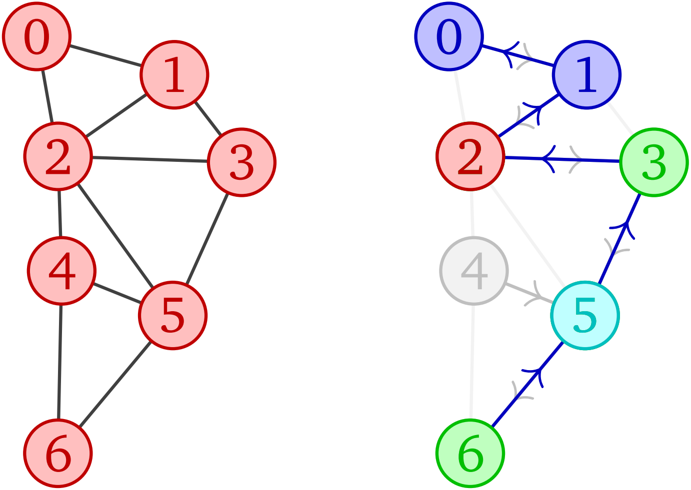


RS Example

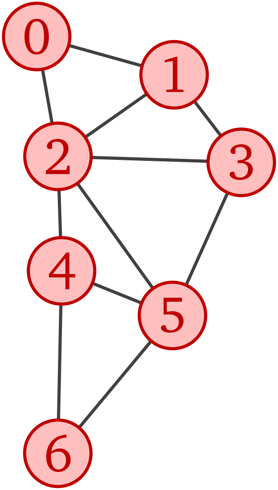


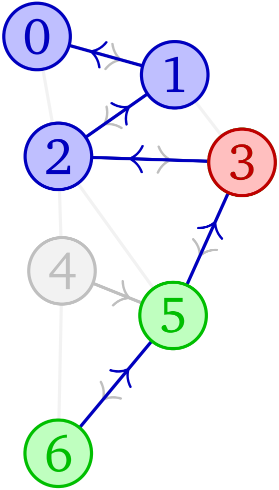


RS Example

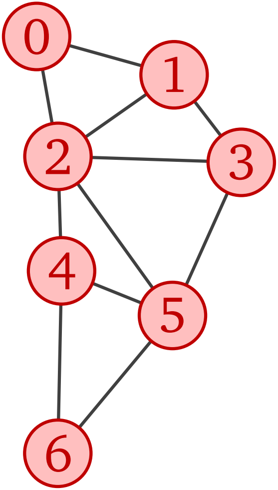


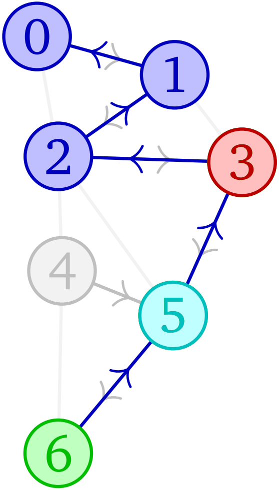
RS Example



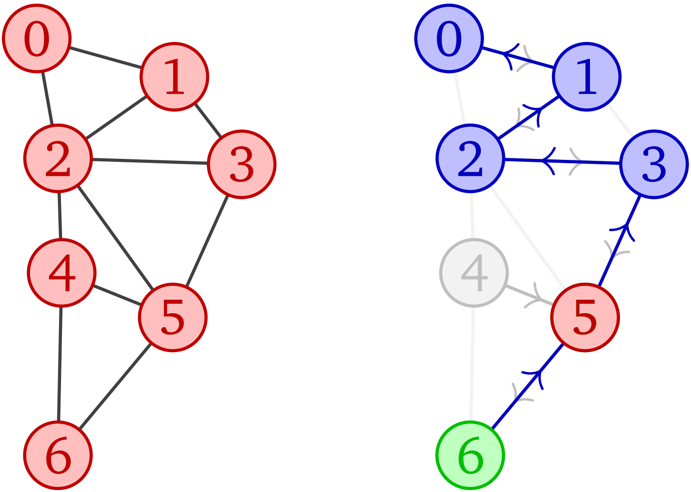


RS Example

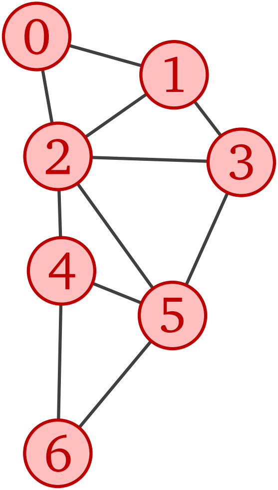


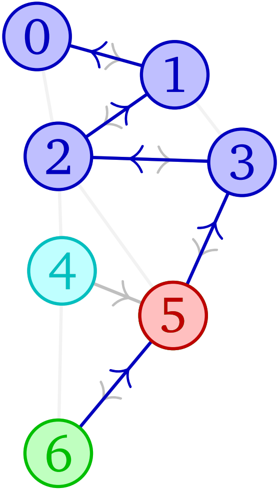


RS Example

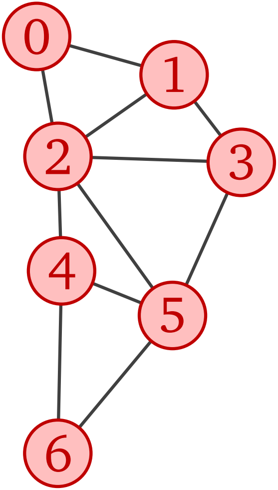


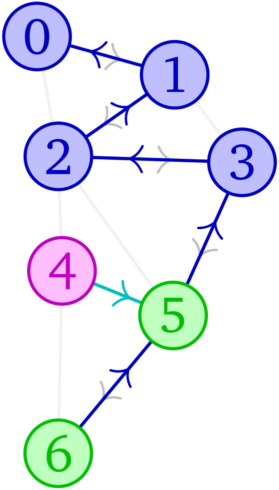
RS Example



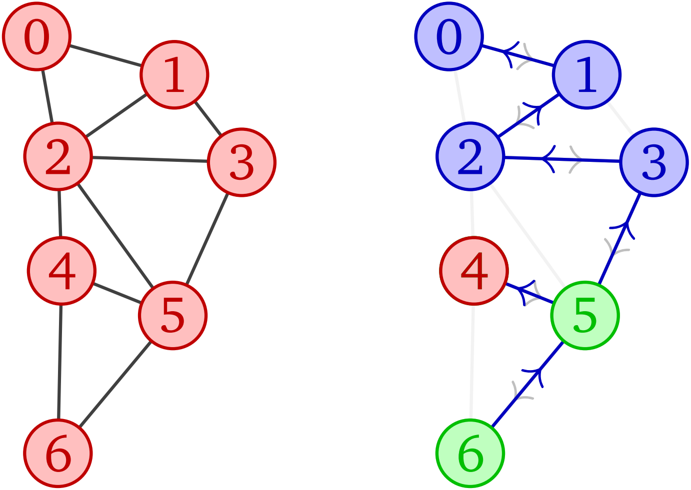


RS Example

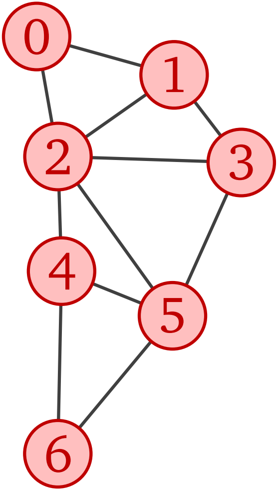


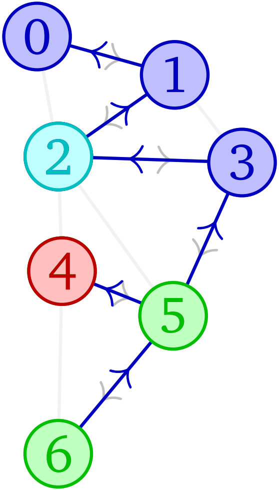


RS Example

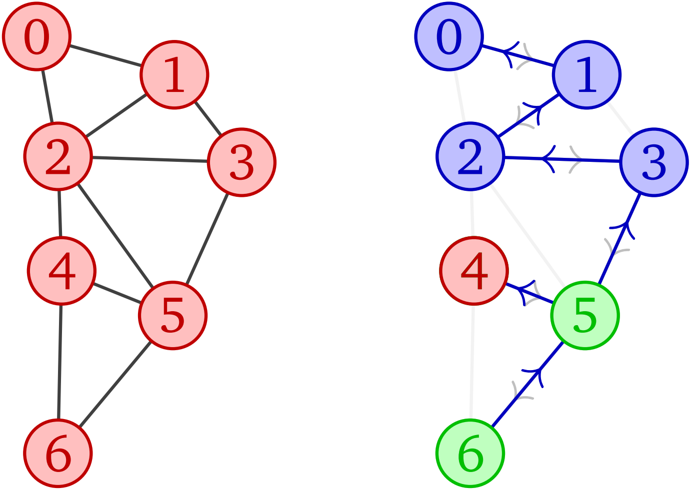


RS Example

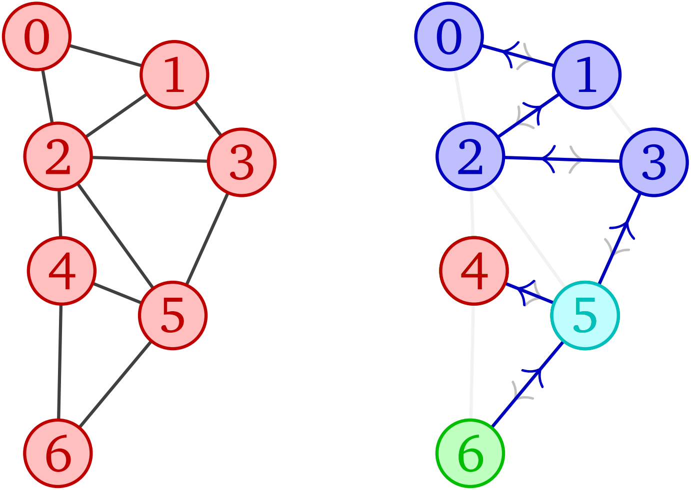




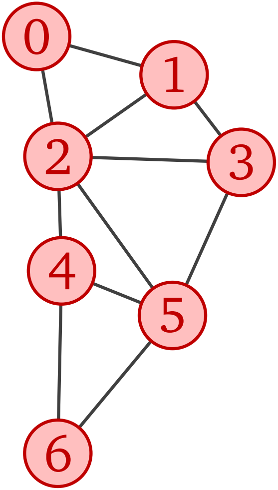
RS Example

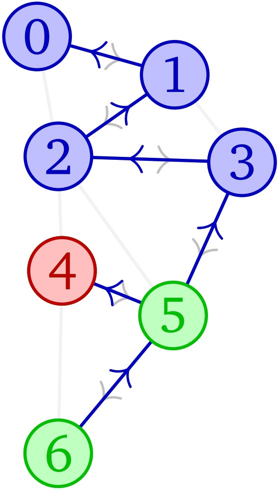


RS Example

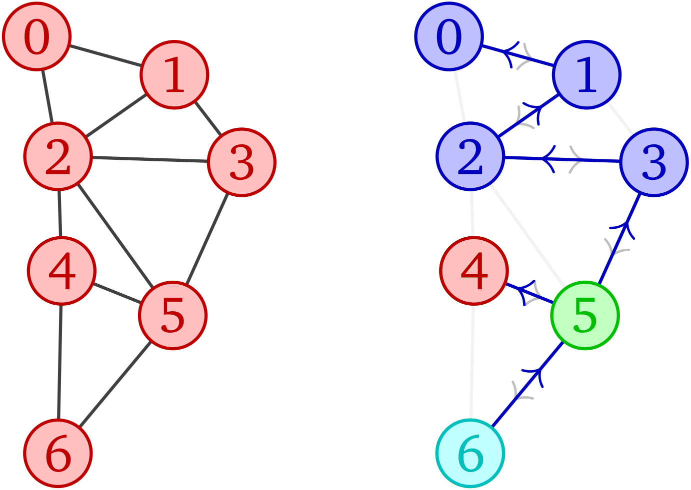


RS Example

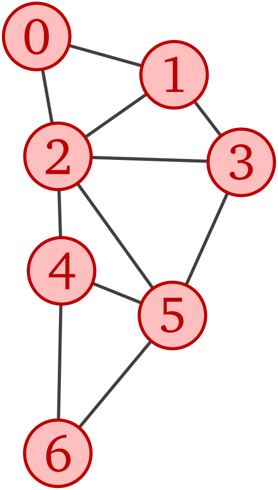


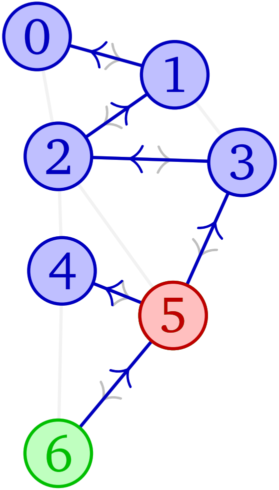


RS Example

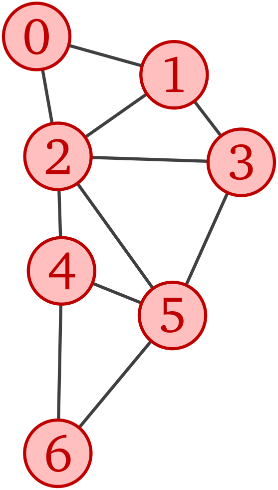


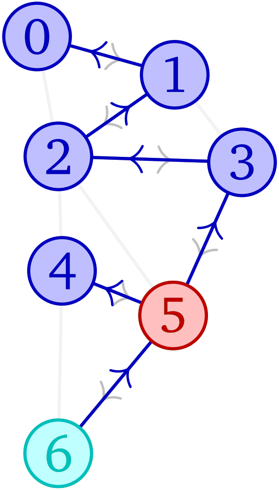
RS Example



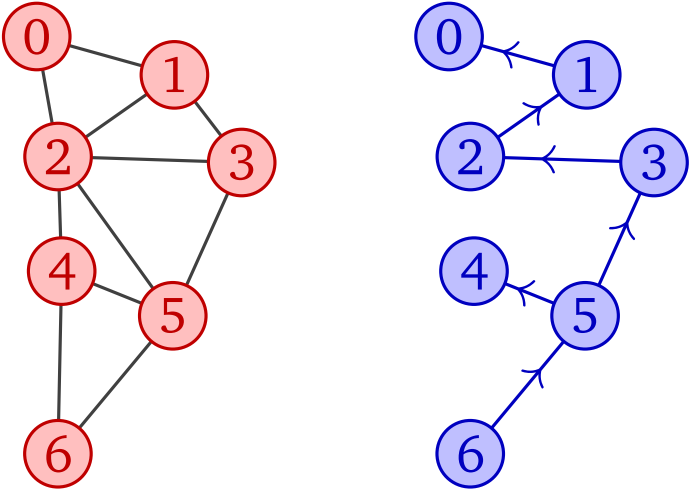


RS Example





RS Example

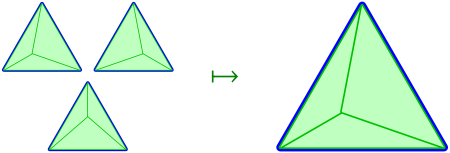


Reverse Search on Orbits

Canonical Representatives

Reverse Search on Orbits

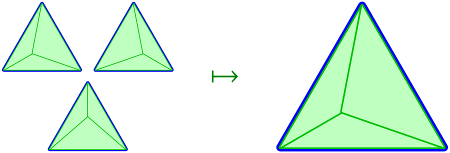
Function "G-Orbits $\longrightarrow$ Elements", e.g.,



Reverse Search on Orbits

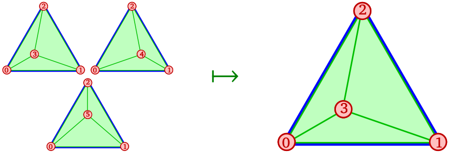
Canonical Representatives

Function "G-Orbits $\longrightarrow$ Elements", e.g.,



RS-Consistent Choice

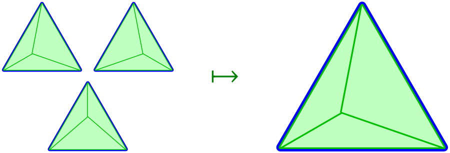
orbit $\mapsto$ objective-minimal sink in orbit, e.g.,



Reverse Search on Orbits

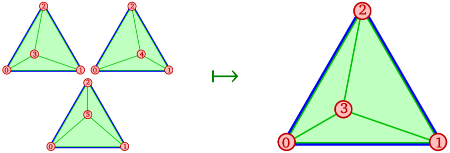
Canonical Representatives

Function “G-Orbits $\longrightarrow$ Elements”, e.g.,



RS-Consistent Choice

orbit $\mapsto$ objective-minimal sink in orbit, e.g.,



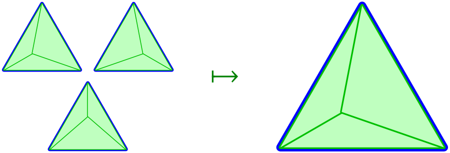
Pivoting Orbits

new pivot := canonical representative $\circ$ old pivot.

Reverse Search on Orbits

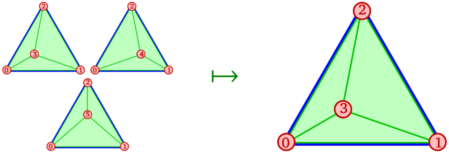
Canonical Representatives

Function “G-Orbits $\longrightarrow$ Elements”, e.g.,



RS-Consistent Choice

orbit $\mapsto$ objective-minimal sink in orbit, e.g.,



Pivoting Orbits

new pivot := canonical representative $\circ$ old pivot.

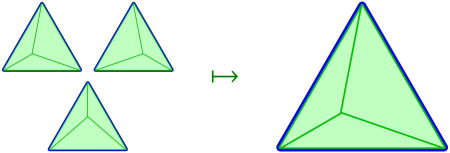
Result

Can enumerate orbits by RS on orbits [Imai et al. 2002].

Reverse Search on Orbits

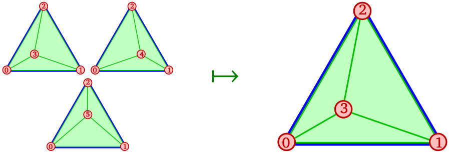
Canonical Representatives

Function "G-Orbits $\longrightarrow$ Elements", e.g.,



RS-Consistent Choice

orbit $\mapsto$ objective-minimal sink in orbit, e.g.,



Pivoting Orbits

new pivot := canonical representative $\circ$ old pivot.

Result

Can enumerate orbits by RS on orbits [Imai et al. 2002].

Bottleneck

Compute canonical representatives.

Agenda

Representation of Objects as Subsets

Representation of Objects as Subsets

Observation

Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Gain Subset Reverse Search (SRS):

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Gain Subset Reverse Search (SRS):

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Gain Subset Reverse Search (SRS):

- Lex-order is an objective with easy opt $\emptyset$

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Gain Subset Reverse Search (SRS):

- Lex-order is an objective with easy opt $\emptyset$
- Removing max-element $\rightsquigarrow$ easily invertible pivot

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Gain Subset Reverse Search (SRS):

- Lex-order is an objective with easy opt $\emptyset$
- Removing max-element $\rightsquigarrow$ easily invertible pivot

$\Longrightarrow$ SRS enumerates subsets.

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Gain Subset Reverse Search (SRS):

- Lex-order is an objective with easy opt $\emptyset$
- Removing max-element $\rightsquigarrow$ easily invertible pivot

$\Longrightarrow$ SRS enumerates subsets.

Crucial Need to recognize complete objects.

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Gain Subset Reverse Search (SRS):

- Lex-order is an objective with easy opt $\emptyset$
- Removing max-element $\rightsquigarrow$ easily invertible pivot

$\Longrightarrow$ SRS enumerates subsets.

Crucial Need to recognize complete objects.

Overhead SRS takes additional time, since

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Gain Subset Reverse Search (SRS):

- Lex-order is an objective with easy opt $\emptyset$
- Removing max-element $\rightsquigarrow$ easily invertible pivot

$\Longrightarrow$ SRS enumerates subsets.

Crucial Need to recognize complete objects.

Overhead SRS takes additional time, since

- all lex-leading subsets of objects are traversed

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Gain Subset Reverse Search (SRS):

- Lex-order is an objective with easy opt $\emptyset$
- Removing max-element $\rightsquigarrow$ easily invertible pivot

$\Longrightarrow$ SRS enumerates subsets.

Crucial Need to recognize complete objects.

Overhead SRS takes additional time, since

- all lex-leading subsets of objects are traversed
- there may be dead-ends w.r.t. lex-extension

Representation of Objects as Subsets

Observation Many objects have a representation as subsets S of $\{1, \ldots, n\}$.

Idea Build objects by lex-extension.

Gain Subset Reverse Search (SRS):

- Lex-order is an objective with easy opt $\emptyset$
- Removing max-element $\rightsquigarrow$ easily invertible pivot

$\Longrightarrow$ SRS enumerates subsets.

Crucial Need to recognize complete objects.

Overhead SRS takes additional time, since

- all lex-leading subsets of objects are traversed
- there may be dead-ends w.r.t. lex-extension
- containment in an object may be difficult to tell early

Generic Algorithm: Symmetric LSRS

Generic Algorithm: Symmetric LSRS

Observation

Subset S lex-min in its orbit
$\implies S \setminus \max S$ lex-min in its orbit.

Generic Algorithm: Symmetric LSRS

Observation

Subset S lex-min in its orbit
$\implies S \setminus \max S$ lex-min in its orbit.

Punch Line

canonical = lex-min $\implies$ canonicals connected

Generic Algorithm: Symmetric LSRS

Observation

Subset S lex-min in its orbit
$\implies S \setminus \max S$ lex-min in its orbit.

Punch Line

canonical = lex-min $\implies$ canonicals connected

Gain

Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:

Generic Algorithm: Symmetric LSRS

Observation

Subset S lex-min in its orbit
$\Longrightarrow S \setminus \max S$ lex-min in its orbit.

Punch Line

canonical = lex-min $\Longrightarrow$ canonicals connected

Gain

Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:

Generic Algorithm: Symmetric LSRS

Observation

Subset S lex-min in its orbit
$\Longrightarrow S \setminus \max S$ lex-min in its orbit.

Punch Line

canonical = lex-min $\Longrightarrow$ canonicals connected

Gain

Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:
Input: a subset S

Generic Algorithm: Symmetric LSRS

Observation Subset S lex-min in its orbit
$\implies S \setminus \max S$ lex-min in its orbit.

Punch Line canonical = lex-min $\implies$ canonicals connected

Gain Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:
Input: a subset S

- if S not lex-extendable to an object, return 0

Generic Algorithm: Symmetric LSRS

Observation Subset S lex-min in its orbit
$\implies S \setminus \max S$ lex-min in its orbit.

Punch Line canonical = lex-min $\implies$ canonicals connected

Gain Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:
Input: a subset S

- if S not lex-extendable to an object, return 0

Generic Algorithm: Symmetric LSRS

Observation Subset S lex-min in its orbit
$\Longrightarrow S \setminus \max S$ lex-min in its orbit.

Punch Line canonical = lex-min $\Longrightarrow$ canonicals connected

Gain Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:
Input: a subset S

- if S not lex-extendable to an object, return 0
- if S not lex-min in its orbit, return 0

Generic Algorithm: Symmetric LSRS

Observation Subset S lex-min in its orbit
$\implies S \setminus \max S$ lex-min in its orbit.

Punch Line canonical = lex-min $\implies$ canonicals connected

Gain Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:
Input: a subset S

- if S not lex-extendable to an object, return 0
- if S not lex-min in its orbit, return 0

Generic Algorithm: Symmetric LSRS

Observation

Subset S lex-min in its orbit
$\implies S \setminus \max S$ lex-min in its orbit.

Punch Line

canonical = lex-min $\implies$ canonicals connected

Gain

Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:
Input: a subset S

- if S not lex-extendable to an object, return 0
- if S not lex-min in its orbit, return 0
- if S is a complete object, return 1

Generic Algorithm: Symmetric LSRS

Observation Subset S lex-min in its orbit
$\implies S \setminus \max S$ lex-min in its orbit.

Punch Line canonical = lex-min $\implies$ canonicals connected

Gain Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:
Input: a subset S

- if S not lex-extendable to an object, return 0
- if S not lex-min in its orbit, return 0
- if S is a complete object, return 1
- for i from $\max S + 1, \dots, n$:

Generic Algorithm: Symmetric LSRS

Observation Subset S lex-min in its orbit
$\implies S \setminus \max S$ lex-min in its orbit.

Punch Line canonical = lex-min $\implies$ canonicals connected

Gain Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:
Input: a subset S

- if S not lex-extendable to an object, return 0
- if S not lex-min in its orbit, return 0
- if S is a complete object, return 1
- for i from $\max S + 1, \dots, n$:
  - increase counter by symLSRS($S \cup \{i\}$)

Generic Algorithm: Symmetric LSRS

Observation

Subset S lex-min in its orbit
$\Longrightarrow S \setminus \max S$ lex-min in its orbit.

Punch Line

canonical = lex-min $\Longrightarrow$ canonicals connected

Gain

Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:
Input: a subset S

- if S not lex-extendable to an object, return 0
- if S not lex-min in its orbit, return 0
- if S is a complete object, return 1
- for i from $\max S + 1, \ldots, n$:
  - increase counter by symLSRS($S \cup \{i\}$)
- return counter

Generic Algorithm: Symmetric LSRS

Observation Subset S lex-min in its orbit
$\Longrightarrow S \setminus \max S$ lex-min in its orbit.

Punch Line canonical = lex-min $\Longrightarrow$ canonicals connected

Gain Symmetric Lexicographic Subset Reverse Search (symLSRS) [equivalent: Pech & Reichard 2009]:
Input: a subset S

- if S not lex-extendable to an object, return 0
- if S not lex-min in its orbit, return 0
- if S is a complete object, return 1
- for i from $\max S + 1, \ldots, n$:
  - increase counter by symLSRS($S \cup \{i\}$)
- return counter

$\longrightarrow$ symLSRS($\emptyset$) lex-enumerates all orbit-lex-min objects

Agenda

Is a Subset Not Lex-Min in its Orbit?

Is a Subset Not Lex-Min in its Orbit?

Idea Exploit lex-min property of S to check $S \cup \{i\}$!

Is a Subset Not Lex-Min in its Orbit?

Idea Exploit lex-min property of S to check $S \cup \{i\}$!

Assumption Order of symmetry group G is managable.

Is a Subset Not Lex-Min in its Orbit?

Idea Exploit lex-min property of S to check $S \cup \{i\}$!

Assumption Order of symmetry group G is managable.

Local Data Store with each subset its critical-element table:

$$\mathrm{crit}_S : \begin{cases} G & \to & \{1, \ldots, n\} \cup \{\infty\}, \\ \pi & \mapsto & \min(S \triangle \pi(S)). \end{cases}$$

Is a Subset Not Lex-Min in its Orbit?

Idea Exploit lex-min property of S to check $S \cup \{i\}$!

Assumption Order of symmetry group G is managable.

Local Data Store with each subset its critical-element table:

$$\mathrm{crit}_S : \begin{cases} G & \longrightarrow & \{1, \ldots, n\} \cup \{\infty\}, \\ \pi & \longmapsto & \min(S \triangle \pi(S)). \end{cases}$$

Observation

$$\text{A symmetry } \pi \text{ lex-decreases a subset } S \iff \mathrm{crit}_S(\pi) \in \pi(S).$$

Ingredient I: Critical Elements

Ingredient I: Critical Elements

Theorem

Let S be a subset that is lex-min in its orbit.
Then for all $i \in \{\max S + 1, \ldots, n\}$ we have:

Ingredient I: Critical Elements

Theorem

Let S be a subset that is lex-min in its orbit.
Then for all $i \in \{\max S + 1, \ldots, n\}$ we have:

$S \cup \{i\}$ is not lex-min in its orbit

$$\Longleftrightarrow$$

there is a $\pi \in G$ with:

Ingredient I: Critical Elements

Theorem Let S be a subset that is lex-min in its orbit.
Then for all $i \in \{\max S + 1, \ldots, n\}$ we have:

$S \cup \{i\}$ is not lex-min in its orbit

$\Longleftrightarrow$

there is a $\pi \in G$ with:

- $\mathrm{crit}_S(\pi) = \infty$ and $\pi(i) < \max S$, or

Ingredient I: Critical Elements

Theorem

Let S be a subset that is lex-min in its orbit.
Then for all $i \in \{\max S + 1, \ldots, n\}$ we have:

$$S \cup \{i\} \text{ is not lex-min in its orbit}$$
$$\iff$$
$$\text{there is a } \pi \in G \text{ with:}$$

- $\mathrm{crit}_S(\pi) = \infty$ and $\pi(i) < \max S$, or
- $\mathrm{crit}_S(\pi) \neq \infty$ and $\pi(i) < \mathrm{crit}_S(\pi)$, or

Ingredient I: Critical Elements

Theorem Let S be a subset that is lex-min in its orbit. Then for all $i \in \{\max S + 1, \dots, n\}$ we have:

$$S \cup \{i\} \text{ is not lex-min in its orbit}$$

$$\iff$$

$$\text{there is a } \pi \in G \text{ with:}$$

- $\mathrm{crit}_S(\pi) = \infty$ and $\pi(i) < \max S$, or
- $\mathrm{crit}_S(\pi) \neq \infty$ and $\pi(i) < \mathrm{crit}_S(\pi)$, or
- $\mathrm{crit}_S(\pi) = \pi(i)$ and $\mathrm{crit}_{S \cup \{i\}}(\pi) \in \pi(S \cup \{i\})$.

Ingredient I: Critical Elements

Theorem Let S be a subset that is lex-min in its orbit. Then for all $i \in \{\max S + 1, \ldots, n\}$ we have:

$S \cup \{i\}$ is not lex-min in its orbit

$$\Longleftrightarrow$$

there is a $\pi \in G$ with:

- $\mathrm{crit}_S(\pi) = \infty$ and $\pi(i) < \max S$, or
- $\mathrm{crit}_S(\pi) \neq \infty$ and $\pi(i) < \mathrm{crit}_S(\pi)$, or
- $\mathrm{crit}_S(\pi) = \pi(i)$ and $\mathrm{crit}_{S\cup\{i\}}(\pi) \in \pi(S \cup \{i\})$.

Gain $\pi(S \cup \{i\})$ only needed if $\mathrm{crit}_S(\pi) = \pi(i)$

Ingredient I: Critical Elements

Theorem Let S be a subset that is lex-min in its orbit. Then for all $i \in \{\max S + 1, \dots, n\}$ we have:

$$S \cup \{i\} \text{ is not lex-min in its orbit}$$
$$\Longleftrightarrow$$
$$\text{there is a } \pi \in G \text{ with:}$$

- $\mathrm{crit}_S(\pi) = \infty$ and $\pi(i) < \max S$, or
- $\mathrm{crit}_S(\pi) \neq \infty$ and $\pi(i) < \mathrm{crit}_S(\pi)$, or
- $\mathrm{crit}_S(\pi) = \pi(i)$ and $\mathrm{crit}_{S \cup \{i\}}(\pi) \in \pi(S \cup \{i\})$.

Gain $\pi(S \cup \{i\})$ only needed if $\mathrm{crit}_S(\pi) = \pi(i)$
$\rightsquigarrow$ roughly $\frac{1}{n}$ of the cases (amortized)

Triangulations as Integer-Subsets

Triangulations as Integer-Subsets

Representation

For a point configuration of n points in rank r:

Triangulations as Integer-Subsets

Representation

For a point configuration of n points in rank r:

- S: the set of r-simplices, lex-ordered

Triangulations as Integer-Subsets

Representation

For a point configuration of n points in rank r:

- $\mathcal{S}$: the set of r-simplices, lex-ordered
- $\mathcal{F}$: the set of interior facets of r simplices, lex-ordered

Triangulations as Integer-Subsets

Representation

For a point configuration of n points in rank r:

- $\mathcal{S}$: the set of r-simplices, lex-ordered
- $\mathcal{F}$: the set of interior facets of r simplices, lex-ordered
- $n_s = |\mathcal{S}|$: no. of simplices

Triangulations as Integer-Subsets

Representation

For a point configuration of n points in rank r:

- $\mathcal{S}$: the set of r-simplices, lex-ordered
- $\mathcal{F}$: the set of interior facets of r simplices, lex-ordered
- $n_s = |\mathcal{S}|$: no. of simplices
- $n_f = |\mathcal{F}|$: no. of interior facets of simplices

Triangulations as Integer-Subsets

Representation

For a point configuration of n points in rank r:

- $\mathcal{S}$: the set of r-simplices, lex-ordered
- $\mathcal{F}$: the set of interior facets of r simplices, lex-ordered
- $n_s = |\mathcal{S}|$: no. of simplices
- $n_f = |\mathcal{F}|$: no. of interior facets of simplices
- $\mathrm{simp} : \{1, \dots, n_s\} \overset{\sim}{\leftrightarrow} \mathcal{S}$: s-index (order-preserving)

Triangulations as Integer-Subsets

Representation

For a point configuration of n points in rank r:

- $\mathcal{S}$: the set of r-simplices, lex-ordered
- $\mathcal{F}$: the set of interior facets of r simplices, lex-ordered
- $n_s = |\mathcal{S}|$: no. of simplices
- $n_f = |\mathcal{F}|$: no. of interior facets of simplices
- $\mathrm{simp} : \{1, \ldots, n_s\} \overset{\sim}{\leftrightarrow} \mathcal{S}$: s-index (order-preserving)
- $\mathrm{facet} : \{1, \ldots, n_f\} \overset{\sim}{\leftrightarrow} \mathcal{F}$: f-index (order-preserving)

Triangulations as Integer-Subsets

Representation

For a point configuration of n points in rank r:

- $\mathcal{S}$: the set of r-simplices, lex-ordered
- $\mathcal{F}$: the set of interior facets of r simplices, lex-ordered
- $n_s = |\mathcal{S}|$: no. of simplices
- $n_f = |\mathcal{F}|$: no. of interior facets of simplices
- $\text{simp} : \{1, \ldots, n_s\} \xrightarrow{\sim} \mathcal{S}$: s-index (order-preserving)
- $\text{facet} : \{1, \ldots, n_f\} \xrightarrow{\sim} \mathcal{F}$: f-index (order-preserving)
- $\mathcal{T} = \{\text{simp}(s) : s \in T\}$ for $T \subseteq \{1, \ldots, n_s\}$

Triangulations as Integer-Subsets

Representation

For a point configuration of n points in rank r:

- $\mathcal{S}$: the set of r-simplices, lex-ordered
- $\mathcal{F}$: the set of interior facets of r simplices, lex-ordered
- $n_s = |\mathcal{S}|$: no. of simplices
- $n_f = |\mathcal{F}|$: no. of interior facets of simplices
- $\mathrm{simp} : \{1, \ldots, n_s\} \overset{\sim}{\leftrightarrow} \mathcal{S}$: s-index (order-preserving)
- $\mathrm{facet} : \{1, \ldots, n_f\} \overset{\sim}{\leftrightarrow} \mathcal{F}$: f-index (order-preserving)
- $\mathcal{T} = \{\mathrm{simp}(s) : s \in T\}$ for $T \subseteq \{1, \ldots, n_s\}$
- $T = \{\text{s-index}(S) : S \in \mathcal{T}\}$ for $\mathcal{T} \subseteq \mathcal{S}$

Triangulations as Integer-Subsets

Representation

For a point configuration of n points in rank r:

- $\mathcal{S}$: the set of r-simplices, lex-ordered
- $\mathcal{F}$: the set of interior facets of r simplices, lex-ordered
- $n_s = |\mathcal{S}|$: no. of simplices
- $n_f = |\mathcal{F}|$: no. of interior facets of simplices
- $\mathrm{simp} : \{1, \dots, n_s\} \overset{\sim}{\leftrightarrow} \mathcal{S}$: s-index (order-preserving)
- $\mathrm{facet} : \{1, \dots, n_f\} \overset{\sim}{\leftrightarrow} \mathcal{F}$: f-index (order-preserving)
- $\mathcal{T} = \{\mathrm{simp}(s) : s \in T\}$ for $T \subseteq \{1, \dots, n_s\}$
- $T = \{\text{s-index}(S) : S \in \mathcal{T}\}$ for $\mathcal{T} \subseteq \mathcal{S}$

Convention

All $s \in \{1, \dots, n_s\}$ and $f \in \{1, \dots, n_f\}$ are called simplices and facets, resp.
All $T \subseteq \{1, \dots, n_s\}$ with pairwise proper intersections are called partial triangulations.

Is a Subset Not Lex-Extendable?

Is a Subset Not Lex-Extendable?

Extendability Check

[Ruppert & Seidel 1992]:

Extendability of partial triangulations is NP-complete.

Is a Subset Not Lex-Extendable?

Extendability Check

[Ruppert & Seidel 1992]:
Extendability of partial triangulations is NP-complete.

Observation

Each interior facet must be covered by additional simplices to complete a triangulation.

Is a Subset Not Lex-Extendable?

Extendability Check

[Ruppert & Seidel 1992]:
Extendability of partial triangulations is NP-complete.

Observation

Each interior facet must be covered by additional simplices to complete a triangulation.

Global Data

Preprocess for each simplex s

Is a Subset Not Lex-Extendable?

Extendability Check

[Ruppert & Seidel 1992]:
Extendability of partial triangulations is NP-complete.

Observation

Each interior facet must be covered by additional simplices to complete a triangulation.

Global Data

Preprocess for each simplex s

- its interior facets $\rightsquigarrow \mathcal{F}(s)$

Is a Subset Not Lex-Extendable?

Extendability Check

[Ruppert & Seidel 1992]:
Extendability of partial triangulations is NP-complete.

Observation

Each interior facet must be covered by additional simplices to complete a triangulation.

Global Data

Preprocess for each simplex s

- its interior facets $\rightsquigarrow \mathcal{F}(s)$
- all simplices with proper intersection $\rightsquigarrow \mathcal{A}(s)$

Is a Subset Not Lex-Extendable?

Extendability Check
[Ruppert & Seidel 1992]:
Extendability of partial triangulations is NP-complete.

Observation
Each interior facet must be covered by additional simplices to complete a triangulation.

Global Data
Preprocess for each simplex s

- its interior facets $\rightsquigarrow \mathcal{F}(s)$
- all simplices with proper intersection $\rightsquigarrow \mathcal{A}(s)$

Local Data
With each partial triangulation T store in lex-order:

Is a Subset Not Lex-Extendable?

Extendability Check

[Ruppert & Seidel 1992]:
Extendability of partial triangulations is NP-complete.

Observation

Each interior facet must be covered by additional simplices to complete a triangulation.

Global Data

Preprocess for each simplex s

- its interior facets $\rightsquigarrow \mathcal{F}(s)$
- all simplices with proper intersection $\rightsquigarrow \mathcal{A}(s)$

Local Data

With each partial triangulation T store in lex-order:

Is a Subset Not Lex-Extendable?

Extendability Check
[Ruppert & Seidel 1992]:
Extendability of partial triangulations is NP-complete.

Observation
Each interior facet must be covered by additional simplices to complete a triangulation.

Global Data
Preprocess for each simplex s

- its interior facets $\rightsquigarrow \mathcal{F}(s)$
- all simplices with proper intersection $\rightsquigarrow \mathcal{A}(s)$

Local Data
With each partial triangulation T store in lex-order:

- the free interior facets $\rightsquigarrow \mathcal{F}(T)$

Is a Subset Not Lex-Extendable?

Extendability Check
[Ruppert & Seidel 1992]:
Extendability of partial triangulations is NP-complete.

Observation
Each interior facet must be covered by additional simplices to complete a triangulation.

Global Data
Preprocess for each simplex s

- its interior facets $\rightsquigarrow \mathcal{F}(s)$
- all simplices with proper intersection $\rightsquigarrow \mathcal{A}(s)$

Local Data
With each partial triangulation T store in lex-order:

- the free interior facets $\rightsquigarrow \mathcal{F}(T)$
- the greater simplices that intersect properly $\rightsquigarrow \mathcal{A}(T)$

Is a Subset Not Lex-Extendable?

Extendability Check
[Ruppert & Seidel 1992]:
Extendability of partial triangulations is NP-complete.

Observation
Each interior facet must be covered by additional simplices to complete a triangulation.

Global Data
Preprocess for each simplex s

- its interior facets $\rightsquigarrow \mathcal{F}(s)$
- all simplices with proper intersection $\rightsquigarrow \mathcal{A}(s)$

Local Data
With each partial triangulation T store in lex-order:

- the free interior facets $\rightsquigarrow \mathcal{F}(T)$
- the greater simplices that intersect properly $\rightsquigarrow \mathcal{A}(T)$

Observation
A partial triangulation T is not lex-extendable

$$\Longleftarrow$$

there is $\{f \in \mathcal{F}(T)\}$ not contained in any $\{s \in \mathcal{A}(T)\}$.

Ingredient II: Lex-pruning/Lex-Breaking

Ingredient II: Lex-pruning/Lex-Breaking

Theorem

A partial triangulation T with free interior facets $\mathcal{F}(T)$ and properly intersecting greater simplices $\mathcal{A}(T)$ is not lex-extendable to a triangulation

$$\Longleftarrow$$

$$\min\{f \in \mathcal{F}(T)\} < \min\left\{\mathcal{F}(\min\{s \in \mathcal{A}(T)\})\right\}.$$

Ingredient II: Lex-pruning/Lex-Breaking

Theorem

A partial triangulation T with free interior facets $\mathcal{F}(T)$ and properly intersecting greater simplices $\mathcal{A}(T)$ is not lex-extendable to a triangulation

$$\Longleftarrow$$

$$\min\{f \in \mathcal{F}(T)\} < \min\left\{\mathcal{F}(\min\{s \in \mathcal{A}(T)\})\right\}.$$

Theorem

A partial triangulation T with free interior facets $\mathcal{F}(T)$ and properly intersecting greater simplices $\mathcal{A}(T)$ is not lex-extendable to a triangulation starting with some $s' \geq s$ in $\mathcal{A}(T)$

$$\Longleftarrow$$

$$\min\{f \in \mathcal{F}(T)\} < \min\left\{\mathcal{F}(s)\right\}.$$

Ingredient II: Lex-pruning/Lex-Breaking

Theorem

A partial triangulation T with free interior facets $\mathcal{F}(T)$ and properly intersecting greater simplices $\mathcal{A}(T)$ is not lex-extendable to a triangulation

$$\Longleftarrow$$

$$\min\{f \in \mathcal{F}(T)\} < \min\Big\{\mathcal{F}(\min\{s \in \mathcal{A}(T)\})\Big\}.$$

Theorem

A partial triangulation T with free interior facets $\mathcal{F}(T)$ and properly intersecting greater simplices $\mathcal{A}(T)$ is not lex-extendable to a triangulation starting with some $s' \geq s$ in $\mathcal{A}(T)$

$$\Longleftarrow$$

$$\min\{f \in \mathcal{F}(T)\} < \min\Big\{\mathcal{F}(s)\Big\}.$$

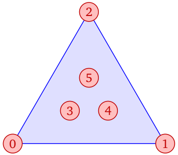
Gain

One integer comparison instead of many subset tests.

Effectivity

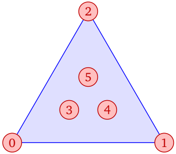
Effectivity

Comparison (with lex-breaking):

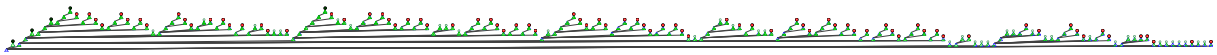


Effectivity

Comparison (with lex-breaking):

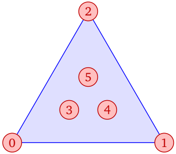


No Pruning

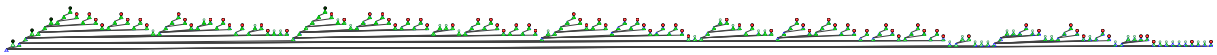


Effectivity

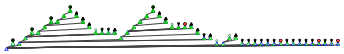
Comparison (with lex-breaking):



No Pruning

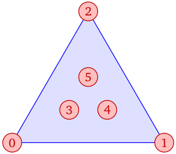


Full Pruning

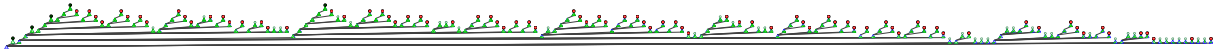


Effectivity

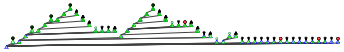
Comparison (with lex-breaking):



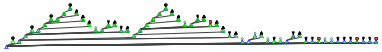
No Pruning



Full Pruning



Lex Pruning



Computational Results for Triangulations

Computational Results for Triangulations

MPTOPCOM
Flip-Based
CPU Times
(16/40 Threads)

[Jordan & Joswig & Kastner 2018]

Point Conf.	# Triang's	# Orbits	CPU time [hh:mm:ss]
$[0, 1]^4$	92,487,256	247,451	00:01:56
$3D_3$ (reg./full/output)	22,201,684,367	925,148,763	96:00:00

Computational Results for Triangulations

MPTOPCOM
Flip-Based
CPU Times
(16/40 Threads)

[Jordan & Joswig & Kastner 2018]

Point Conf.	# Triang's	# Orbits	CPU time [hh:mm:ss]
$[0,1]^4$	92,487,256	247,451	00:01:56
$3D_3$ (reg./full/output)	22,201,684,367	925,148,763	96:00:00

TOPCOM 1.0.8
Subset-Based
CPU Times
(16 Threads)

[R. 2022]

Point Conf.	# Triang's	# Orbits	CPU time [hh:mm:ss]
$[0,1]^4$	92,487,256	247,451	00:00:04
$3D_3$ (output)	22,201,684,367	925,148,763	01:05:11
$3D_3$ (count)	22,201,684,367	925,148,763	00:21:02
$3D_3$ (regular)	21,861,522,799	910,974,879	20:21:53
$3D_3$ (full)	511,052,427	21,302,400	00:01:01
$3D_3$ (unimod.)	346,903,379	14,459,488	00:00:39
Dodecahedron	1,533,079,037,570	12,775,757,027	11:11:48
Pyritohedron	32,734,029,351,118	1,363,918,758,719	692:30:04
$\Delta_5 \times \Delta_3$	442,472,050,753,920	25,606,173,722	1313:57:17 (M1Max8t)

Bonus Track I

Bonus Track I

For Raman

Triangulations with only simplices of min. vol.
of generalized hypersimplices [Manecke et al. 2020]:

Bonus Track I

For Raman

Triangulations with only simplices of min. vol.
of generalized hypersimplices [Manecke et al. 2020]:

- $\Delta(5, 1, 3)$ has 27,780 (250 classes)

Bonus Track I

For Raman

Triangulations with only simplices of min. vol. of generalized hypersimplices [Manecke et al. 2020]:

- $\Delta(5,1,3)$ has 27,780 (250 classes)
- $\Delta(5,1,4)$ has 5 (1 class)

Bonus Track I

For Raman

Triangulations with only simplices of min. vol.
of generalized hypersimplices [Manecke et al. 2020]:

- $\Delta(5, 1, 3)$ has 27,780 (250 classes)
- $\Delta(5, 1, 4)$ has 5 (1 class)
- $\Delta(6, 1, 3)$ has more than 245,074,320 (340,381 classes)

Bonus Track I

For Raman

Triangulations with only simplices of min. vol. of generalized hypersimplices [Manecke et al. 2020]:

- $\Delta(5, 1, 3)$ has 27,780 (250 classes)
- $\Delta(5, 1, 4)$ has 5 (1 class)
- $\Delta(6, 1, 3)$ has more than 245,074,320 (340,381 classes)
- $\Delta(6, 1, 4)$ has more than 249,295,320 (347,613 classes)

Bonus Track I

For Raman

Triangulations with only simplices of min. vol. of generalized hypersimplices [Manecke et al. 2020]:

- $\Delta(5, 1, 3)$ has 27,780 (250 classes)
- $\Delta(5, 1, 4)$ has 5 (1 class)
- $\Delta(6, 1, 3)$ has more than 245,074,320 (340,381 classes)
- $\Delta(6, 1, 4)$ has more than 249,295,320 (347,613 classes)
- $\Delta(6, 2, 4)$ has more than 7,248,961,080 (10,068,279 classes)

Bonus Track II

Bonus Track II

Other Results

Enumeration of (co-)circuits (different lex-min check):

Bonus Track II

Other Results

Enumeration of (co-)circuits (different lex-min check):

- $[0,1]^8$ has
  38,636,185,528,212,416 circuits in 3,858,105,362 classes
  (CPU: 163:37:00)
  (asked by Lisa Lamberti and Komei Fukuda)

Bonus Track II

Other Results

Enumeration of (co-)circuits (different lex-min check):

- $[0, 1]^8$ has
  38,636,185,528,212,416 circuits in 3,858,105,362 classes
  (CPU: 163:37:00)
  (asked by Lisa Lamberti and Komei Fukuda)
- $[0,1]^9$ has
  448,691,419,804,586 cocircuits in 3,899,720 classes
  (CPU: 13:30:12)
  (extends [Aichholzer & Aurnhammer 1996])

Bonus Track II

Other Results

Enumeration of (co-)circuits (different lex-min check):

- $[0, 1]^8$ has
  38,636,185,528,212,416 circuits in 3,858,105,362 classes
  (CPU: 163:37:00)
  (asked by Lisa Lamberti and Komei Fukuda)
- $[0,1]^9$ has
  448,691,419,804,586 cocircuits in 3,899,720 classes
  (CPU: 13:30:12)
  (extends [Aichholzer & Aurnhammer 1996])

Sideline

Necessary conditions for lex-extendability

Bonus Track II

Other Results

Enumeration of (co-)circuits (different lex-min check):

- $[0, 1]^8$ has 38,636,185,528,212,416 circuits in 3,858,105,362 classes (CPU: 163:37:00) (asked by Lisa Lamberti and Komei Fukuda)
- $[0,1]^9$ has 448,691,419,804,586 cocircuits in 3,899,720 classes (CPU: 13:30:12) (extends [Aichholzer & Aurnhammer 1996])

Sideline

Necessary conditions for lex-extendability

- found for cocircuits

Bonus Track II

Other Results

Enumeration of (co-)circuits (different lex-min check):

- $[0,1]^8$ has
  38,636,185,528,212,416 circuits in 3,858,105,362 classes
  (CPU: 163:37:00)
  (asked by Lisa Lamberti and Komei Fukuda)
- $[0,1]^9$ has
  448,691,419,804,586 cocircuits in 3,899,720 classes
  (CPU: 13:30:12)
  (extends [Aichholzer & Aurnhammer 1996])

Sideline

Necessary conditions for lex-extendability

- found for cocircuits
- but not so far for circuits.

Agenda

Conclusions/Questions

Conclusions/Questions

Conclusions

Enumeration of orbits of triangulations accelerated by
“**Geometry meets Combinatorics**”:

Conclusions/Questions

Conclusions

Enumeration of orbits of triangulations accelerated by
"**Geometry meets Combinatorics**":

- critical-element tables for lex-min check

Conclusions/Questions

Conclusions

Enumeration of orbits of triangulations accelerated by "**Geometry meets Combinatorics**":

- critical-element tables for lex-min check
- minimal-element comparison for lex-extendability check

Conclusions/Questions

Conclusions

Enumeration of orbits of triangulations accelerated by
“Geometry meets Combinatorics”:

- critical-element tables for lex-min check
- minimal-element comparison for lex-extendability check

Questions

Potential further research:

Conclusions/Questions

Conclusions

Enumeration of orbits of triangulations accelerated by
"**Geometry meets Combinatorics**":

- critical-element tables for lex-min check
- minimal-element comparison for lex-extendability check

Questions

Potential further research:

- Investigate the complexity of symLSRS.

Conclusions/Questions

Conclusions

Enumeration of orbits of triangulations accelerated by "**Geometry meets Combinatorics**":

- critical-element tables for lex-min check
- minimal-element comparison for lex-extendability check

Questions

Potential further research:

- Investigate the complexity of symLSRS.
- Apply symLSRS to more examples.

Conclusions/Questions

Conclusions

Enumeration of orbits of triangulations accelerated by
"**Geometry meets Combinatorics**":

- critical-element tables for lex-min check
- minimal-element comparison for lex-extendability check

Questions

Potential further research:

- Investigate the complexity of symLSRS.
- Apply symLSRS to more examples.
- Represent flip-graph exploration in terms of subsets.