

Übung 4 – Algorithmen II

Moritz Laupichler, Nikolai Maas – {moritz.laupichler, nikolai.maas}@kit.edu
https://algo2.iti.kit.edu/AlgorithmenII_WS23.php

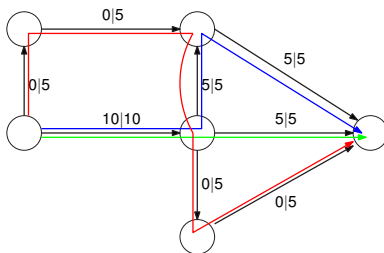
Institut für Theoretische Informatik - Algorithm Engineering

```
    result = current_weight;
    return true;
}

for( EdgeID eid = graph.edgeBegin( current ); eid != graph.edgeEnd( current ); ++eid ){
    const Edge & edge = graph.getEdge( eid );
    COUNTING( statistic_data.inc( DijkstraStatisticData::TOUCHED_EDGES ); )
    if( edge.forward ){
        COUNTING( statistic_data.inc( DijkstraStatisticData::RELAXED_EDGES ); )
        weight new_weight = edge.weight + current_weight;
        GUARANTEE( new_weight >= current_weight, std::runtime_error, "Weight overflow detected." );
        if( !priority_queue.isReached( edge.target ) ){
            COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_EDGES ); )
            COUNTING( statistic_data.inc( DijkstraStatisticData::REACHED_NODES ); )
            priority_queue.push( edge.target, new_weight );
        } else {
            if( priority_queue.getCurrentKey( edge.target ) > new_weight ){
                COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_NODES ); )
                priority_queue.decreaseKey( edge.target, new_weight );
            }
        }
    }
}
```

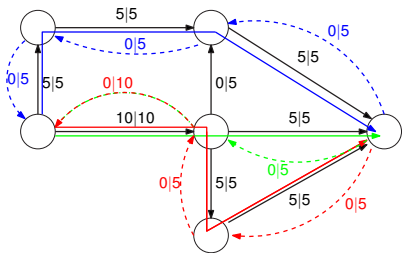
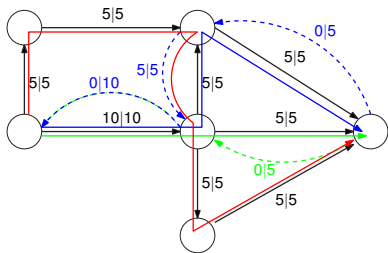
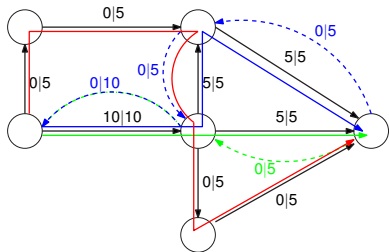
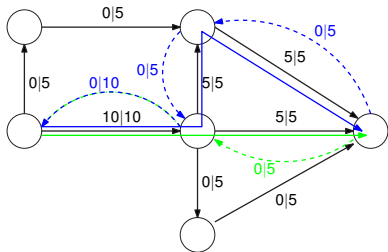
- Maximale Flüsse
 - Ford-Fulkerson
 - Dinitz
 - Preflow-Push

Ford-Fulkerson



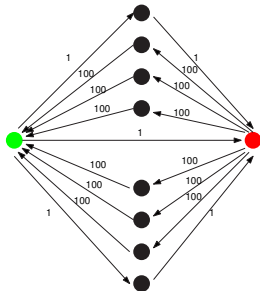
- Verwalten von Restkapazitäten
- Modellierung und Erkennung von “Gegenflüssen”
- $c^f(e) = c(e) - f(e)$: Restkapazität
Hier: Fluss $f(e)$ und Gesamtkapazität $c(e)$
- $c^f(e^{\text{rev}}) = f(e)$: Restkapazität in Gegenrichtung
- Keine Kanten mit Gewicht 0
- Flüsse über Kanten e und e^{rev} erlaubt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

Flüsse und Ford-Fulkerson



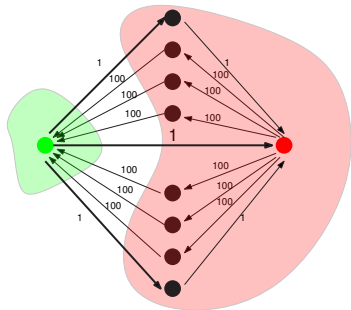
Max Flow - Min Cut

Max Flow - Min Cut



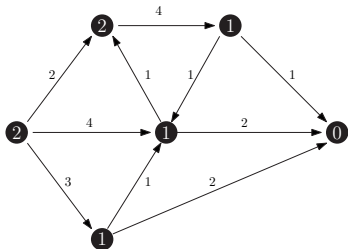
Max Flow - Min Cut

- S - T -Schnitte betrachten nur Kanten $S \rightarrow T$
- Kanten $T \rightarrow S$ werden **nicht** berücksichtigt
- Flow muss durch alle möglichen S - T -Schnitte
⇒ **Max Flow = Min S - T -Cut**

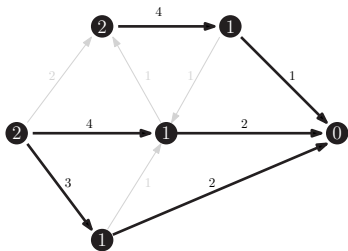


Dinitz

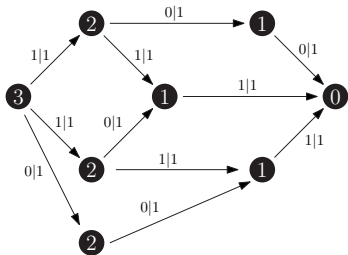
- Geben Distanz im Residualgraphen zur Senke t an (hop-based)
- Rückwärtsgerichtete Breitensuche
 - Start bei Knoten t
 - Layer: Knoten mit gleicher Distanz zu s im BFS-Baum
- *Layer Graph* (Schichtgraph)



- Knotenmenge $V_S = V$
- $E' = \{e = (u, v) \in E \mid f(e) < c(e), d(u) = d(v) + 1\}$
- $E'^{rev} = \{e^{rev} = (v, u)^{rev} \mid f(v, u) > 0, d(u) = d(v) + 1\}$
- Kantenmenge $E_S = E' \cup E'^{rev} \Rightarrow$ azyklisch



- Kein weiterer Fluss möglich
"Auf jedem Pfad ist mindestens eine Kante ausgelastet"
- *Blocking flow als atomare Operation*
 - Berechnung auf Schichtgraph
 - Kein Residualgraph
 - Kein Rückfluss möglich
- i.A. nicht maximal auf Schichtgraph und Residualgraph



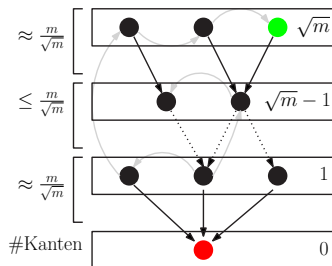
- $\#breakthrough \leq m$
 - Jedes breakthrough saturiert mindestens eine Kante
⇒ Kein breakthrough über saturierte Kante mehr möglich
 - **Laufzeit** $O(n)$
- $\#retreat \leq m$
 - Jedes retreat löscht eine Kante
 - **Laufzeit** $O(1)$
- $\#extends \leq \#retreats + n \cdot \#breakthrough$
 - **Retreat:**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

- Pro Phase
 - Rückwärtsgerichtete Breitensuche: **Laufzeit** $O(n + m)$
 - Blockierender Fluss: **Laufzeit** $O(nm)$
- Phase in $O(nm)$
- #Phasen $\leq n$ (Beweis in Vorlesung)
(Lemma 1: Jede Phase erhöht Label von s um mindestens 1)
 \Rightarrow **Gesamtlaufzeit** $O(n^2m)$

Amortisierte Kosten

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

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



Preflow-Push

Preflow-Push

Wiederholung

Bezeichnungen

■ aktiver Knoten

- Knoten v aktiv gdw. $\text{excess}(v) = \text{inflow}(v) - \text{outflow}(v) > 0$

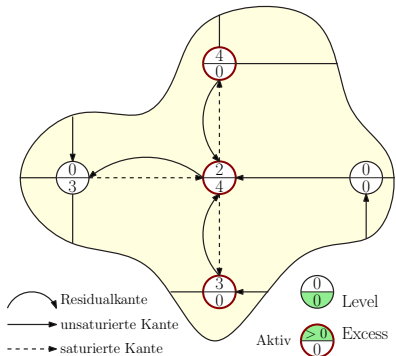
■ gültige Kante

- Kante $(v, w) \in G^f$ ist gültig, wenn $\text{Level } d(v) = d(w) + 1$

allgemeiner Ablauf

1. wähle **aktiven Knoten** v
2. falls **gültige Kante** (v, w) existiert: **push**
 - schiebe Fluss entlang (v, w)
3. ansonsten: **relabel**

- Operationen nur auf aktiven Knoten
- aktive Knoten haben excess
inflow > outflow
- **push**
 - schiebe Fluss entlang Kante (u, v)
 - Voraussetzung: $d(u) = d(v) + 1$
- **relabel**
 - erhöht Level eines Knoten
 - nur zulässig wenn *push* vom Knoten nicht möglich
 \Rightarrow erhält $d(u) \leq d(v) + 1$ für alle (u, v) im Residualgraph



Preflow-Push

Wiederholung

Bezeichnungen

■ aktiver Knoten

- Knoten v aktiv gdw. $\text{excess}(v) = \text{inflow}(v) - \text{outflow}(v) > 0$

■ gültige Kante

- Kante $(v, w) \in G^f$ ist gültig, wenn $\text{Level } d(v) = d(w) + 1$

allgemeiner Ablauf

1. wähle aktiven Knoten v

WELCHEN?

2. falls gültige Kante (v, w) existiert: push

WELCHE?

- schiebe Fluss entlang (v, w)

WIEVIEL?

$$f'_{(v,w)} = f_{(v,w)} + \min\{c_{(v,w)}^f, \text{excess}(v)\}$$

3. ansonsten: relabel

- erhöhe Level von v

WIEVIEL?

$$d(v) = d(v) + 1$$

Unterschiedliche Auswahl des **aktiven Knotens**:

- *generic preflow-push* $\mathcal{O}(n^2 m)$
- *FIFO preflow-push* $\mathcal{O}(n^3)$
- *highest-level preflow-push* $\mathcal{O}(n^2 \sqrt{m})$

Unterschiedliches **relabel**:

- *aggressive local relabeling*
- *global relabeling*
- *gap heuristic*

⇒ nur **Heuristiken**, aber in der Praxis deutliche Beschleunigung!

Relabeling

- *aggressive local relabeling*
- *global relabeling*
- *gap heuristic*

Knotenauswahl

- *two-phase approach*

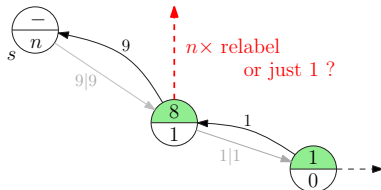
■ aggressive local relabeling

- erhöhe Level in einem Schritt, so dass gültige Kante existiert

$$d(v) = 1 + \min_{(v,w) \in E^f} d(w)$$

$d(w) \geq d(v)$, wenn keine gültige Kante in G^f existiert!

Beispiel 2



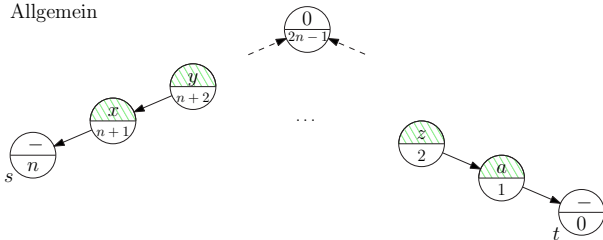
■ global relabeling

- setze Levels auf

$$d(v) = \begin{cases} \mu(v, t) & \text{falls } t \text{ von } v \text{ erreichbar} \\ n + \mu(v, s) & \text{sonst, falls } s \text{ von } v \text{ erreichbar} \\ 2n - 1 & \text{sonst} \end{cases}$$

- Berechnung der Distanzen $\mu(v, \cdot)$ über Breitensuche in $\mathcal{O}(m)$
nur alle $\Omega(m)$ Schritte, damit Kosten $\mathcal{O}(1)$ pro Schritt

Allgemein

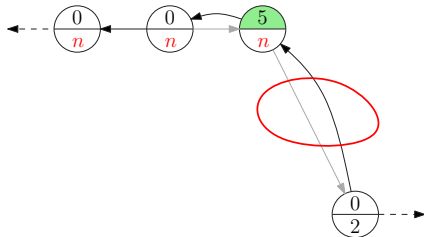


■ *gap heuristic*

- wird ein Level durch $\text{relabel}(v)$ leer, setze Level von v und aller von v erreichbaren Knoten auf

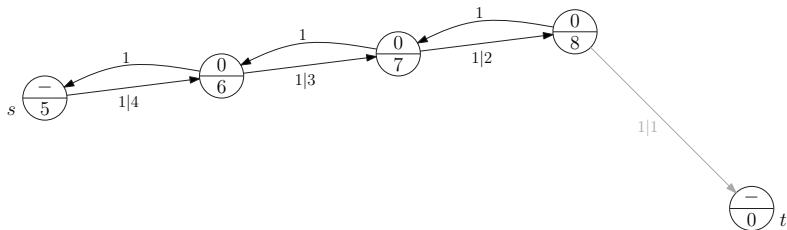
$$d(w) = \max\{d(w), n\}$$

Lücke kann nie mehr überwunden werden, Fluss nur noch zurück zu s



■ *two-phase approach*

- Phase 1: wähle nur Knoten Level $d(v) < n$ aus
⇒ erzeugt *maximum preflow*
⇒ korrekter Fluss in t
- Phase 2: nur noch Knoten mit Level $d(v) \geq n$ übrig
⇒ Fluss nur nach s möglich



Ende!



Feierabend!