Efficient Planarity Testing

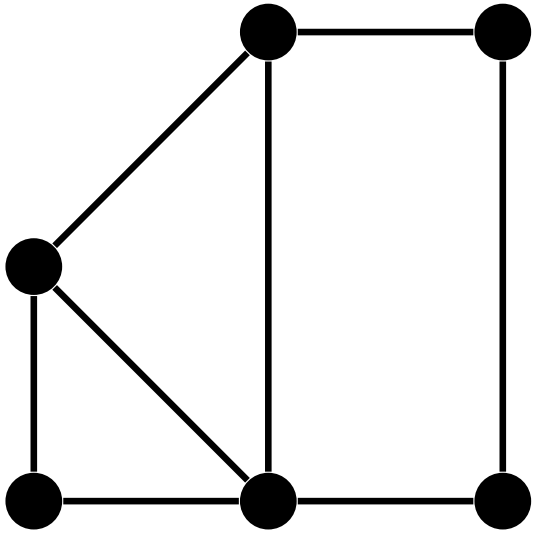
Seminar *Algorithms on Sparse Graphs*

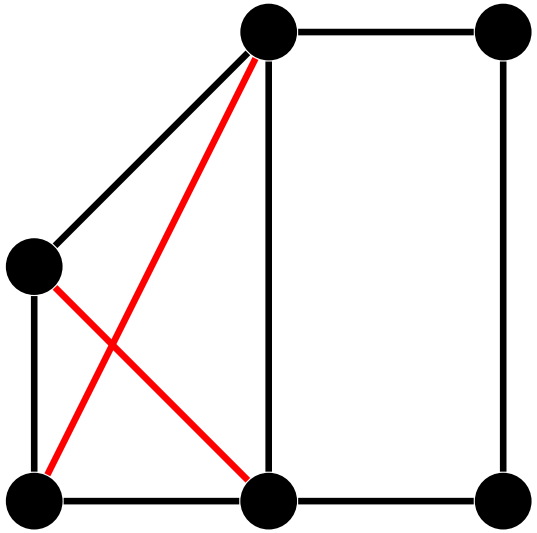
Kevin Behrens

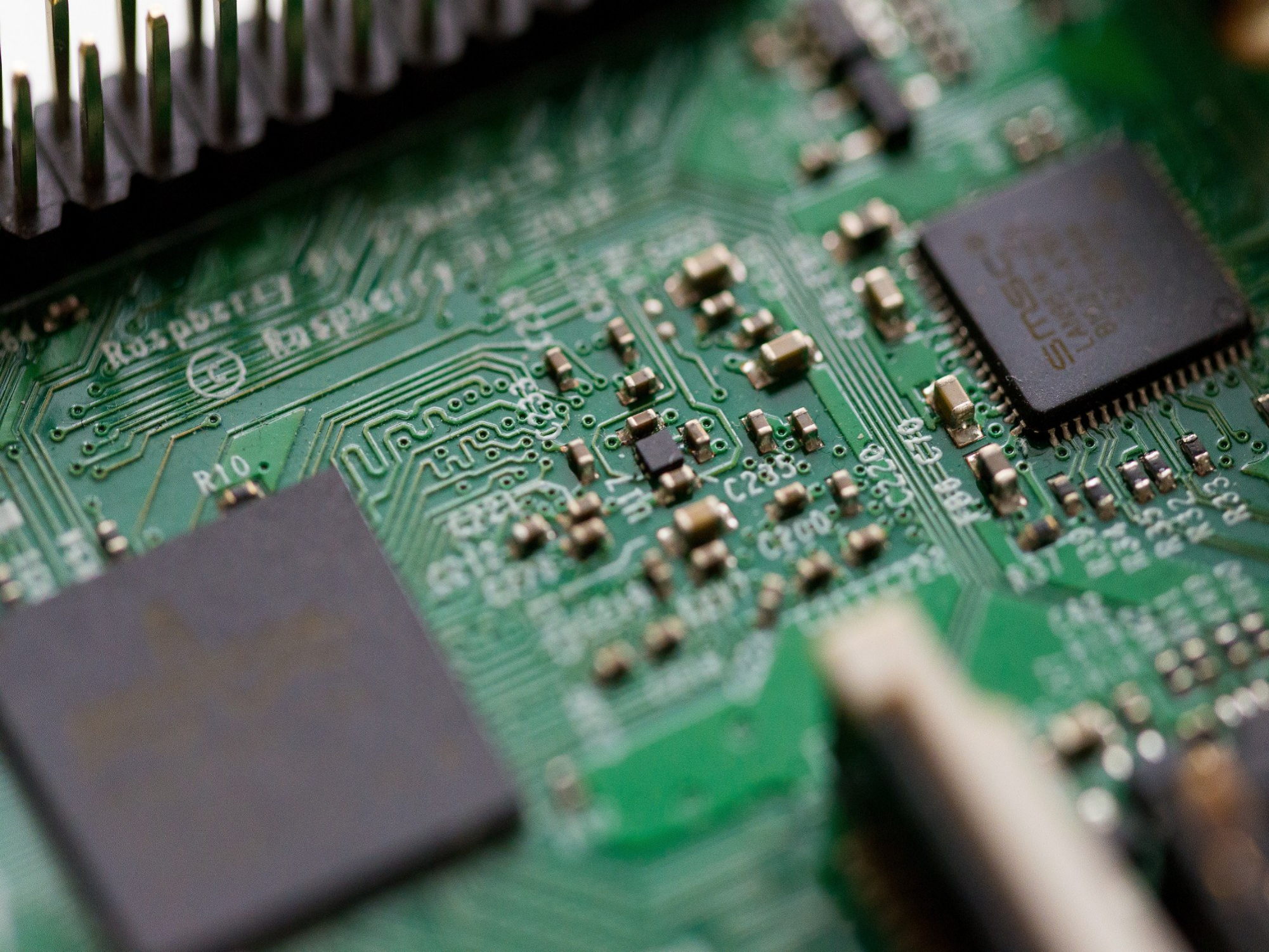
26. April 2018

Inhalt

Was ist ein planarer Graph?






R10
C233
C200

Motivation

Es existieren zahlreiche Anwendungsfälle:

- Städte- und Straßenplanung (bspw. *Autobahnkreuze*)
- Leiterplatinen Design
- Chemie: Kanonische Molekulardarstellung

Zusätzlich besitzen planare Graphen Eigenschaften, die einige Probleme vereinfachen:

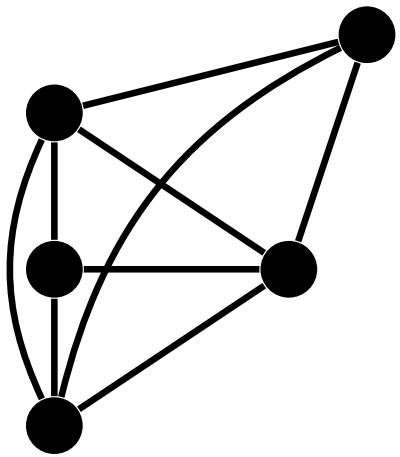
- Effiziente Speicherung
- Planare Graphen sind immer 4-färbbar. (4-COLOR)
- Algorithmen für spezifische Probleme sind einfacher. (bspw. MAX CUT)

Ziel

Effizienter Algorithmus für einen Planaritätstest von beliebigen Graphen

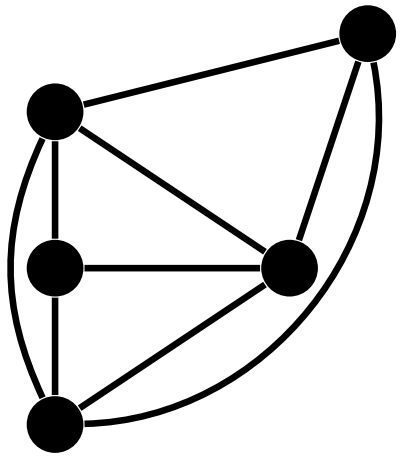
Unterschied: Planarer Graph / Planare Zeichnung

Planarer Graph



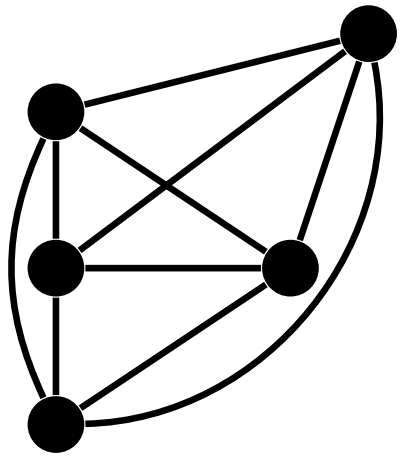
Nicht-Planare Zeichnung

Planarer Graph



Planare Zeichnung

Nicht-Planarer Graph



Nicht-Planare Zeichnung

Planarität

Definition

Planarität von Graphen. Ein Graph G ist planar, g.d.w. wenn mindestens eine Einbettung der Knoten und Kanten in die Ebene (planare Zeichnung) existiert, so dass gilt:

1. Jeder Knoten v wird auf jeweils einen Punkt abgebildet.
2. Jede Kante (v, w) wird auf eine einfache Kurve abgebildet, wobei die Knoten die Endpunkte der Kurve bilden.
3. Die Einbettungen der Kanten schneiden sich nur in gemeinsamen Endpunkten (Knoten).

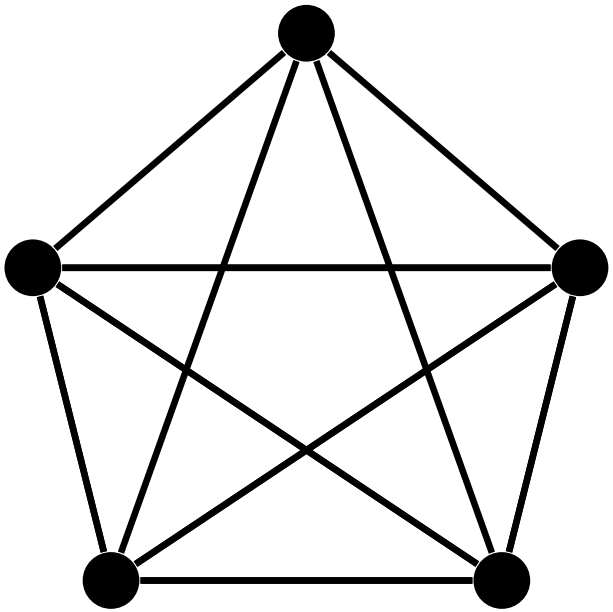
Euler's Theorem

Definition

Euler's Theorem (1758). Bei einem planaren Graphen G mit V Knoten und E Kanten, ist die Kantenanzahl wie folgt beschränkt:

$$E \leq 3V - 3$$

Satz von Kuratowski



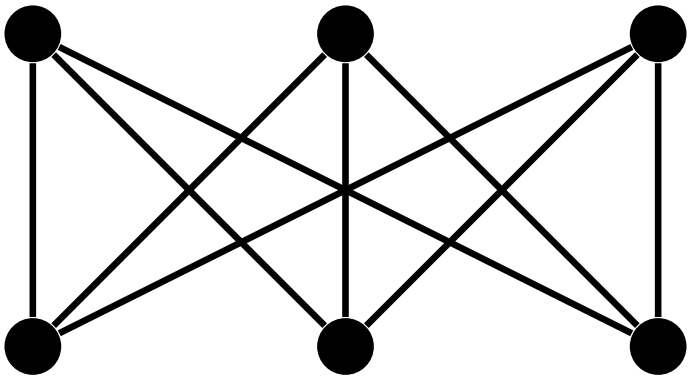


Abbildung: Minimale Nicht-Planare Graphen: K_5 und $K_{3,3}$

Satz von Kuratowski

Definition

Satz von Kuratowski (1930). Ein Graph G ist genau dann planar, wenn G keinen Teilgraphen enthält der K_5 oder $K_{3,3}$ entspricht oder ein Unterteilungsgraph dieser ist.

Unterteilungsgraph

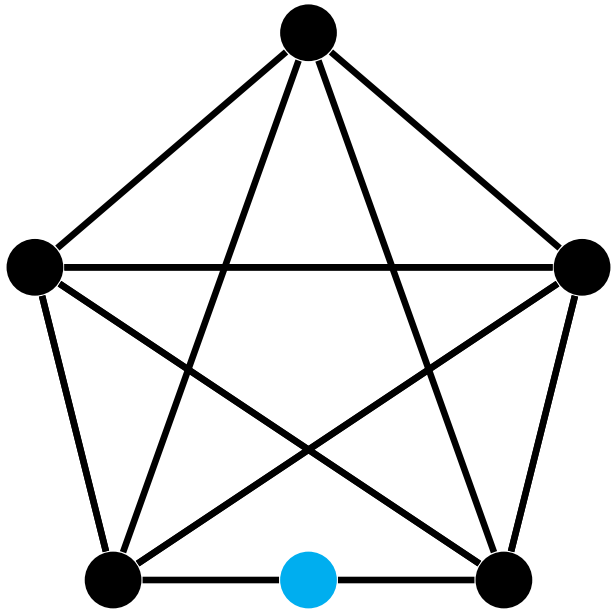


Abbildung: Unterteilungsgraph von K_5

Unterteilungsgraph

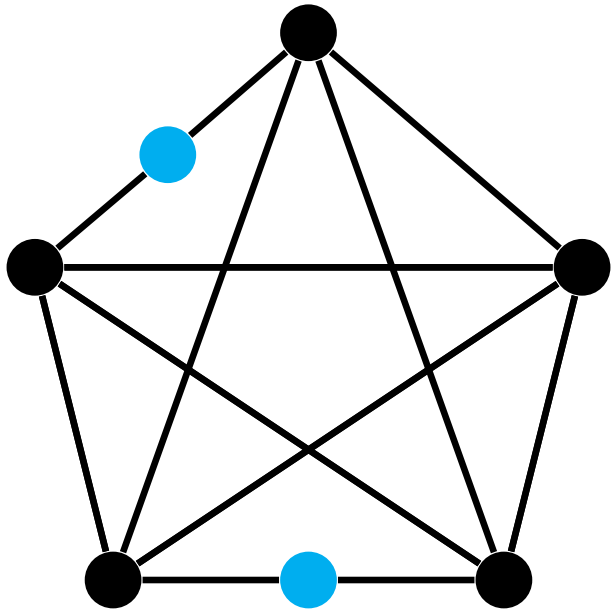


Abbildung: Unterteilungsgraph von K_5

Historie

Bisher verfolgte Strategie:
Sukzessiver Aufbau einer planaren Einbettung eines Graphen durch Hinzunahme von Knoten und Kanten.

Jahr	Algorithmus	Laufzeit
1961	Auslander und Parter + (Goldstein)	$\mathcal{O}(n^3)$
1964	Demoucron, Malgrange und Pertuiset	$\mathcal{O}(n^2)$
1967	Lempel, Even und Cederbaum	$\mathcal{O}(n^2)$
1974	**Hopcroft und Tarjan**	$\mathcal{O}(n)$

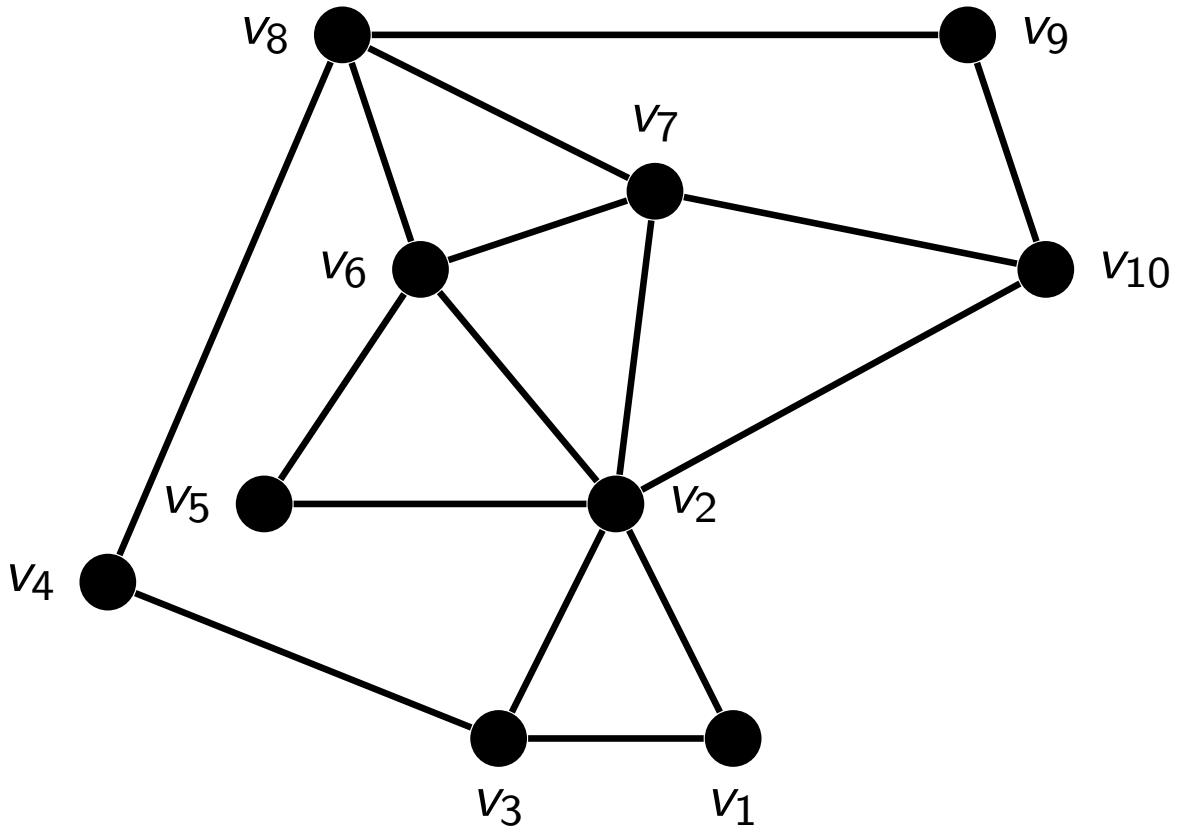
Struktur des Algorithmus

- Sukzessiver Aufbau einer Einbettung von Pfaden mithilfe von Tiefensuche

Ablauf

1. Graphen mit "zu vielen" Kanten aussortieren
2. Aufteilung des Graphen in Zusammenhangskomponenten
3. Umstrukturierung des Graphen in einen Tiefensuchbaum
4. Kreis im Graphen finden und in die Ebene einbetten
5. Planaritätstest für jeden "Teil" durch rekursive Anwendung

Beispielgraph



Anwendung von Euler's Theorem

Definition

Euler's Theorem (1758). Bei einem planaren Graphen G mit V Knoten und E Kanten, ist die Kantenanzahl wie folgt beschränkt:

$$E \leq 3V - 3$$

In unserem Fall:
$V = 10$ und $E = 16$. Daraus folgt:

$$E \leq 3 \cdot V - 3$$
$$16 \leq 3 \cdot 10 - 3$$
$$16 \leq 27 \checkmark$$

Aufteilung in Komponenten

Definition

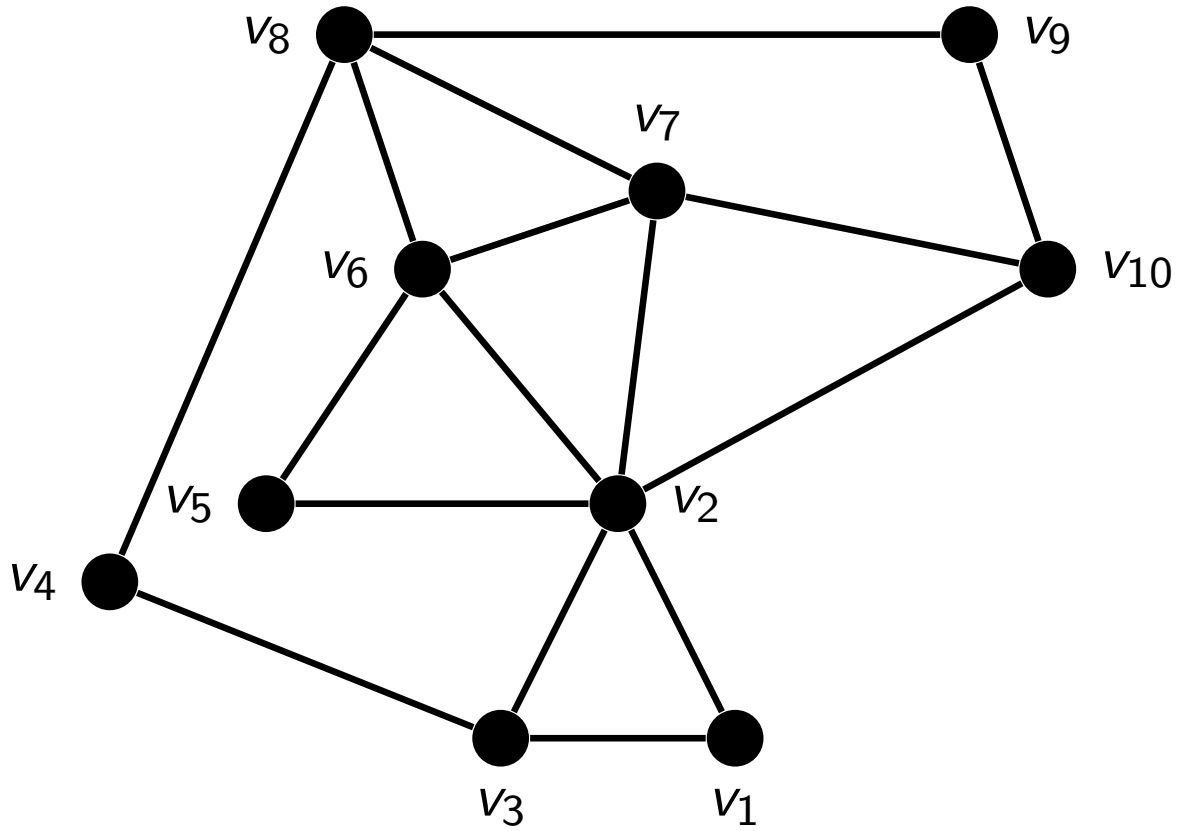
2-Zusammenhang (Biconnected Components). Ein Graph G ist genau dann 2-zusammenhängend, wenn er keine Knoten enthält die einen *Trenner* darstellt. *Trenner* sind Knoten, die bei Entfernung aus dem Graphen, die Anzahl der Zusammenhangskomponenten erhöhen würden.

Definition

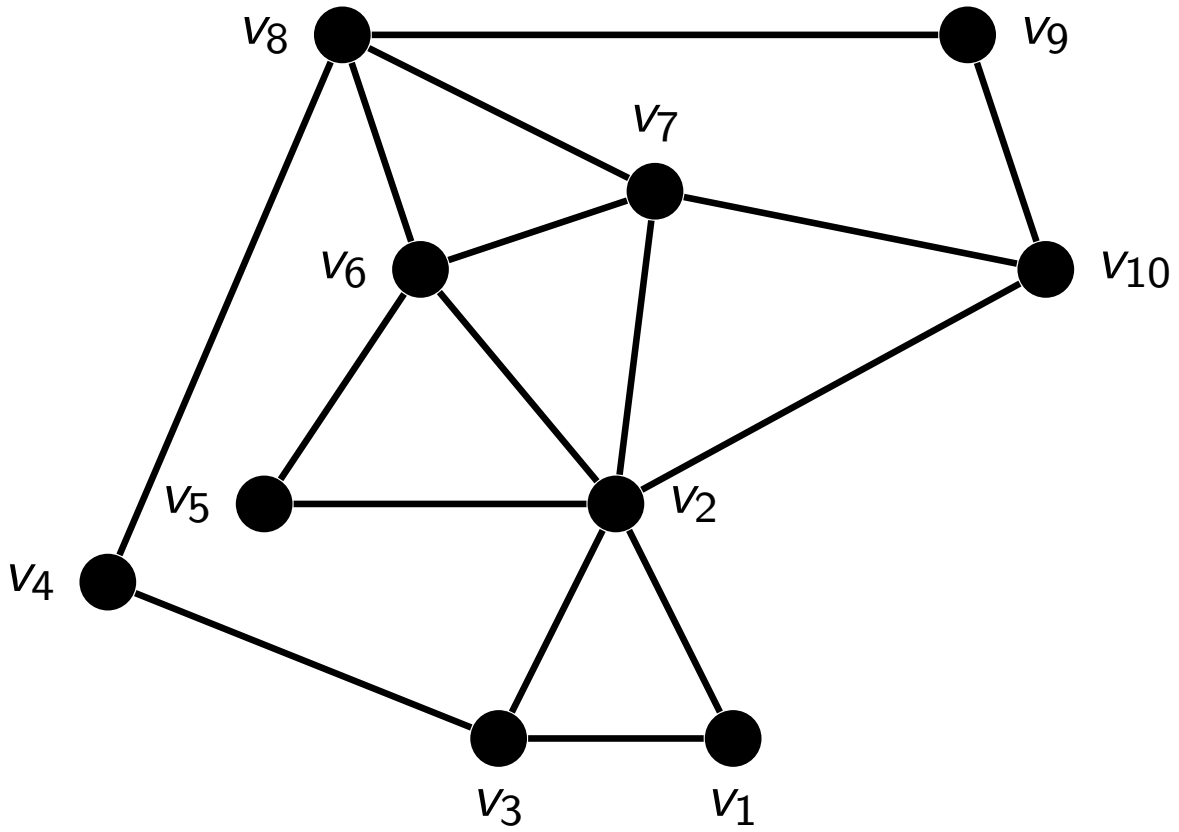
Planarität mit 2-Zusammenhang. Ein Graph G ist genau dann planar, wenn alle seine 2-Zusammenhangskomponenten planar sind.

Beispiel: Komponenten

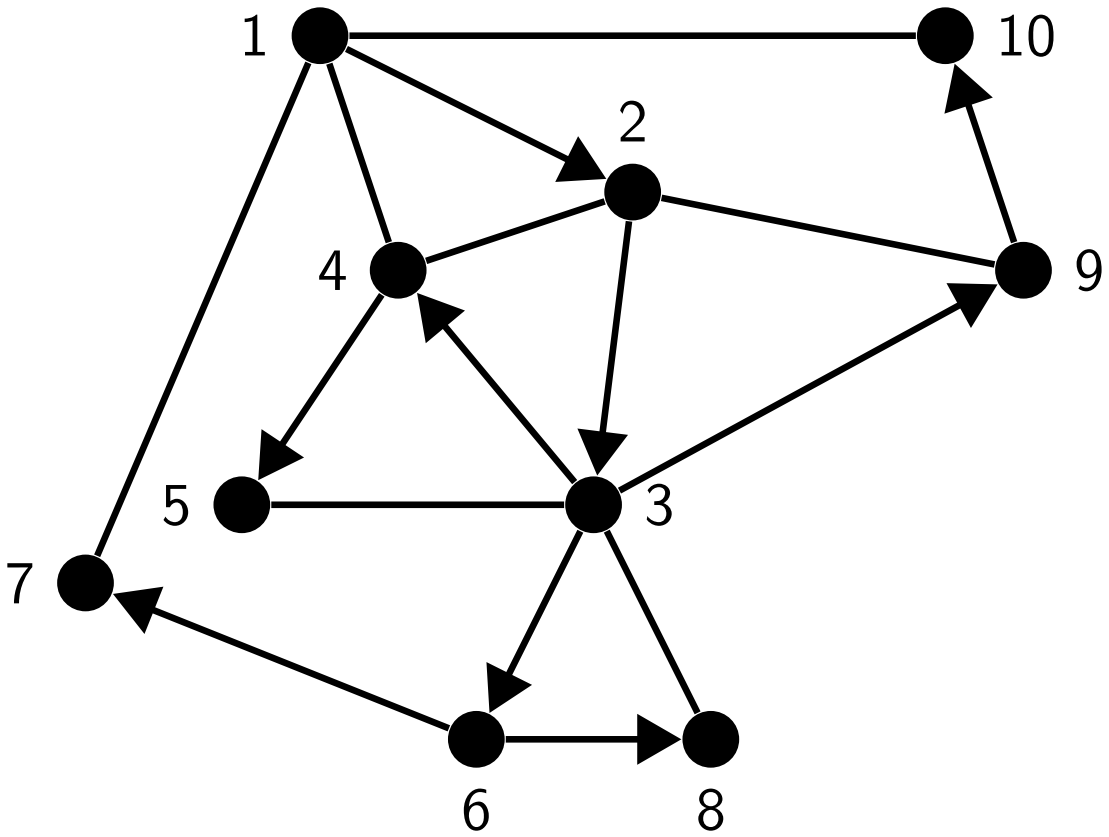
Hier: 1 ZHK und keine Trenner. Somit ist G 2-zusammenhängend.



Tiefensuche



Tiefensuche



Partionierung

Definition

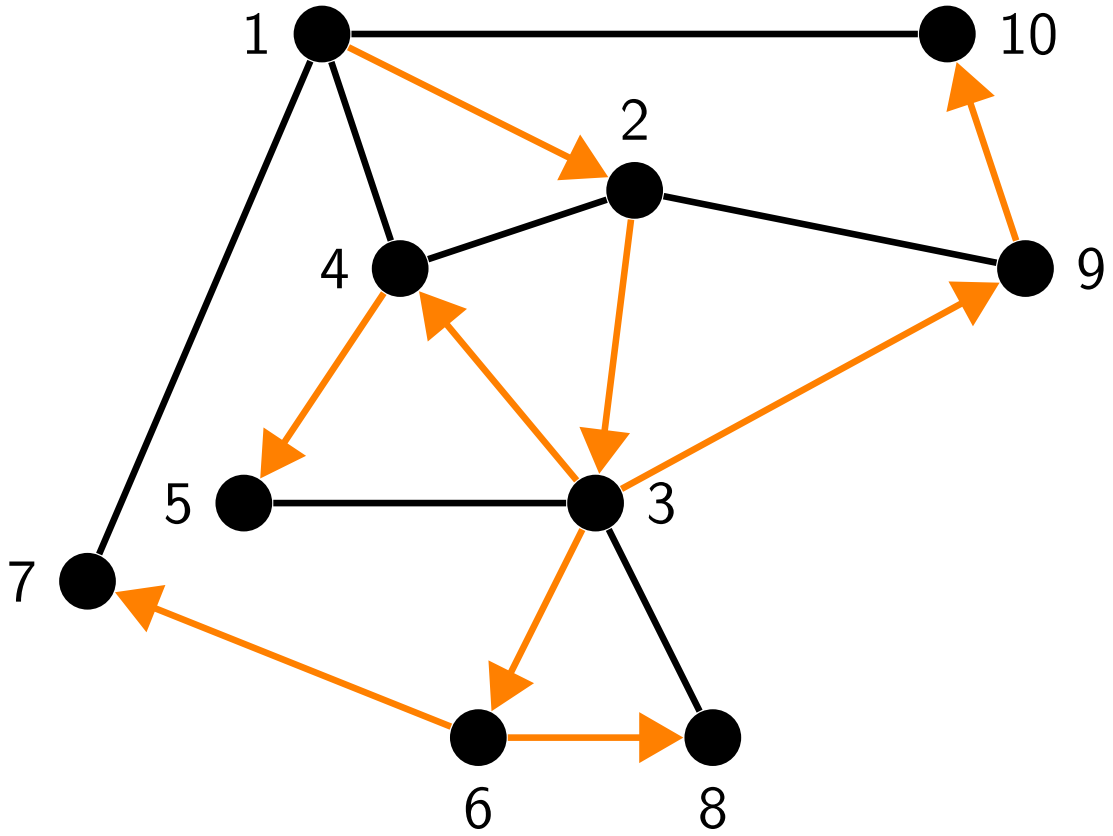
Baumkanten (tree arcs). Kanten von Graph G, die bei der Tiefensuche genutzt wurden um den Graphen zu durchlaufen. Diese Kanten repräsentieren den Tiefensuchbaum.

Definition

Rückwärtskanten (fronds / back-edges). Kanten von Graph G, die bei der Tiefensuche nicht genutzt wurden und von einem Knoten zu einem anderen Knoten führen, welcher bereits *vorher besucht wurde*.

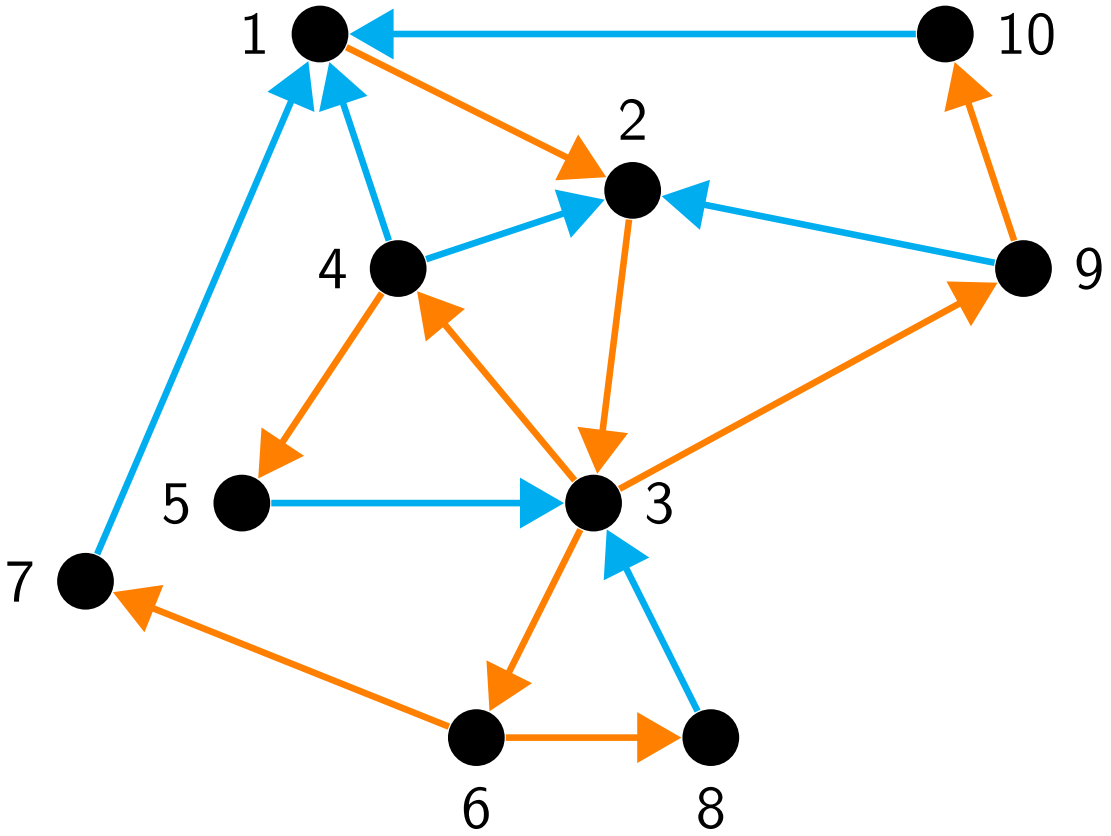
Partionierung

Markierung der Baumkanten und Rückwärtskanten:

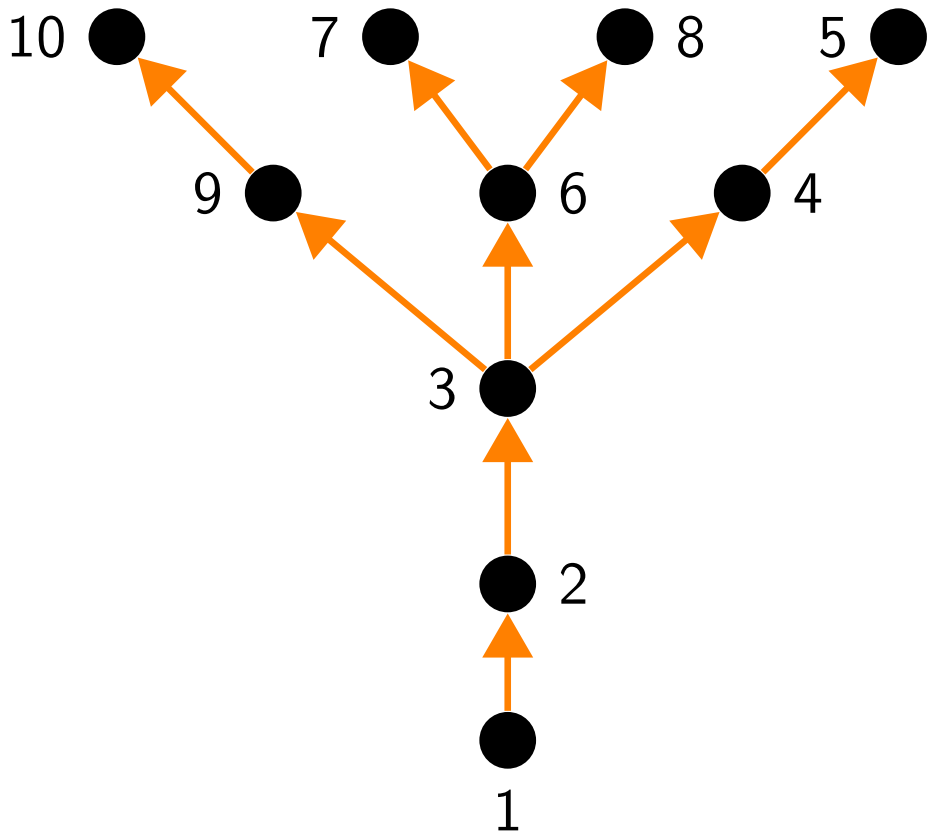


Partionierung

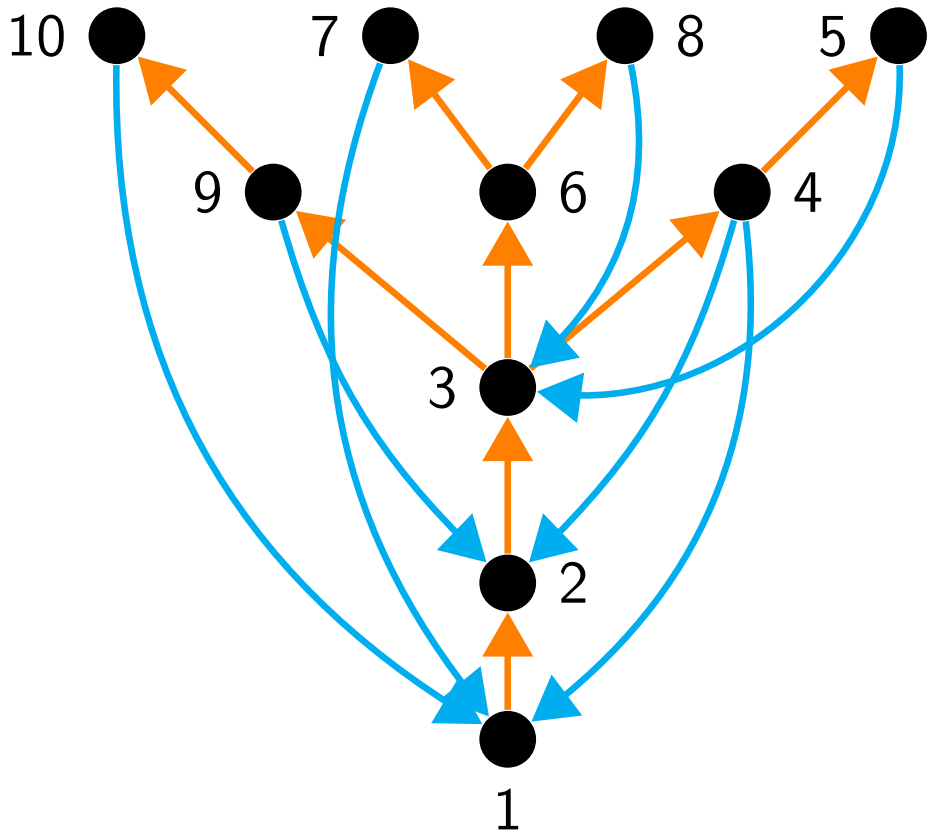
Markierung der Baumkanten und Rückwärtskanten:



Tiefensuchbaum



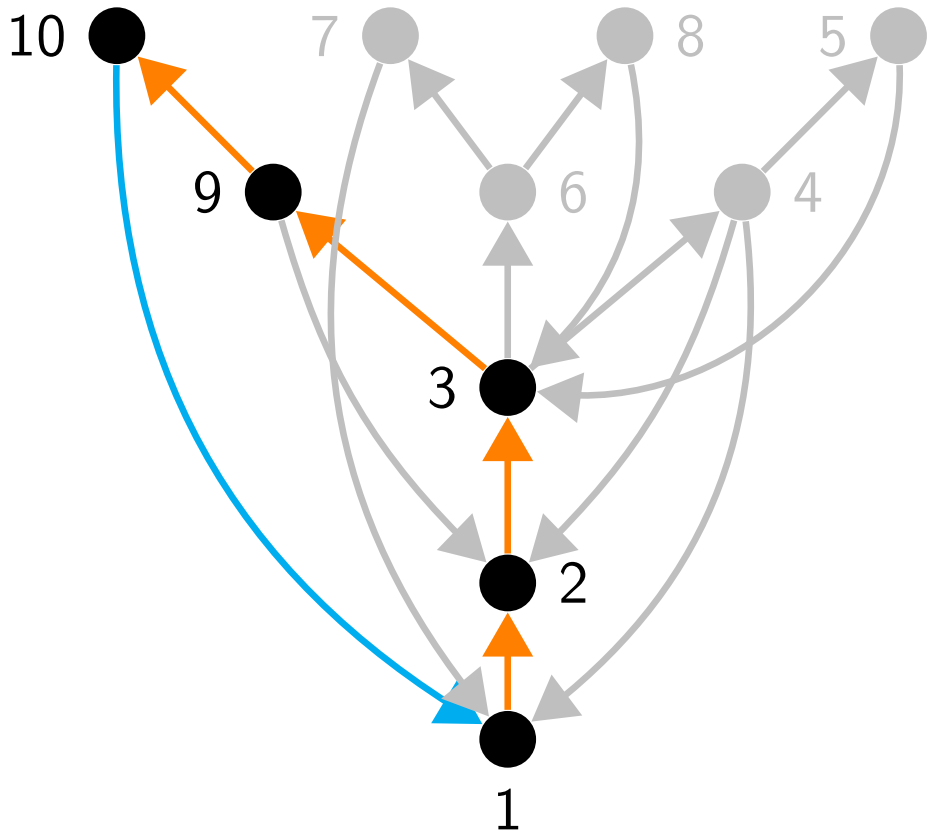
Tiefensuchbaum (Palm Tree)



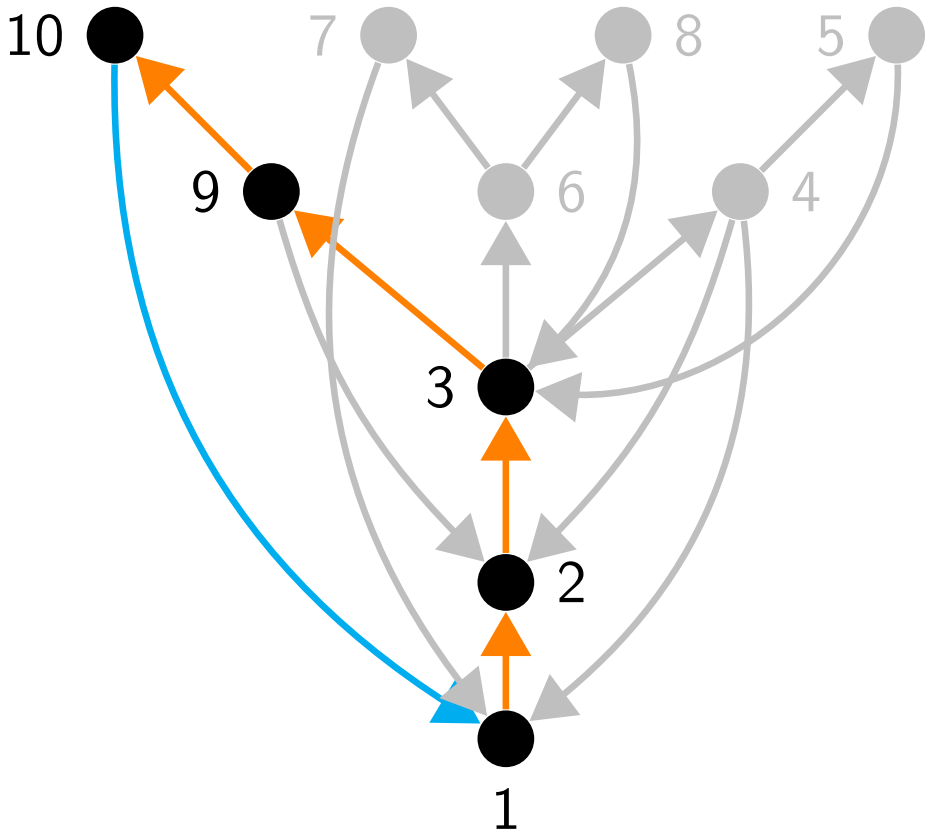
Pathfinding

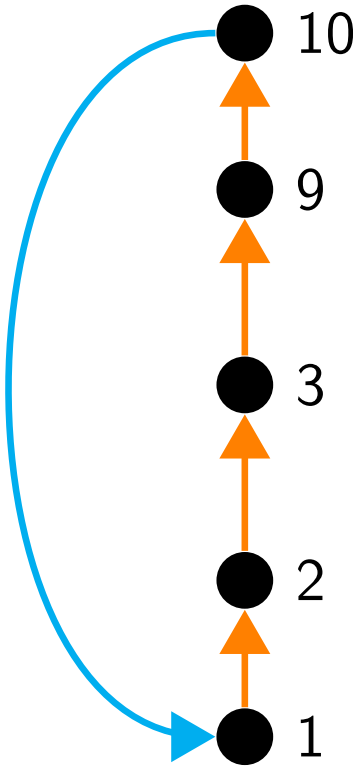
- Für jeden Aufruf des Algorithmus: Finden eines Zykels im zu testenden "Teil"
- Pfad besteht aus einer Sequenz von Baumkanten gefolgt von *einer* Rückwärtskante
- Vorgehensweise: Erweiterte Tiefensuche
- Besondere Reihenfolge zur Suche der Pfade, um die Effizienz zu steigern

Zykel

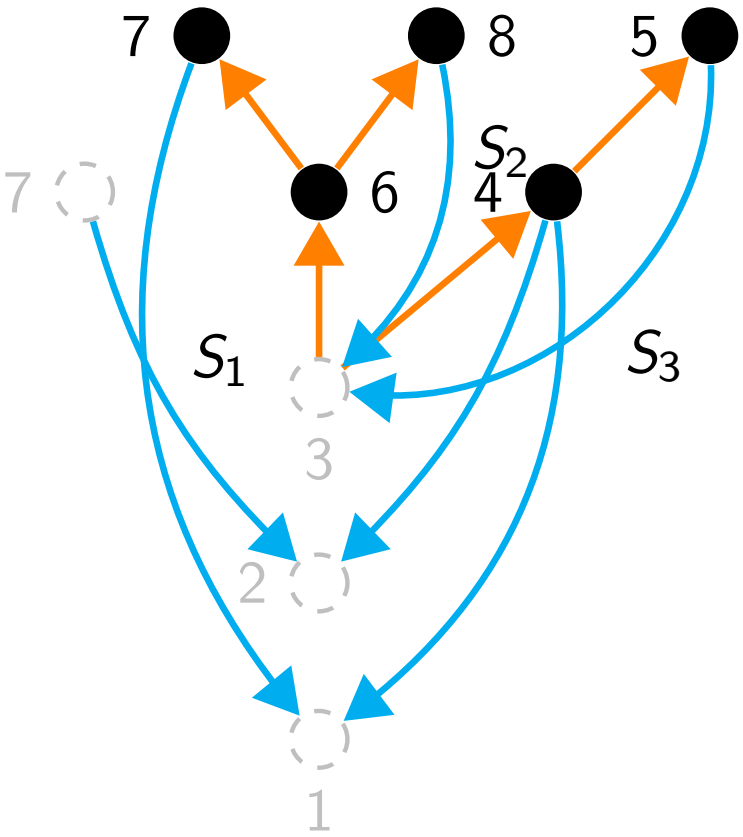


Einbettung des Zykels

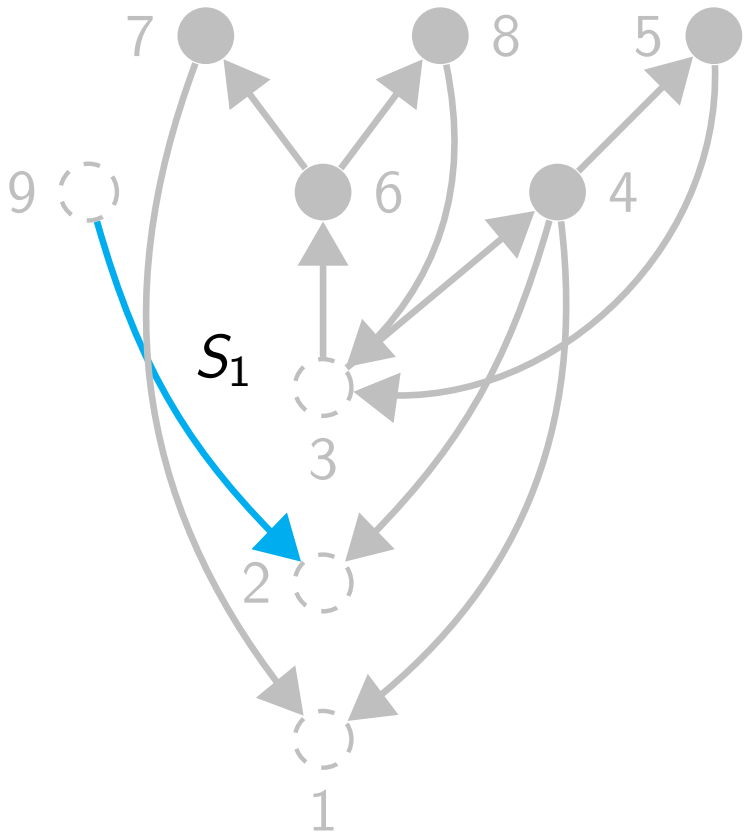




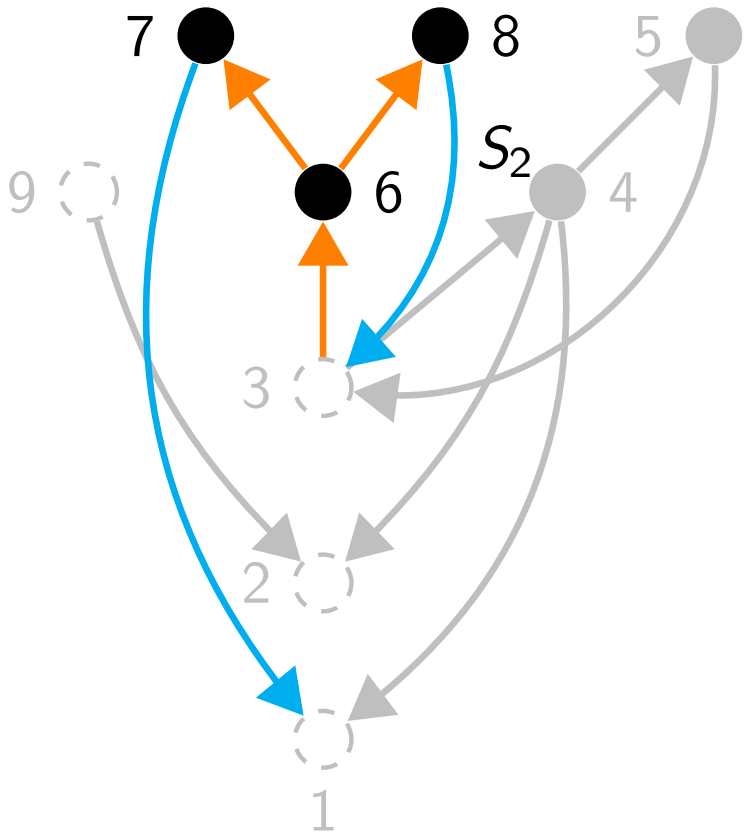
Segmente



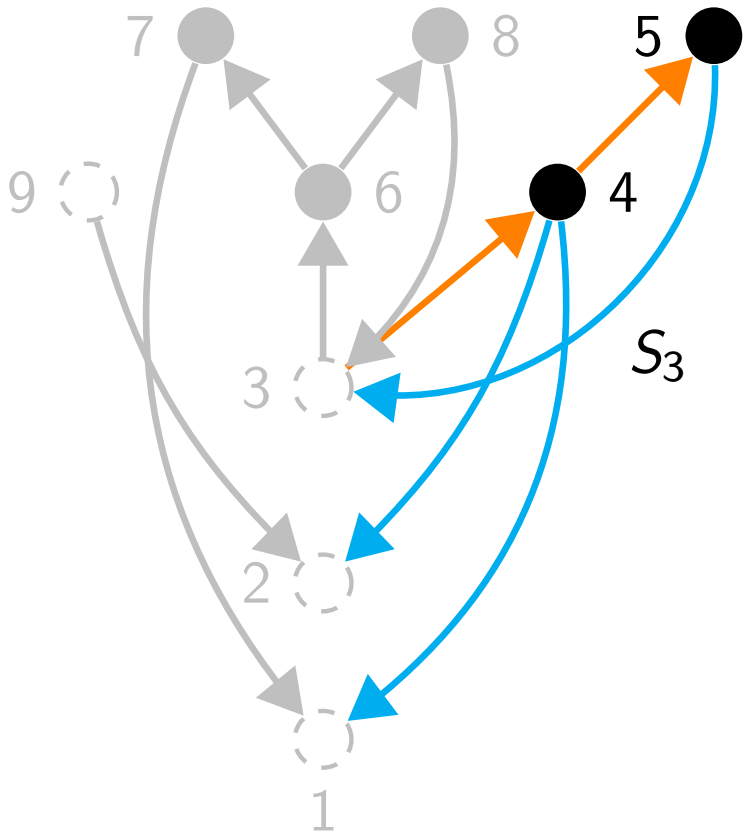
Segment S_1



Segment S_2



Segment S_3



Segmente

Durch Entfernen des gefundenen Zykels zerfällt der Graph in verschiedene Segmente:

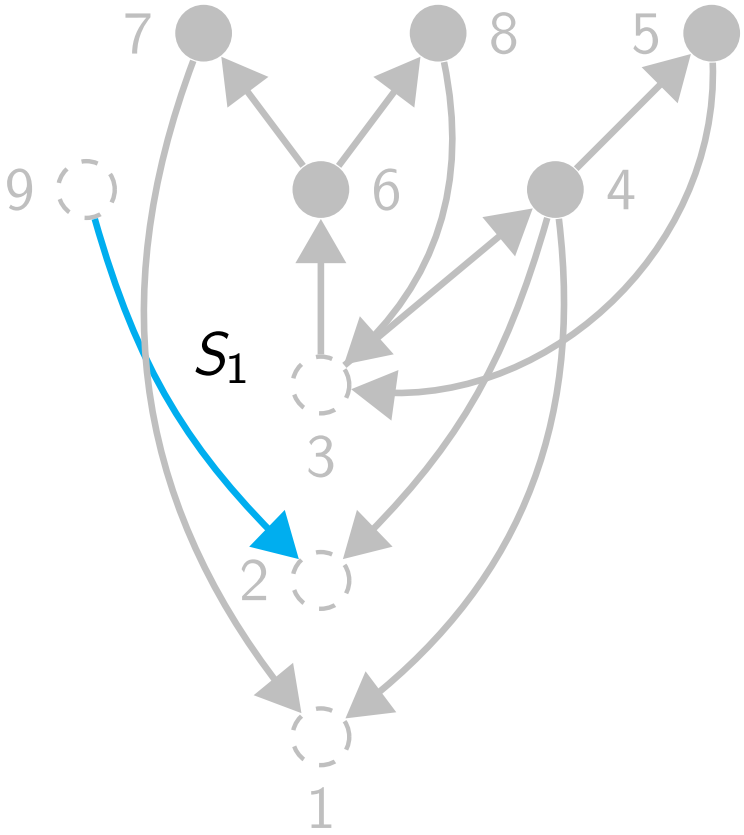
Definition

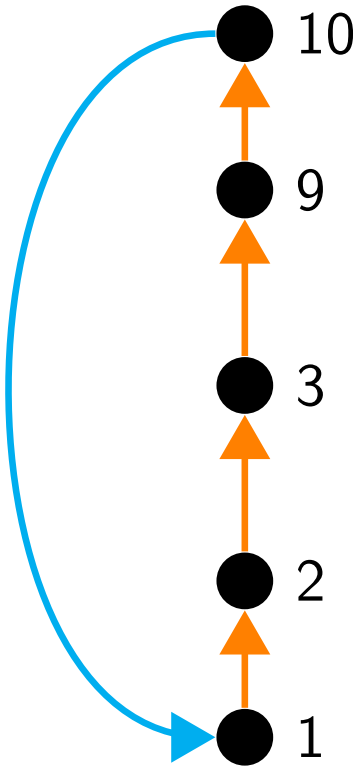
Segment (pieces). Ein Segement ist entweder eine einzelne Rückwärtskante (v, w) oder eine Baumkante (v, w) inklusive eines Subbaumes mit Wurzel w und allen zugehörigen Rückwärtskanten.

Einbettung

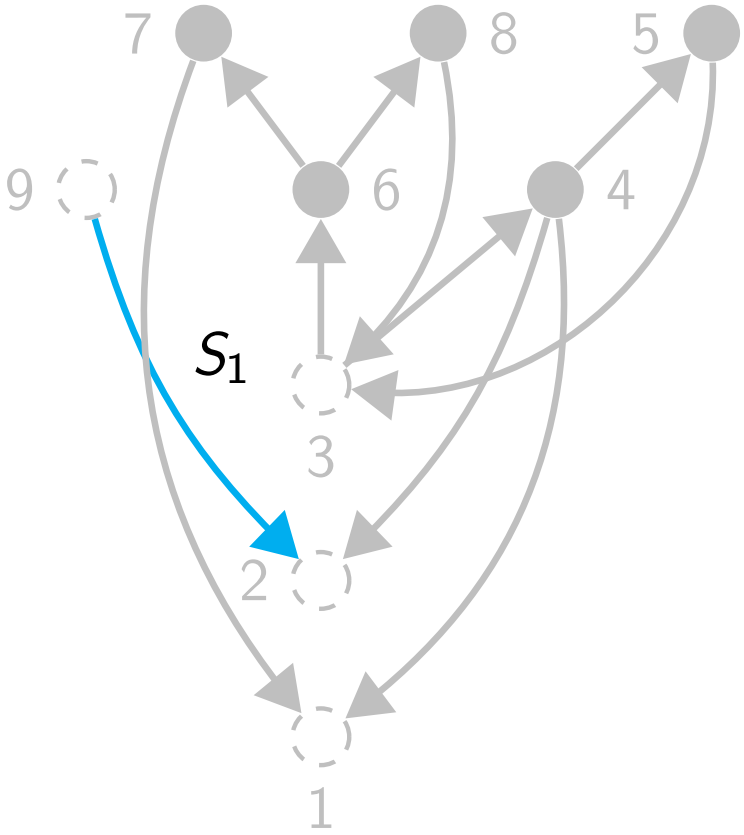
- Rekursives Anwenden des Algorithmus auf die einzelnen Segmente
- Einbettung der Segmente zum Zeitpunkt der “Entdeckung” (Pathfinding)
- Ein Segment muss im Ganzen auf einer Seite des Zykels eingebettet werden
- Wenn Rückwärtskanten eine Einbettung verhindern, versuche durch Verschieben von (Teil-)Segmenten Einbettung zu ermöglichen
- Falls planare Einbettung nicht möglich, ist der Graph nicht planar!

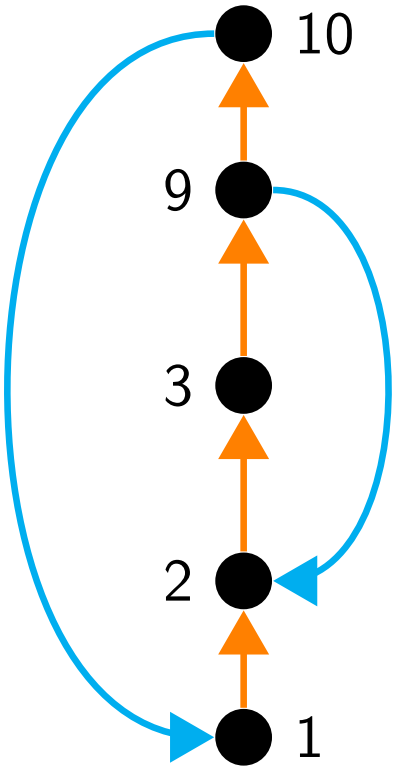
Einbettung



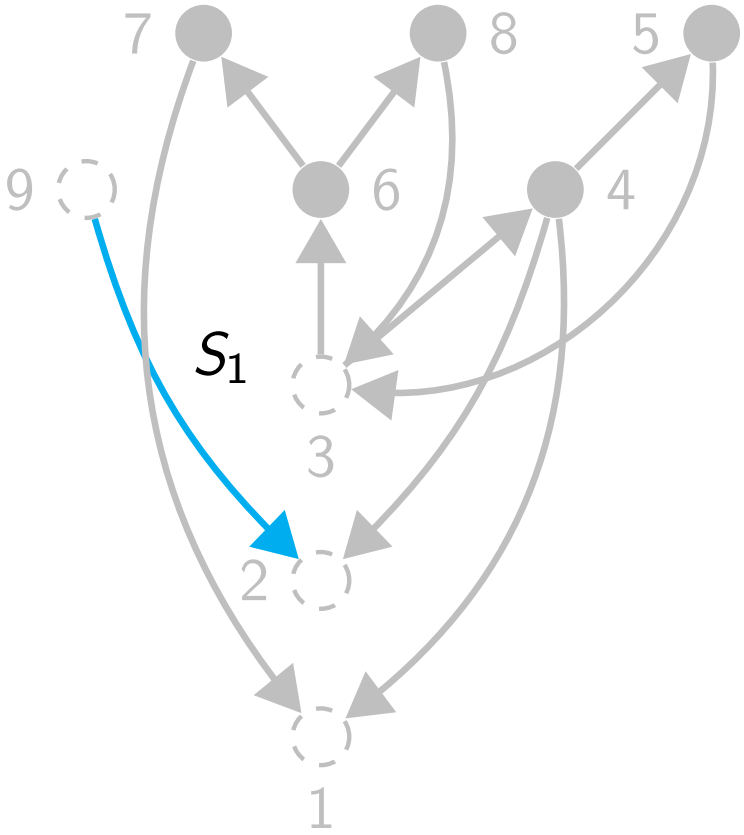


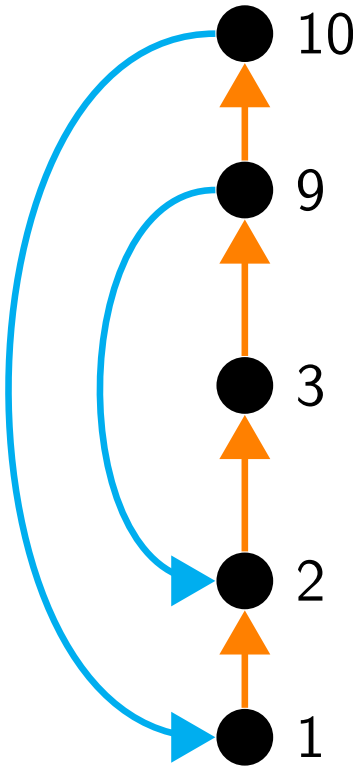
Einbettung



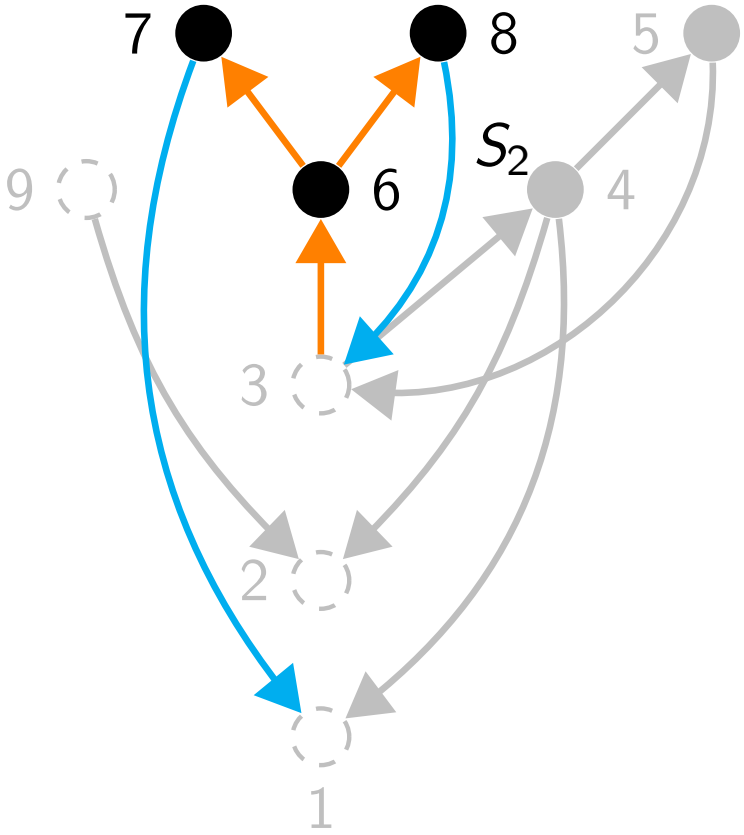


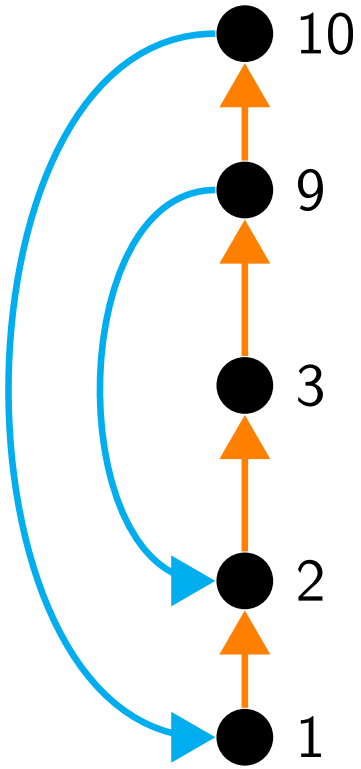
Einbettung



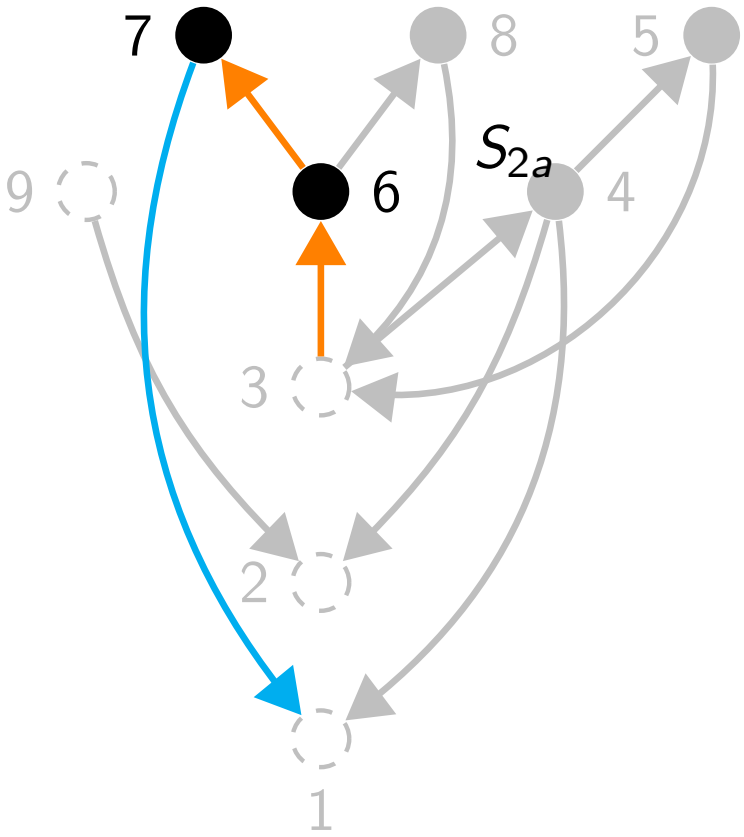


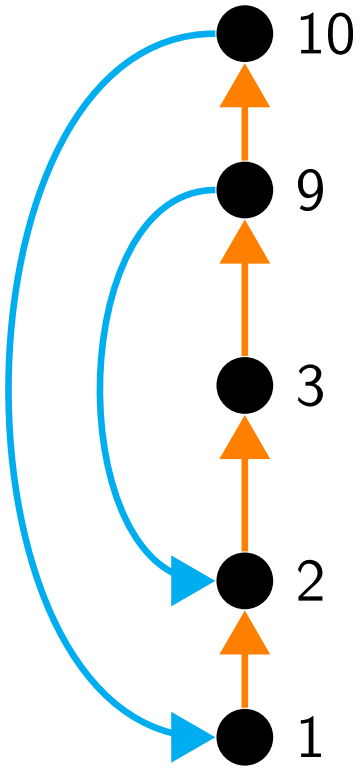
Einbettung



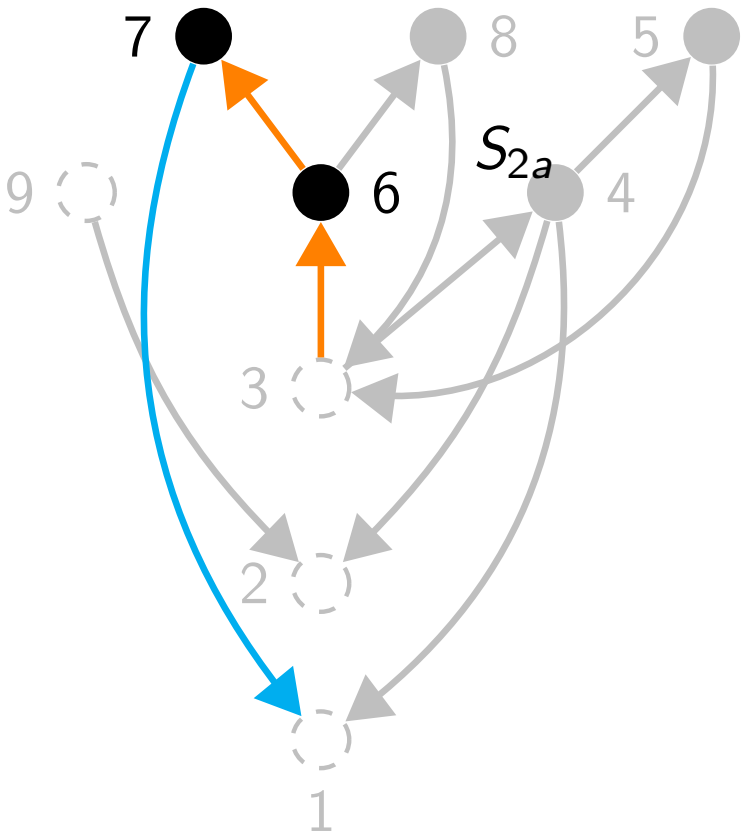


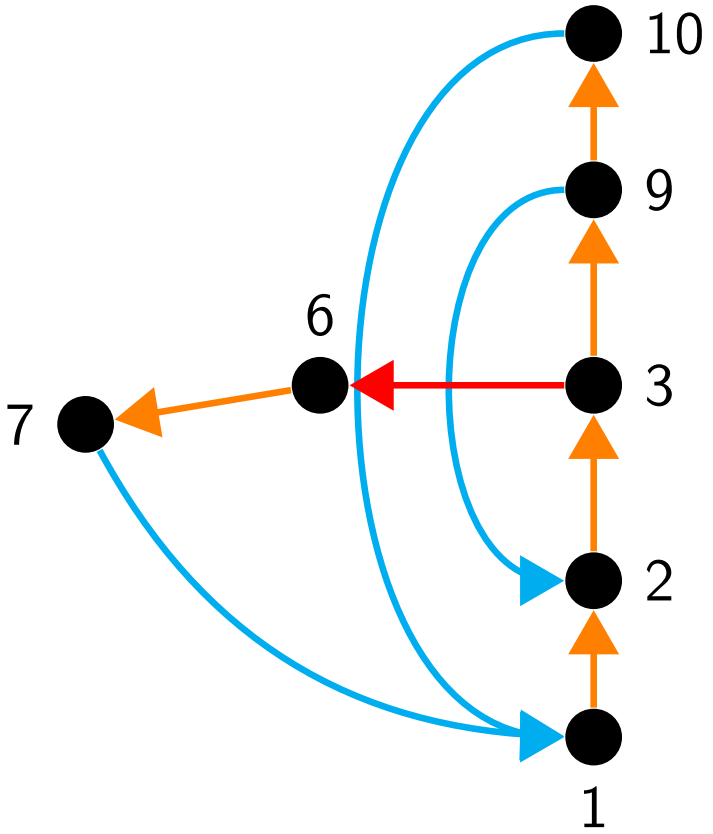
Einbettung



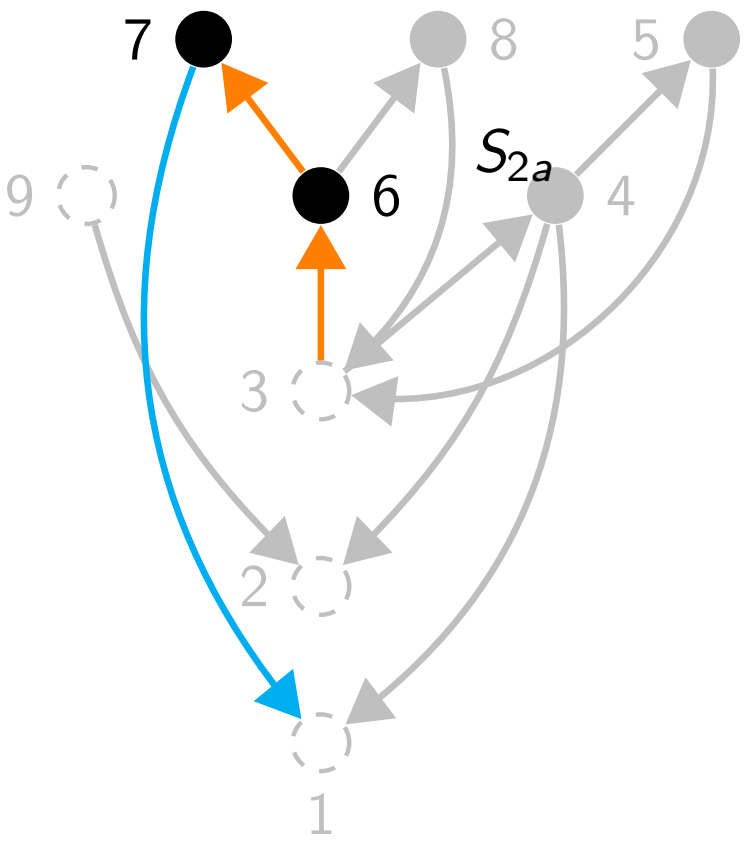


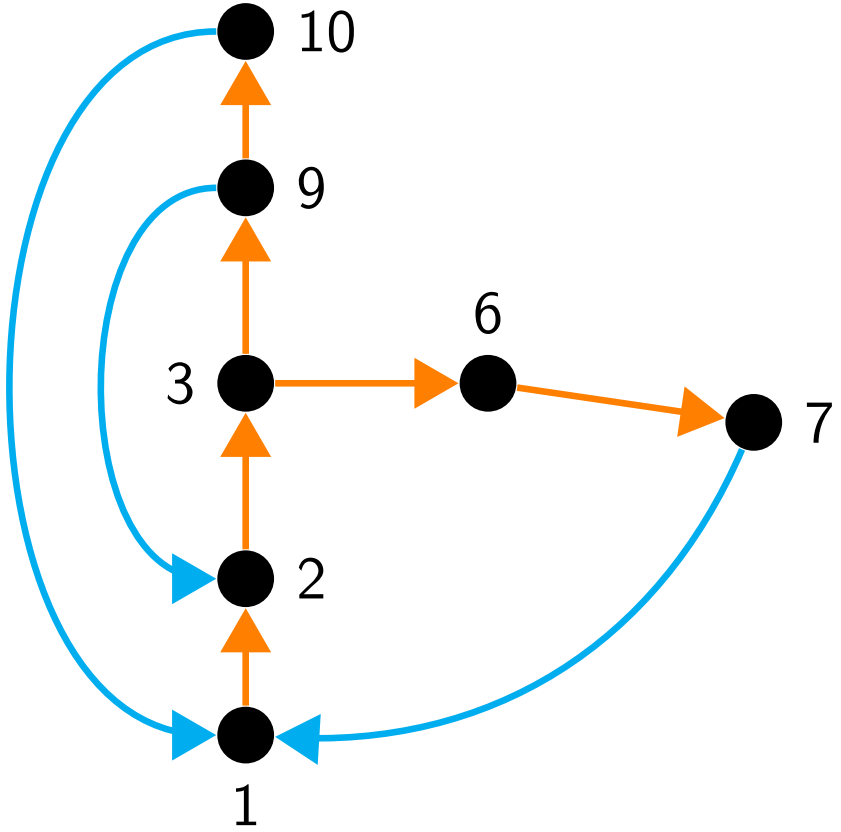
Einbettung



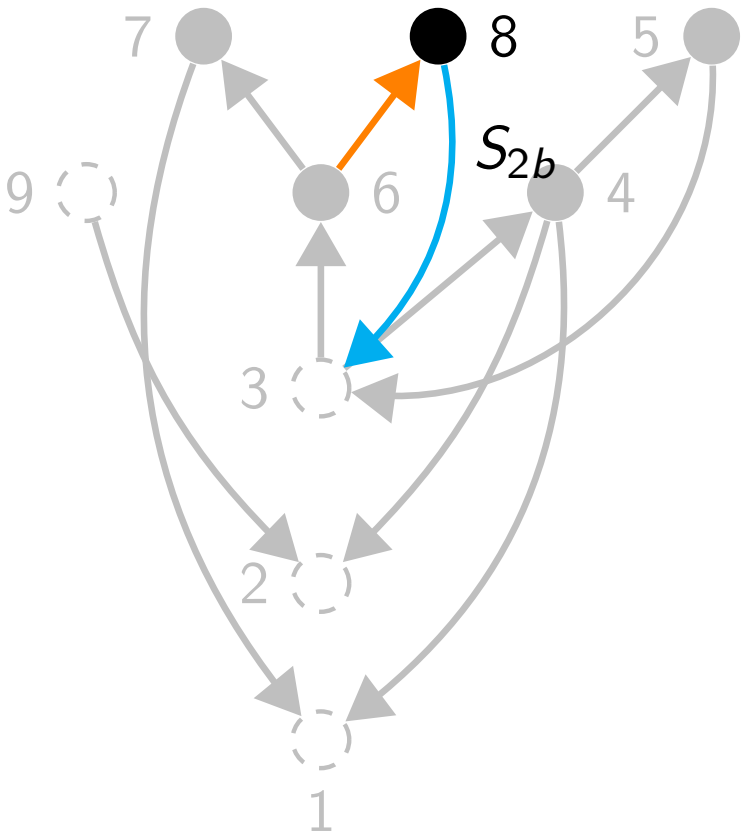


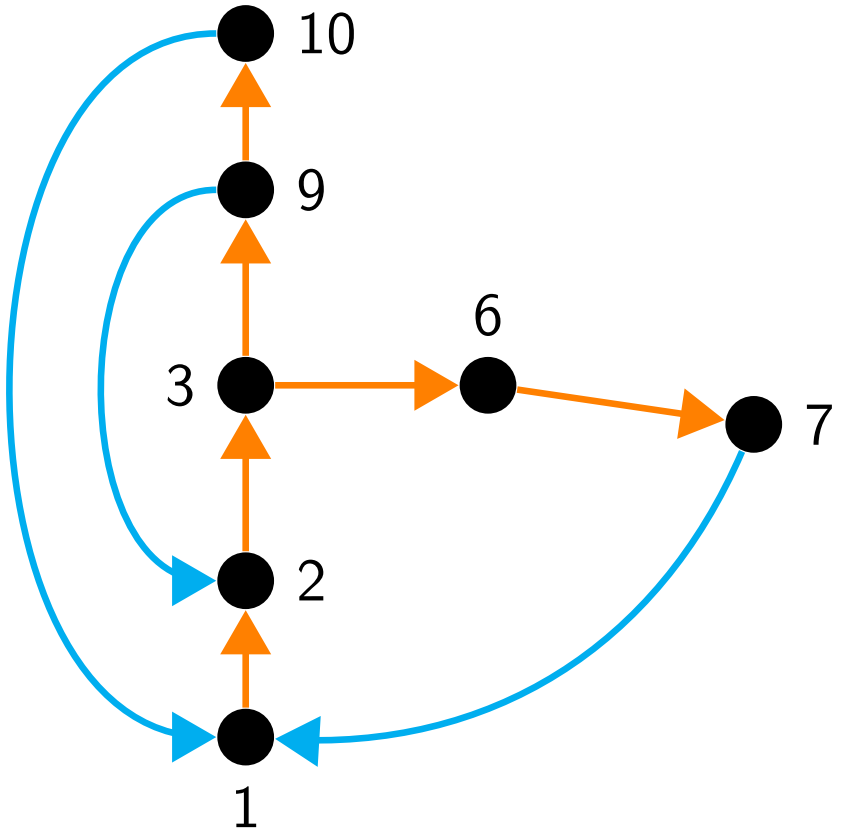
Einbettung



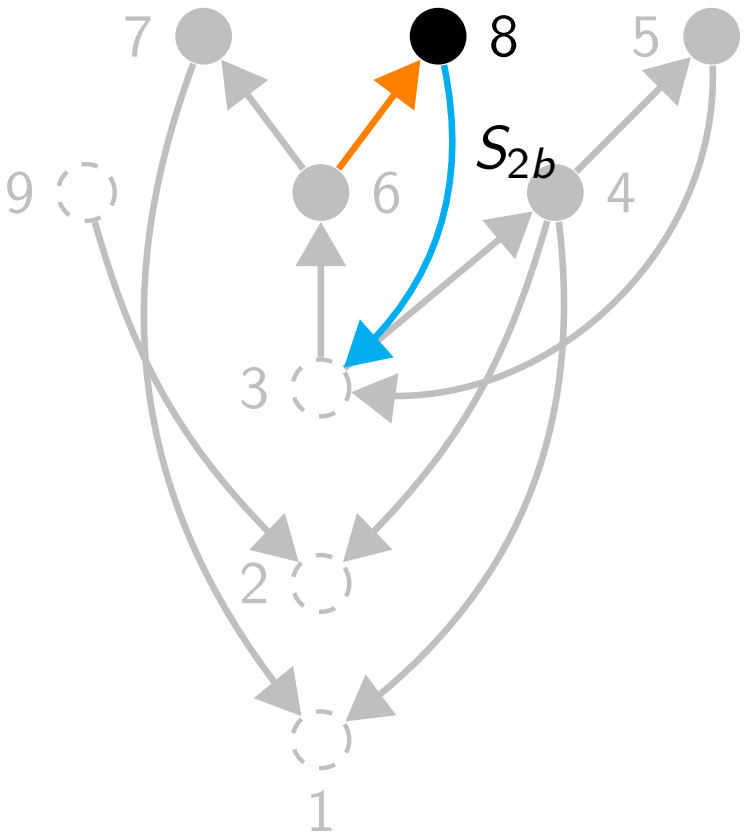


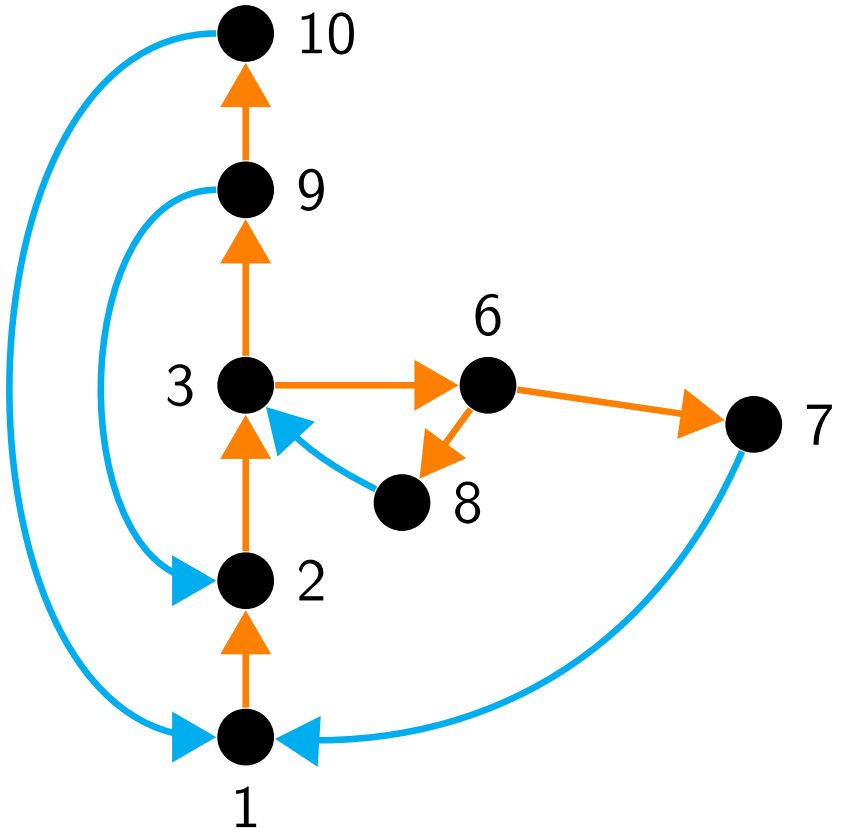
Einbettung



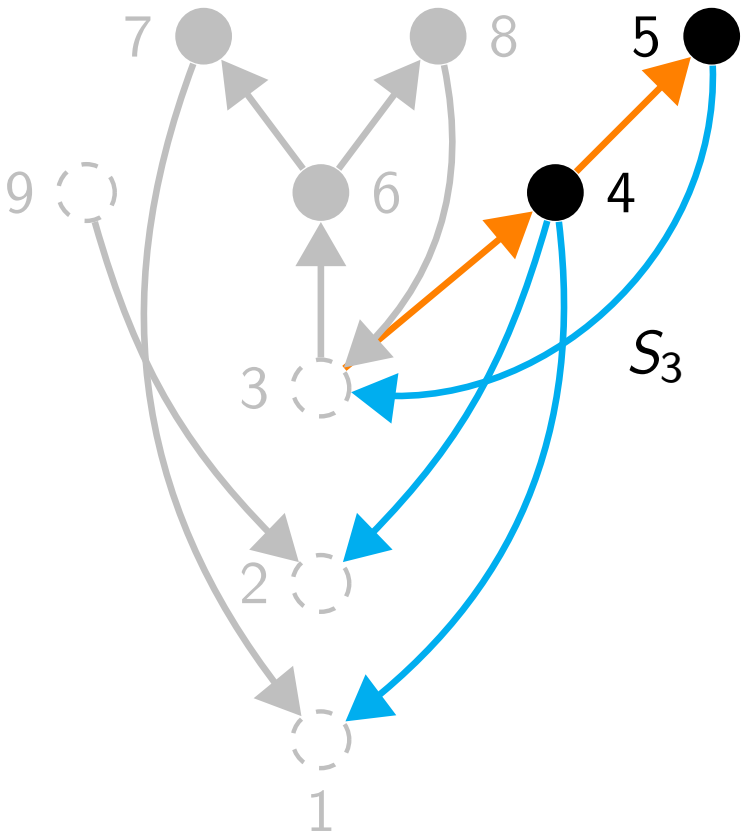


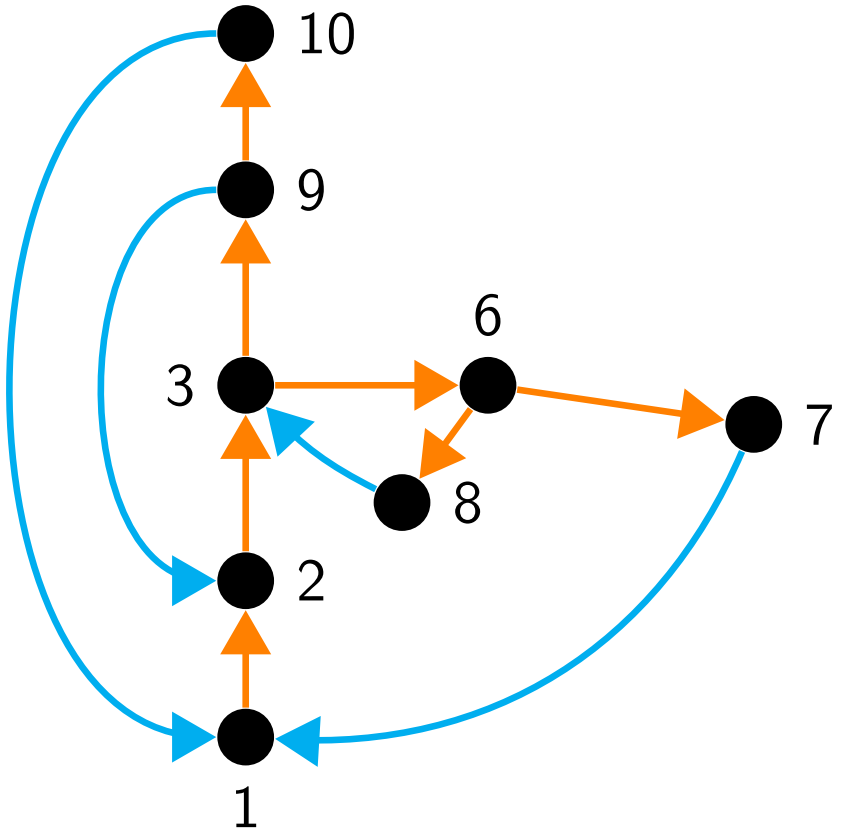
Einbettung



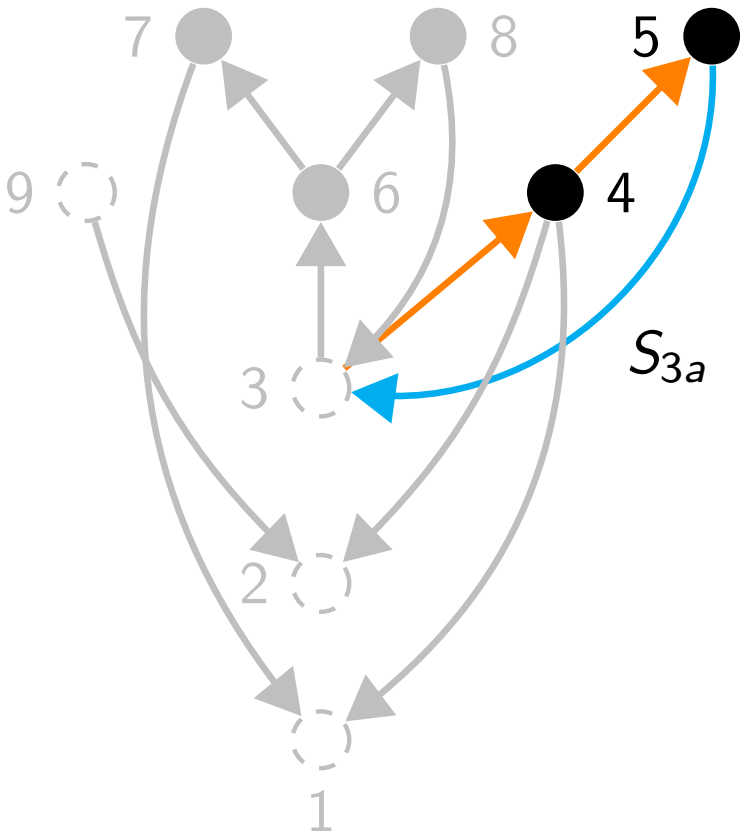


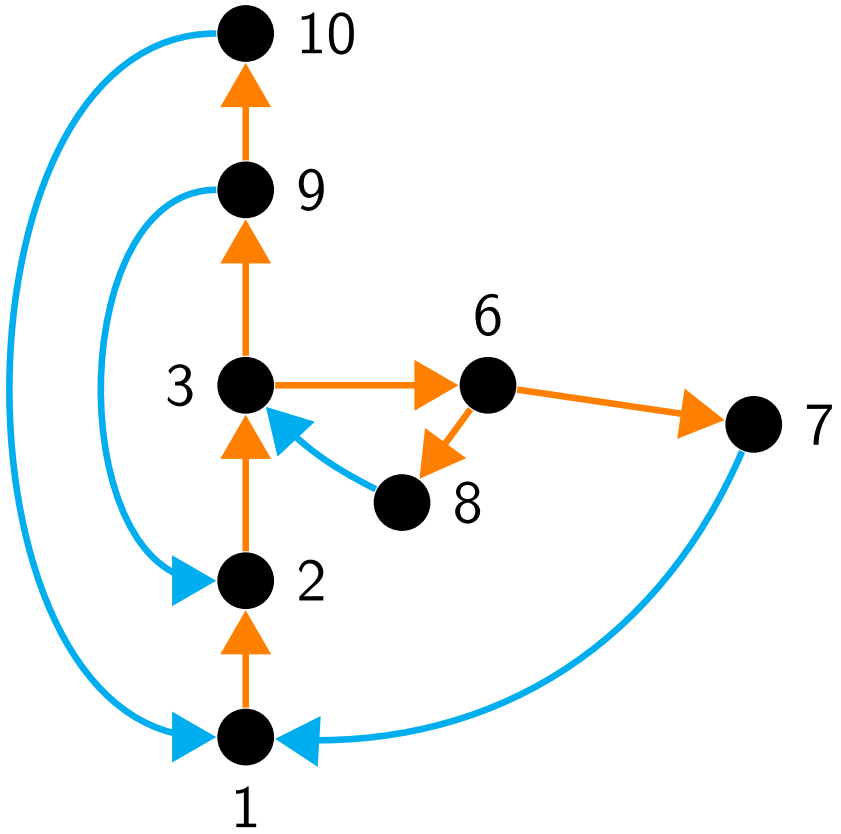
Einbettung



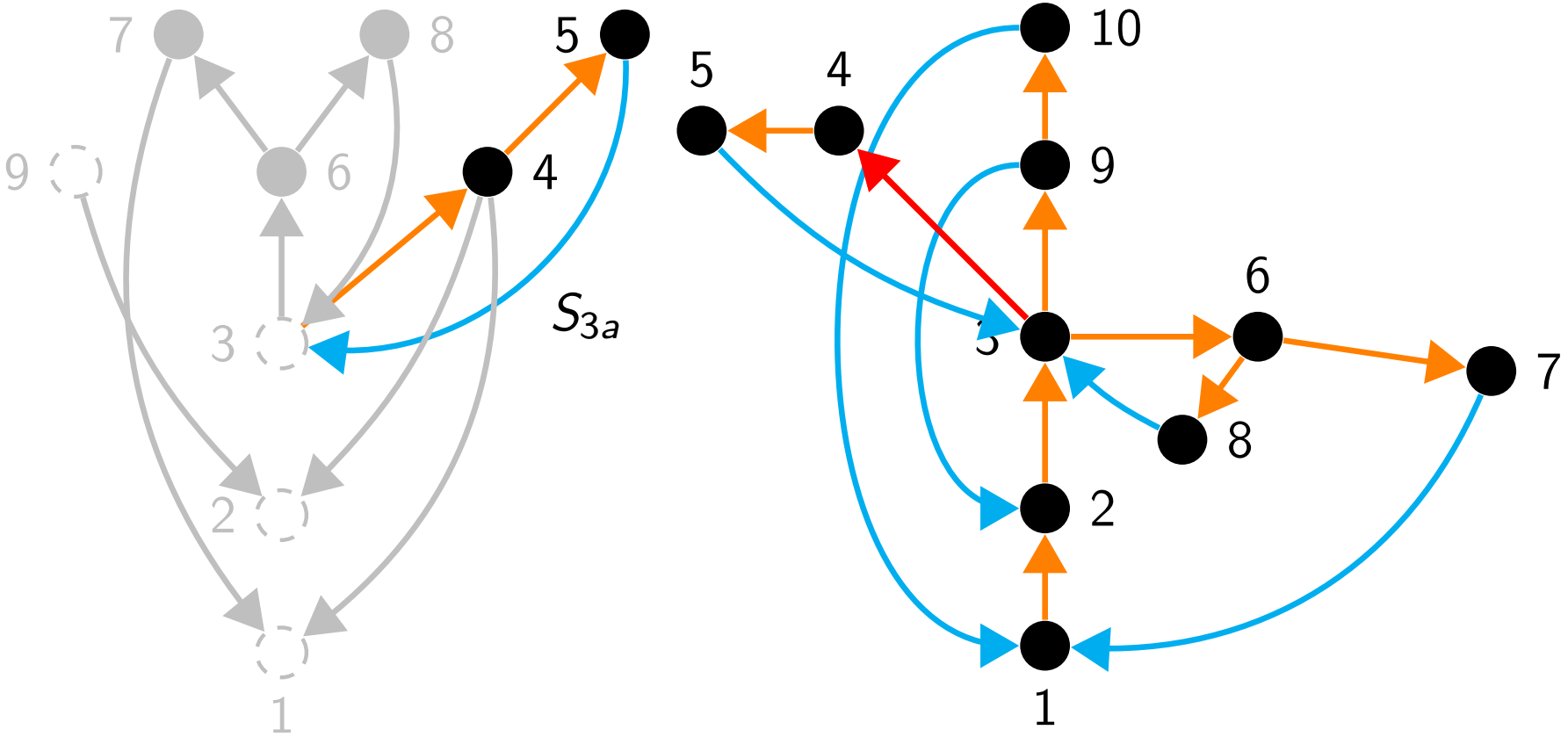


Einbettung

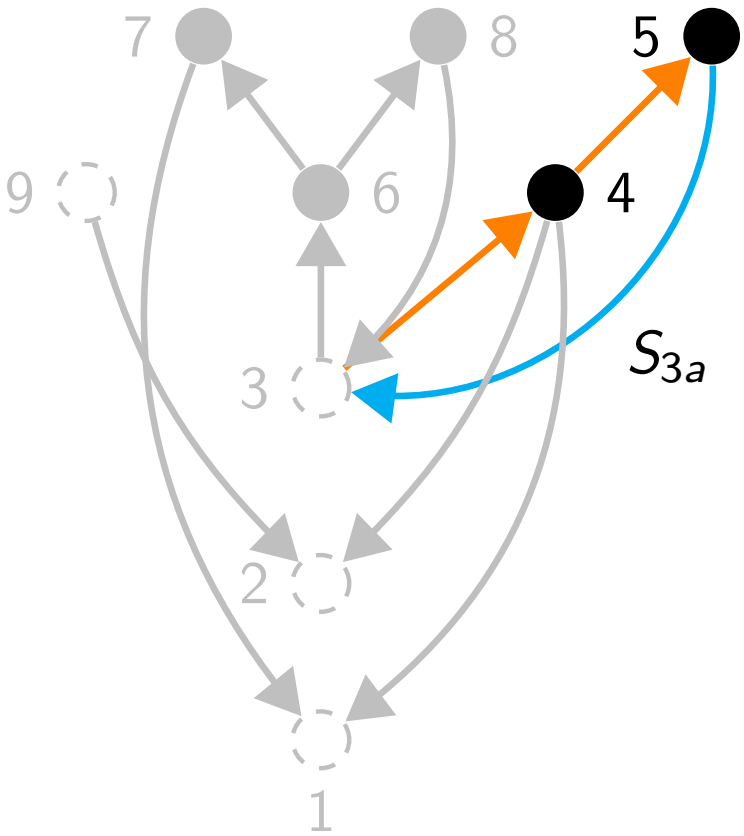


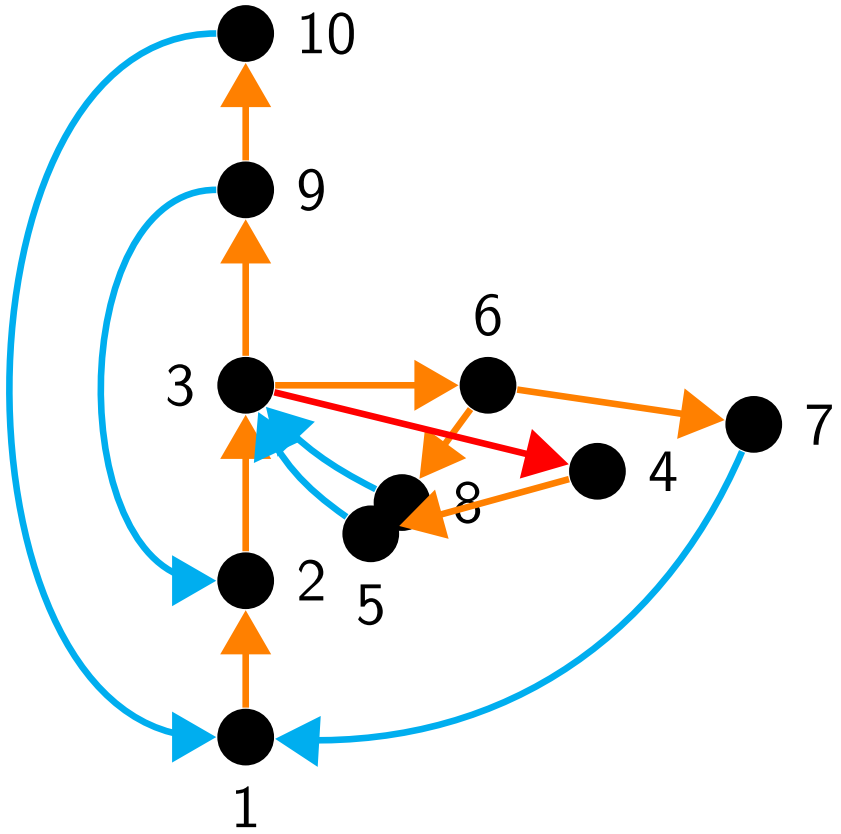


Einbettung

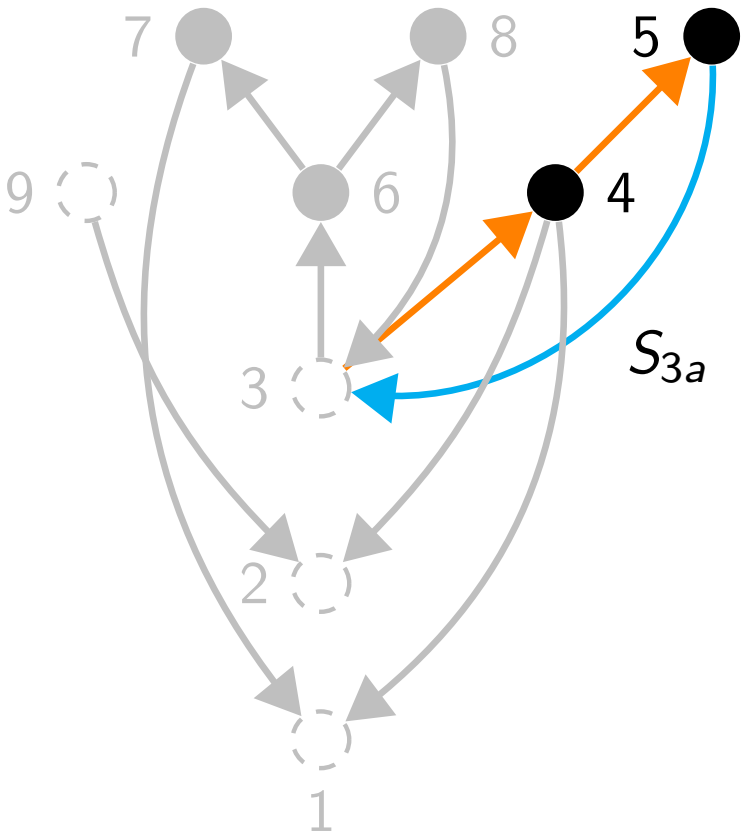


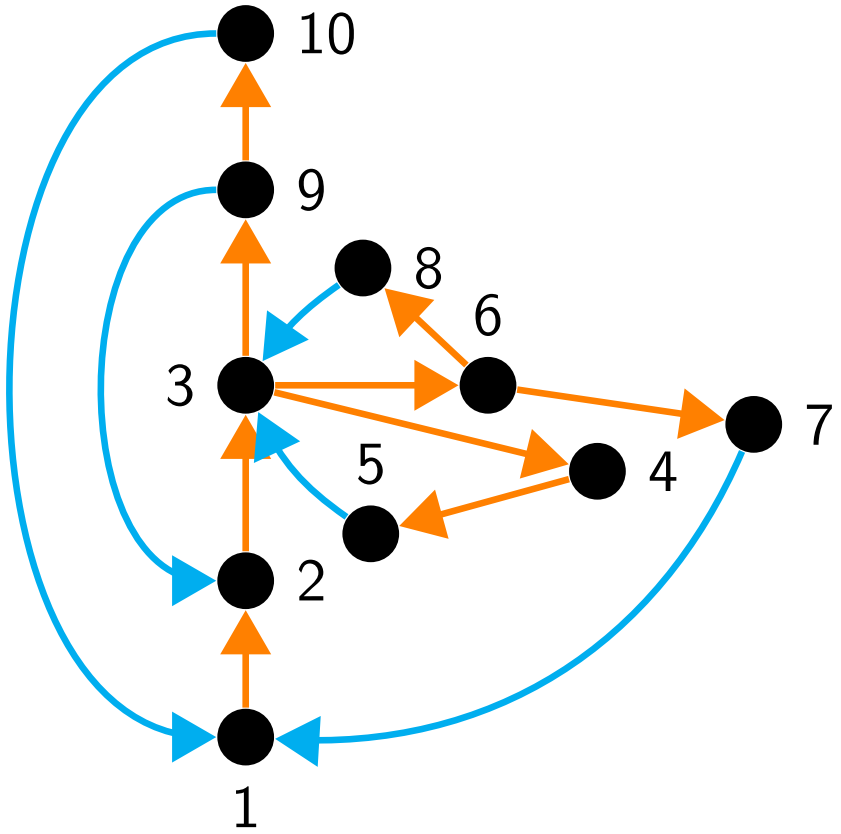
Einbettung



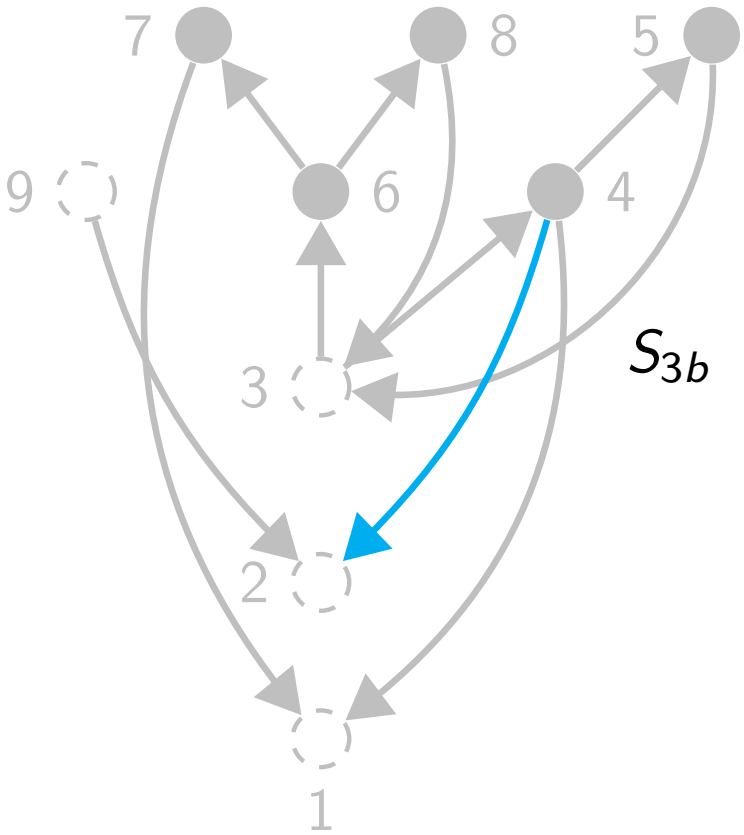


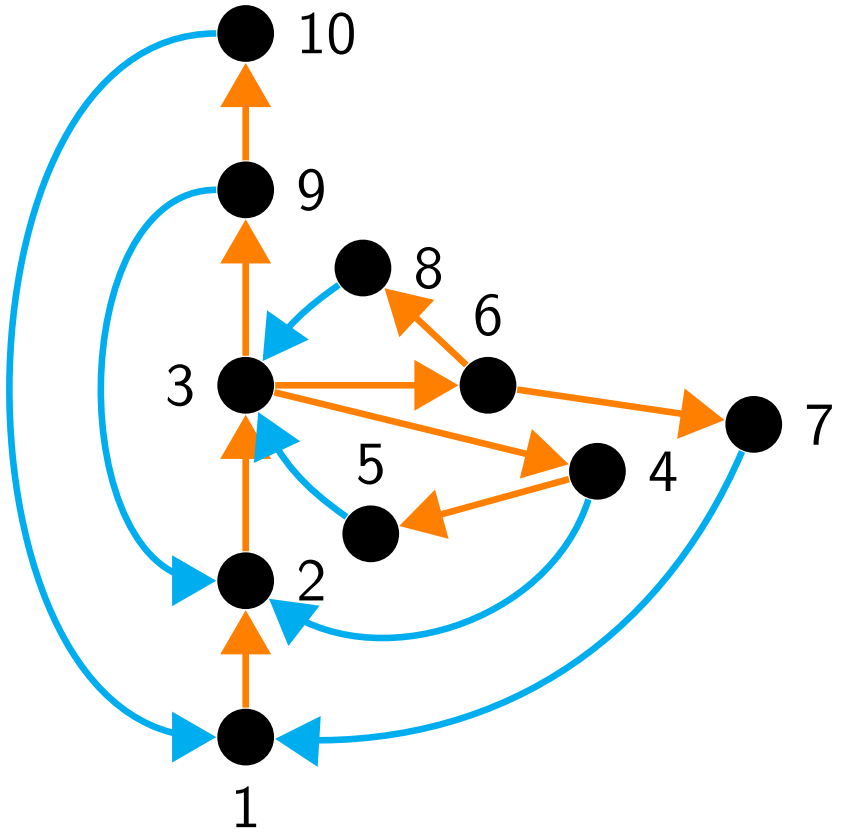
Einbettung



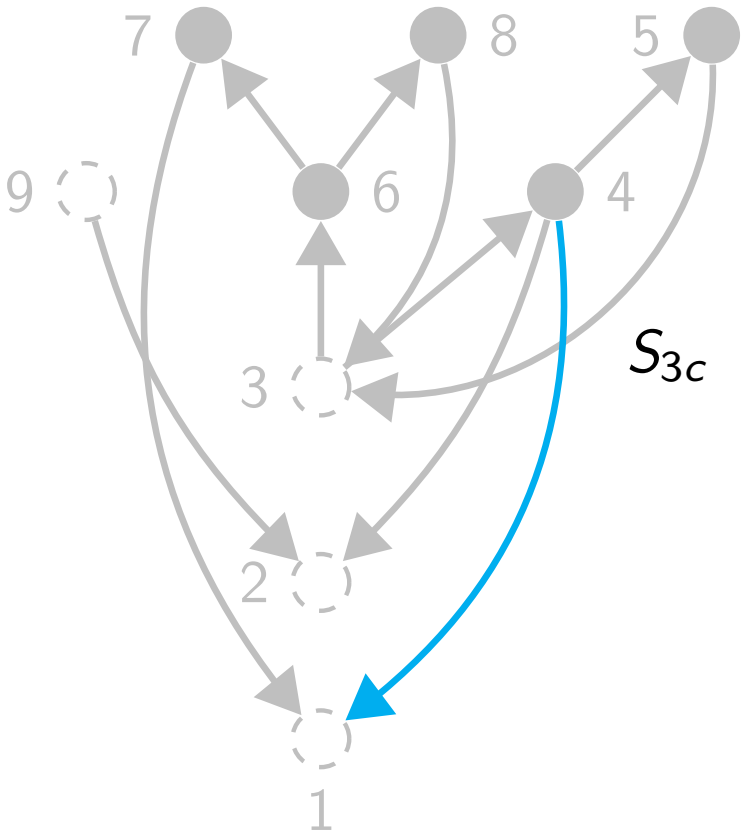


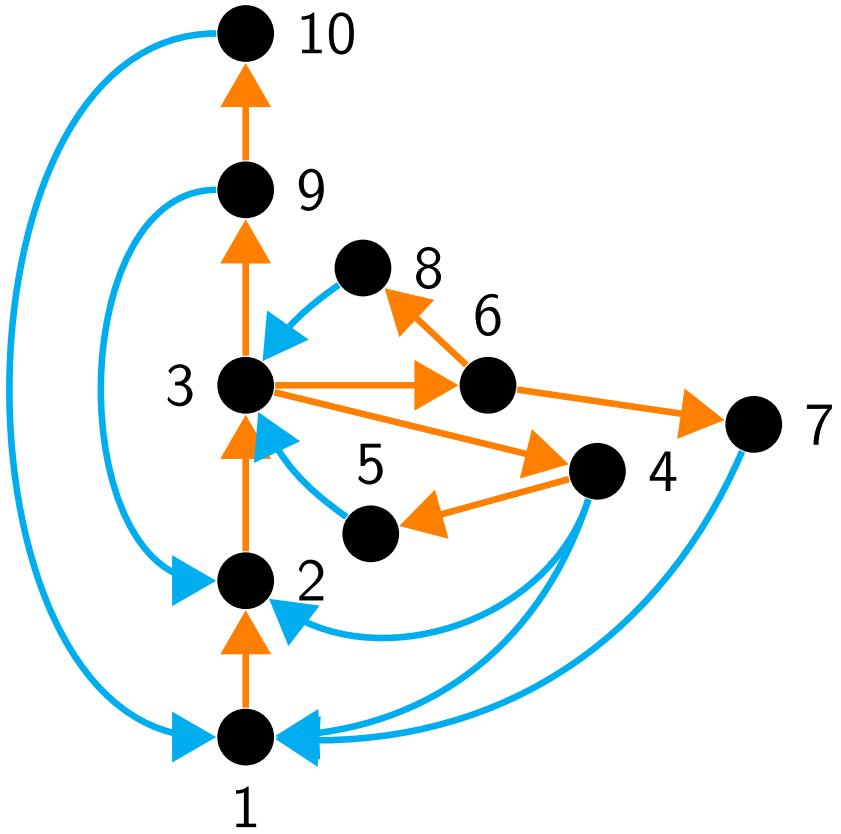
Einbettung





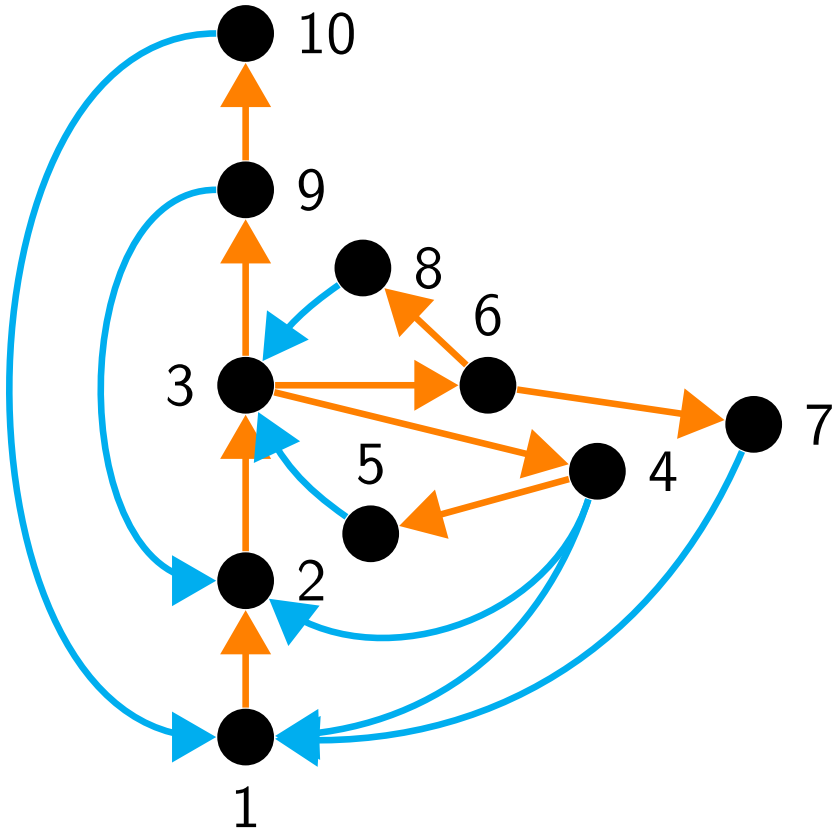
Einbettung





Ergebnis

Einbettung aller Segmente möglich $\Rightarrow$ Graph G ist planar!



Laufzeit

Sei G ein ungerichteter Graph mit V Knoten und E Kanten.

Kanten zählen (Euler's Theorem)	$\mathcal{O}(V+E)$
2-Zusammenhangskomponenten finden	$\mathcal{O}(V+E)$
Umstrukturierung Palm Tree	$\mathcal{O}(V+E)$
Pathfinding	$\mathcal{O}(V+E)$
Embedding	$\mathcal{O}(V+E)$
Planaritätstest nach Hopcroft und Tarjan	$\mathcal{O}(V)$

Implementation

- Anstatt Adjazenzmatrix, Nutzung von Adjazenzlisten

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$A_{v1} = \{2, 3\}$$
$$A_{v2} = \{1, 3\}$$
$$A_{v3} = \{1\}$$

Definition

LOWPTx(v) Werte. Entspricht dem x-ten niedrigsten Knoten der über eine Rückwärtskante von einem Nachfolger von v erreicht werden kann.

- Vor Pathfinding: Sortierung der Adjanzenzlisten nach $\mathrm{LOWPT1}()$ / $\mathrm{LOWPT2}()$ Werten
- Beim Embedding: Nutzung von Stacks L und R und *Blocks*

Diskussion

- Vorgehensweise: Sukzessive Einbettung von einzelnen Pfaden aus dem Graphen, bis planare Einbettung erreicht
- Relativ komplex, sowohl in den konzeptuellen Belangen als auch in Implementierung
- Algorithmus gibt keine planare Einbettung aus, lediglich Charakterisierung.
- Erst 1996 später: Erweiterung durch Mehlhorn und Mutzel (Ausgabe einer planaren Einbettung in $\mathcal{O}(n)$).

Neue Entwicklungen

In den letzten 40 Jahren: Versuche der Vereinfachung des Algorithmus und seiner Implementierung:

Jahr	Algorithmus	Laufzeit
1974	**Hopcroft und Tarjan**	$\mathcal{O}(n)$
1976	Booth und Lueker	$\mathcal{O}(n)$
1985	de Fraysseix, de Mendez und Rosenstiehl	$\mathcal{O}(n)$
1993	Shih und Hsu	$\mathcal{O}(n)$
2004	Boyer und Myrvold	$\mathcal{O}(n)$
2014	Mondshein und Schmitz	$\mathcal{O}(n)$

Derzeit State-of-the-Art: de Fraysseix, de Mendez und Rosenstiehl (FMR) & Boyer und Myrvold

Geschafft!

Vielen Dank für die Aufmerksamkeit!

Fragen, Feedback, Unklarheiten... ?

Quellen

- Hopcroft, J., & Tarjan, R. (1974). Efficient planarity testing. *Journal of the ACM (JACM), 21(4)*, 549-568.
- Mehlhorn, K., & Mutzel, P. (1996). On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm. *Algorithmica, 16(2)*, 233-242.
- Deo, N. (1976). Note on Hopcroft and Tarjan's planarity algorithm. *Journal of the ACM (JACM), 23(1)*, 74-75.
- Tollis, I. G. (2003). Lecture Notes on Planarity Testing And Construction Of Planar Embedding.
- Patrignani, M. (2013). Planarity Testing and Embedding.