Übersicht

Starke Zusammenhangskomponenten

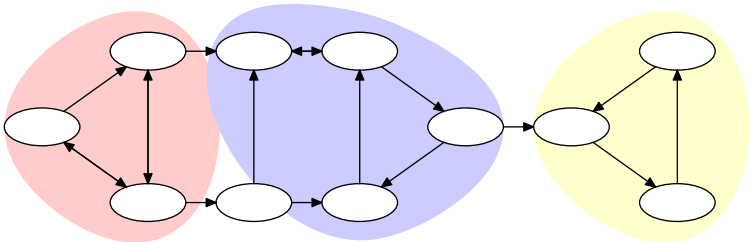
Definition

Es sei $G = (V, E)$ ein gerichteter Graph. Ein Pfad der Länge k von u_1 nach u_{k+1} ist eine Folge $(u_1, u_2), (u_2, u_3), \ldots, (u_k, u_{k+1})$ von Kanten aus E wobei $u_1, \ldots, u_{k+1}$ paarweise verschieden sind.

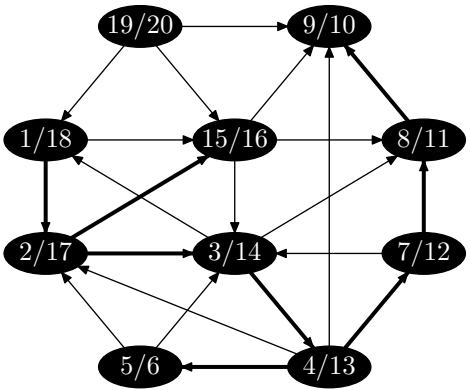
Eine Menge $C \subseteq V$ ist eine starke Zusammenhangskomponente, wenn es Pfade zwischen allen Paaren von Knoten in C gibt und es keine echte Obermenge von C mit dieser Eigenschaft gibt.

Die starken Komponenten sind eine Verfeinerung der Zusammenhangskomponenten des entsprechenden ungerichteten Graphen.

Starke Komponenten – Beispiel



Es gibt genau vier starke Komponenten.

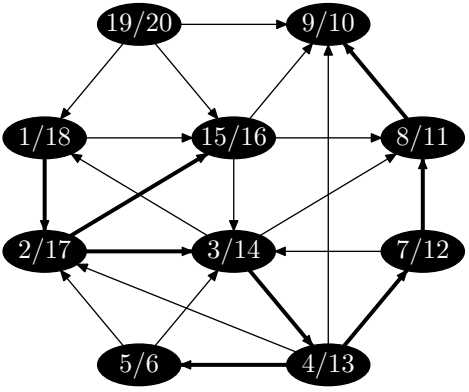


Lemma

Der Knoten mit der größten finish time nach einer DFS ist in einer Quellenkomponente.

Beweis.

Er ist Wurzel eines DFS-Baums T. Wenn es einen anderen DFS-Baum gäbe, von dem eine Kante nach T führte, dann müßte dieser danach besucht werden. □



Quellenkomponente: Starke Komponente, in die keine Kante hineinführt

Senkenkomponente: Starke Komponente, aus der keine Kante herausführt

G_r: Wie G, aber Richtung der Kanten umgekehrt

Korollar

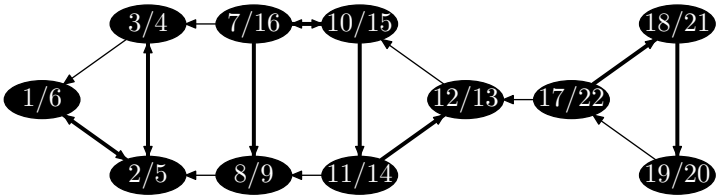
Der Knoten mit der größten finish time nach einer DFS in G_r ist in einer Senkenkomponente von G.

Die starken Komponenten von G und G_r sind identisch.

Starke Komponenten

Lemma

Gegeben sei ein gerichteter Graph G mit einem DFS-Wald.
Der DFS-Wald bleibt samt discover und finish times ein möglicher DFS-Wald, wenn die Quellenkomponente entfernt wird, die den Knoten mit maximaler finish time enthält.

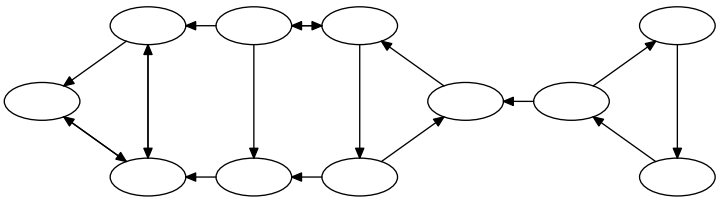


Beweis.

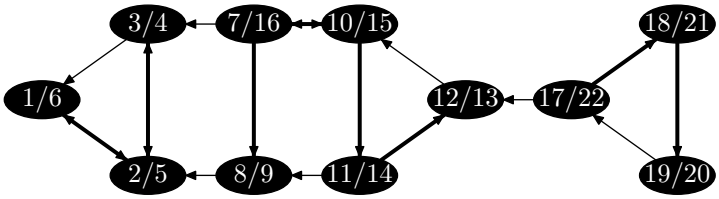
Der Baum spannt die ganze starke Komponente auf. Er wurde als letzter besucht. □

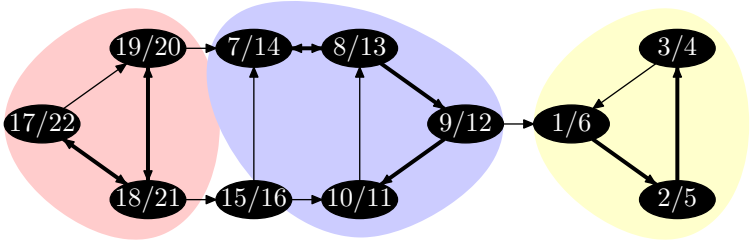
Starke Komponenten – Algorithmus von Kosaraju

1. Bilde G_r (Kanten umdrehen)
2. Führe Tiefensuche auf G_r aus
3. Ordne die Knoten nach absteigender finish time $f(v)$
4. Führe in dieser Reihenfolge Tiefensuche auf G aus
5. Jeder Baum im DFS-Wald spannt eine starke Komponente auf



Starke Komponenten – Algorithmus von Kosaraju



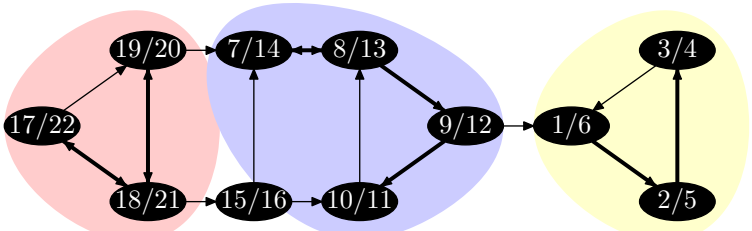


Theorem

Der Algorithmus von Kosaraju berechnet die starken Komponenten eines gerichteten Graphen.

Beweis.

Die DFS auf G beginnt in einem Knoten einer Senkenkomponente von G. Dabei entsteht ein DFS-Baum, der diese starke Komponente aufspannt.
Die DFS fährt dann wieder in einer Senkenkomponente des Restgraphen fort und so weiter. □



Übersicht

Topologisches Sortieren

Ein reflexiv-transitive Hüllen eines DAG entspricht einer Halbordnung.

Definition

Es sei M eine Menge. Wir sagen $\leq \subseteq M \times M$ ist eine Halbordnung auf M, wenn

1. Für alle $x \in M$ gilt $x \leq x$
2. Für alle $x, y \in M$ gilt $x \leq y \wedge y \leq x \Rightarrow x = y$
3. Für alle $x, y, z \in M$ gilt $x \leq y \wedge y \leq z \Rightarrow x \leq z$

Topologisches Sortieren:

Bette eine Halbordnung in eine Ordnung ein.

Ordne die Knoten eines DAG so, daß keine Kante von einem größeren zu einem kleineren Knoten führt.

Topologisches Sortieren – Anwendungen

Topologisches Sortieren kann viele Probleme lösen.

- In welcher Reihenfolge sollen Vorlesungen gehört werden?
- Wie baut man ein kompliziertes Gerät?
- Entwurf von Klassenhierarchien.
- Task scheduling.
- Tabellenkalkulation.
- Optimierung beim Compilieren.
- ...

Frage: Kann jede Halbordnung topologisch sortiert werden?

Topologisches Sortieren

Theorem

Jede endliche Halbordnung kann in eine (totale) Ordnung eingebettet werden.

Beweis.

Sei $\leq \subseteq M \times M$ eine endliche Halbordnung auf M.
Da M endlich ist, dann gibt es ein $x \in M$, so daß es kein $y \in M$ mit $y \neq x$ und $y \leq x$ gibt.
Wir können die Ordnung mit x als kleinstem Element beginnen und dann eine Ordnung von $M \setminus \{x\}$ anfügen.
Durch Induktion sieht man, daß dies eine (totale) Ordnung auf M ist. □

Frage: Was ist mit unendlichen Mengen?

Topologisches Sortieren

Theorem

G sei ein DAG. Wenn wir die Knoten von G nach den finish times einer DFS umgekehrt anordnen, sind sie topologisch sortiert.

Beweis.

Es seien u und v Knoten von G mit $f(u) < f(v)$
Daher kommt u nach v in der konstruierten Ordnung.

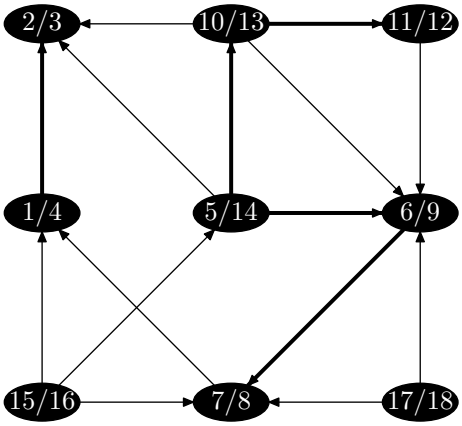
Wir zeigen, daß es keine Kante von u nach v gibt:

Falls $d(v) < d(u)$ müßte es eine Rückwärtskante sein, die es nicht gibt (DAG).

Also $d(v) > f(u)$ und v blieb weiß bis zur finish time von u.

Nur möglich, wenn keine Kante von u nach v. □

Topologisches Sortieren – Beispiel



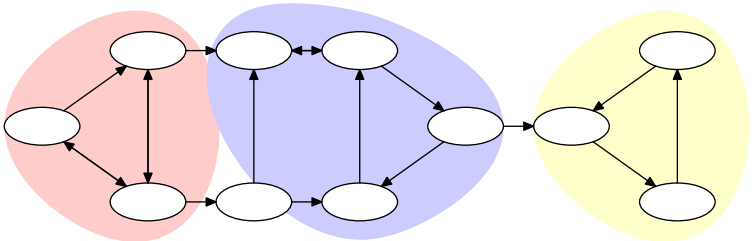
Topologisches Sortieren in $O(|V| + |E|)$ Schritten.

Übersicht

s-t Connectivity

Gegeben: Knoten s und t in gerichtetem Graph

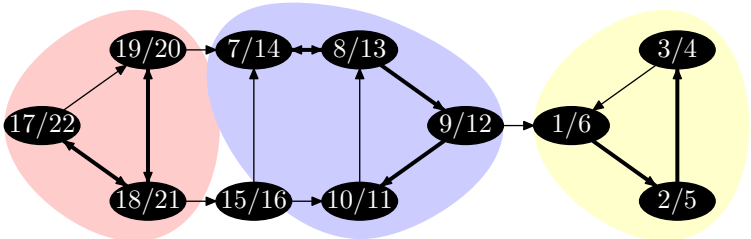
Frage: Ist t von s erreichbar (durch einen gerichteten Pfad)?



s-t Connectivity

Führe eine DFS aus, starte bei s.

Pfad von s nach $t \iff f(t) < f(s)$.

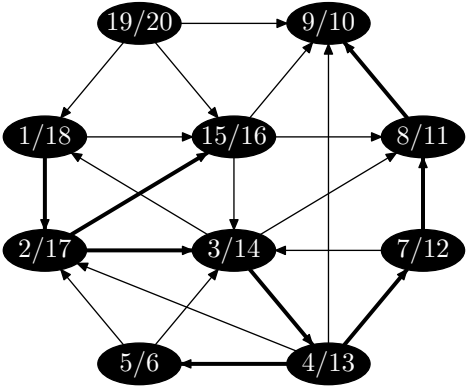


Beweis: Wenn s schwarz wird, sind alle von s erreichbaren Knoten schwarz.

s-t Connectivity

Theorem

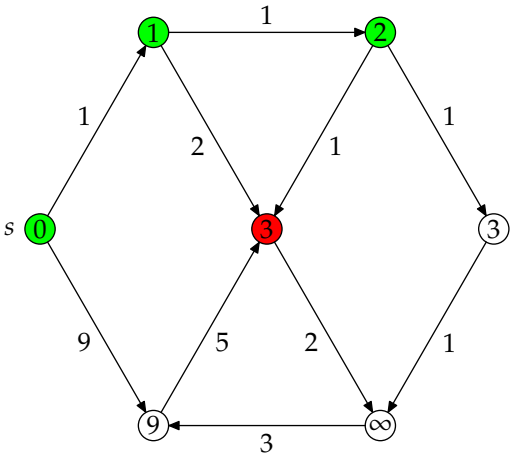
Gegeben sei ein gerichteter Graph $G = (V, E)$ und $s \in V$.
Wir können in linearer Zeit alle von s erreichbaren Knoten finden.



Single Source Shortest Paths

Gegeben ein gerichteter Graph $G = (V, E)$ mit nicht-negativen Kantengewichten $\mathit{length} : E \to \mathbf{Q}$ und ein Knoten $s \in V$, finde die kürzesten Wege von s zu allen Knoten.

- Wir lösen das Problem mit dynamischer Programmierung.
- Menge F von Knoten, deren Abstand bekannt ist.
- Anfangs ist $F = \{s\}$.
- F wird in jeder Iteration größer.
- Invariante: Kein Knoten $v \notin F$ hat kleineren Abstand zu s als jeder Knoten in F.



- Grüne Knoten: F
- Roter Knoten aktiv, relaxiert seine Nachbarn
- Weiße Knoten enthalten Abstand zu s über grüne Knoten.

Korrektheit

Lemma

- *Jeder grüne Knoten enthält den Abstand von s.*
- *Jeder weiße Knoten enthält Abstand von s über grüne Knoten.*

Beweis.

Sei v einer weißer Knoten, dessen Beschriftung minimal ist.
Betrachte einen kürzesten Pfad von s nach v und den ersten weißen Knoten u auf diesem Pfad.
Es gilt $u = v$, da der Abstand von s zu u mindestens so groß ist wie der Abstand von s zu v.
Wenn ein Knoten grün wird, garantiert die Relaxation, daß die zweite Bedingung weiter gilt. □