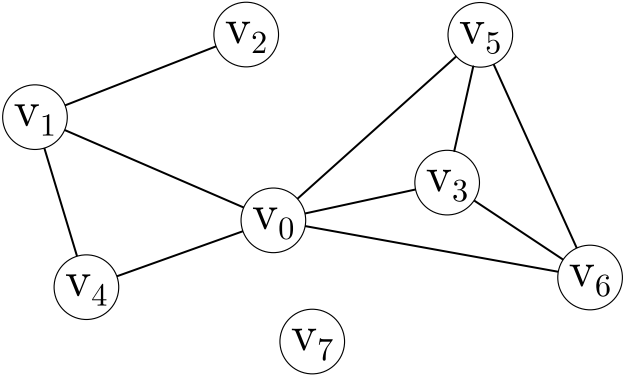
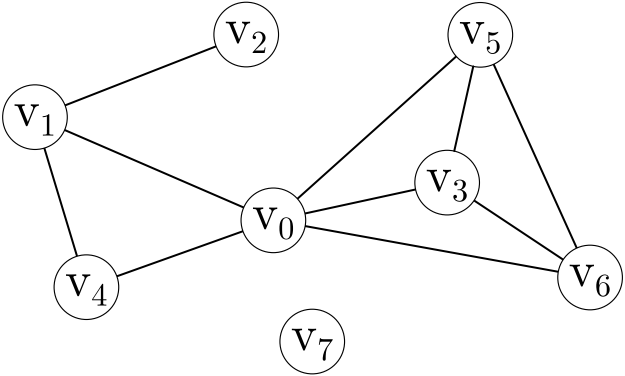


v_2
v_5
v_1
v_3
v_0
v_4
v_6
v_7

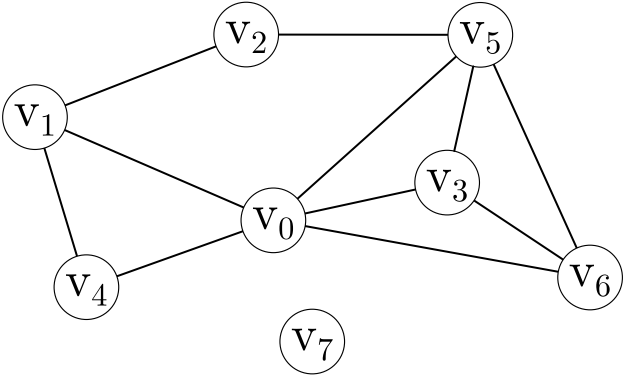
Chordaler Graph

G ist chordal, wenn jeder Kreis mit mindestens 4 Knoten eine Sehne besitzt.



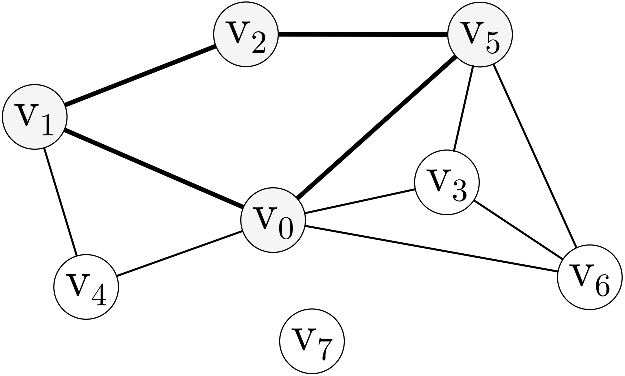
Chordaler Graph

G ist chordal, wenn jeder Kreis mit mindestens 4 Knoten eine Sehne besitzt.



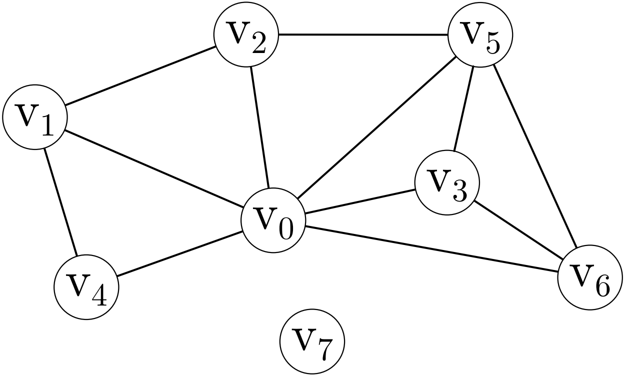
Chordaler Graph

G ist chordal, wenn jeder Kreis mit mindestens 4 Knoten eine Sehne besitzt.



Chordaler Graph

G ist chordal, wenn jeder Kreis mit mindestens 4 Knoten eine Sehne besitzt.



Platzeffiziente Erkennung von chordalen Graphen

Seminar „Algorithmen und Datenstrukturen“

Konstantin Fickel

Sommersemester 2018

Outline

Outline

Outline

Outline

Sankar Deep Chakraborty and

Srinivasa Rao Satti. Space-efficient algorithms for maximum cardinality search, stack bfs, queue BFS and applications.
In *Computing and Combinatorics - 23rd International Conference, COCOON 2017, Hong Kong, China, August 3-5, 2017, Proceedings*, pages 87–98, 2017

Space-Efficient Algorithms for Maximum Cardinality Search, Stack BFS, Queue BFS and Applications

Sankardeep Chakraborty[1(✉)] and Srinivasa Rao Satti[2]

[1] The Institute of Mathematical Sciences, CIT Campus, Taramani, Chennai 600113, India
sankardeep@imsc.res.in
[2] Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, South Korea
ssrao@cse.snu.ac.kr

Abstract. Following the recent trends of designing space efficient algorithms for fundamental algorithmic graph problems, we present several time-space tradeoffs for performing Maximum Cardinality Search (MCS), Stack Breadth First Search (Stack BFS), and Queue Breadth First Search (Queue BFS) on a given input graph. As applications of these results, we also provide space-efficient implementations for testing if a given undirected graph is chordal, reporting an independent set, and a proper coloring of a given chordal graph among others. Finally, we also show how two other seemingly different graph problems and their algorithms have surprising connection with MCS with respect to designing space efficient algorithms.

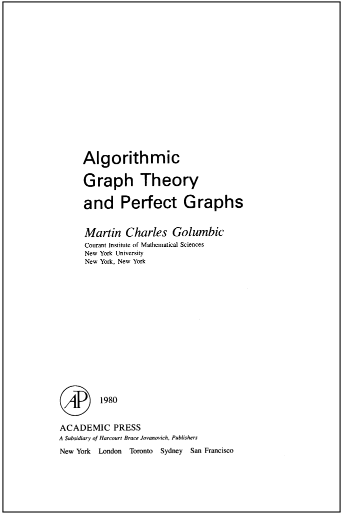
1 Introduction

Space efficient algorithms are becoming increasingly important owing to their applications in the presence of rapid growth of "big data". Another reason for the importance of space efficient algorithms is the proliferation of specialized handheld devices and embedded systems that have a limited supply of memory. As a consequence, algorithms that are oblivious to space constraint are not desired in such scenario. Even if mobile devices and embedded systems are designed with large supply of memory, it might be useful to restrict the number of write operations. For example, on flash memory, writing is a costly operation in terms of speed, and it also reduces the reliability and longevity of the memory. Keeping all these constraints in mind, it makes sense to consider algorithms that do not modify the input and use only a limited amount of work space. Such computational model has been proposed in algorithmic literature to study space efficient algorithms, and is known as the read-only memory (ROM) model. Following the recent trend, in this article, we focus on the space requirement for implementing some fundamental graph algorithms in such settings.

There is already a rich history of designing space efficient algorithms in the read-only memory model. In computational complexity theory, L is the complexity class [1] containing decision problems that can be solved by a deterministic

Y. Cao and J. Chen (Eds.): COCOON 2017, LNCS 10392, pp. 87–98, 2017.
DOI: 10.1007/978-3-319-62389-4_8

Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*.
Elsevier, 2004

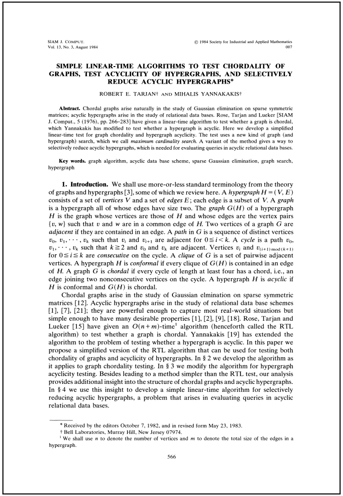

Algorithmic
Graph Theory
and Perfect Graphs

Martin Charles Golumbic
Courant Institute of Mathematical Sciences
New York University
New York, New York

1980

ACADEMIC PRESS
A Subsidiary of Harcourt Brace Jovanovich, Publishers
New York London Toronto Sydney San Francisco

Robert E. Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, July 1984



SIAM J. COMPUT.
Vol. 13, No. 3, August 1984

007

SIMPLE LINEAR-TIME ALGORITHMS TO TEST CHORDALITY OF GRAPHS, TEST ACYCLICITY OF HYPERGRAPHS, AND SELECTIVELY REDUCE ACYCLIC HYPERGRAPHS*

ROBERT E. TARJAN† AND MIHALIS YANNAKAKIS†

Abstract. Chordal graphs arise naturally in the study of Gaussian elimination on sparse symmetric matrices; acyclic hypergraphs arise in the study of relational data bases. Rose, Tarjan and Lueker [SIAM J. Comput., 5 (1976), pp. 266–283] have given a linear-time algorithm to test whether a graph is chordal, which Yannakakis has modified to test whether a hypergraph is acyclic. Here we develop a simplified linear-time test for graph chordality and hypergraph acyclicity. The test uses a new kind of graph (and hypergraph) search, which we call *maximum cardinality search*. A variant of the method gives a way to selectively reduce acyclic hypergraphs, which is needed for evaluating queries in acyclic relational data bases.

Key words. graph algorithm, acyclic data base scheme, sparse Gaussian elimination, graph search, hypergraph

1. Introduction. We shall use more-or-less standard terminology from the theory of graphs and hypergraphs [3], some of which we review here. A *hypergraph* $H=(V, E)$ consists of a set of *vertices* V and a set of *edges* E; each edge is a subset of V. A *graph* is a hypergraph all of whose edges have size two. The *graph* $G(H)$ of a hypergraph H is the graph whose vertices are those of H and whose edges are the vertex pairs $\{v, w\}$ such that v and w are in a common edge of H. Two vertices of a graph G are *adjacent* if they are contained in an edge. A *path* in G is a sequence of distinct vertices $v_0, v_1, \cdots, v_k$ such that v_i and v_{i+1} are adjacent for $0 \leqq i < k$. A *cycle* is a path $v_0, v_1, \cdots, v_k$ such that $k \geqq 2$ and v_0 and v_k are adjacent. Vertices v_i and $v_{(i+1) \bmod (k+1)}$ for $0 \leqq i \leqq k$ are *consecutive* on the cycle. A *clique* of G is a set of pairwise adjacent vertices. A hypergraph H is *conformal* if every clique of $G(H)$ is contained in an edge of H. A graph G is *chordal* if every cycle of length at least four has a chord, i.e., an edge joining two nonconsecutive vertices on the cycle. A hypergraph H is *acyclic* if H is conformal and $G(H)$ is chordal.

Chordal graphs arise in the study of Gaussian elmination on sparse symmetric matrices [12]. Acyclic hypergraphs arise in the study of relational data base schemes [1], [7], [21]; they are powerful enough to capture most real-world situations but simple enough to have many desirable properties [1], [2], [9], [18]. Rose, Tarjan and Lueker [15] have given an $O(n+m)$-time[1] algorithm (henceforth called the RTL algorithm) to test whether a graph is chordal. Yannakakis [19] has extended the algorithm to the problem of testing whether a hypergraph is acyclic. In this paper we propose a simplified version of the RTL algorithm that can be used for testing both chordality of graphs and acyclicity of hypergraphs. In § 2 we develop the algorithm as it applies to graph chordality testing. In § 3 we modify the algorithm for hypergraph acyclicity testing. Besides leading to a method simpler than the RTL test, our analysis provides additional insight into the structure of chordal graphs and acyclic hypergraphs. In § 4 we use this insight to develop a simple linear-time algorithm for selectively reducing acyclic hypergraphs, a problem that arises in evaluating queries in acyclic relational data bases.

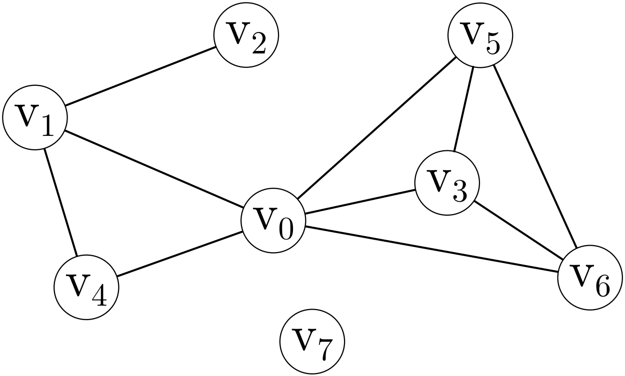
* Received by the editors October 7, 1982, and in revised form May 23, 1983.

† Bell Laboratories, Murray Hill, New Jersey 07974.

[1] We shall use n to denote the number of vertices and m to denote the total size of the edges in a hypergraph.

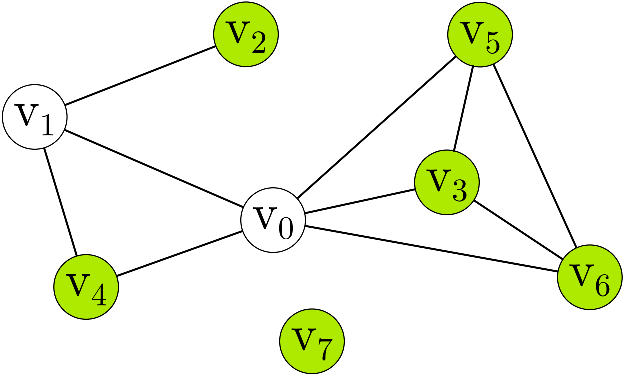
Simplizialer Knoten

Ein simplizialer Knoten v in einem Graphen G ist ein Knoten, dessen Nachbarn einen vollständigen Teilgraphen $G\left[\mathrm{Adj}\left(v\right)\right]$ induzieren.



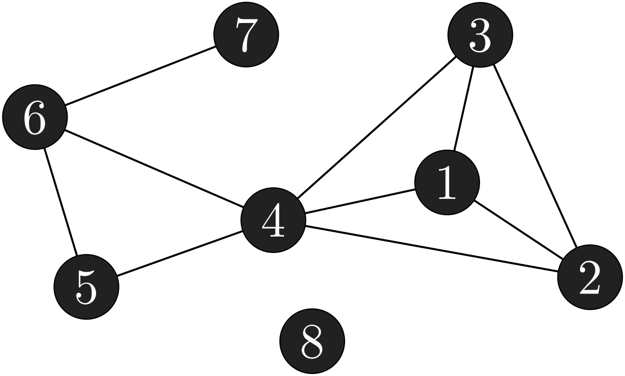
Simplizialer Knoten

Ein simplizialer Knoten v in einem Graphen G ist ein Knoten, dessen Nachbarn einen vollständigen Teilgraphen $G\left[\mathrm{Adj}\left(v\right)\right]$ induzieren.



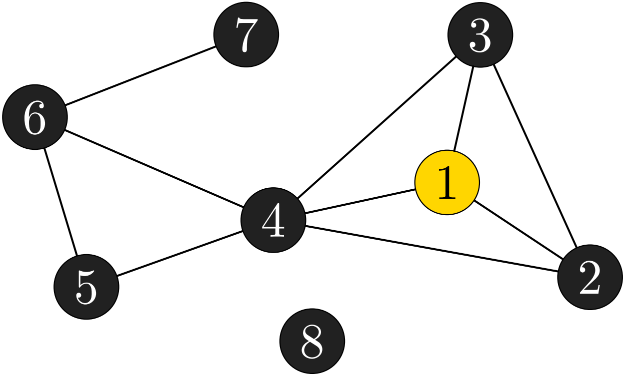
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



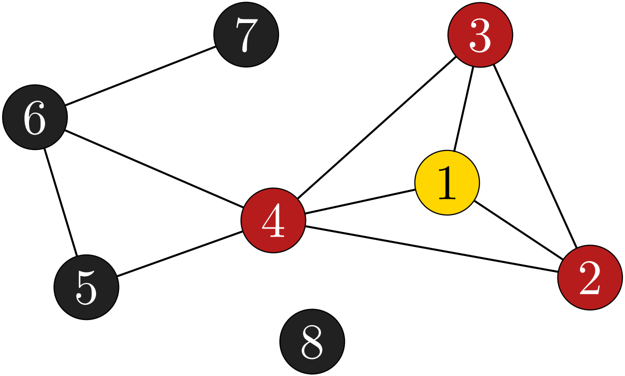
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



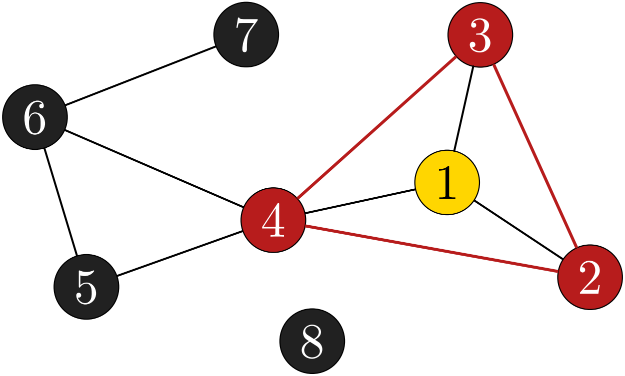
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



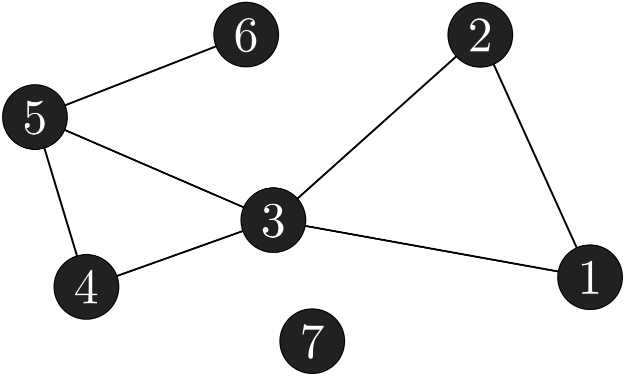
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



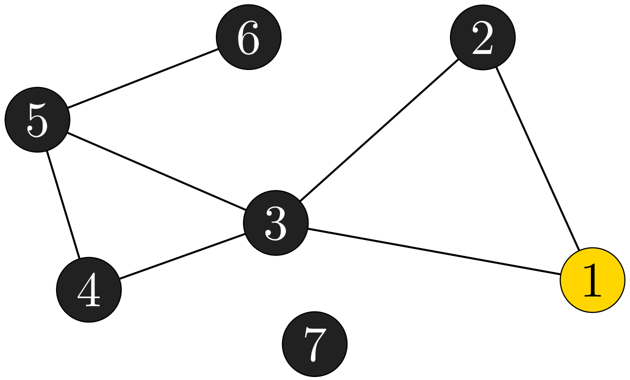
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



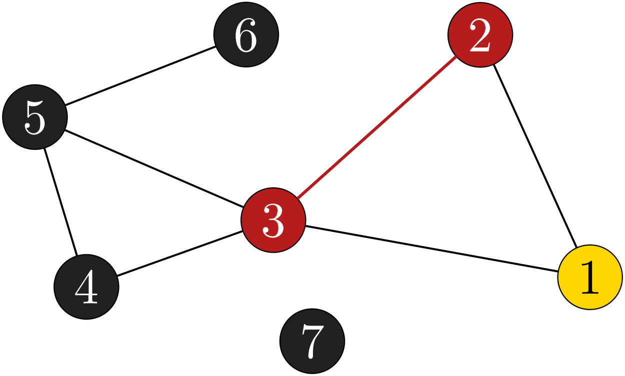
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



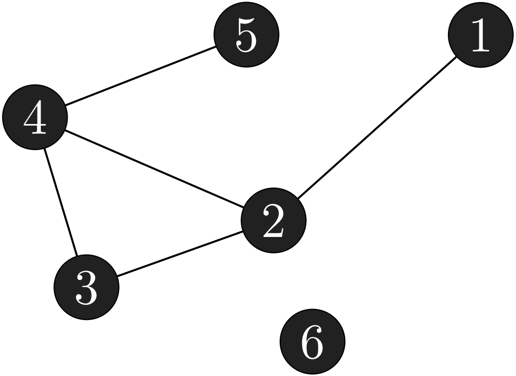
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



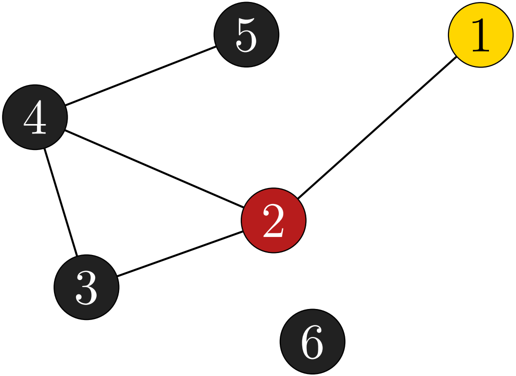
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



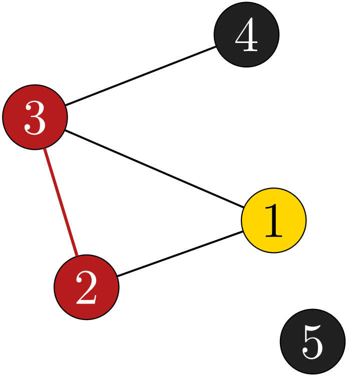
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \dots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \dots, \sigma(|V|)\}]$ ist.



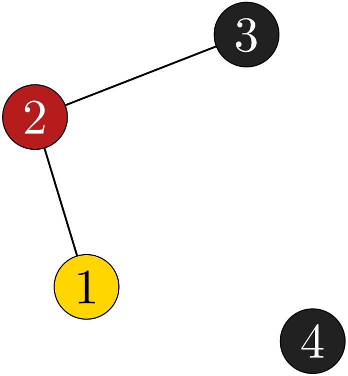
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



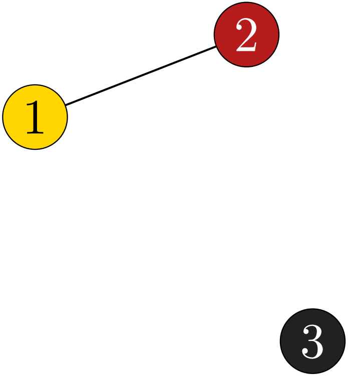
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.

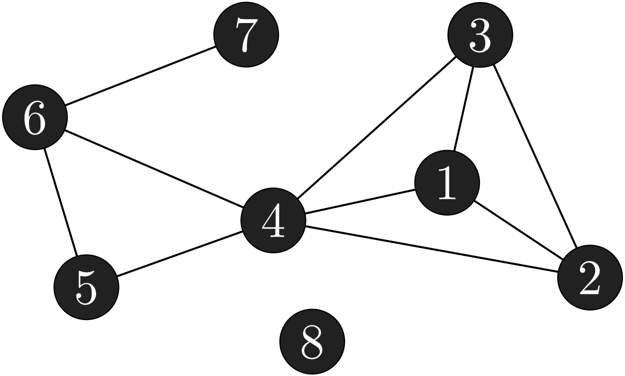
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \dots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \dots, \sigma(|V|)\}]$ ist.

1

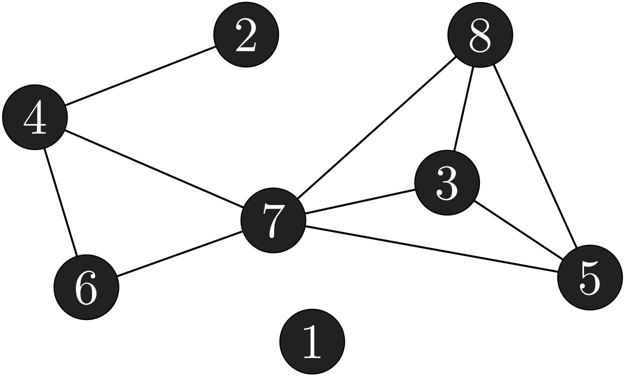
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



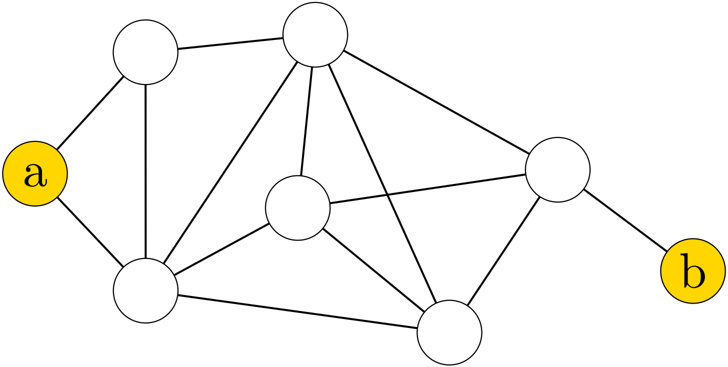
Perfekte Eliminations-Reihenfolge

Eine Reihenfolge σ auf $G = (V, E)$ ist eine perfekte Eliminations-Reihenfolge, wenn für jedes $i \in \{1, \ldots, |V|\}$ der Knoten $\sigma(i)$ simplizial im Teilgraph $G[\{\sigma(i), \ldots, \sigma(|V|)\}]$ ist.



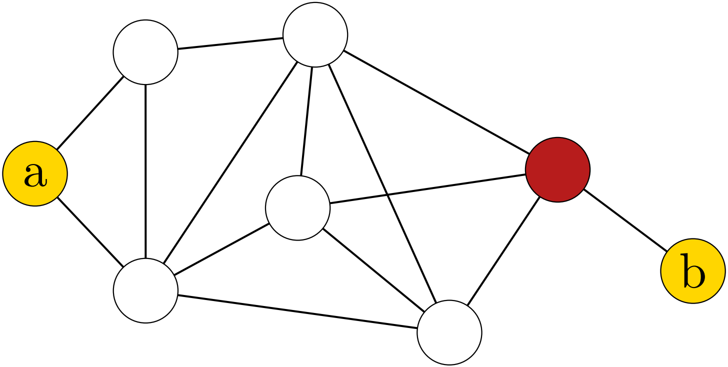
Knotentrenner

Ein Knotentrenner S der Knoten $a, b \in V$ ist eine Teilmenge $S \subseteq V$ mit $a, b \notin S$, bei dem sich a und b in verschiedenen Zusammenhangskomponenten von $G[V \setminus S]$ befinden.



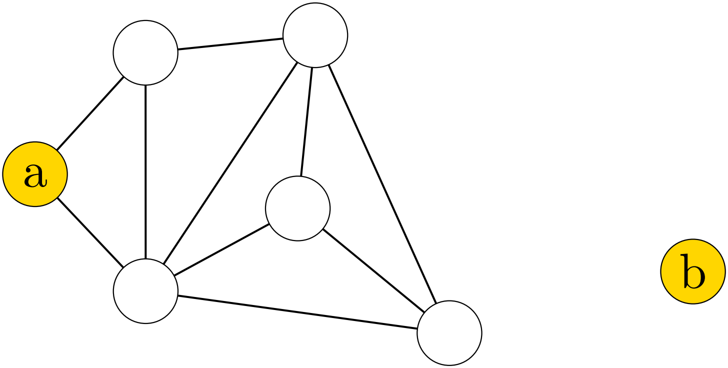
Knotentrenner

Ein Knotentrenner S der Knoten $a, b \in V$ ist eine Teilmenge $S \subseteq V$ mit $a, b \notin S$, bei dem sich a und b in verschiedenen Zusammenhangskomponenten von $G[V \setminus S]$ befinden.



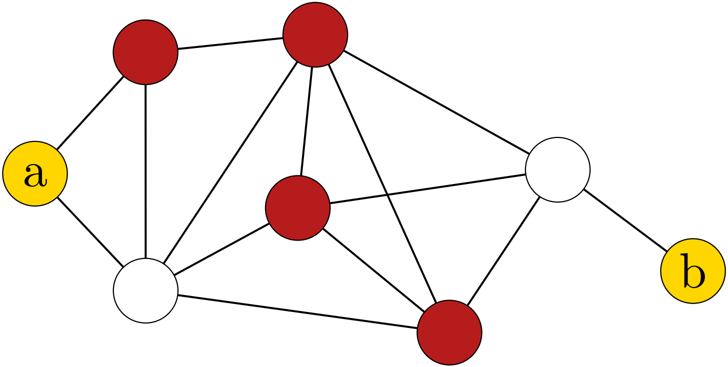
Knotentrenner

Ein Knotentrenner S der Knoten $a, b \in V$ ist eine Teilmenge $S \subseteq V$ mit $a, b \notin S$, bei dem sich a und b in verschiedenen Zusammenhangskomponenten von $G[V \setminus S]$ befinden.



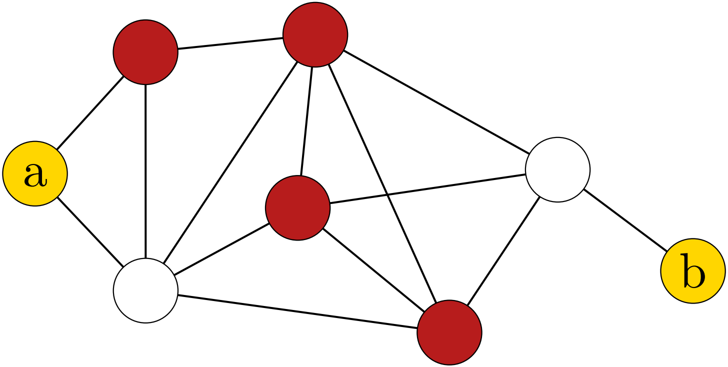
Knotentrenner

Ein Knotentrenner S der Knoten $a, b \in V$ ist eine Teilmenge $S \subseteq V$ mit $a, b \notin S$, bei dem sich a und b in verschiedenen Zusammenhangskomponenten von $G[V \setminus S]$ befinden.



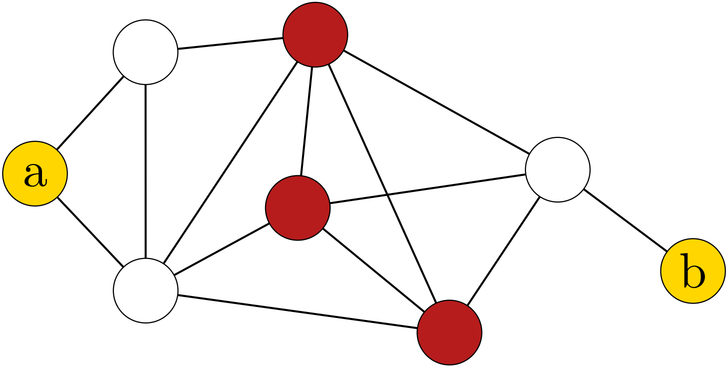
Minimaler Knotentrenner

S ist ein minimaler Knotentrenner, falls diese Aussage für kein $T \subsetneq S$ gilt.



Minimaler Knotentrenner

S ist ein minimaler Knotentrenner, falls diese Aussage für kein $T \subsetneq S$ gilt.

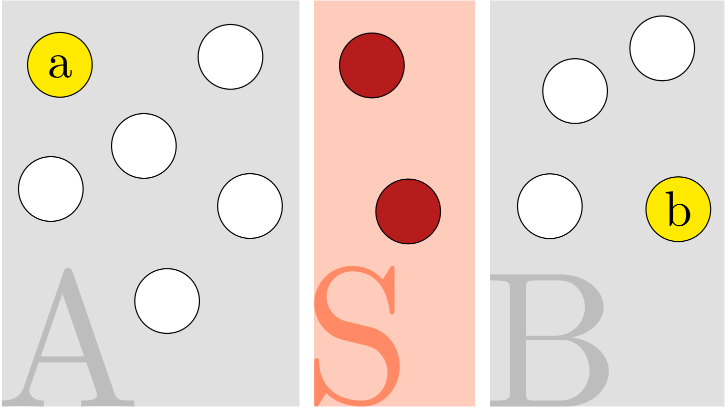


Lemma 1

Sei $G = (V, E)$ ein chordaler Graph. Dann ist der induzierte Teilgraph $G[S]$ zu jedem minimalen Knotentrenner $S \subseteq V$ vollständig.

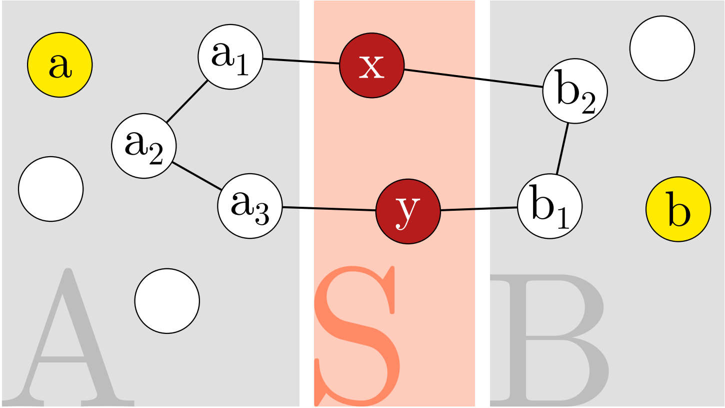
Lemma 1

Sei $G = (V, E)$ ein chordaler Graph. Dann ist der induzierte Teilgraph $G[S]$ zu jedem minimalen Knotentrenner $S \subseteq V$ vollständig.



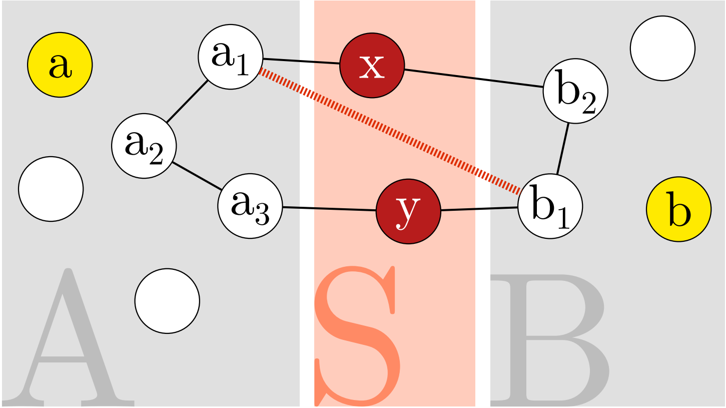
Lemma 1

Sei $G = (V, E)$ ein chordaler Graph. Dann ist der induzierte Teilgraph $G[S]$ zu jedem minimalen Knotentrenner $S \subseteq V$ vollständig.



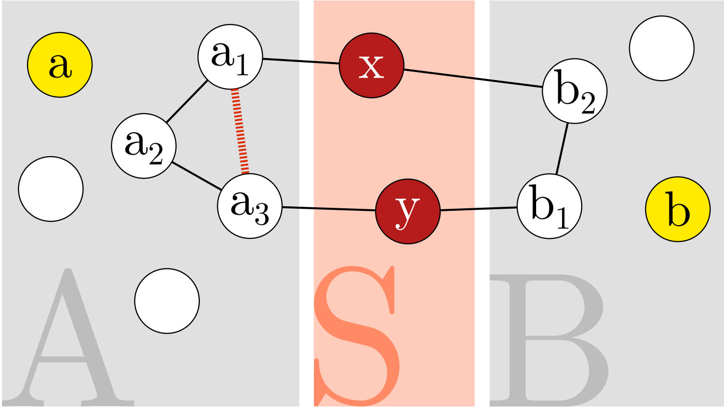
Lemma 1

Sei $G = (V, E)$ ein chordaler Graph. Dann ist der induzierte Teilgraph $G[S]$ zu jedem minimalen Knotentrenner $S \subseteq V$ vollständig.



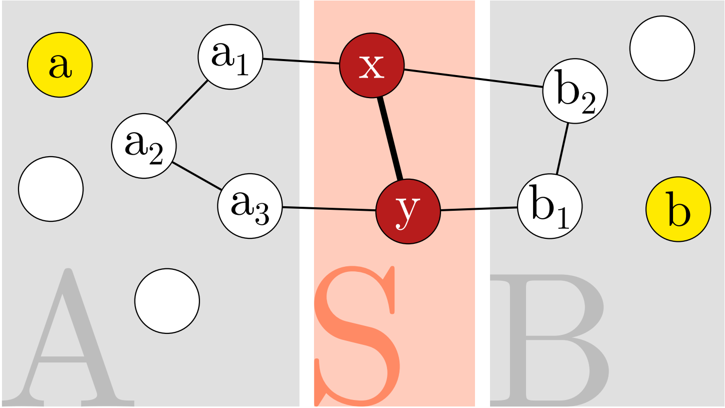
Lemma 1

Sei $G = (V, E)$ ein chordaler Graph. Dann ist der induzierte Teilgraph $G[S]$ zu jedem minimalen Knotentrenner $S \subseteq V$ vollständig.



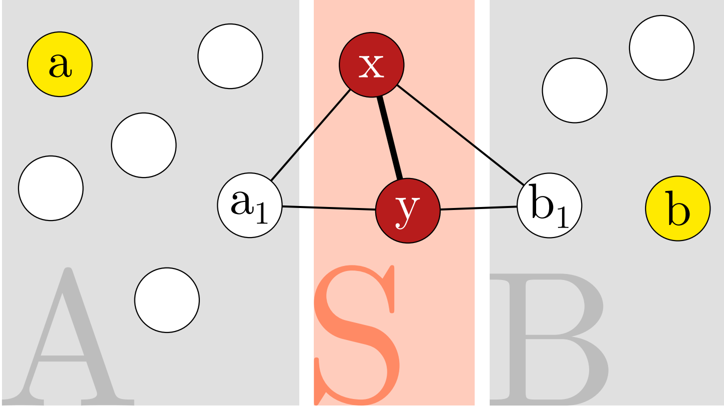
Lemma 1

Sei $G = (V, E)$ ein chordaler Graph. Dann ist der induzierte Teilgraph $G[S]$ zu jedem minimalen Knotentrenner $S \subseteq V$ vollständig.



Lemma 1

Sei $G = (V, E)$ ein chordaler Graph. Dann ist der induzierte Teilgraph $G[S]$ zu jedem minimalen Knotentrenner $S \subseteq V$ vollständig.



Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

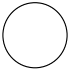
IV Für jeden chordalen Graphen $G = (V, E)$ mit $|V| \leq t$ gelte, dass G entweder vollständig ist oder zwei nicht-benachbarte simpliziale Knoten besitzt.

Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

IV Für jeden chordalen Graphen $G = (V, E)$ mit $|V| \leq t$ gelte, dass G entweder vollständig ist oder zwei nicht-benachbarte simpliziale Knoten besitzt.

$t = 1$

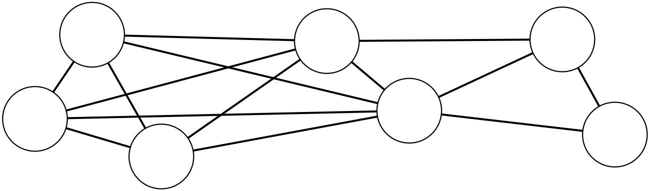


Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

IV Für jeden chordalen Graphen $G = (V, E)$ mit $|V| \leq t$ gelte, dass G entweder vollständig ist oder zwei nicht-benachbarte simpliziale Knoten besitzt.

$t \Rightarrow t + 1$

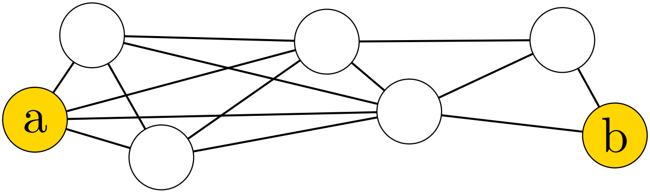


Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

IV Für jeden chordalen Graphen $G = (V, E)$ mit $|V| \leq t$ gelte, dass G entweder vollständig ist oder zwei nicht-benachbarte simpliziale Knoten besitzt.

$t \Rightarrow t + 1$

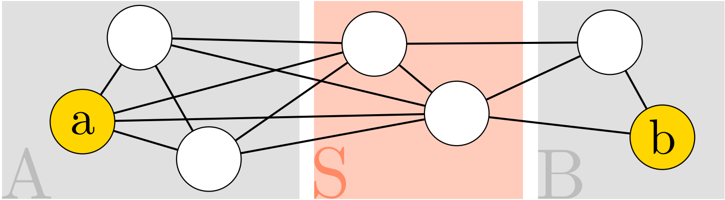


Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

IV Für jeden chordalen Graphen $G = (V, E)$ mit $|V| \leq t$ gelte, dass G entweder vollständig ist oder zwei nicht-benachbarte simpliziale Knoten besitzt.

$t \Rightarrow t + 1$

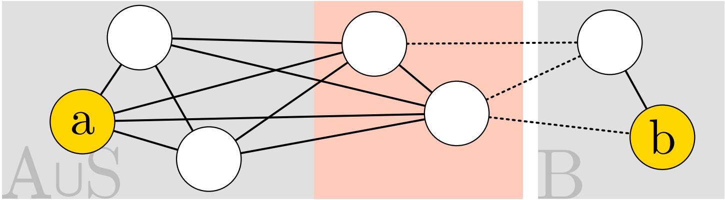


Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

IV Für jeden chordalen Graphen $G = (V, E)$ mit $|V| \leq t$ gelte, dass G entweder vollständig ist oder zwei nicht-benachbarte simpliziale Knoten besitzt.

$t \Rightarrow t + 1$

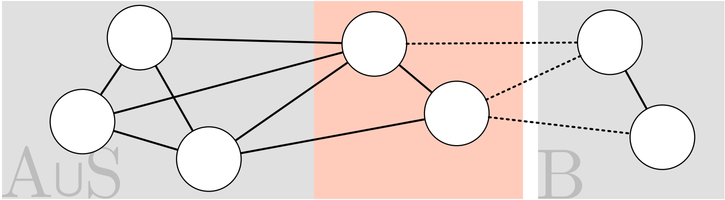


Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

IV Für jeden chordalen Graphen $G = (V, E)$ mit $|V| \leq t$ gelte, dass G entweder vollständig ist oder zwei nicht-benachbarte simpliziale Knoten besitzt.

$t \Rightarrow t + 1$

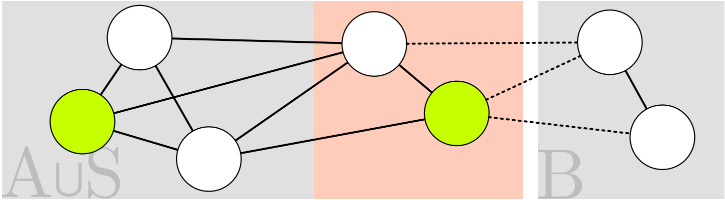


Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

IV Für jeden chordalen Graphen $G = (V, E)$ mit $|V| \leq t$ gelte, dass G entweder vollständig ist oder zwei nicht-benachbarte simpliziale Knoten besitzt.

$t \Rightarrow t + 1$

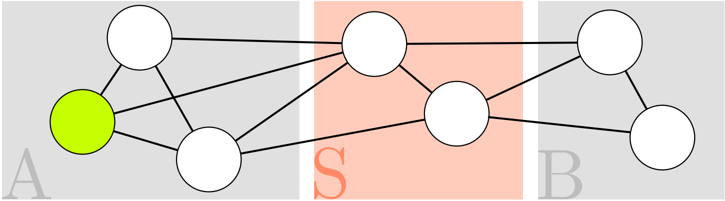


Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

IV Für jeden chordalen Graphen $G = (V, E)$ mit $|V| \leq t$ gelte, dass G entweder vollständig ist oder zwei nicht-benachbarte simpliziale Knoten besitzt.

$t \Rightarrow t + 1$

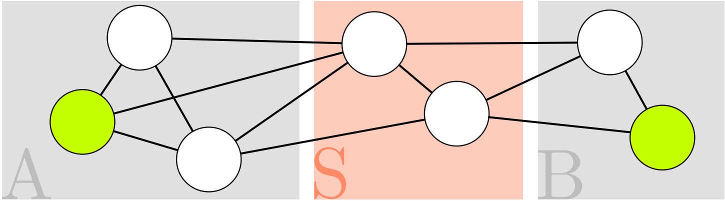


Lemma 2

Jeder chordale Graph $G = (V, E)$ besitzt einen simplizialen Knoten. Falls G nicht vollständig ist, dann besitzt dieser zwei nicht-benachbarte simpliziale Knoten.

IV Für jeden chordalen Graphen $G = (V, E)$ mit $|V| \leq t$ gelte, dass G entweder vollständig ist oder zwei nicht-benachbarte simpliziale Knoten besitzt.

$t \Rightarrow t + 1$



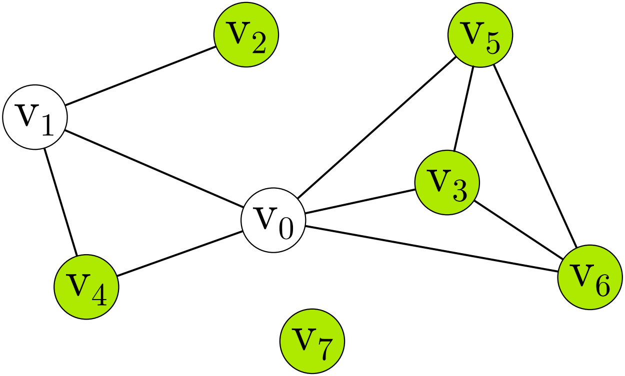
Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

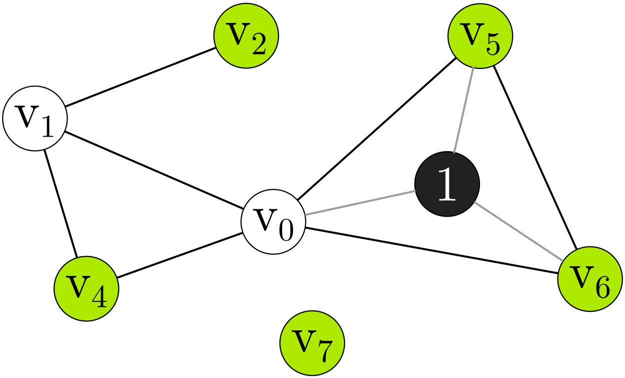
$\Rightarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

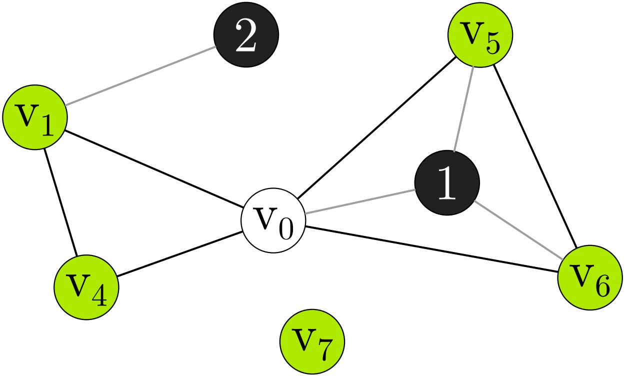
$\Rightarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

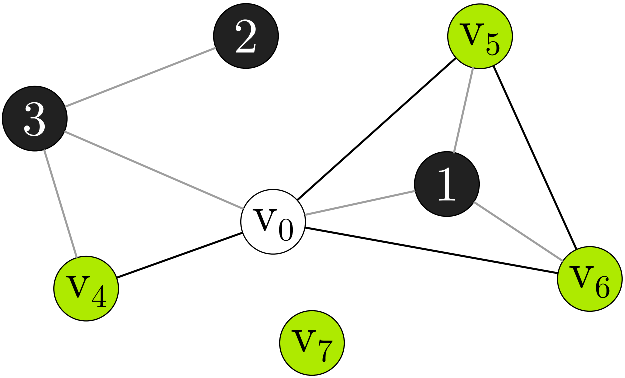
$\Rightarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

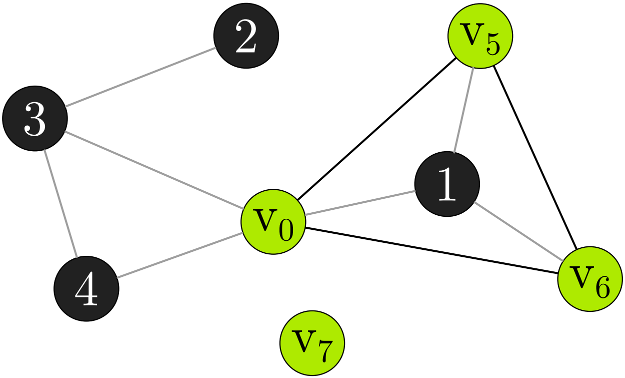
$\Rightarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

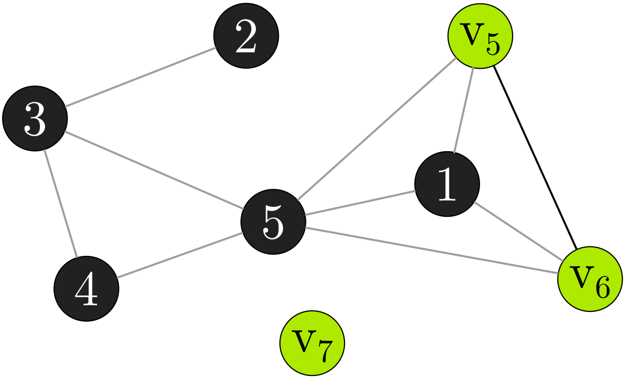
$\Rightarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

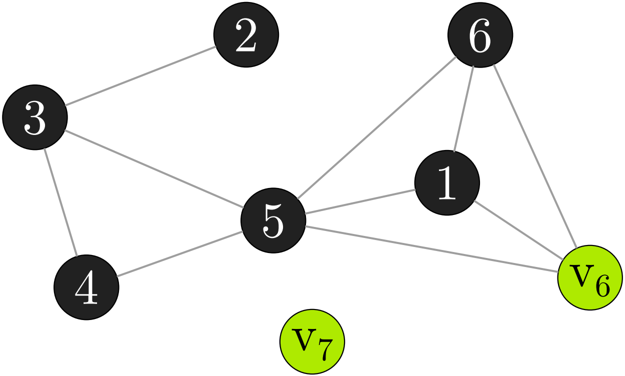
$\Rightarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

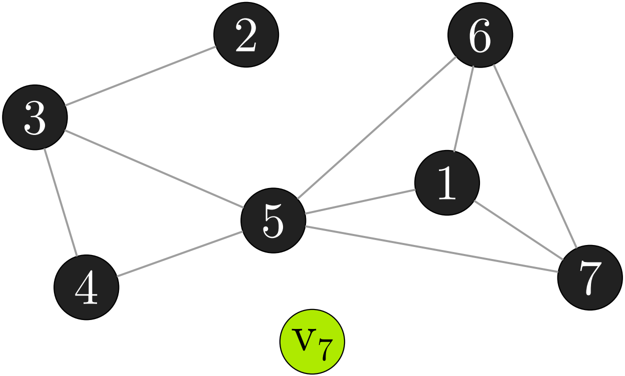
$\Rightarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

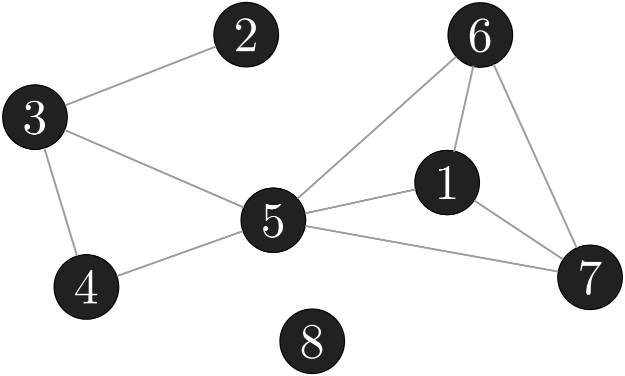
$\Rightarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

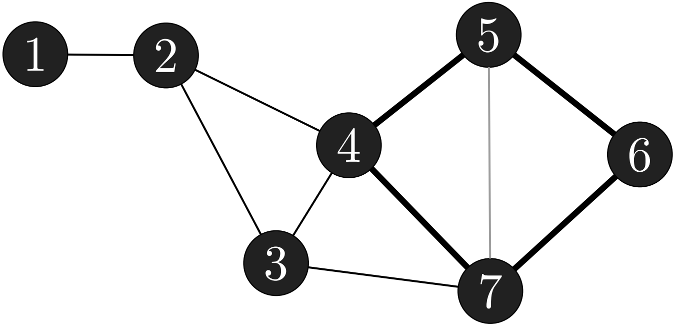
$\Rightarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

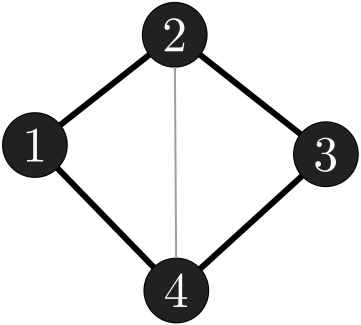
$\Leftarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

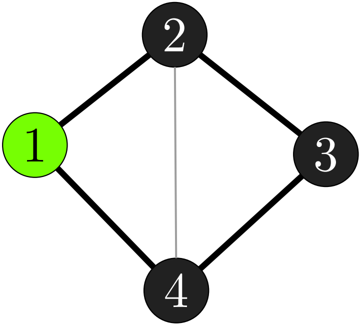
$\Leftarrow$



Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

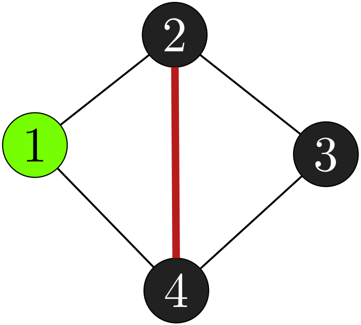
$\Leftarrow$



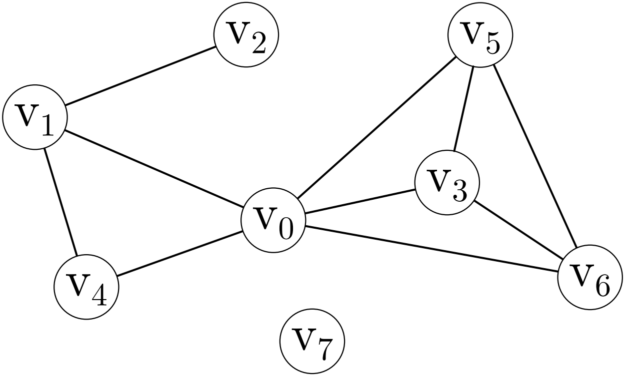
Satz 1

Sei G ein ungerichteter Graph. G ist chordal genau dann, wenn für G eine perfekte Eliminations-Reihenfolge σ existiert.

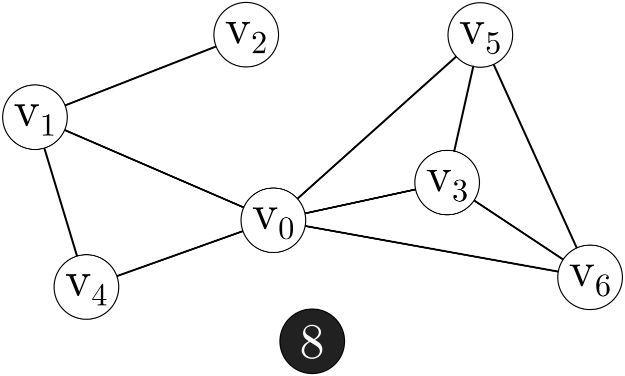
$\Leftarrow$



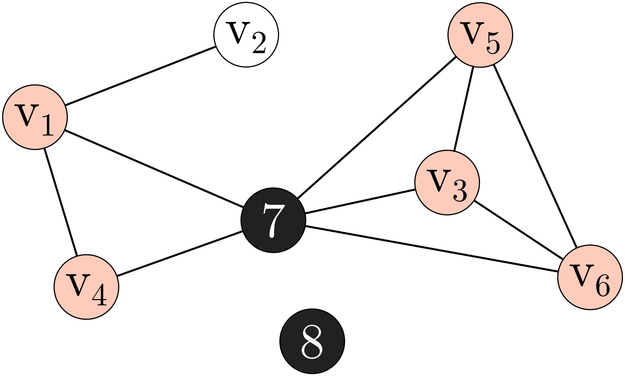
Kardinalitätssuche



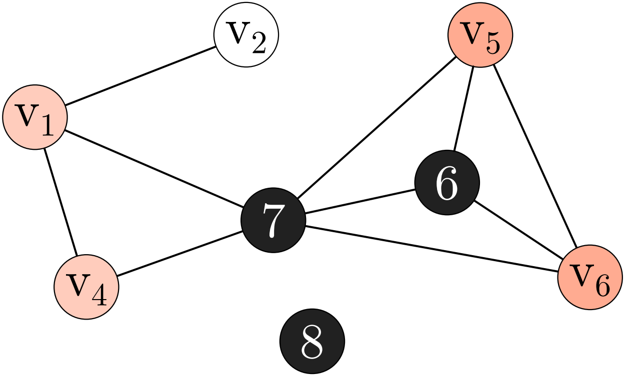
Kardinalitätssuche



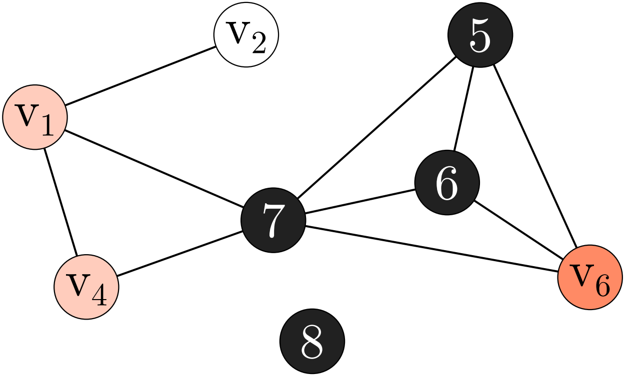
Kardinalitätssuche



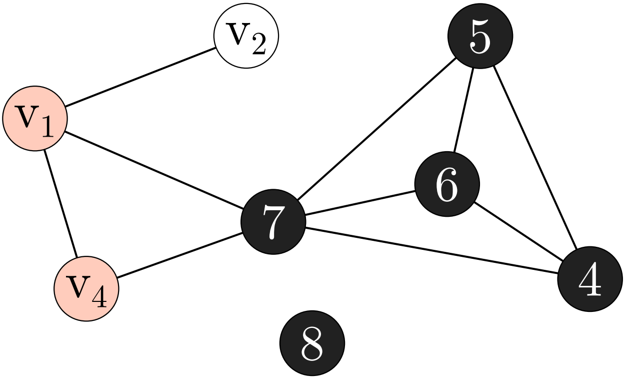
Kardinalitätssuche



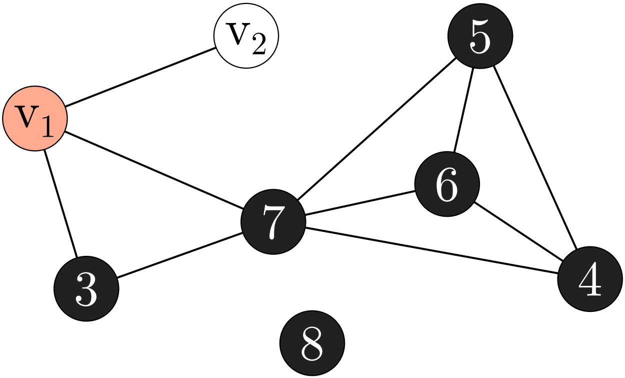
Kardinalitätssuche



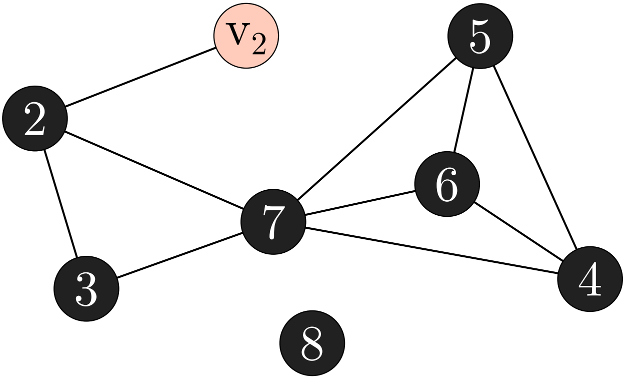
Kardinalitätssuche



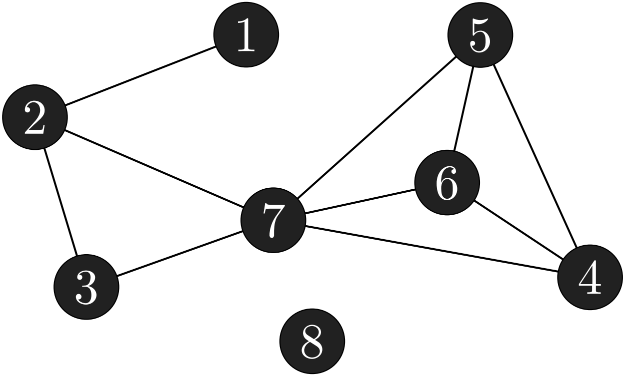
Kardinalitätssuche



Kardinalitätssuche



Kardinalitätssuche

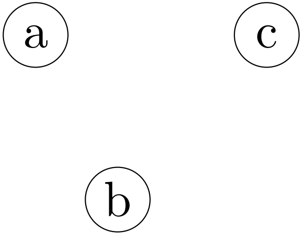


Reihenfolgeneigenschaft P

Eine Reihenfolge σ besitzt die Eigenschaft P, falls für alle Knoten $a, b, c \in V$ mit $\sigma^{-1}(a) < \sigma^{-1}(b) < \sigma^{-1}(c)$ und $c \in \text{Adj}(a) \setminus \text{Adj}(b)$ ein weiterer Knoten $x \in \text{Adj}(b) \setminus \text{Adj}(a)$ mit $\sigma^{-1}(b) < \sigma^{-1}(x)$ existiert.

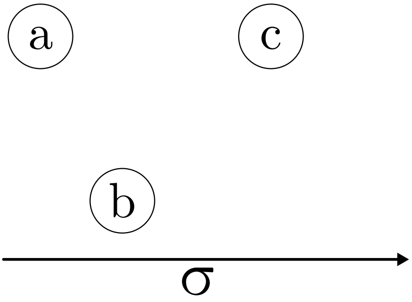
Reihenfolgeneigenschaft P

Eine Reihenfolge σ besitzt die Eigenschaft P, falls für alle Knoten $a, b, c \in V$ mit $\sigma^{-1}(a) < \sigma^{-1}(b) < \sigma^{-1}(c)$ und $c \in \mathrm{Adj}(a) \setminus \mathrm{Adj}(b)$ ein weiterer Knoten $x \in \mathrm{Adj}(b) \setminus \mathrm{Adj}(a)$ mit $\sigma^{-1}(b) < \sigma^{-1}(x)$ existiert.



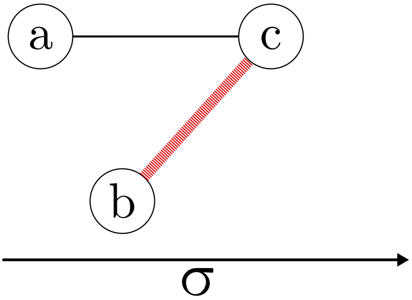
Reihenfolgeneigenschaft P

Eine Reihenfolge σ besitzt die Eigenschaft P, falls für alle Knoten $a, b, c \in V$ mit $\sigma^{-1}(a) < \sigma^{-1}(b) < \sigma^{-1}(c)$ und $c \in \operatorname{Adj}(a) \setminus \operatorname{Adj}(b)$ ein weiterer Knoten $x \in \operatorname{Adj}(b) \setminus \operatorname{Adj}(a)$ mit $\sigma^{-1}(b) < \sigma^{-1}(x)$ existiert.



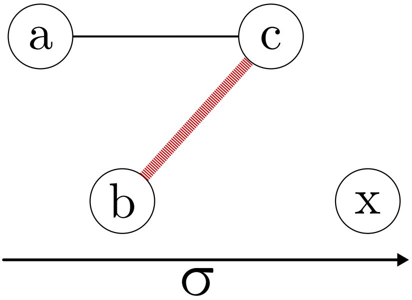
Reihenfolgeneigenschaft P

Eine Reihenfolge σ besitzt die Eigenschaft P, falls für alle Knoten $a, b, c \in V$ mit $\sigma^{-1}(a) < \sigma^{-1}(b) < \sigma^{-1}(c)$ und $c \in \mathrm{Adj}(a) \setminus \mathrm{Adj}(b)$ ein weiterer Knoten $x \in \mathrm{Adj}(b) \setminus \mathrm{Adj}(a)$ mit $\sigma^{-1}(b) < \sigma^{-1}(x)$ existiert.



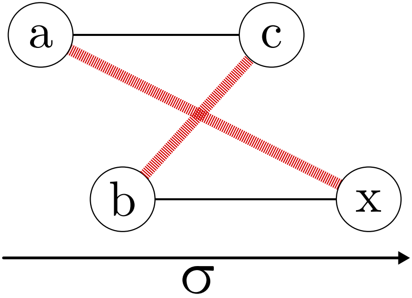
Reihenfolgeneigenschaft P

Eine Reihenfolge σ besitzt die Eigenschaft P, falls für alle Knoten $a, b, c \in V$ mit $\sigma^{-1}(a) < \sigma^{-1}(b) < \sigma^{-1}(c)$ und $c \in \text{Adj}(a) \setminus \text{Adj}(b)$ ein weiterer Knoten $x \in \text{Adj}(b) \setminus \text{Adj}(a)$ mit $\sigma^{-1}(b) < \sigma^{-1}(x)$ existiert.



Reihenfolgeneigenschaft P

Eine Reihenfolge σ besitzt die Eigenschaft P, falls für alle Knoten $a, b, c \in V$ mit $\sigma^{-1}(a) < \sigma^{-1}(b) < \sigma^{-1}(c)$ und $c \in \mathrm{Adj}(a) \setminus \mathrm{Adj}(b)$ ein weiterer Knoten $x \in \mathrm{Adj}(b) \setminus \mathrm{Adj}(a)$ mit $\sigma^{-1}(b) < \sigma^{-1}(x)$ existiert.

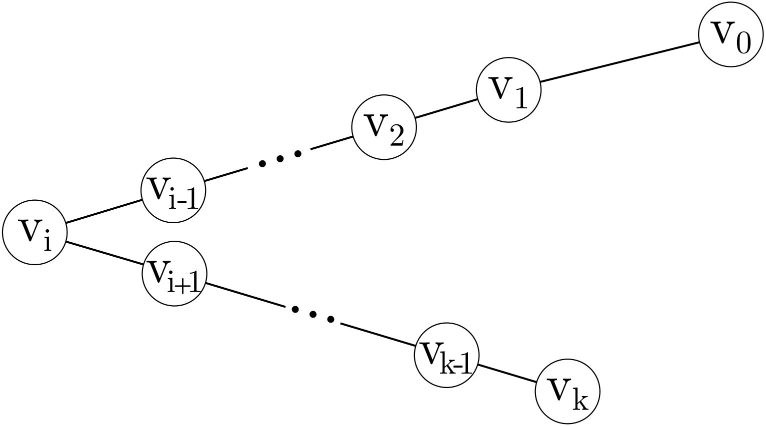


Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.

Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.

Pfadeigenschaft Q

Ein Pfad $\pi = (v_0, \ldots, v_k)$ in G mit $k \geq 2$ besitzt die Eigenschaft Q genau dann, wenn folgende Aussagen erfüllt sind:

Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.

Pfadeigenschaft Q

Ein Pfad $\pi = (v_0, \ldots, v_k)$ in G mit $k \geq 2$ besitzt die Eigenschaft Q genau dann, wenn folgende Aussagen erfüllt sind:

- π besitzt keine Zwischenverbindungen

Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.

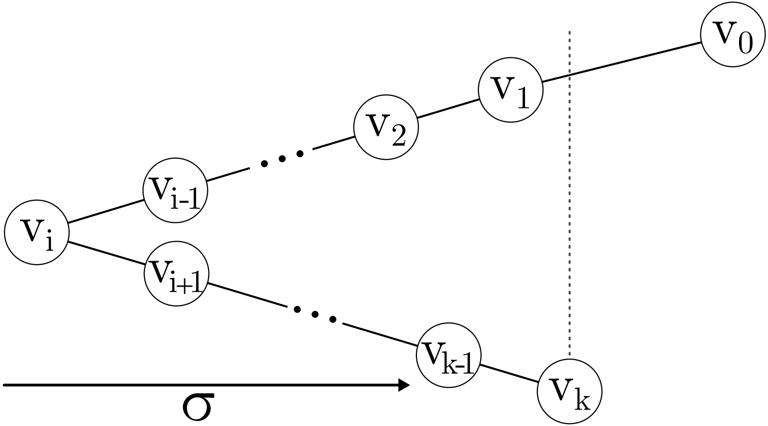
Pfadeigenschaft Q

Ein Pfad $\pi = (v_0, \ldots, v_k)$ in G mit $k \geq 2$ besitzt die Eigenschaft Q genau dann, wenn folgende Aussagen erfüllt sind:

- π besitzt keine Zwischenverbindungen
- Für ein $i \in \{1, 2, \ldots, k-1\}$ sei $\sigma^{-1}(v_0) > \sigma^{-1}(v_k) > \sigma^{-1}(v_1) > \sigma^{-1}(v_2) > \ldots > \sigma^{-1}(v_i)$ und $\sigma^{-1}(v_i) < \sigma^{-1}(v_{i+1}) < \ldots < \sigma^{-1}(v_k)$

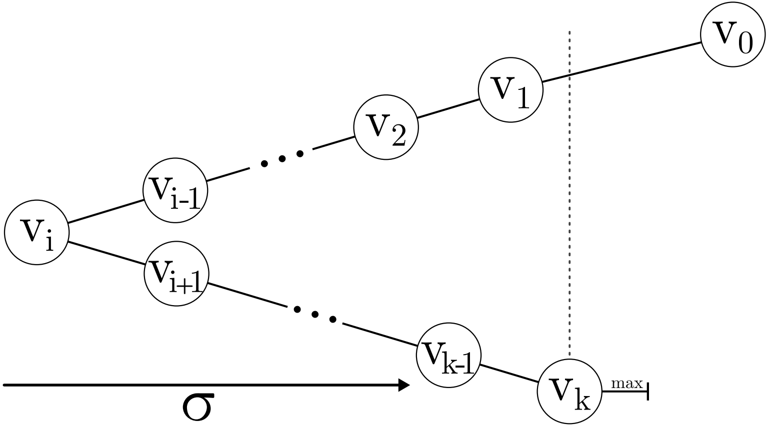
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



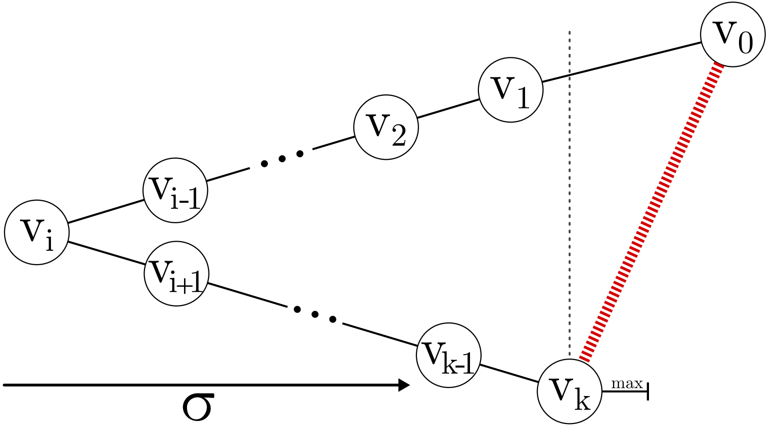
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



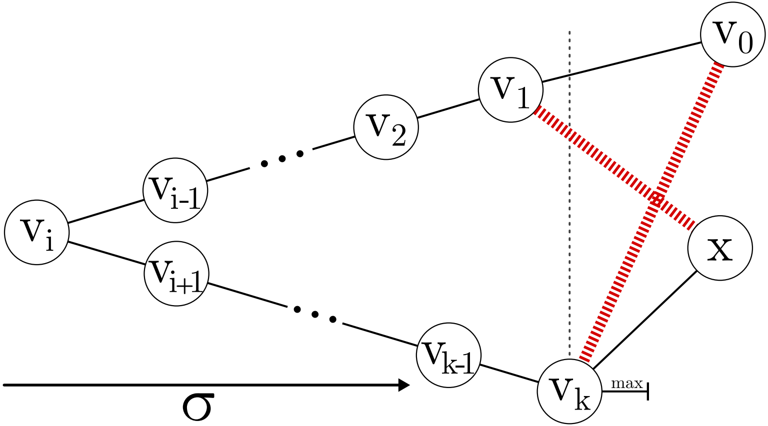
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



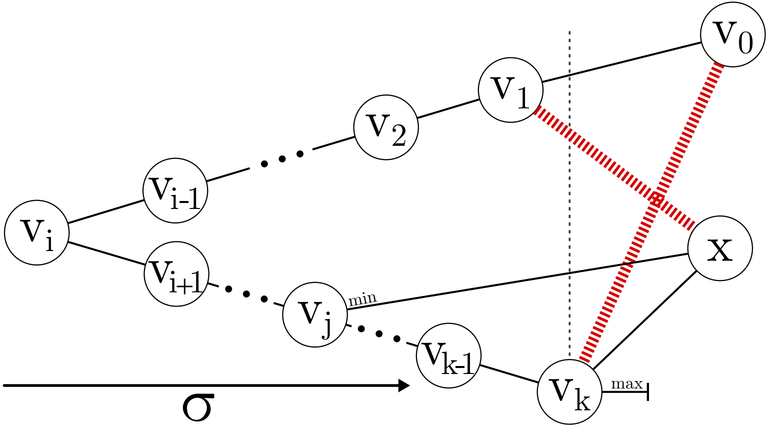
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



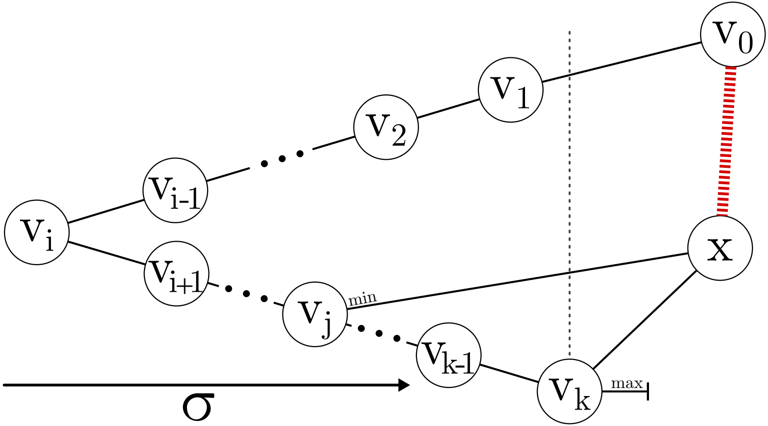
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



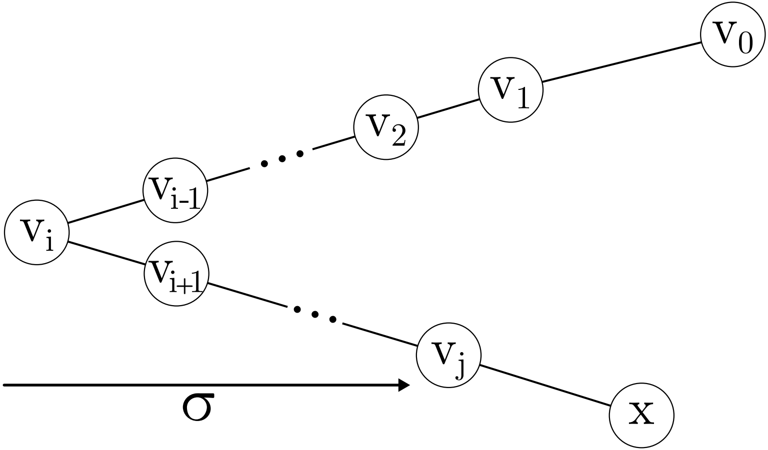
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



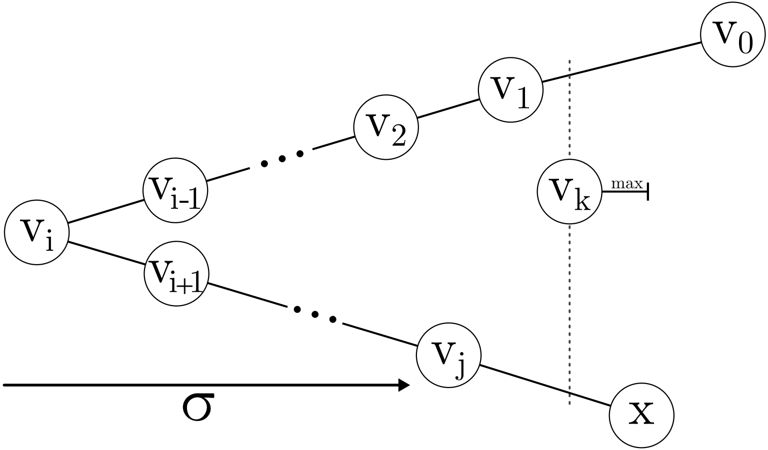
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



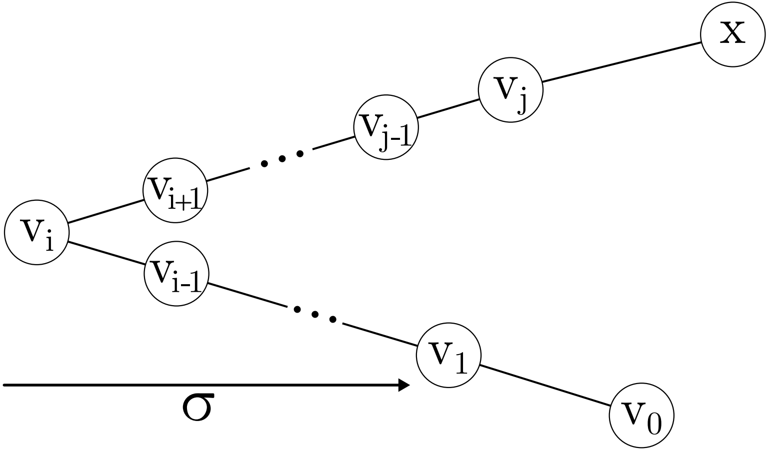
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



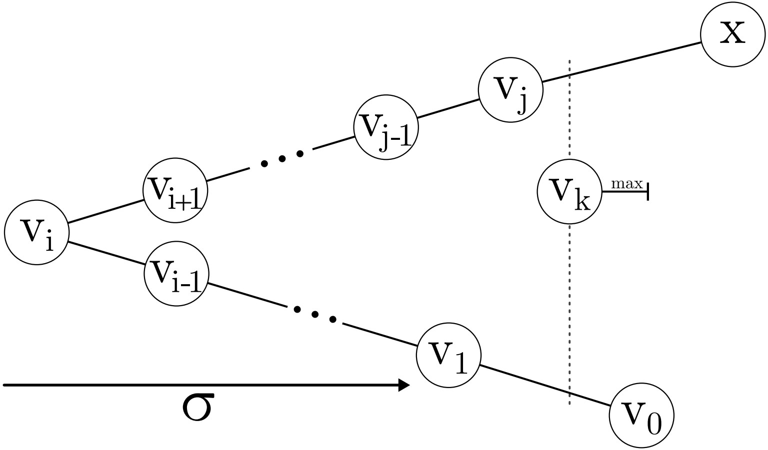
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



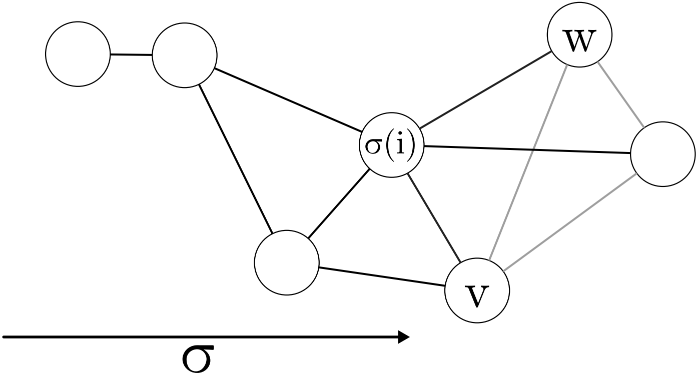
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



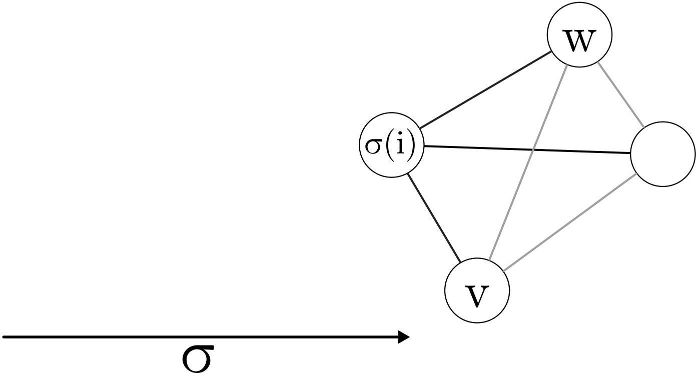
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



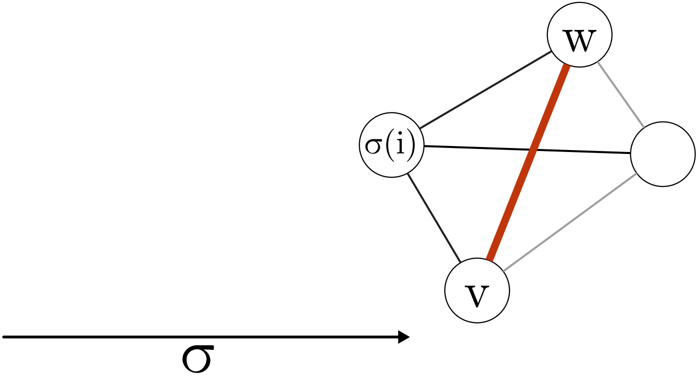
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



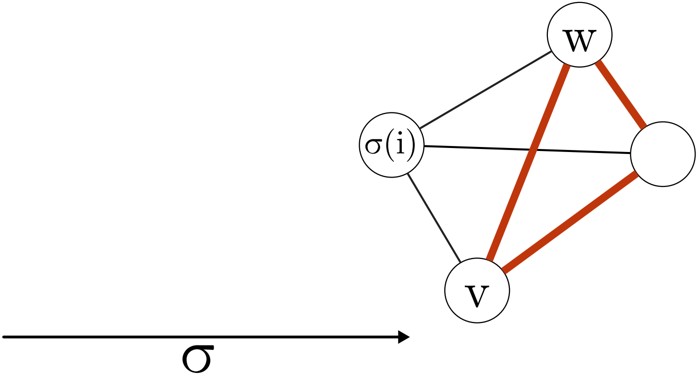
Satz 3

Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



Satz 3

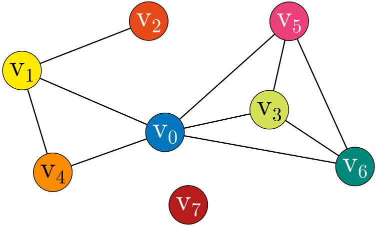
Falls die Reihenfolge σ von den Knoten von G die Eigenschaft P erfüllt, so ist σ eine perfekte Eliminations-Reihenfolge.



for $i \in \{0, \ldots, n-1\}$ **do**
  S $[i] \longleftarrow \emptyset$

for $v \in V$ **do**
  füge v in S $[0]$ ein;
  M $[v] \longleftarrow$ Position von v in S $[0]$;
  N $[v] \longleftarrow 0$;

$j \longleftarrow 0$;

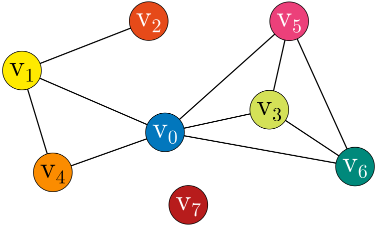


N=[v_i → Enthaltende Menge in S] M=[v_i → Speicherstelle in S]

$$S = [\ {}^{0}\{\ \},\ {}^{1}\{\ \},\ {}^{2}\{\ \},\ {}^{3}\{\ \ldots$$

$$\sigma = [\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ ]$$

for $i \in \{n, \ldots, 1\}$ **do**
- $v \longleftarrow$ entferne einen Knoten aus $\mathtt{S}[j]$;
- $\sigma[i] \longleftarrow v$;
- $\mathtt{N}[v] \longleftarrow -1$;
- $j \longleftarrow j + 1$;
- **for** $w \in \mathrm{Adj}(v)$ *mit* $\mathtt{N}[w] \geq 0$ **do**
  - $k \longleftarrow N[w]$;
  - lösche w aus $\mathtt{S}$ an Position $\mathtt{M}[w]$ aus der Menge $\mathtt{S}[k]$;
  - füge w in $\mathtt{S}[k+1]$ ein;
  - $\mathtt{N}[w] \longleftarrow k + 1$;
  - $\mathtt{M}[w] \longleftarrow$ neue Postion von v in $\mathtt{S}[k]$;
- **while** $j \geq 0 \wedge \mathtt{S}[j] = \emptyset$ **do**
  - $j \longleftarrow j - 1$;



N=[v_i → Enthaltende Menge in S]

M=[v_i → Speicherstelle in S]

S=[$^0\{\ \}$, $^1\{\ \}$, $^2\{\ \}$, $^3\{$ …

σ = [1 2 3 4 5 6 7 8]

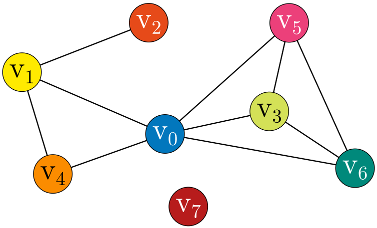
for $i \in \{0, \ldots, n-1\}$ **do**
    $\mathtt{S}[i] \longleftarrow \emptyset$

for $v \in V$ **do**
    füge v in $\mathtt{S}[0]$ ein;
    $\mathtt{M}[v] \longleftarrow$ Position von v in $\mathtt{S}[0]$;
    $\mathtt{N}[v] \longleftarrow 0$;

$j \longleftarrow 0$;

for $i \in \{n, \ldots, 1\}$ **do**
    $v \longleftarrow$ entferne einen Knoten aus $\mathtt{S}[j]$;
    $\sigma[i] \longleftarrow v$;
    $\mathtt{N}[v] \longleftarrow -1$;
    $j \longleftarrow j+1$;
    for $w \in \mathrm{Adj}(v)$ *mit* $\mathtt{N}[w] \geq 0$ **do**
        $k \longleftarrow N[w]$;
        lösche w aus $\mathtt{S}$ an Position $\mathtt{M}[w]$ aus der Menge $\mathtt{S}[k]$;
        füge w in $\mathtt{S}[k+1]$ ein;
        $\mathtt{N}[w] \longleftarrow k+1$;
        $\mathtt{M}[w] \longleftarrow$ neue Postion von v in $\mathtt{S}[k]$;
    while $j \geq 0 \wedge \mathtt{S}[j] = \emptyset$ **do**
        $j \longleftarrow j-1$;

for $v \in V$ **do**
  `B` $[v] \longleftarrow$ `false;`



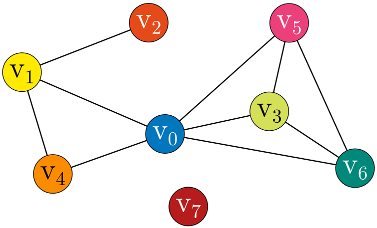
B = [v_0 v_1 v_2 v_3 v_4 v_5 v_6 v_7]

u =

c = 0 1 2 3 4 5 ...

σ = [1 2 3 4 5 6 7 8]

```
for i ∈ {n, ..., 1} do
    u ⟵ wähle beliebigen Knoten x ∈ V;
    c ⟵ 0;
    for v ∈ V mit B[v] = false do
        k ⟵ 0;
        for w ∈ Adj(v) mit B[w] = true do
            k ⟵ k + 1;
        if k ≥ c then
            u ⟵ v;
            c ⟵ k;
    B[u] = true;
    σ[i] ⟵ u;
```



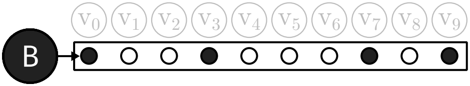
B = [v_0 v_1 v_2 v_3 v_4 v_5 v_6 v_7]

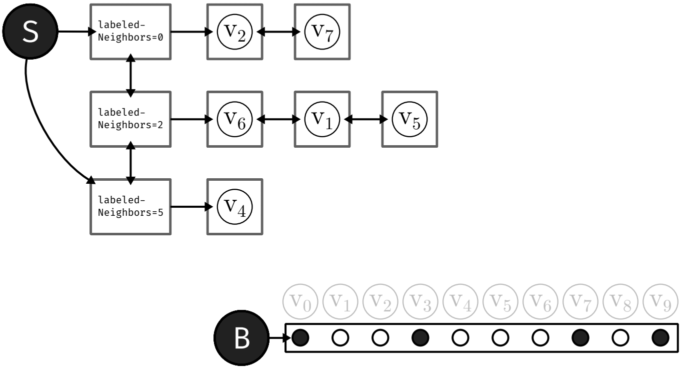
u =

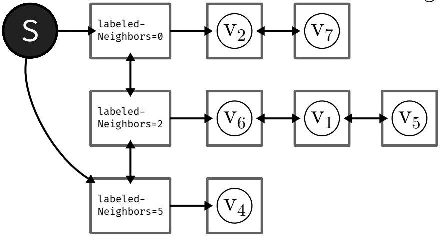
c = 0 1 2 3 4 5 ...

σ = [1 2 3 4 5 6 7 8]

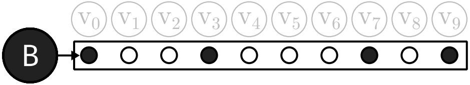
```
for v ∈ V do
    B [v] ⟵ false;

for i ∈ {n, . . . , 1} do
    u ⟵ wähle beliebigen Knoten x ∈ V;
    c ⟵ 0;
    for v ∈ V mit B [v] = false do
        k ⟵ 0;
        for w ∈ Adj (v) mit B [w] = true do
            k ⟵ k + 1;

        if k ≥ c then
            u ⟵ v;
            c ⟵ k;

    B [u] = true;
    σ [i] ⟵ u;
```


v0
v1
v2
v3
v4
v5
v6
v7
v8
v9
B

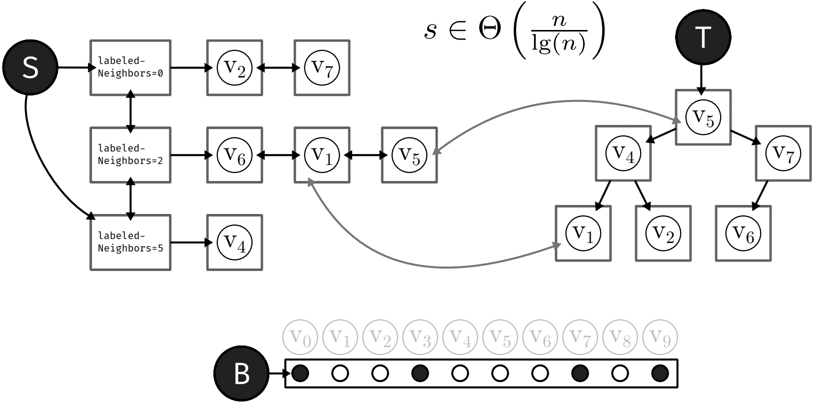

S
labeled-Neighbors=0
labeled-Neighbors=2
labeled-Neighbors=5
v_2 v_7
v_6 v_1 v_5
v_4
B
v_0 v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9



$$s \in \Theta\left(\frac{n}{\lg(n)}\right)$$



$$s \in \Theta\left(\frac{n}{\lg(n)}\right)$$



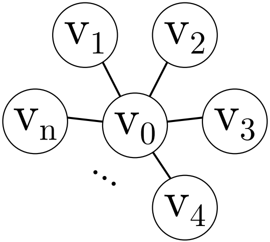
$$\mathcal{O}\left(\frac{m^2}{n} + m \cdot \lg(n)\right)$$

$$\mathcal{O}\left(\frac{m^2}{n} + m \cdot \lg(n)\right)$$

$$\mathcal{O}\left(\sum_{w \in V} |\mathrm{Adj}(w)|^2\right) = \mathcal{O}\left(\frac{m^2}{n}\right)$$

$$\mathcal{O}\left(\frac{m^2}{n} + m \cdot \lg(n)\right)$$

$$\mathcal{O}\left(\sum_{w \in V} |\mathrm{Adj}(w)|^2\right) = \mathcal{O}\left(\frac{m^2}{n}\right)$$

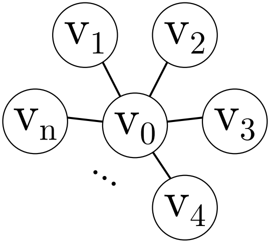


$\mathcal{O}\left(\frac{m^2}{n} + m \cdot \lg(n)\right)$

$\mathcal{O}\left(\sum_{w \in V} |\mathrm{Adj}(w)|^2\right) = \mathcal{O}\left(\frac{m^2}{n}\right)$

$$\sum_{v \in V} |\mathrm{Adj}(v)|^2 =$$

$$1 \cdot (n-1)^2 + (n-1) \cdot 1^2 \in \mathcal{O}\left(n^2\right)$$



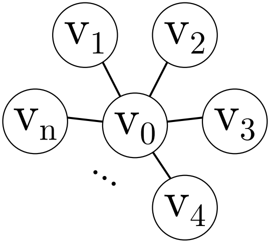
$$\mathcal{O}\left(\frac{m^2}{n} + m \cdot \lg(n)\right)$$

$$\mathcal{O}\left(\sum_{w \in V} |\mathrm{Adj}(w)|^2\right) = \mathcal{O}\left(\frac{m^2}{n}\right)$$

$$\sum_{v \in V} |\mathrm{Adj}(v)|^2 =$$

$$1 \cdot (n-1)^2 + (n-1) \cdot 1^2 \in \mathcal{O}\left(n^2\right)$$

$$\frac{(n-1)^2}{n} \in \mathcal{O}(n)$$



$$\mathcal{O}\left(\frac{m^2}{n} + m \cdot \lg(n)\right)$$

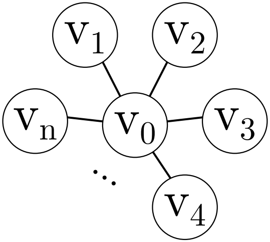
$$\mathcal{O}\left(\sum_{w \in V} |\mathrm{Adj}(w)|^2\right) = \mathcal{O}\left(\frac{m^2}{n}\right)$$

$$\sum_{v \in V} |\mathrm{Adj}(v)|^2 =$$

$$1 \cdot (n-1)^2 + (n-1) \cdot 1^2 \in \mathcal{O}\left(n^2\right)$$

$$\frac{(n-1)^2}{n} \in \mathcal{O}(n)$$

$$\mathcal{O}\left(\sum_{w \in V} |\mathrm{Adj}(w)|^2\right) \subseteq \mathcal{O}\left(m^2\right),$$



$\mathcal{O}\left(\frac{m^2}{n} + m \cdot \lg(n)\right)$

$\mathcal{O}\left(\sum_{w \in V} |\mathrm{Adj}(w)|^2\right) = \mathcal{O}\left(\frac{m^2}{n}\right)$

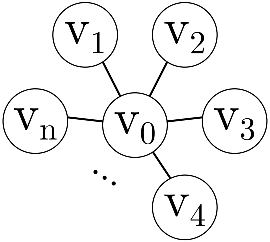
$\sum_{v \in V} |\mathrm{Adj}(v)|^2 =$

$1 \cdot (n-1)^2 + (n-1) \cdot 1^2 \in \mathcal{O}(n^2)$

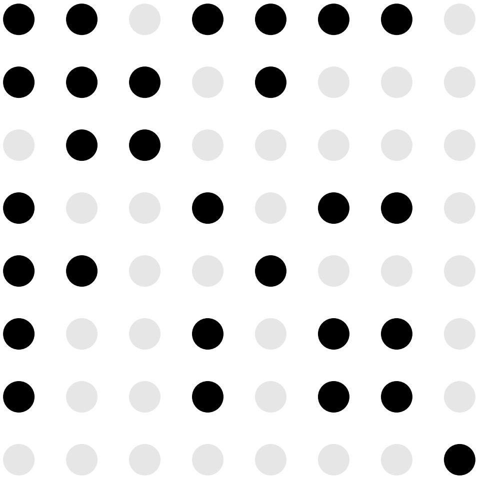
$\frac{(n-1)^2}{n} \in \mathcal{O}(n)$

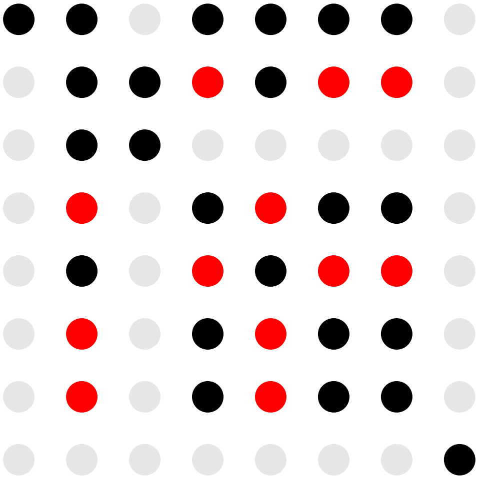
$\mathcal{O}\left(\sum_{w \in V} |\mathrm{Adj}(w)|^2\right) \subseteq \mathcal{O}(m^2)$,

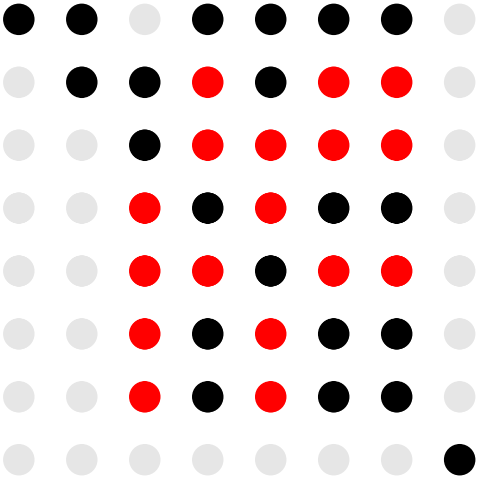
$\mathcal{O}\left(m^2 + m \cdot \lg(n)\right)$

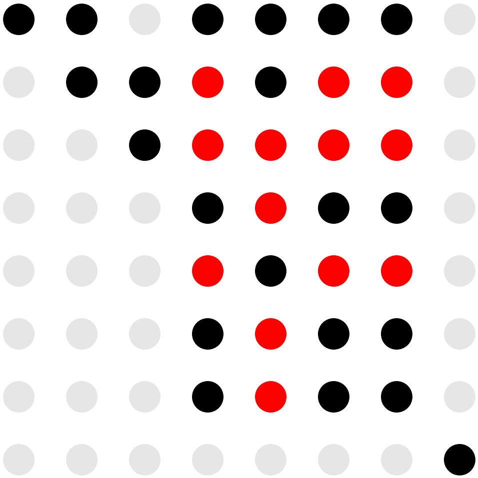


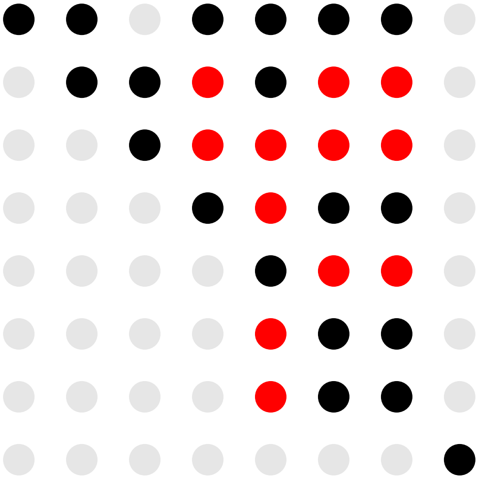
$$\begin{bmatrix} 9 & 2 & 0 & 2 & 1 & 1 & 2 & 0 \\ 2 & 8 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 7 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 8 & 0 & 1 & 3 & 0 \\ 1 & 1 & 0 & 0 & 9 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 7 & 2 & 0 \\ 2 & 0 & 0 & 3 & 0 & 2 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix}$$

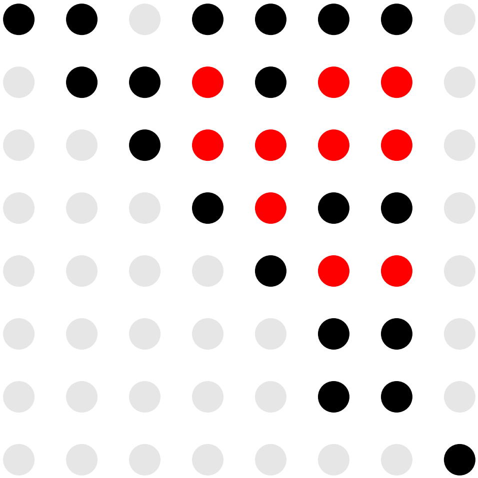


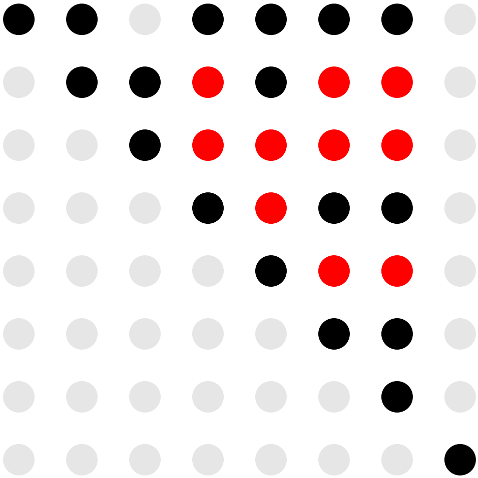




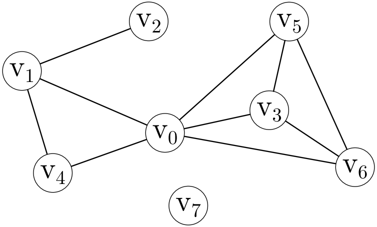




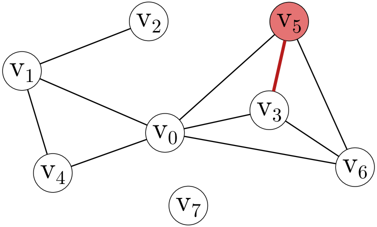




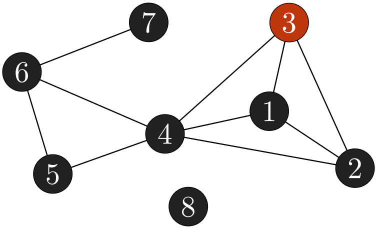
$$A = \begin{bmatrix} 9 & 2 & 0 & 2 & 1 & 1 & 2 & 0 \\ 2 & 8 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 7 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 8 & 0 & 1 & 3 & 0 \\ 1 & 1 & 0 & 0 & 9 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 7 & 2 & 0 \\ 2 & 0 & 0 & 3 & 0 & 2 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix}$$



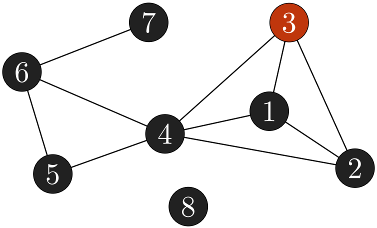
$$A = \begin{bmatrix} 9 & 2 & 0 & 2 & 1 & 1 & 2 & 0 \\ 2 & 8 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 7 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 8 & 0 & 1 & 3 & 0 \\ 1 & 1 & 0 & 0 & 9 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 7 & 2 & 0 \\ 2 & 0 & 0 & 3 & 0 & 2 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix}$$

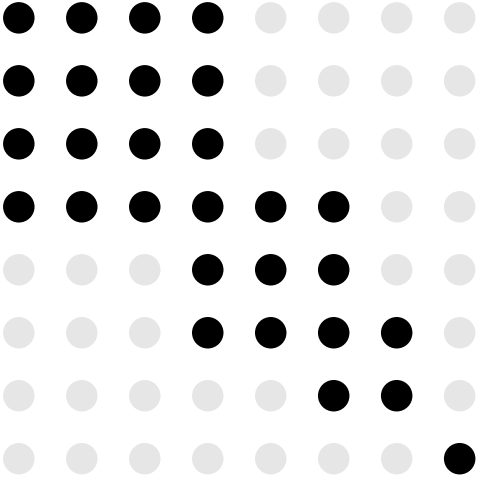


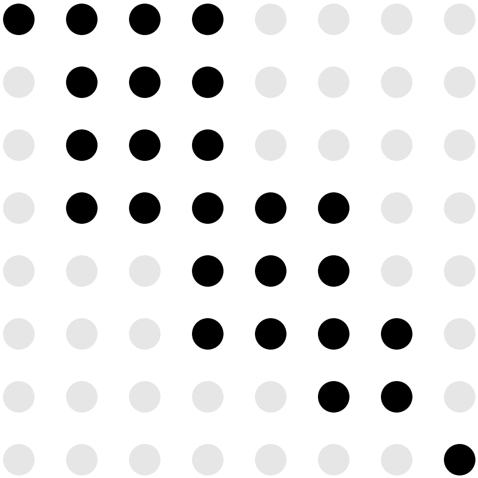
$$A = \begin{bmatrix} 9 & 2 & 0 & 2 & 1 & 1 & 2 & 0 \\ 2 & 8 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 7 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 8 & 0 & 1 & 3 & 0 \\ 1 & 1 & 0 & 0 & 9 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 7 & 2 & 0 \\ 2 & 0 & 0 & 3 & 0 & 2 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix}$$

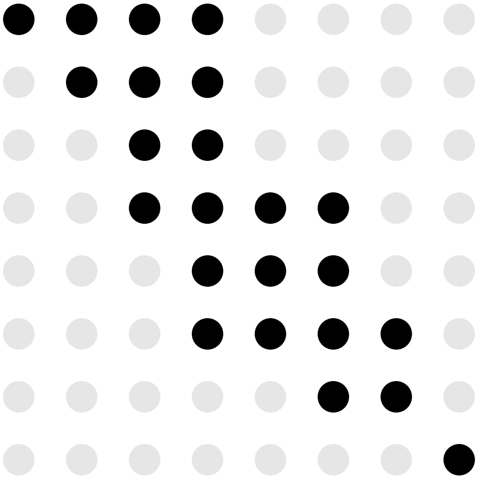


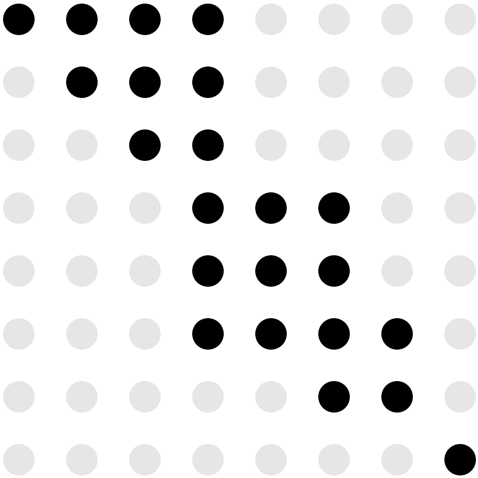
$$A^* = \begin{bmatrix} 8 & 3 & 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 9 & 2 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 7 & 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 9 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 9 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 8 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix}$$

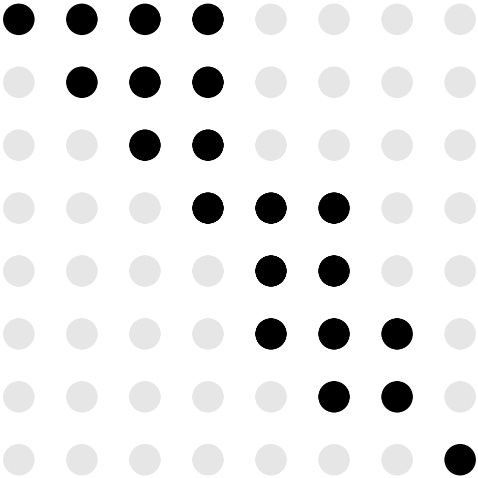


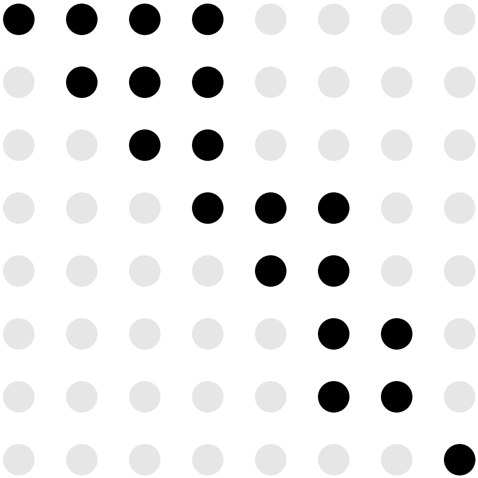


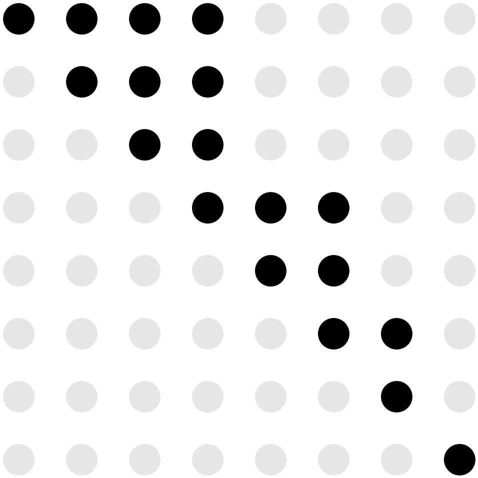












$$\begin{bmatrix} 8 & 3 & 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 9 & 2 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 7 & 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 9 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 9 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 8 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{bmatrix}$$