Karlsruher Institut für Technologie

Übung 4 – Algorithmen II

Daniel Seemaier, Tobias Heuer – *daniel.seemaier@kit.edu, tobias.heuer@kit.edu*

`http://algo2.iti.kit.edu/AlgorithmenII_WS20.php`

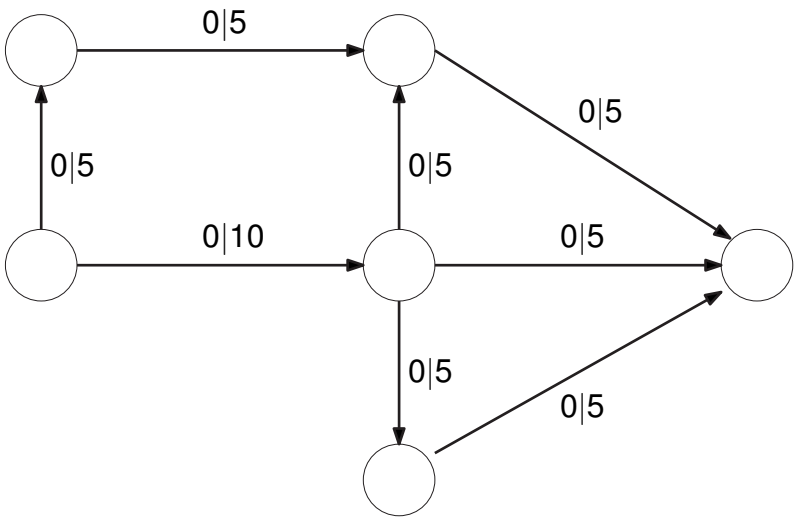
Institut für Theoretische Informatik - Algorithmik II

Themenübersicht

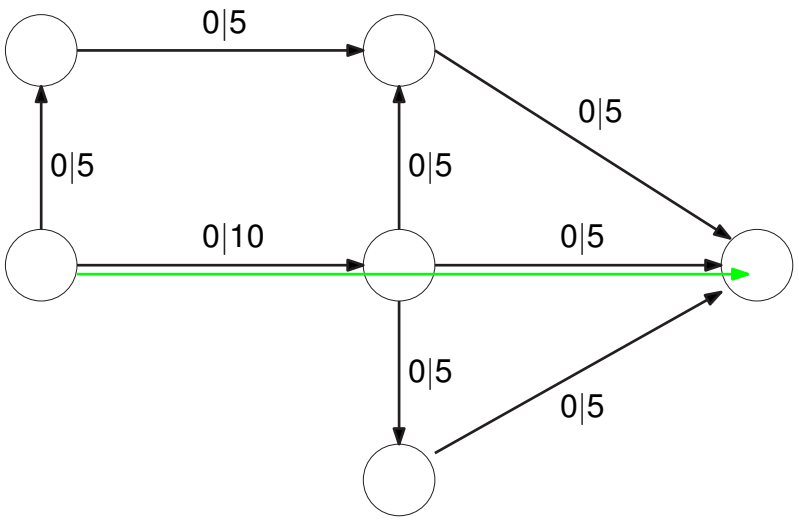
- Ford Fulkerson
  - Erhöhende Pfade
  - Residualgraph
- Max-Flow und Min-Cut
- Dinitz Algorithmus
  - Distanz Label
  - Layergraph
  - Blocking Flow

Ford Fulkerson

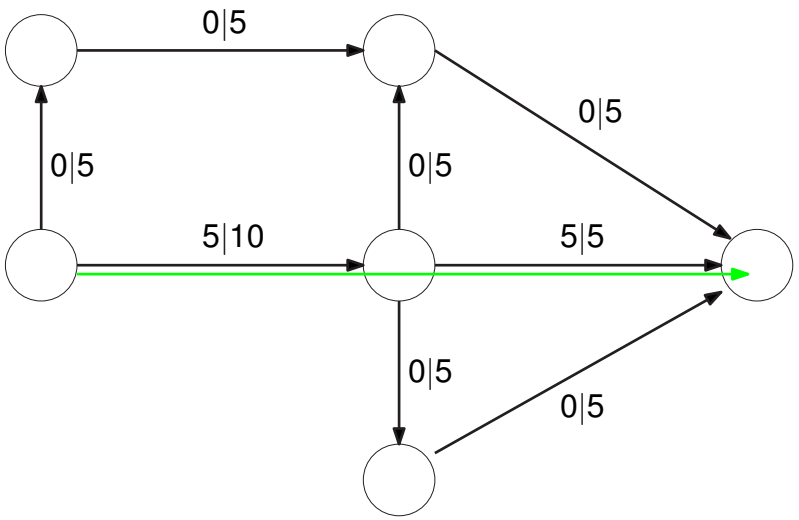
Flüsse und Ford Fulkerson



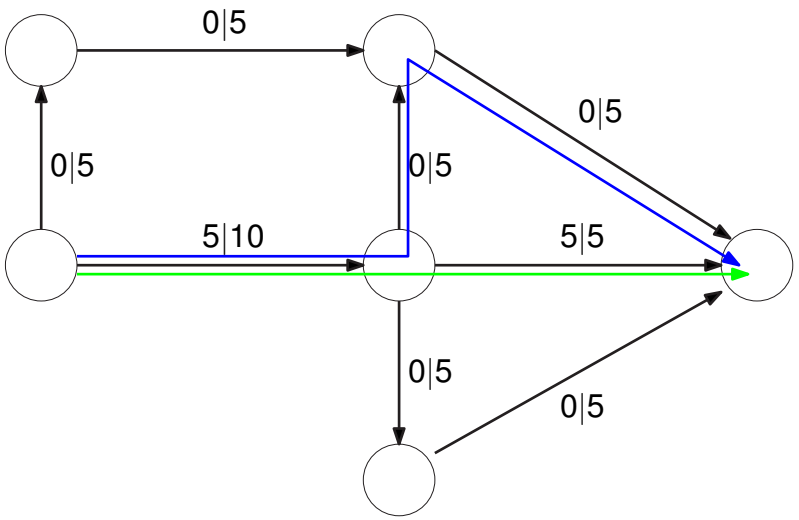
Flüsse und Ford Fulkerson



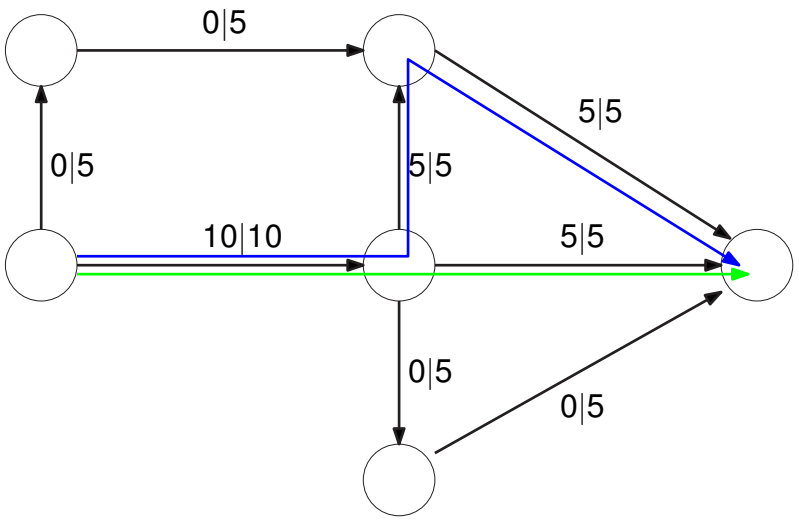
Flüsse und Ford Fulkerson



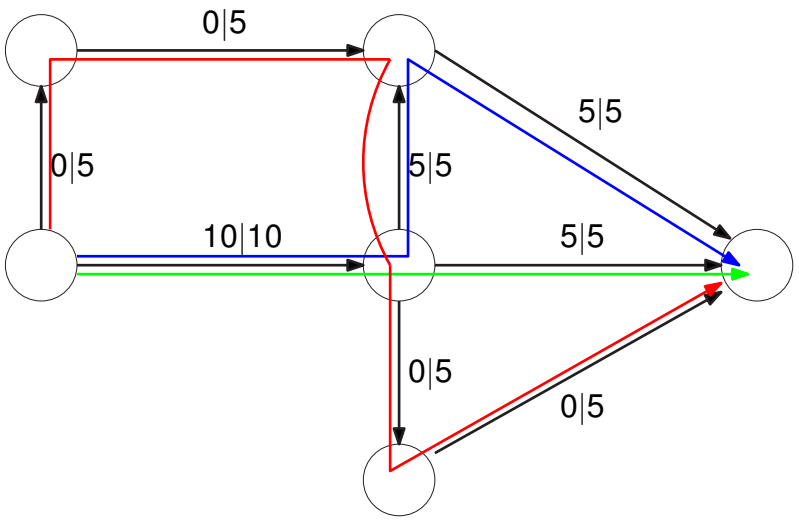
Flüsse und Ford Fulkerson



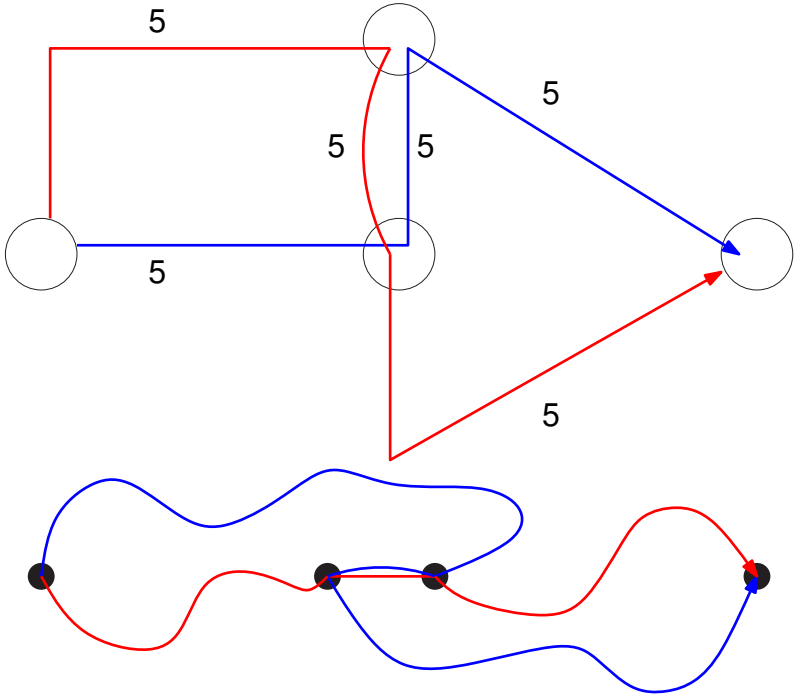
Flüsse und Ford Fulkerson



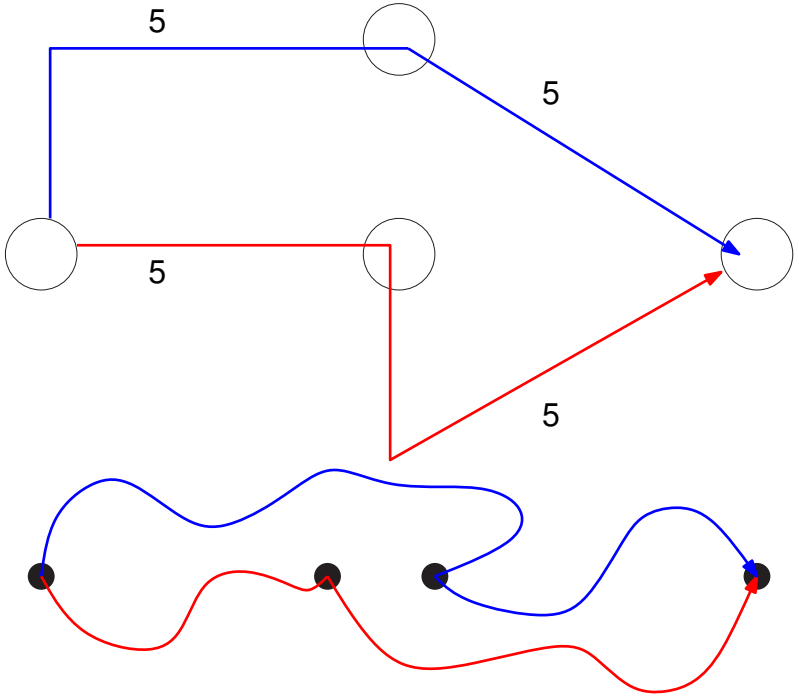
Flüsse und Ford Fulkerson



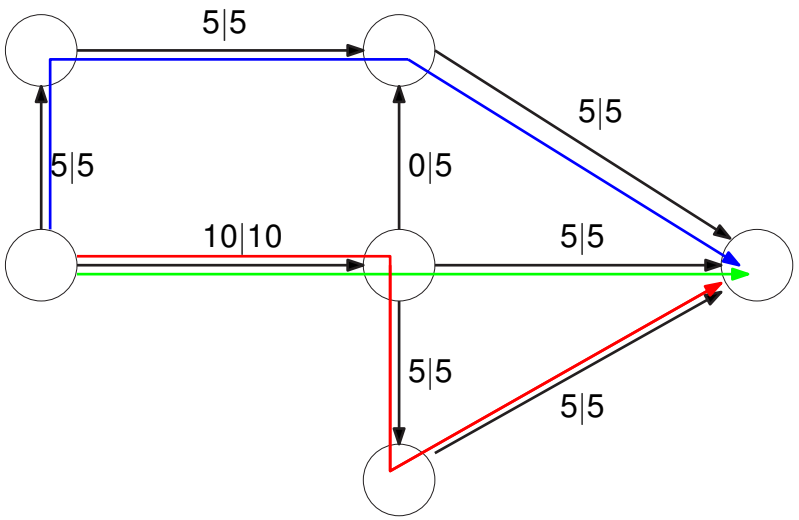
Flüsse und Ford Fulkerson



Flüsse und Ford Fulkerson



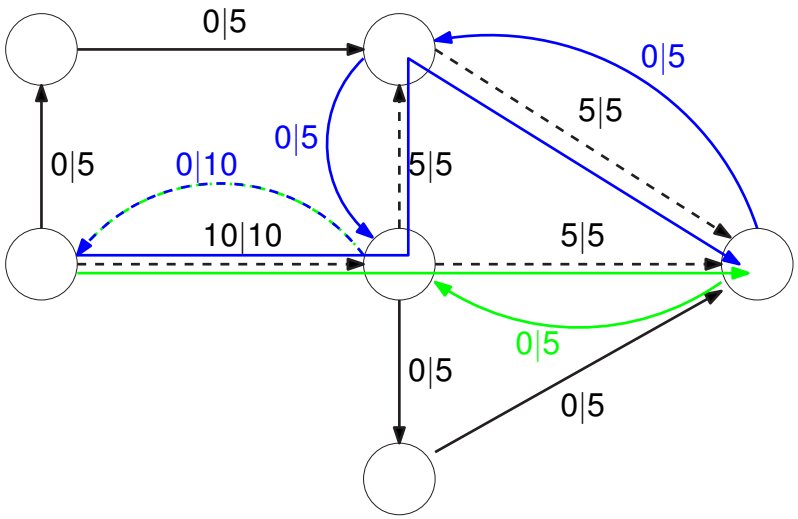
Flüsse und Ford Fulkerson



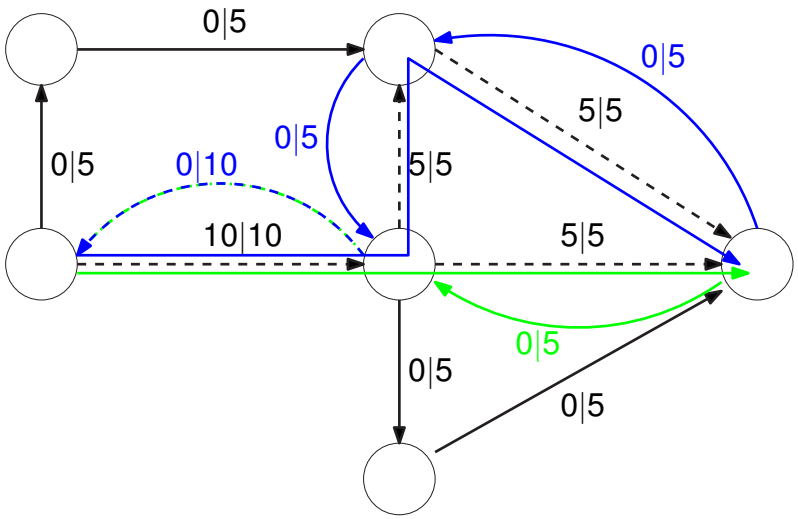
Residualgraph

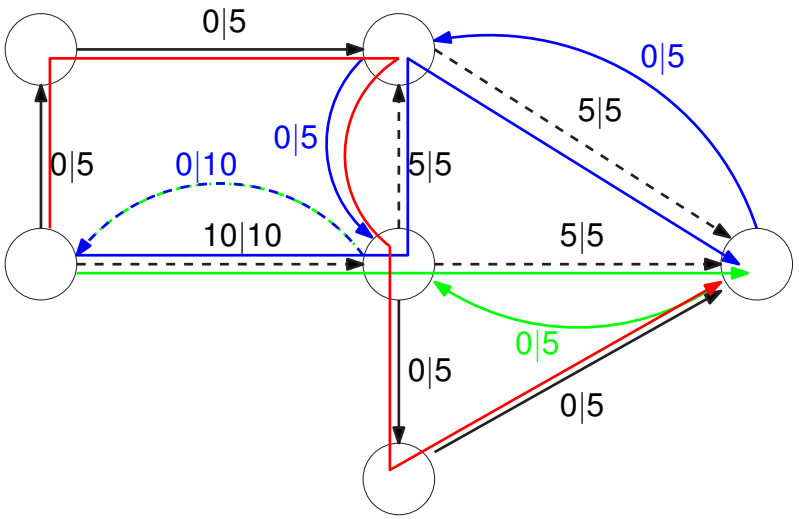
- Verwalten von Restkapazitäten
- Modellierung und Erkennung von “Gegenflüssen”
- $c^f(e) = c(e) - f(e)$: Restkapazität
  Hier: Fluss $f(e)$ und Gesamtkapazität $c(e)$
- $c^f(e^{\text{rev}}) = f(e)$: Fluss über Kante e
  Hier: Restkapazität und Gesamtkapazität von e
- Keine 0-Gewicht Kanten
- Flüsse über Kanten e und e^{rev} erlaubt
  - Fluss über Kante $\rightarrow$ Update beider Kanten

Flüsse und Ford Fulkerson

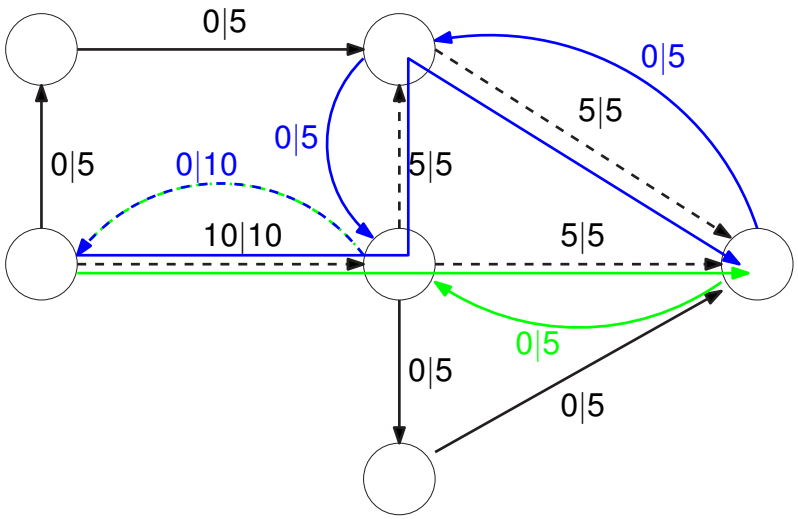


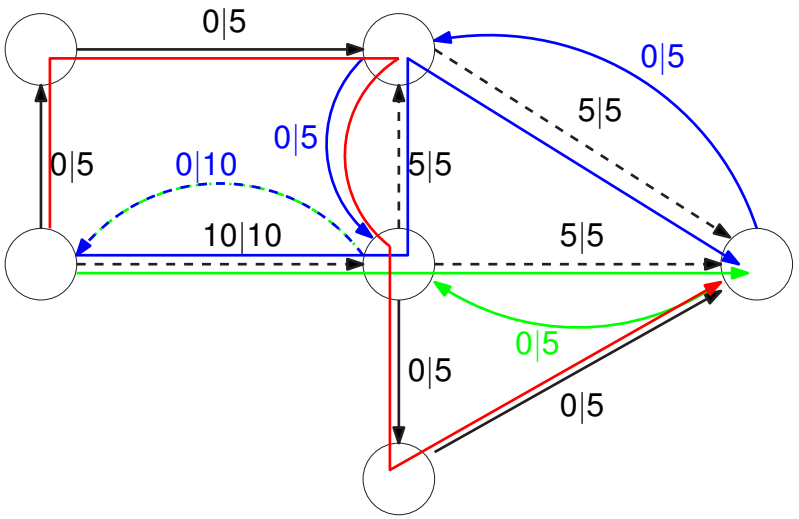
Flüsse und Ford Fulkerson

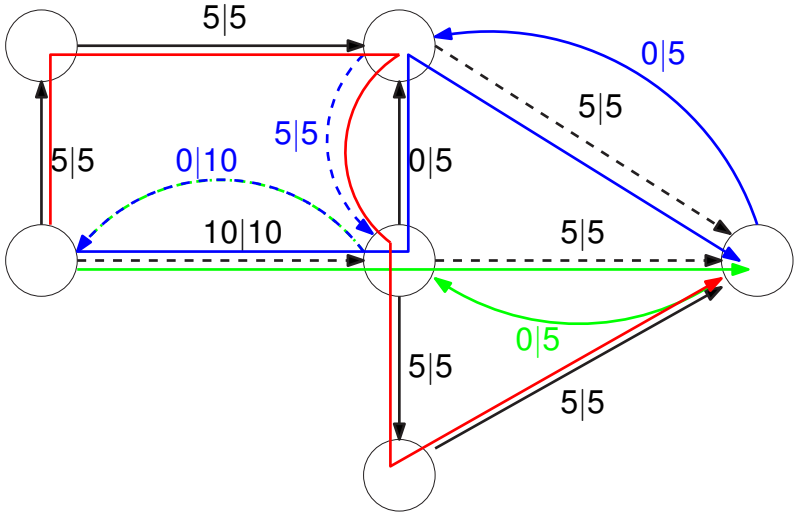




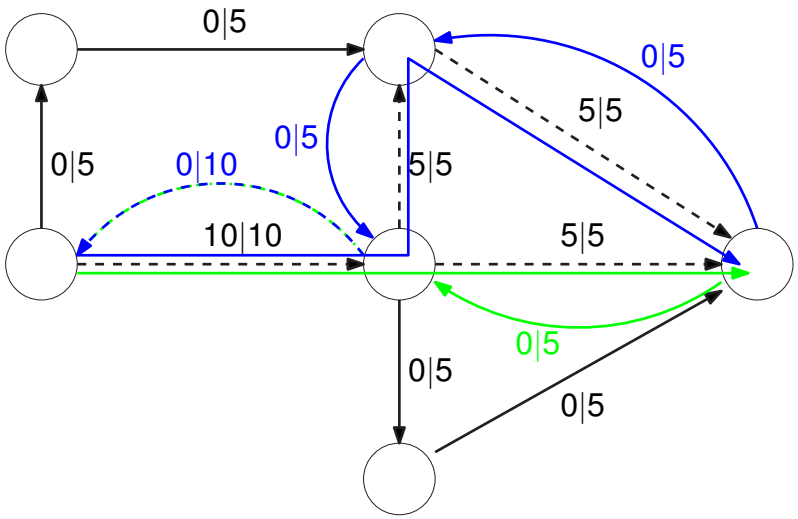
Flüsse und Ford Fulkerson

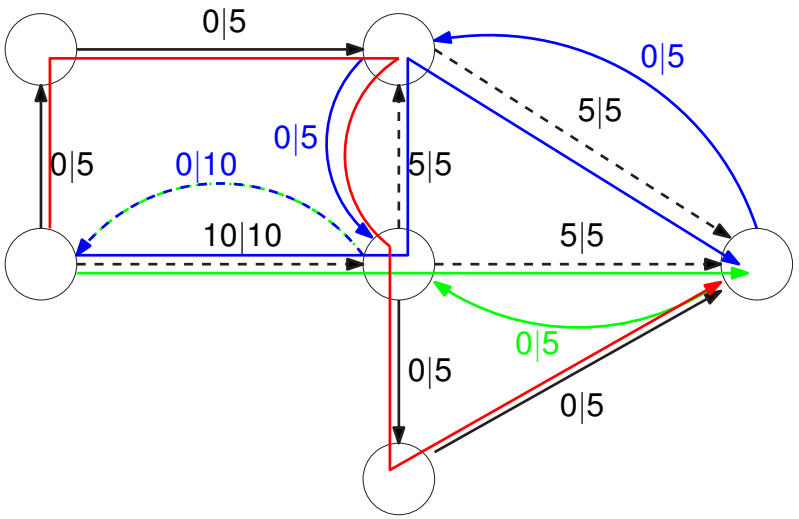


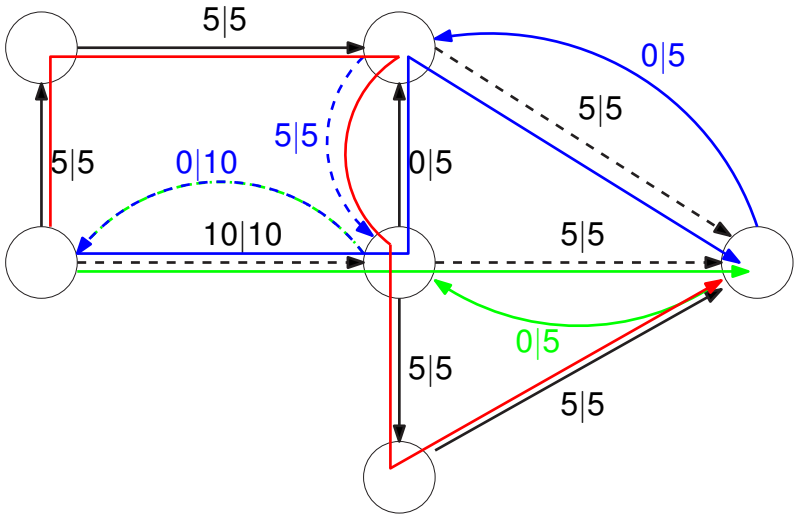


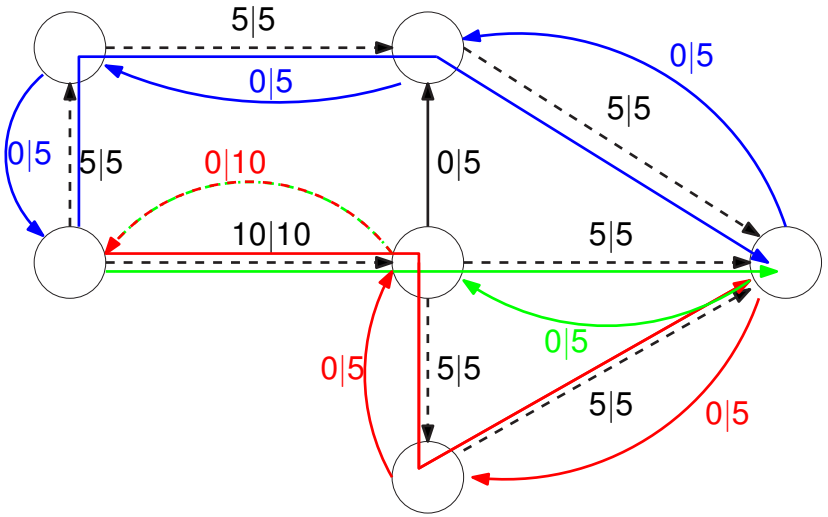


Flüsse und Ford Fulkerson



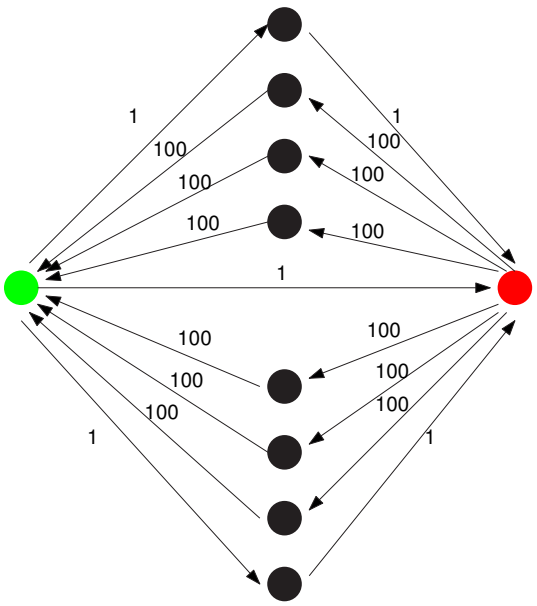






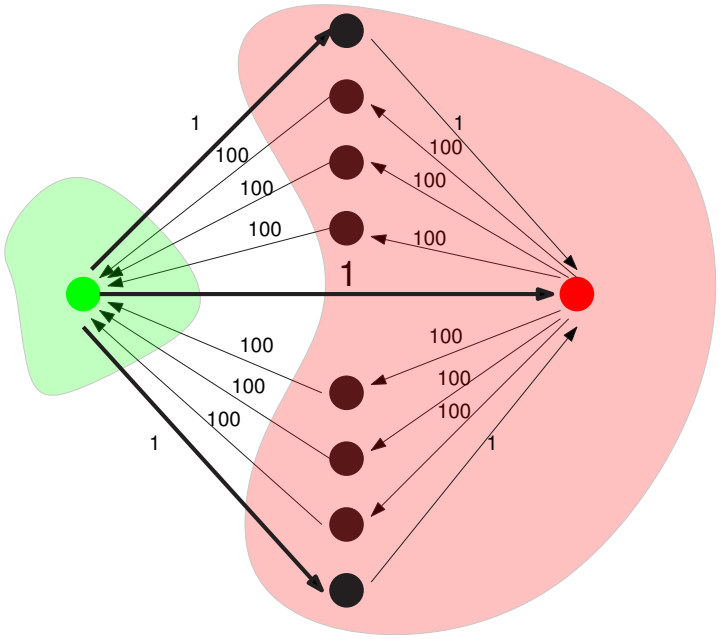
Max Flow - Min Cut

Max Flow - Min Cut



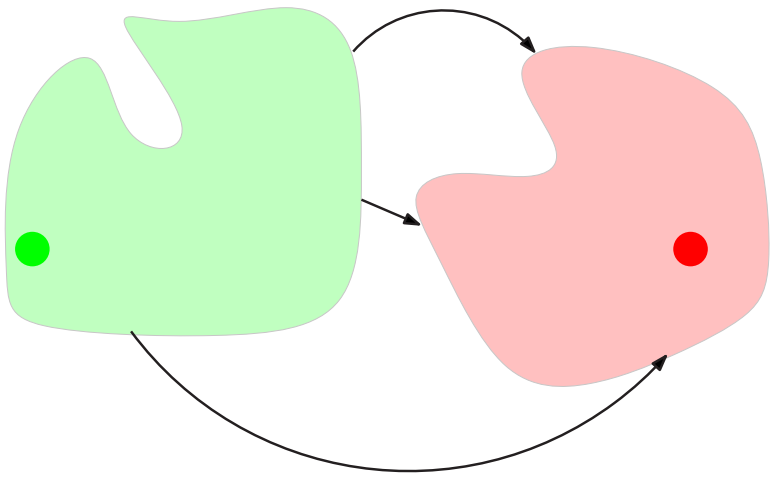
Max Flow - Min Cut

- S-T-Schnitte betrachten nur Kanten $S \to T$
- Kanten $T \to S$ werden nicht berücksichtigt
- s und t werden durch alle möglichen S-T-Schnitte getrennt
- Flow muss von s nach t, auch durch alle möglichen S-T-Schnitte $\to$ Max Flow = Min S-T-Cut



Max Flow - Min Cut

- S-T-Schnitte betrachten nur Kanten $S \to T$
- Kanten $T \to S$ werden nicht berücksichtigt
- s und t werden durch alle möglichen S-T-Schnitte getrennt
- Flow muss von s nach t, auch durch alle möglichen S-T-Schnitte $\to$ Max Flow = Min S-T-Cut



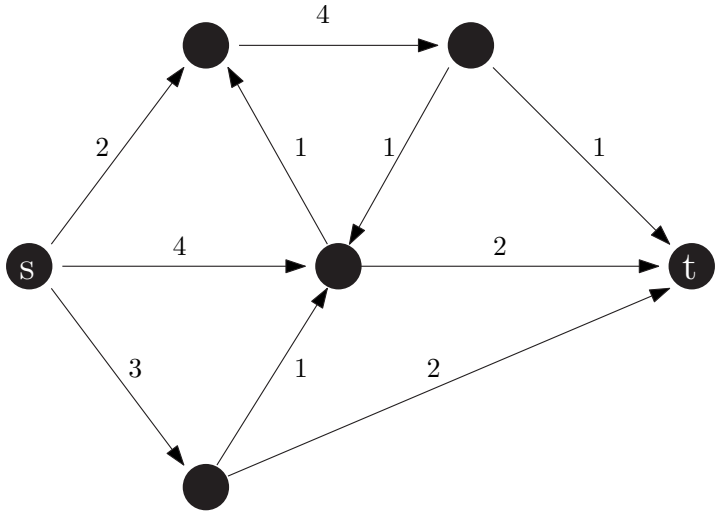
Dinitz Algorithmus

Dinitz

Distanz Label

- Geben Distanz im Residualgraphen (hop-based) zur Senke *t* an
- Rückwärtsgerichtete Breitensuche
  - Start bei Knoten *t*
  - Layer: Knoten mit gleicher Distanz zu *s* im BFS-Baum
  - Knoten in einem Layer: gleiches Label
- *Layered graph*

  auch: kürzeste Wege Netzwerk, Schichtgraph

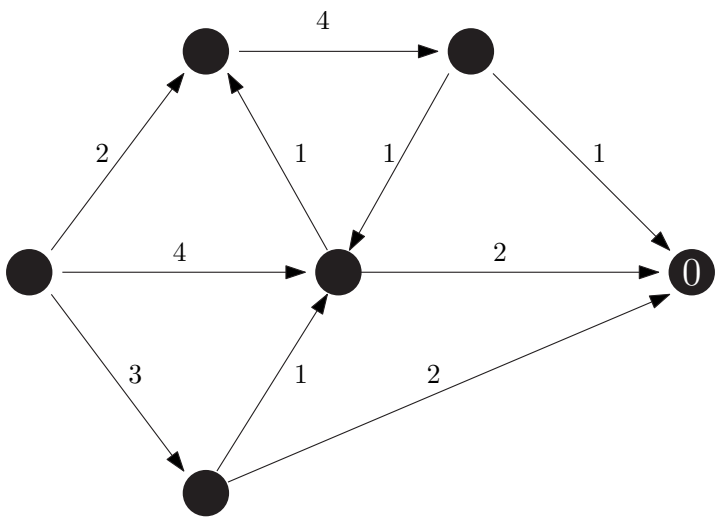


Dinitz

Distanz Label

- Geben Distanz im Residualgraphen (hop-based) zur Senke t an
- Rückwärtsgerichtete Breitensuche
  - Start bei Knoten t
  - Layer: Knoten mit gleicher Distanz zu s im BFS-Baum
  - Knoten in einem Layer: gleiches Label
- *Layered graph*

  auch: kürzeste Wege Netzwerk, Schichtgraph

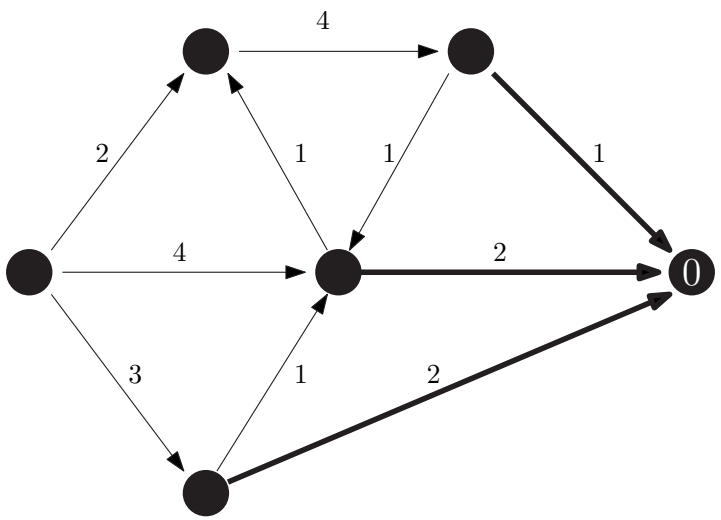


Dinitz

Distanz Label

- Geben Distanz im Residualgraphen (hop-based) zur Senke t an
- Rückwärtsgerichtete Breitensuche
  - Start bei Knoten t
  - Layer: Knoten mit gleicher Distanz zu s im BFS-Baum
  - Knoten in einem Layer: gleiches Label
- *Layered graph*

  auch: kürzeste Wege Netzwerk, Schichtgraph

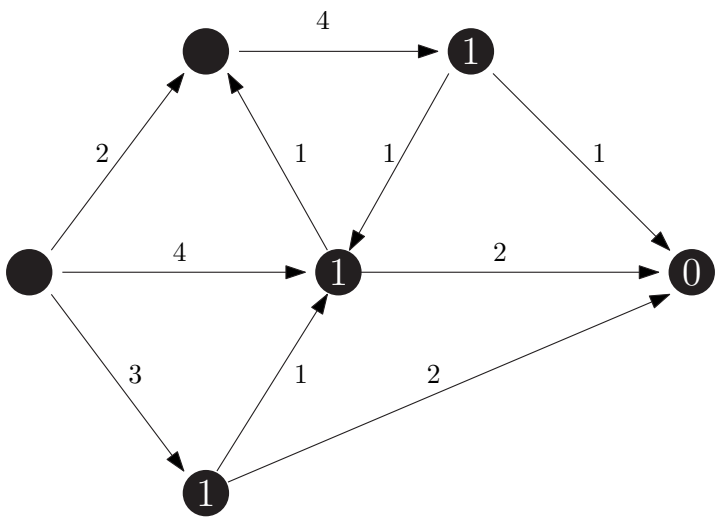


Dinitz

Distanz Label

- Geben Distanz im Residualgraphen (hop-based) zur Senke t an
- Rückwärtsgerichtete Breitensuche
  - Start bei Knoten t
  - Layer: Knoten mit gleicher Distanz zu s im BFS-Baum
  - Knoten in einem Layer: gleiches Label
- *Layered graph*

  auch: kürzeste Wege Netzwerk, Schichtgraph

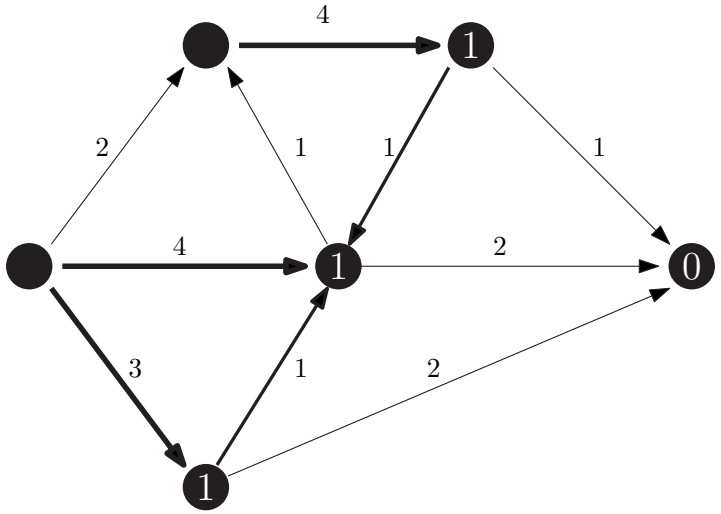


Dinitz

Distanz Label

- Geben Distanz im Residualgraphen (hop-based) zur Senke t an
- Rückwärtsgerichtete Breitensuche
  - Start bei Knoten t
  - Layer: Knoten mit gleicher Distanz zu s im BFS-Baum
  - Knoten in einem Layer: gleiches Label
- *Layered graph*

  auch: kürzeste Wege Netzwerk, Schichtgraph

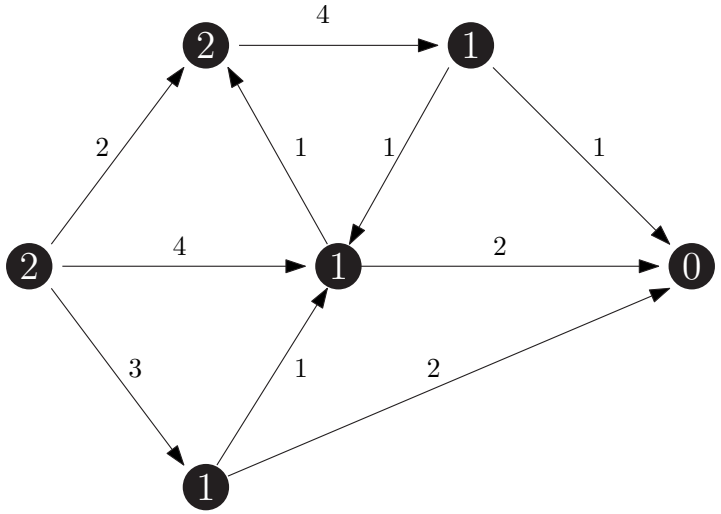


Dinitz

Distanz Label

- Geben Distanz im Residualgraphen (hop-based) zur Senke t an
- Rückwärtsgerichtete Breitensuche
  - Start bei Knoten t
  - Layer: Knoten mit gleicher Distanz zu s im BFS-Baum
  - Knoten in einem Layer: gleiches Label
- *Layered graph*

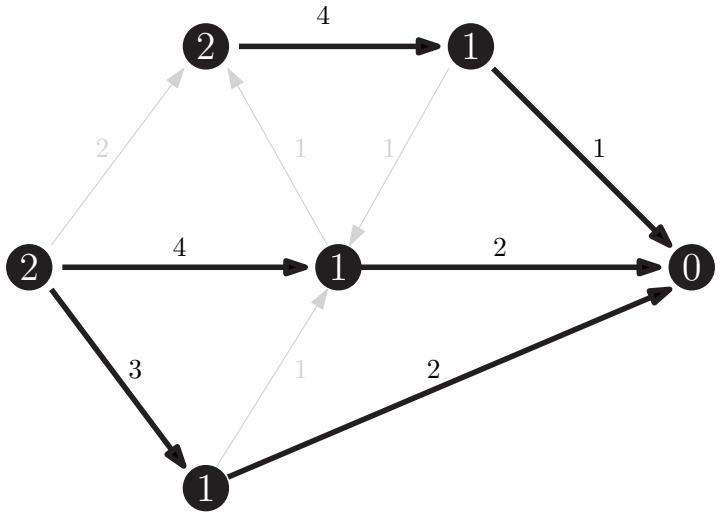
  auch: kürzeste Wege Netzwerk, Schichtgraph



Dinitz

Schichtgraph für Graph $G = (V, E)$

- Knotenmenge $V_S = V$
- $E' = \{e = (u, v) \in E | f(e) < c(e),$
  $d(u) = d(v) + 1\}$
  $E'^{rev} = \{e^{rev} = (v, u)^{rev} | f(v, u) > 0,$
  $d(u) = d(v) + 1\}$
  Kantenmenge $E_S = E' \cup E'^{rev}$
- betrachte Analogie zu Edmonds-Karp (kürzeste erhöhende Wege)

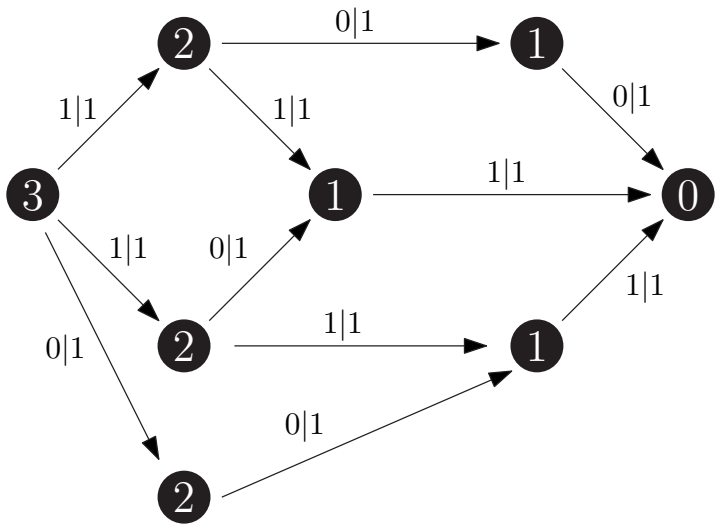


Dinitz

Blockierender Fluss

- Kein weiterer Fluss möglich

  "Auf jedem Weg durch den Graphen mindestens eine Kante bis zur maximalen Kapazität ausgelastet ist"
- *Blocking flow als atomare Operation*
  - Berechnung auf Schichtgraph
  - Kein Residualgraph
  - Kein Rückfluss möglich
- i.A. nicht maximal auf Schichtgraph und Redisualgraph

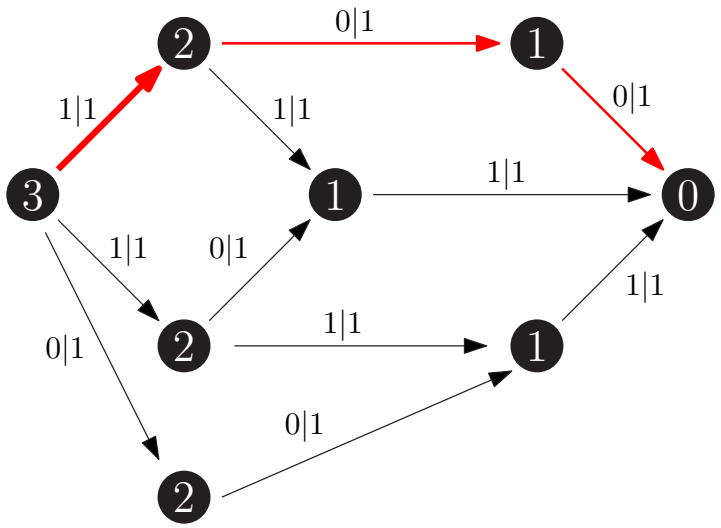


Dinitz

Blockierender Fluss

- Kein weiterer Fluss möglich

  "Auf jedem Weg durch den Graphen mindestens eine Kante bis zur maximalen Kapazität ausgelastet ist"
- *Blocking flow als atomare Operation*
  - Berechnung auf Schichtgraph
  - Kein Residualgraph
  - Kein Rückfluss möglich
- i.A. nicht maximal auf Schichtgraph und Redisualgraph

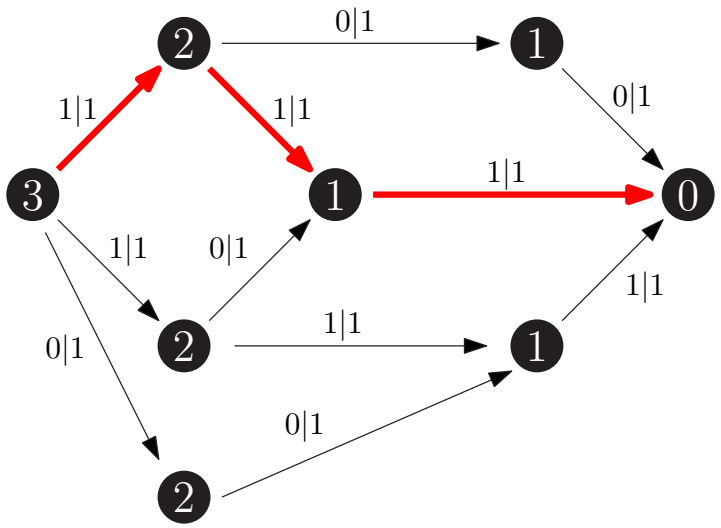


Dinitz

Blockierender Fluss

- Kein weiterer Fluss möglich

  "Auf jedem Weg durch den Graphen mindestens eine Kante bis zur maximalen Kapazität ausgelastet ist"
- *Blocking flow als atomare Operation*
  - Berechnung auf Schichtgraph
  - Kein Residualgraph
  - Kein Rückfluss möglich
- i.A. nicht maximal auf Schichtgraph und Redisualgraph

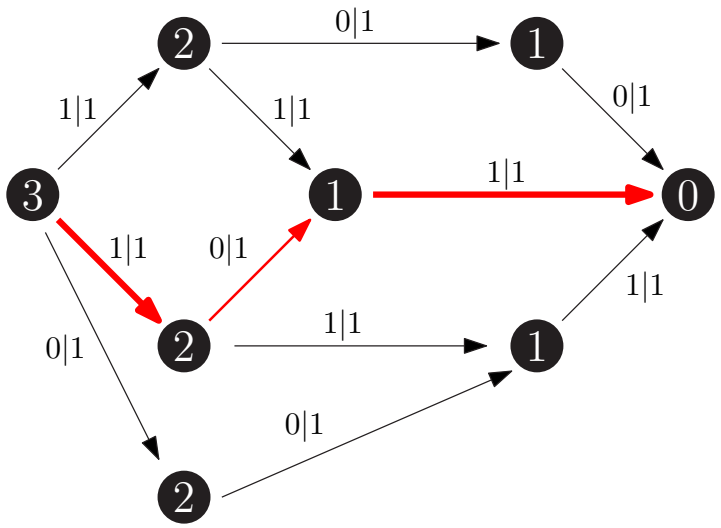


Dinitz

Blockierender Fluss

- Kein weiterer Fluss möglich

  "Auf jedem Weg durch den Graphen mindestens eine Kante bis zur maximalen Kapazität ausgelastet ist"
- *Blocking flow als atomare Operation*
  - Berechnung auf Schichtgraph
  - Kein Residualgraph
  - Kein Rückfluss möglich
- i.A. nicht maximal auf Schichtgraph und Redisualgraph

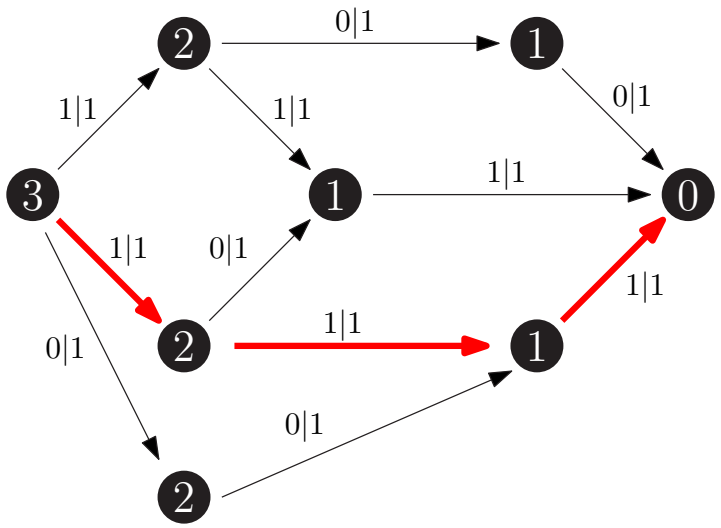


Dinitz

Blockierender Fluss

- Kein weiterer Fluss möglich

  "Auf jedem Weg durch den Graphen mindestens eine Kante bis zur maximalen Kapazität ausgelastet ist"
- *Blocking flow als atomare Operation*
  - Berechnung auf Schichtgraph
  - Kein Residualgraph
  - Kein Rückfluss möglich
- i.A. nicht maximal auf Schichtgraph und Redisualgraph

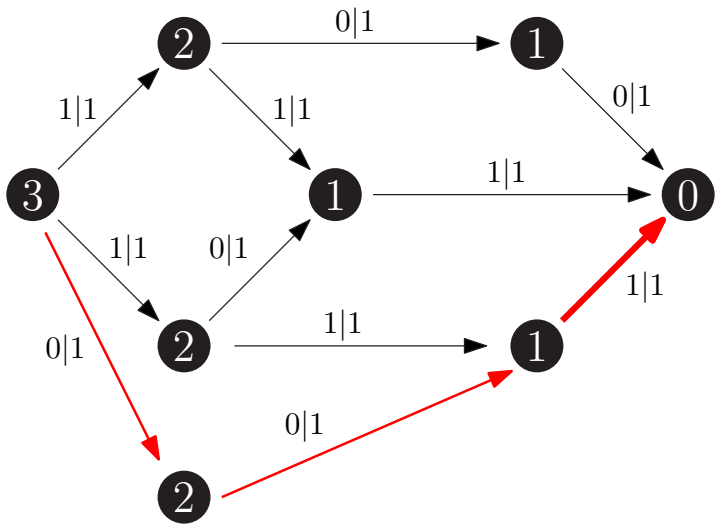


Dinitz

Blockierender Fluss

- Kein weiterer Fluss möglich

  "Auf jedem Weg durch den Graphen mindestens eine Kante bis zur maximalen Kapazität ausgelastet ist"
- *Blocking flow als atomare Operation*
  - Berechnung auf Schichtgraph
  - Kein Residualgraph
  - Kein Rückfluss möglich
- i.A. nicht maximal auf Schichtgraph und Redisualgraph



Dinitz

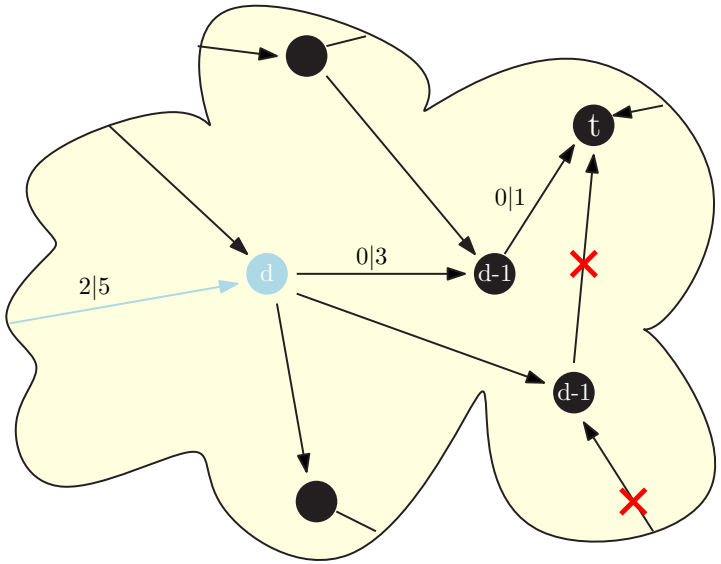
Blockierender Fluss: Operationen

- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - extend - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - retreat - Sackgasse gefunden, gehe zurück, lösche Kante
  - breakthrough - Tiefensuche hat Senke erreicht, lösche saturierte Kanten

Dinitz

Blockierender Fluss: Operationen

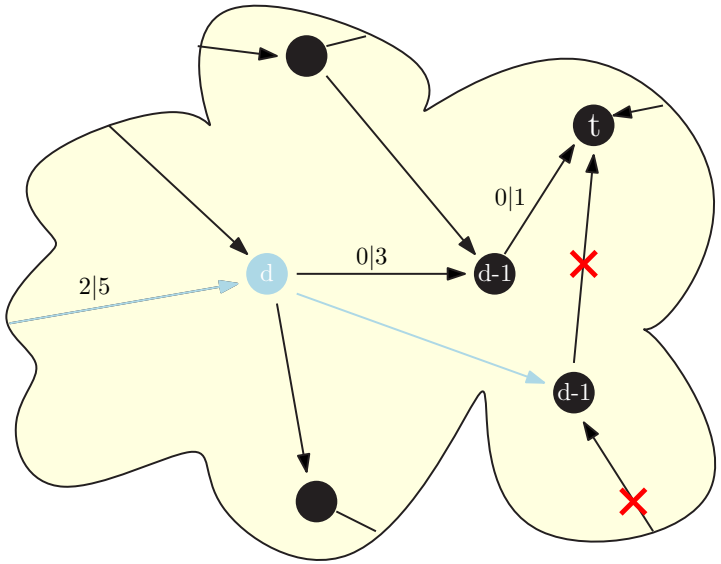
- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - **extend** - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - retreat - Sackgasse gefunden, gehe zurück, lösche Kante
  - breakthrough - Tiefensuche hat Senke erreicht, lösche saturierte Kanten



Dinitz

Blockierender Fluss: Operationen

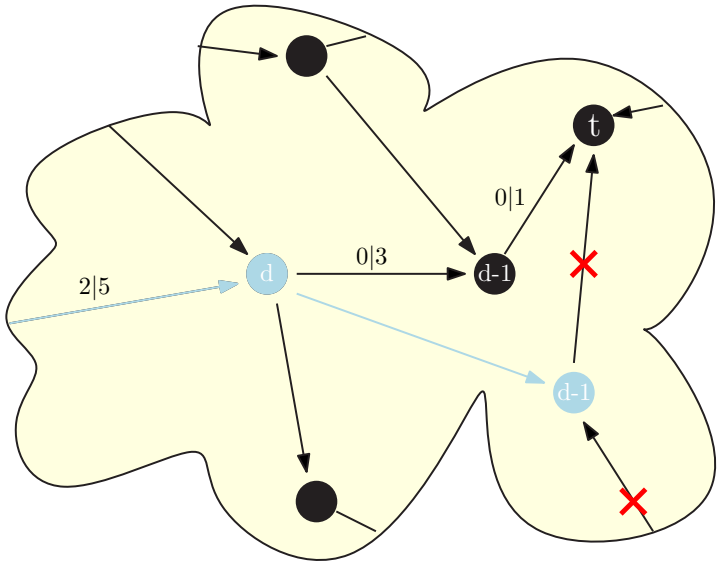
- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - **extend** - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - retreat - Sackgasse gefunden, gehe zurück, lösche Kante
  - breakthrough - Tiefensuche hat Senke erreicht, lösche saturierte Kanten



Dinitz

Blockierender Fluss: Operationen

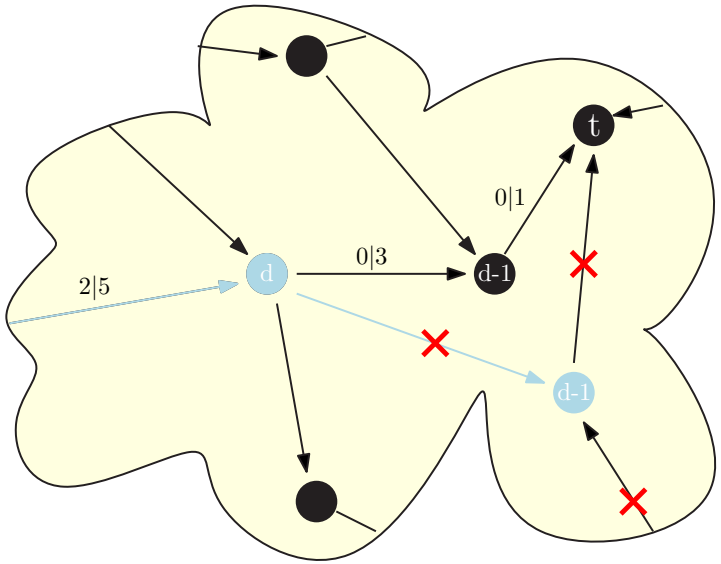
- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - **extend** - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - retreat - Sackgasse gefunden, gehe zurück, lösche Kante
  - breakthrough - Tiefensuche hat Senke erreicht, lösche saturierte Kanten



Dinitz

Blockierender Fluss: Operationen

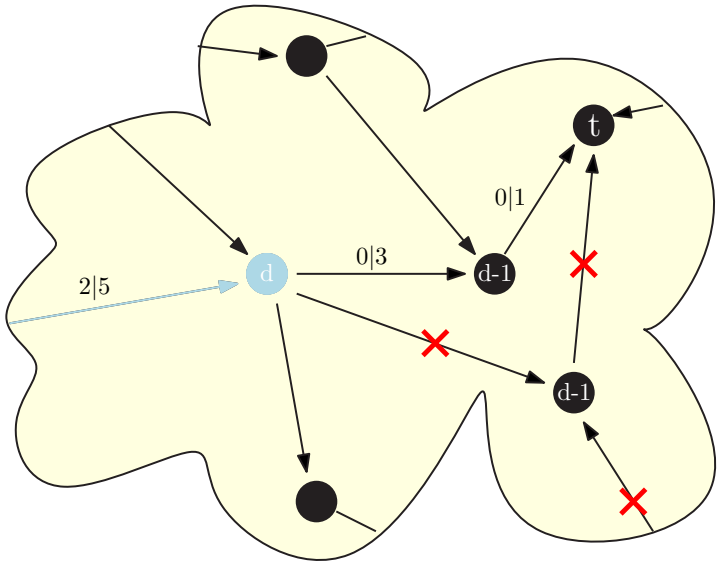
- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - extend - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - **retreat** - Sackgasse gefunden, gehe zurück, lösche Kante
  - breakthrough - Tiefensuche hat Senke erreicht, lösche saturierte Kanten



Dinitz

Blockierender Fluss: Operationen

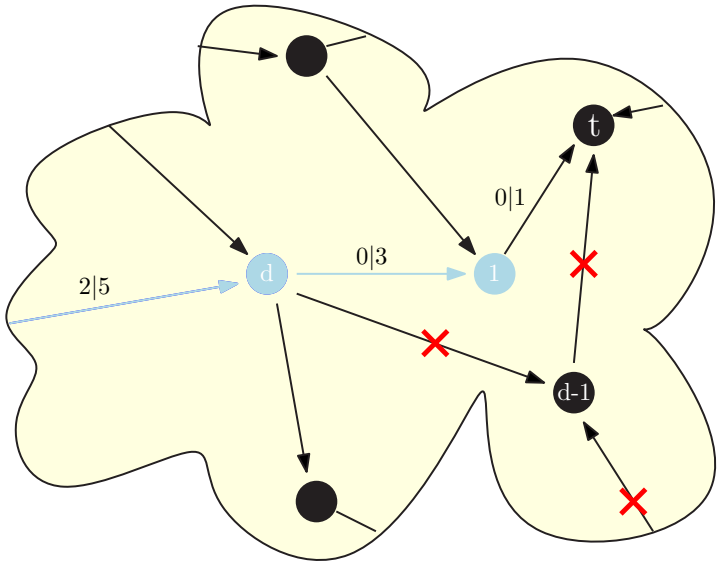
- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - extend - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - **retreat** - Sackgasse gefunden, gehe zurück, lösche Kante
  - breakthrough - Tiefensuche hat Senke erreicht, lösche saturierte Kanten



Dinitz

Blockierender Fluss: Operationen

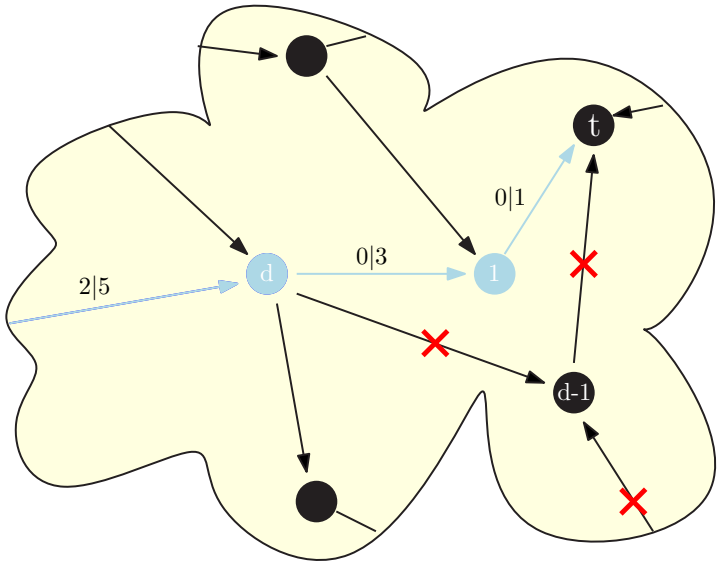
- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - **extend** - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - retreat - Sackgasse gefunden, gehe zurück, lösche Kante
  - breakthrough - Tiefensuche hat Senke erreicht, lösche saturierte Kanten



Dinitz

Blockierender Fluss: Operationen

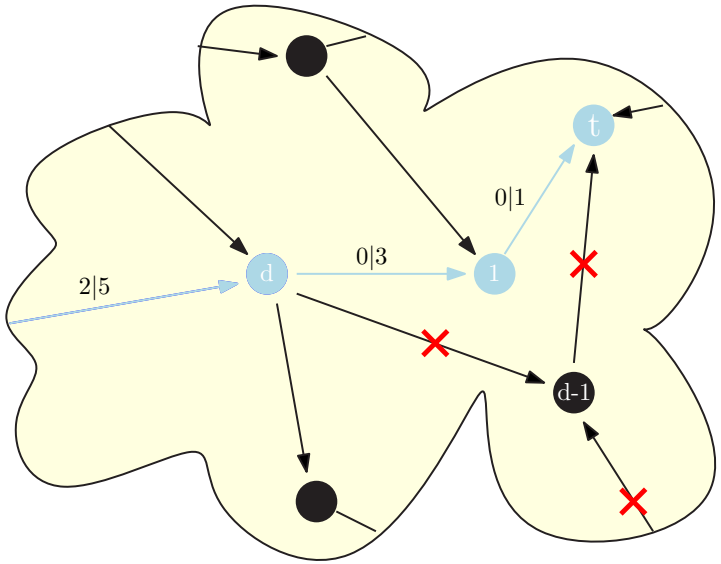
- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - **extend** - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - retreat - Sackgasse gefunden, gehe zurück, lösche Kante
  - breakthrough - Tiefensuche hat Senke erreicht, lösche saturierte Kanten



Dinitz

Blockierender Fluss: Operationen

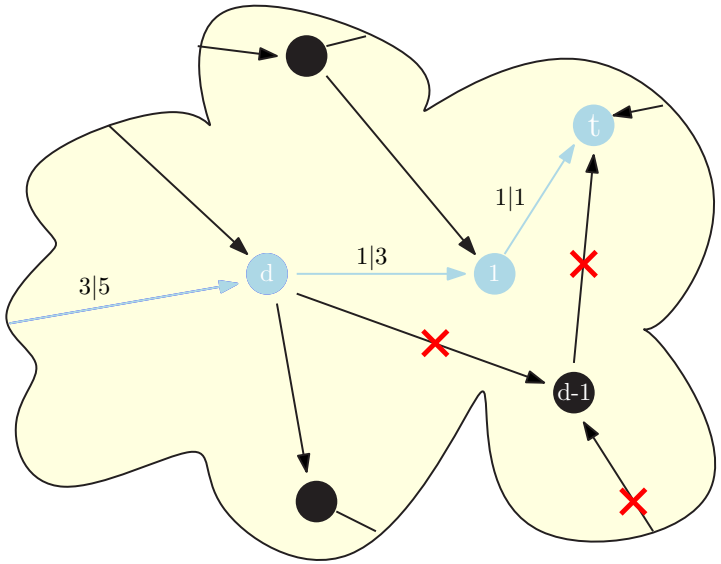
- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - extend - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - retreat - Sackgasse gefunden, gehe zurück, lösche Kante
  - **breakthrough** - Tiefensuche hat Senke erreicht, lösche saturierte Kanten



Dinitz

Blockierender Fluss: Operationen

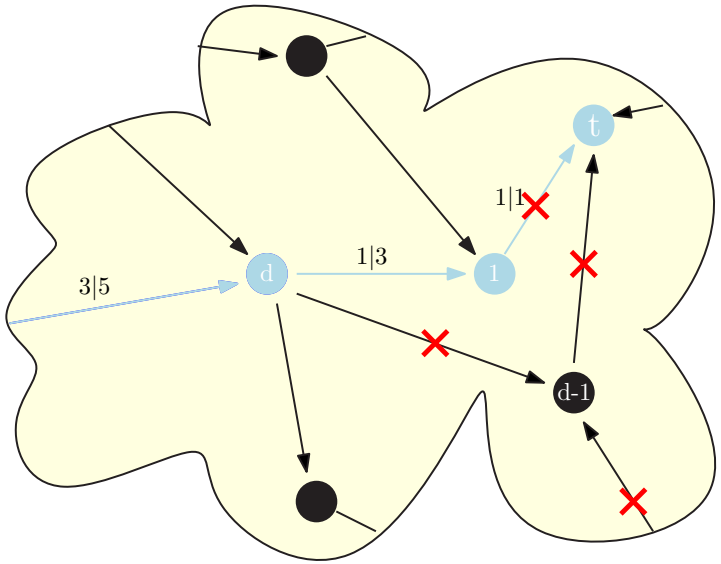
- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - extend - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - retreat - Sackgasse gefunden, gehe zurück, lösche Kante
  - **breakthrough** - Tiefensuche hat Senke erreicht, lösche saturierte Kanten



Dinitz

Blockierender Fluss: Operationen

- *Blocking flow*: Berechnung basiert auf Tiefensuche von Knoten s
- Drei Operationen
  - extend - gehe einen Knoten näher ans Ziel (Schichtgraph)
  - **retreat** - Sackgasse gefunden, gehe zurück, lösche Kante
  - breakthrough - Tiefensuche hat Senke erreicht, lösche saturierte Kanten



Dinitz

Kosten pro Blockierender Fluss

- #breakthrough $\leq m$
  - Jedes breakthrough saturiert mindestens eine Kante
    $\rightarrow$ Kein breakthrough über saturierte Kante mehr möglich
  - **Laufzeit** $O(n)$
- #retreat $\leq m$
  - Jedes retreat löscht eine Kante
  - **Laufzeit** $O(1)$
- #extends $\leq$ #retreats $+ n \cdot$ #breakthrough
  - **Retreat:** Vorher **ein extend**
    Ohne breakthrough nur retreats
  - **Breakthrough:** Vorher $\leq n$ **erfolgreiche extends**
    Schichtgraph schließt Kreise aus
  - **Laufzeit** $O(1)$
- Blockierender Fluss in $O(nm)$

Dinitz

Laufzeit

- Pro Phase
  - Rückwärtsgerichtete Breitensuche: **Laufzeit** $O(n+m)$
  - Blockierender Fluss: **Laufzeit** $O(nm)$
- Phase in $O(nm)$
- #Phasen $\leq n$

  (Lemma 1: Jede Phase erhöht Label von s um mindestens 1)

  $\rightarrow$ Gesamtlaufzeit $O(n^2 m)$

Dinitz

Laufzeit

- Pro Phase
  - Rückwärtsgerichtete Breitensuche: **Laufzeit** $O(n+m)$
  - Blockierender Fluss: **Laufzeit** $O(nm)$
- Phase in $O(nm)$
- #Phasen $\leq n$ (Beweis, wie in Vorlesung, nicht hier)

  (Lemma 1: Jede Phase erhöht Label von s um mindestens 1)

  $\rightarrow$ Gesamtlaufzeit $O(n^2 m)$

Dinitz

Kosten pro Phase - *Unit Capacity Network*

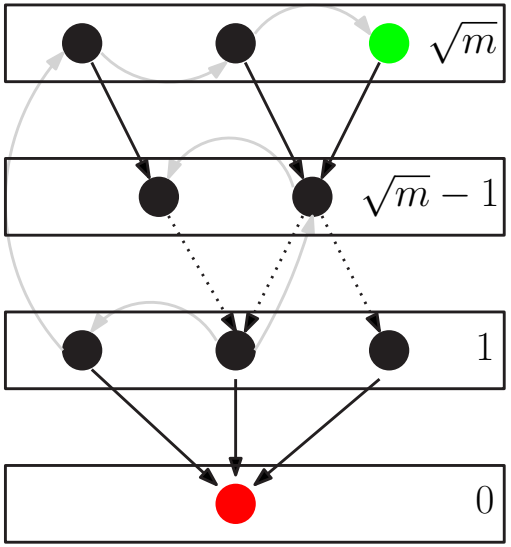
Amortisierte Kosten

- Retreat/breakthrough
  - Löscht alle beteiligten Kanten
    $\rightarrow$ Jede Kante nur an einem Retreat/breakthrough beteiligt
- Extend
  - Pro Kante ein *extend*
  - Bezahlt für Breakthrough und Retreat
  - #extends $= O(m)$
  - Kosten $O(1)$
- Phase in $O(m + n)$

Dinitz

Anzahl Phasen - *Unit Capacity Network*

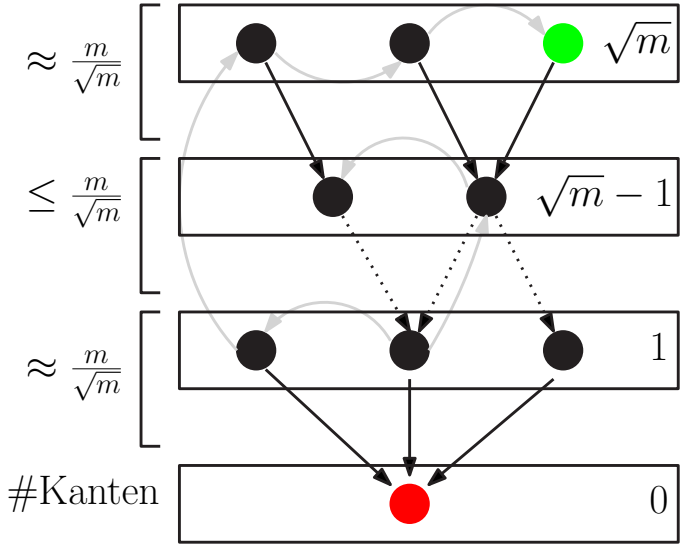
- Nach $\sqrt{m}$ Phasen
  - Graph hat mindestens $\sqrt{m}$ Layer
    (Lemma 1: Jede Phase erhöht Label von s um mindestens 1)
- Erwartet $m/\sqrt{m}$ Kanten pro Layer
  $\rightarrow$ es gibt Layer i mit höchstens $\sqrt{m}$ Kanten zu Layer $i-1$
- Induziert Schnitt (im Residualgraphen) zwischen $S = \{v | d(v) \geq i\}$ und $T = V \setminus S$, Kapazität $\leq \sqrt{m}$
- Jede Phase erhöht Fluss um ≥ 1
  $\rightarrow$ Zusätzlich $\leq \sqrt{m}$ Phasen



Dinitz

Anzahl Phasen - *Unit Capacity Network*

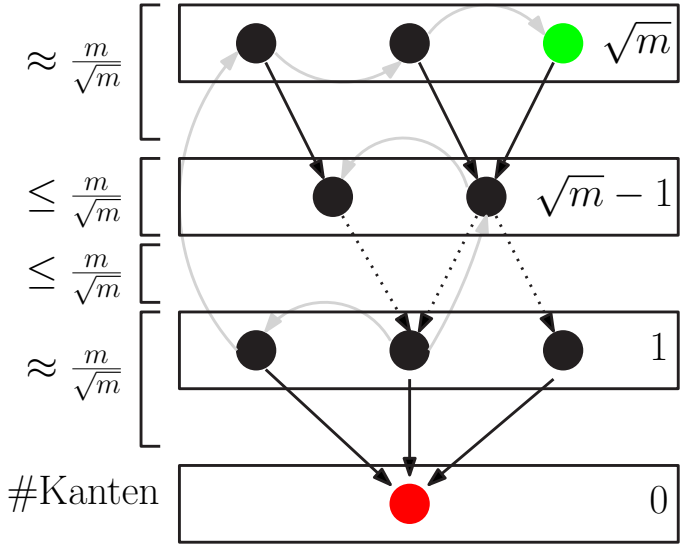
- Nach $\sqrt{m}$ Phasen
  - Graph hat mindestens $\sqrt{m}$ Layer
    (Lemma 1: Jede Phase erhöht Label von s um mindestens 1)
- Erwartet $m/\sqrt{m}$ Kanten pro Layer
  $\rightarrow$ es gibt Layer i mit höchstens $\sqrt{m}$ Kanten zu Layer $i-1$
- Induziert Schnitt (im Residualgraphen) zwischen $S = \{v | d(v) \geq i\}$ und $T = V \setminus S$, Kapazität $\leq \sqrt{m}$
- Jede Phase erhöht Fluss um ≥ 1
  $\rightarrow$ Zusätzlich $\leq \sqrt{m}$ Phasen



Dinitz

Anzahl Phasen - *Unit Capacity Network*

- Nach $\sqrt{m}$ Phasen
  - Graph hat mindestens $\sqrt{m}$ Layer
    (Lemma 1: Jede Phase erhöht Label von s um mindestens 1)
- Erwartet $m/\sqrt{m}$ Kanten pro Layer
  $\rightarrow$ es gibt Layer i mit höchstens $\sqrt{m}$ Kanten zu Layer $i-1$
- Induziert Schnitt (im Residualgraphen) zwischen $S = \{v | d(v) \geq i\}$ und $T = V \setminus S$, Kapazität $\leq \sqrt{m}$
- Jede Phase erhöht Fluss um ≥ 1
  $\rightarrow$ Zusätzlich $\leq \sqrt{m}$ Phasen



Ende!

Feierabend!