DHBW Lörrach

Duale Hochschule Baden-Württemberg

Algorithmen und Komplexität
TIF 21A/B
Dr. Bruno Becker

9. Optimierungsprobleme für Graphen

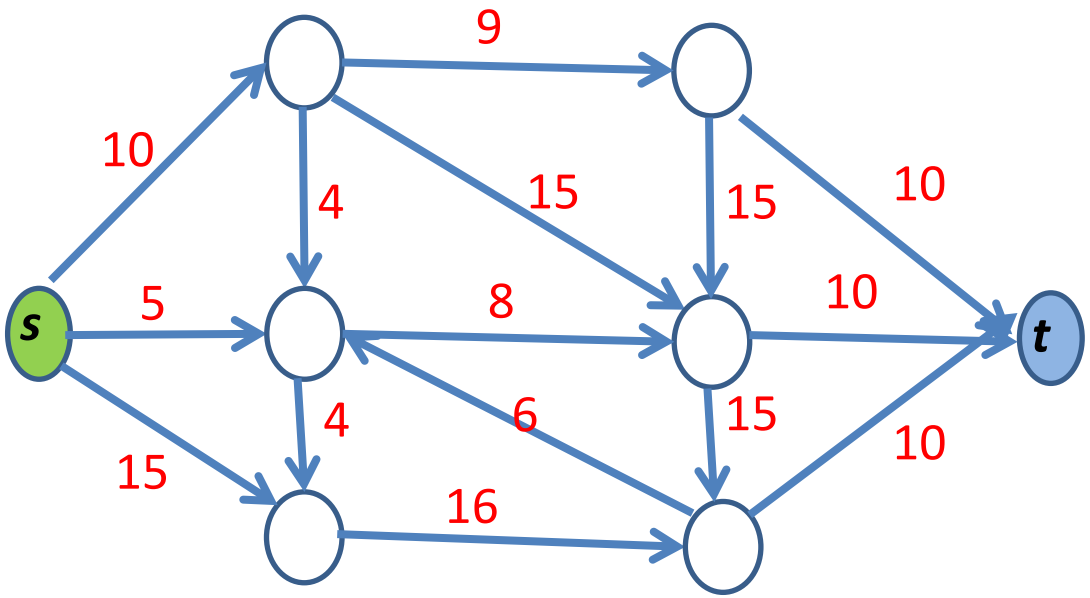
9.3. Maximaler Fluss

Maximaler Fluss

- **Flussnetzwerke und Flüsse**
- Der Ford-Fulkerson-Algorithmus
- Schnitte und Flüsse

Anwendungsbeispiel für maximalen Fluss

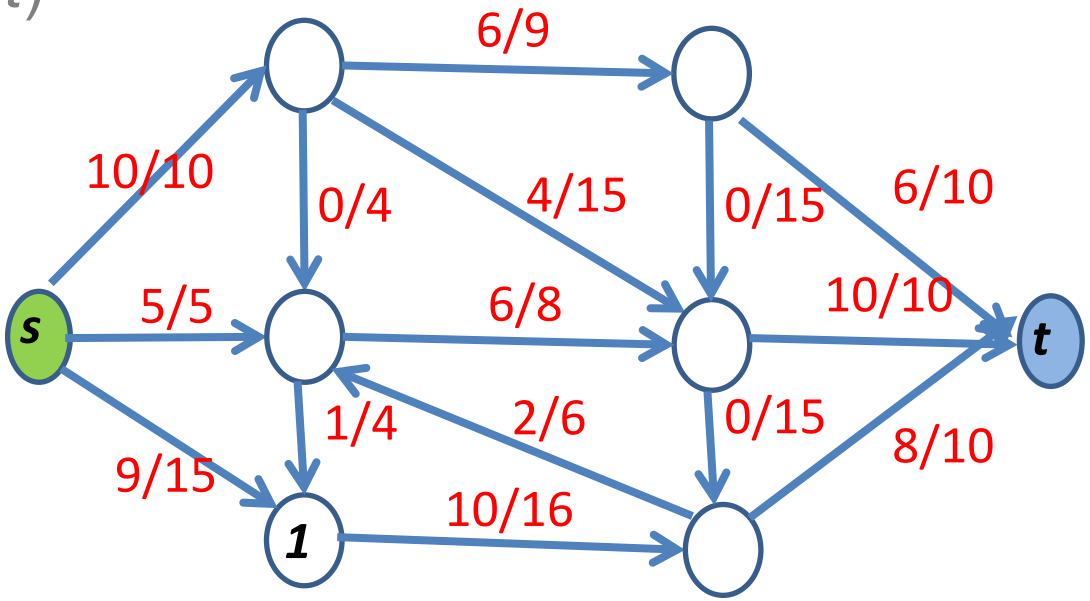
- Datennetzwerk mit mehreren Abschnitten
- Kantengewichte: Bandbreite des Abschnitts
- Wie erreicht man maximale Bandbreite von *s* nach *t* ?



Fluss (*flow*) in einem Flussnetzwerk

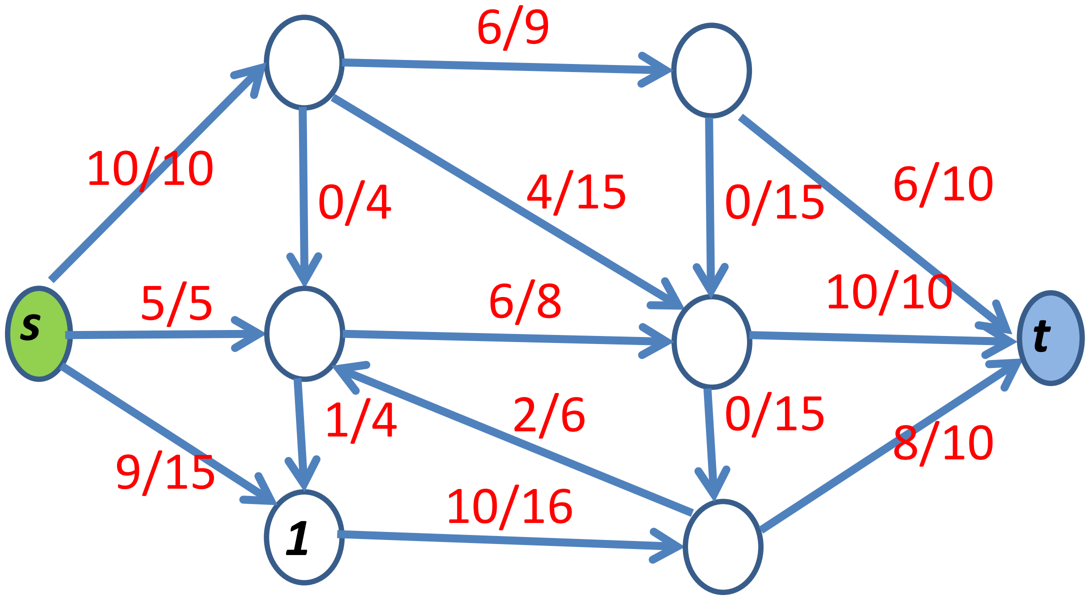
Funktion $f : E \to \mathbb{R}$, ordnet jeder Kante *e* einen Flusswert zu, sodass:

1. **Kapazitätsbeschränkung:** $0 \leq f(e) \leq c(e)$
2. **Flusserhaltung:** eingehender Fluss= ausgehender Fluss für jeden Knoten (außer *s* und *t*)



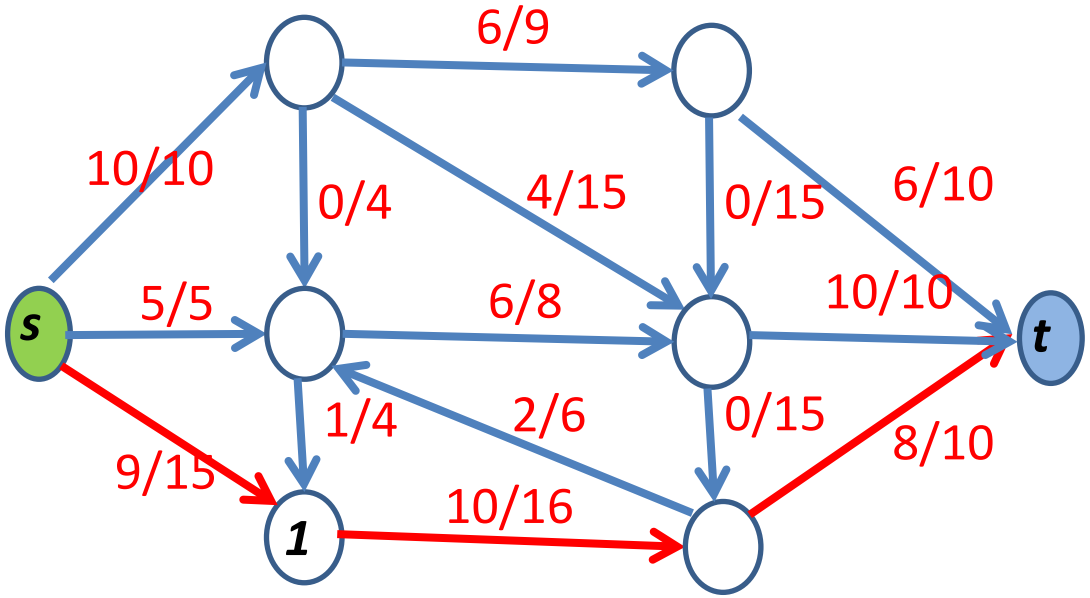
Wert eines Flusses, maximaler Fluss

- **Wert eines Flusses** = eingehender Fluss in *t* = ausgehender Fluss in *s*
- **Max-Flow-Problem:** Finde einen Fluss von maximalem Wert in einem Netzwerk



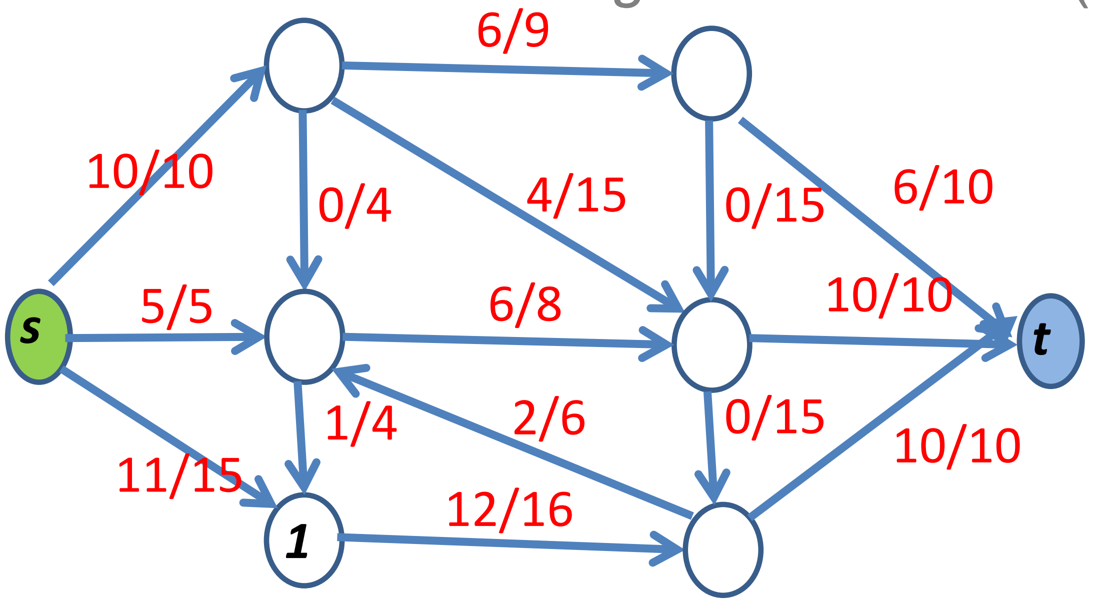
Wie kann man einen gegebenen Fluss verbessern?

- **Einfacher Fall:** Finde einen ***gerichteten*** Pfad von *s* nach *t*, in dem alle Kanten noch freie Kapazitäten haben
- Erhöhe Fluss um ***Engpass-Kapazität*** **:** Minimum der freien Kapazitäten auf dem Erweiterungspfad



Erweiterungspfad

- **Erweiterungspfad (*augmenting path*):** *Ungerichteter* Pfad von *s* nach *t*, in dem der Fluss:
  1. Auf allen Vorwärtskanten *erhöht* werden kann (nicht voll)
  2. Auf allen Rückwärtskanten *erniedrigt* werden kann (nicht leer)



Maximaler Fluss

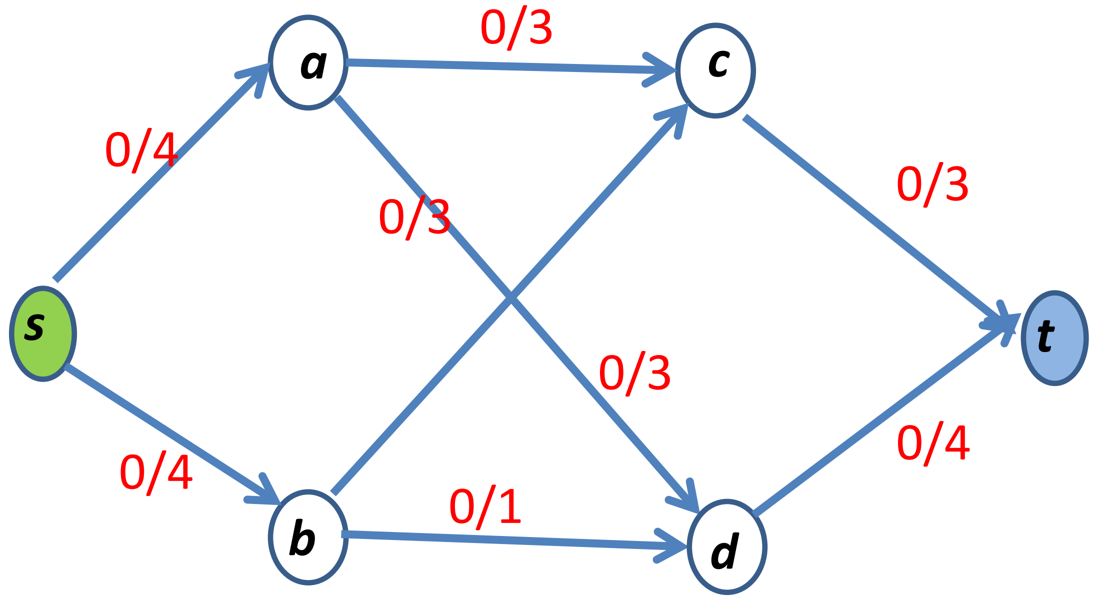
- Flussnetzwerke und Flüsse
- **Der Ford-Fulkerson-Algorithmus**
- Schnitte und Flüsse

Ford-Fulkerson Algorithmus

1. *Starte mit Fluss 0;*
2. *Solange es einen Erweiterungspfad gibt*
   2.1. Finde einen Erweiterungspfad;
   2.2 Berechne Engpasskapazität
   2.3 Erhöhe Fluss auf dem Erweiterungspfad um die Engpasskapazität

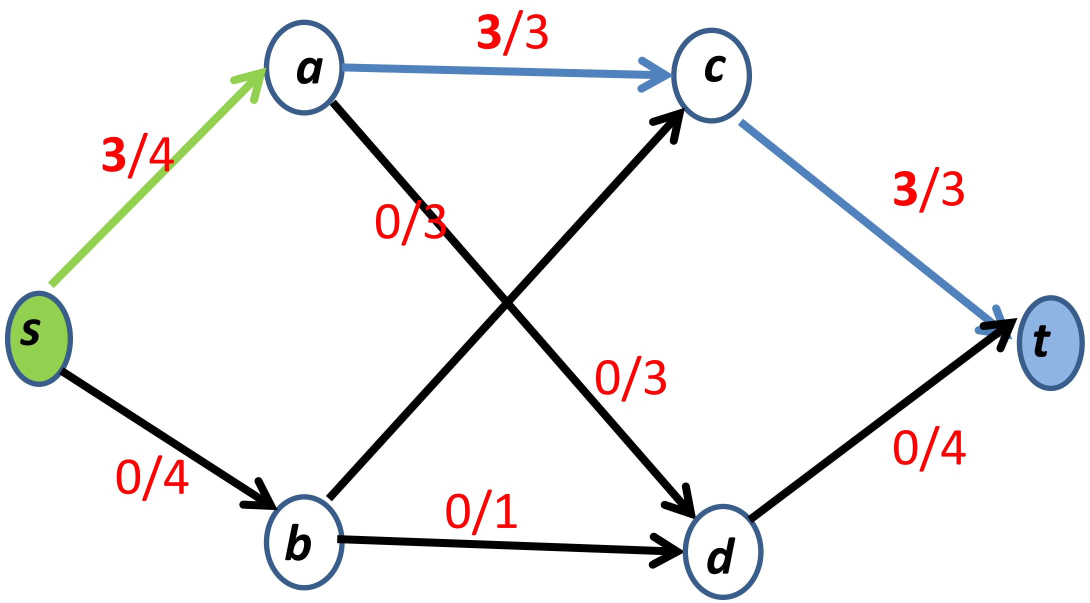
Beispiel zum Algorithmus von Ford-Fulkerson

- Start mit Fluss 0
- Suche Erweiterungspfad von *s* nach *t*
- Möglichkeiten: *sact, sadt, sbct, sbdt.*



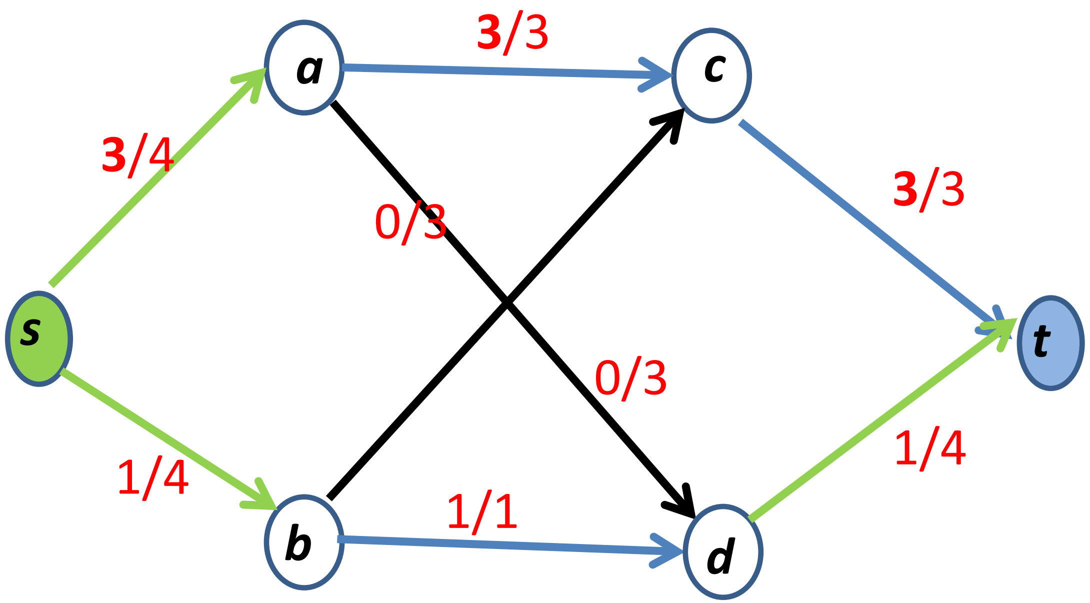
Beispiel zum Algorithmus von Ford-Fulkerson (2)

- Auswahl: *sact* (zufällig); Erhöhung Fluss um Engpass-Kapazität (3)
- Blaue Kanten voll ausgelastet – dürfen nur rückwärts verwendet werden
- Schwarze Kanten unbenutzt – dürfen nur vorwärts verwendet werden
- Grüne Kanten teilweise ausgelastet- Nutzung in beiden Richtungen



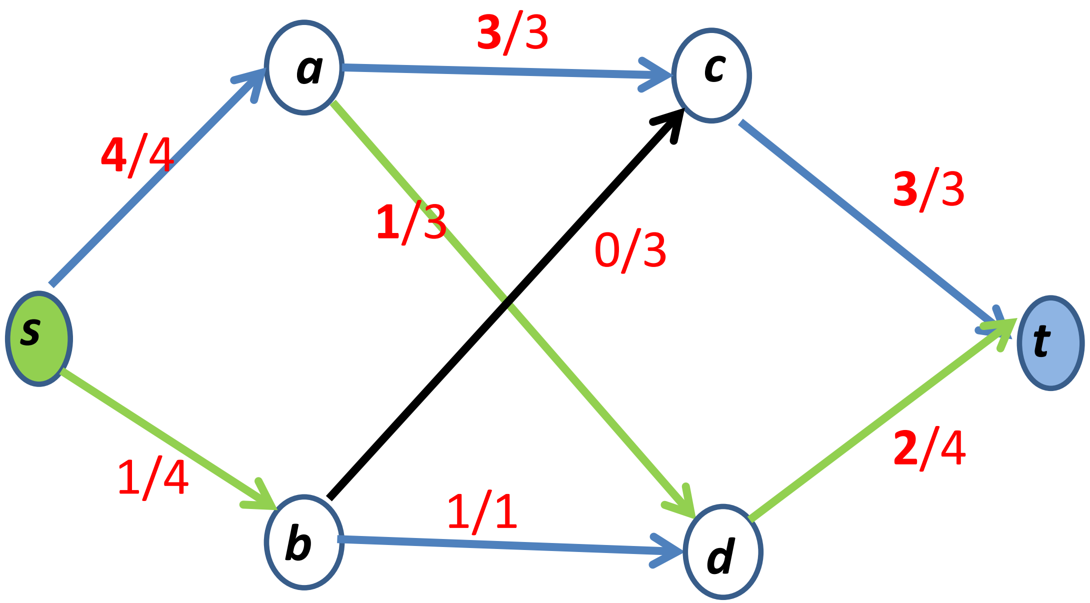
Beispiel zum Algorithmus von Ford-Fulkerson (3)

- Mögliche Pfade: *sadt, sbdt, sbcadt*
- Wähle: *sbdt*, Engpass-Kapazität 1
- Neuer Fluss hat Wert 4



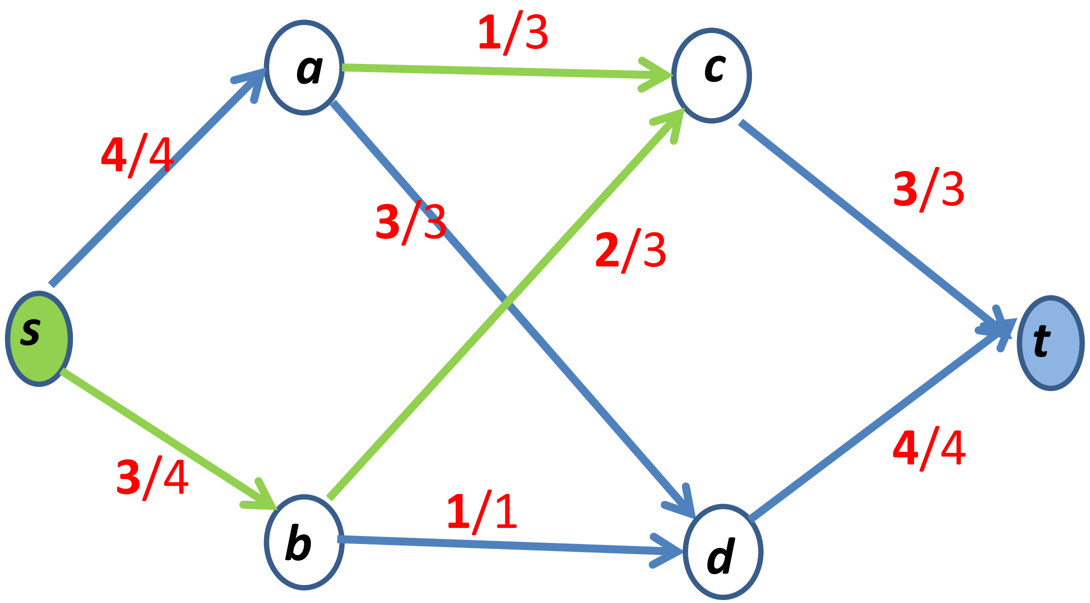
Beispiel zum Algorithmus von Ford-Fulkerson (4)

- Im Anschluss gibt es zwei mögliche Pfade: *sadt, sbcadt*
- Wähle: *sadt*, Engpass-Kapazität 1
- Neuer Fluss hat Wert 5



Beispiel zum Algorithmus von Ford-Fulkerson (5)

- Im Anschluss gibt es einen mögliche Pfad: *sbcadt*
- Engpass-Kapazität 2
- Neuer Fluss hat Wert 7
- *Danach gibt es keinen Erweiterungspfad mehr! → Maximaler Fluss erreicht*



Ford-Fulkerson Algorithmus

1. *Starte mit Fluss 0;*
2. *Solange es einen Erweiterungspad gibt*
   2.1. Finde einen Erweiterungspfad;
   2.2 Berechne Engpasskapazität
   2.3 Erhöhe Fluss auf dem Erweiterungspfad um die Engpasskapazität

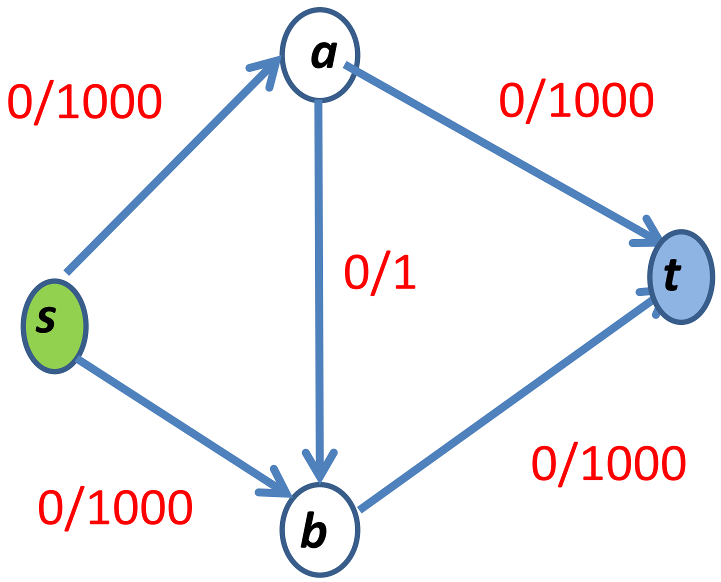
Fragen:

- Wie findet man algorithmisch einen Erweiterungspfad?
- Terminiert Algorithmus immer? Wenn ja, nach wievielen Erweiterungen?
- Falls Algorithmus terminiert, ist das Ergebnis immer maximaler Fluss?

Terminierung von Ford-Fulkerson Algorithmus

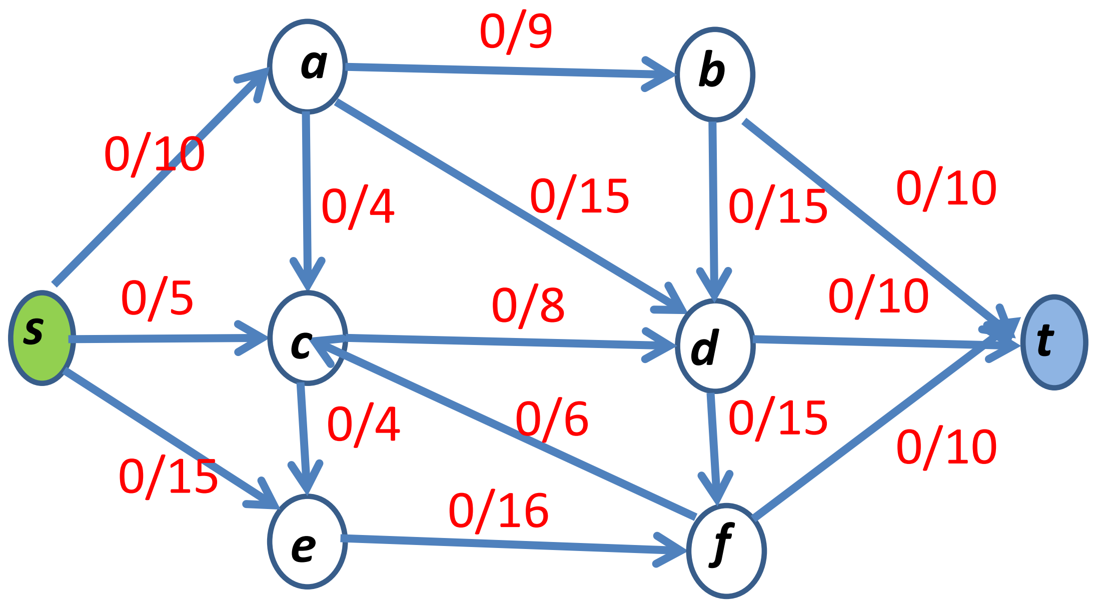
- *Ganzzahlige* Kapazitäten
  - Terminiert immer, da Fluss bei jedem Schleifendurchlauf um mindestens 1 erhöht wird
- *Rationale* Kapazitäten
  - Multiplikation mit Hauptnenner aus allen Kantengewichten ergibt ganzzahlige Kapazitäten → Terminiert immer
- *Irrationale* Kapazitäten
  - Es gibt Fälle, in denen der Algorithmus **nicht** terminiert
  - In der Praxis keine Bedeutung → Begrenzung auf rationale Kapazitäten

Ungünstiges Beispiel für Ford-Fulkerson

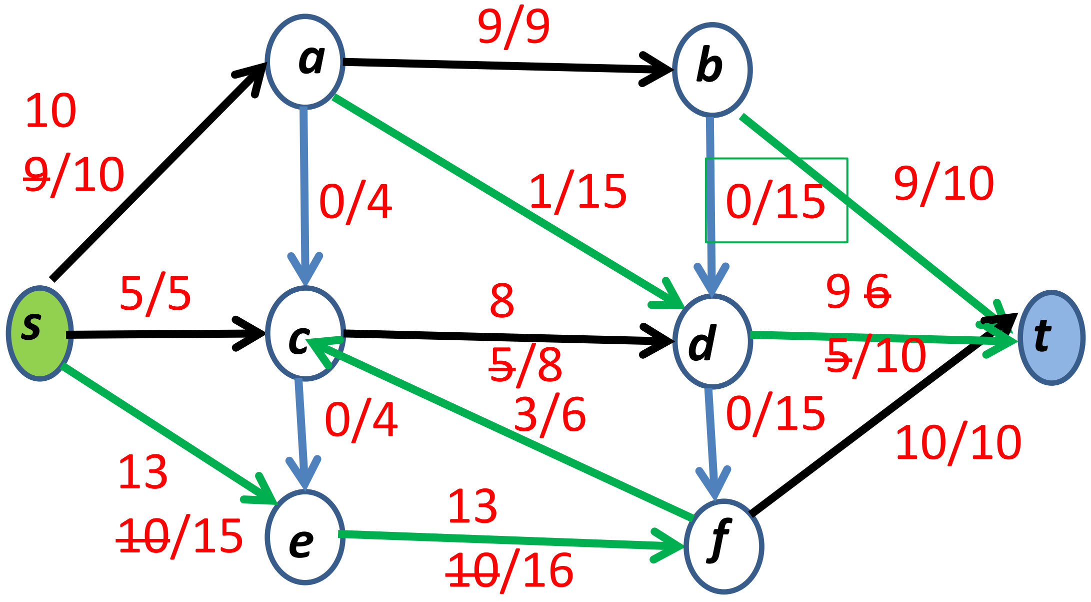


- Im Worstcase:
  - z.B Tiefensuche: Anzahl Erweiterungsschritte **= Maxflow-Wert**!
- Gegenmaßnahme: Erweiterungspfad mit Breitensuche: Immer möglichst wenig Pfeile → Aufwand unabhängig vom Maxflow-Wert!

Beispiel 1 als Übung



Beispiel 1 als Übung



1. s-e-f-t: Engpass f-t 10 F=10
2. s-a-b-t: Engpass a-b 9   F = 19
3. s-c-d-t: Engpass s-c 5   F=24
4. s-a-d-t: Engpass s-a 1 F=25
5. s-e-f-c-d-t: Engpass c-d 3 F=28

Weiterer Pfad?
Geht nur über s-e-f-c -> Ende

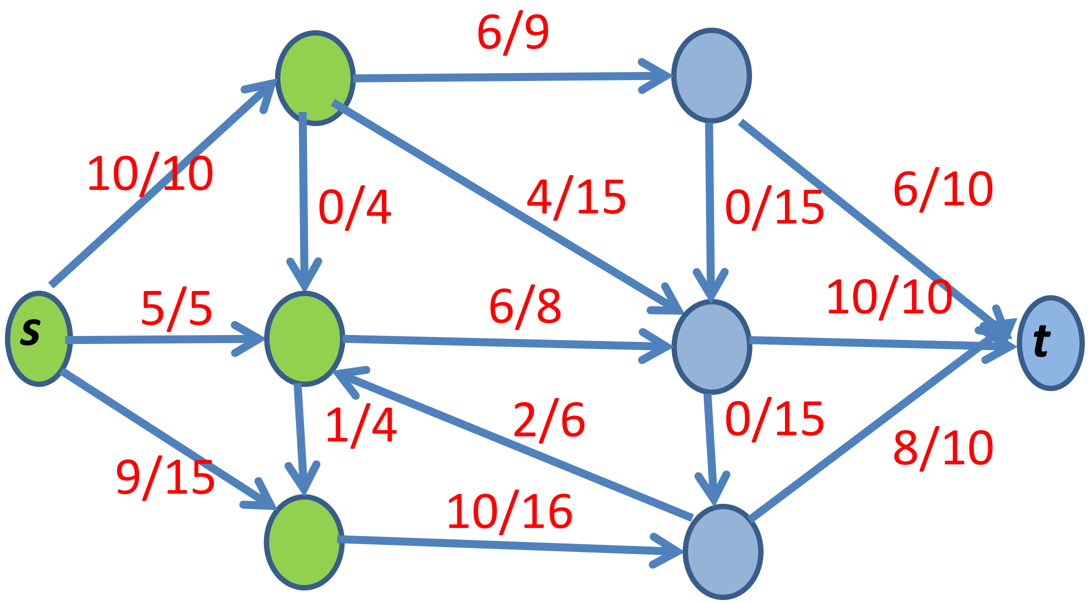
Wie kann man sicherstellen, dass man Maximalen Fluss gefunden hat?

Maximaler Fluss

- Flussnetzwerke und Flüsse
- Der Ford-Fulkerson-Algorithmus
- **Schnitte und Flüsse**

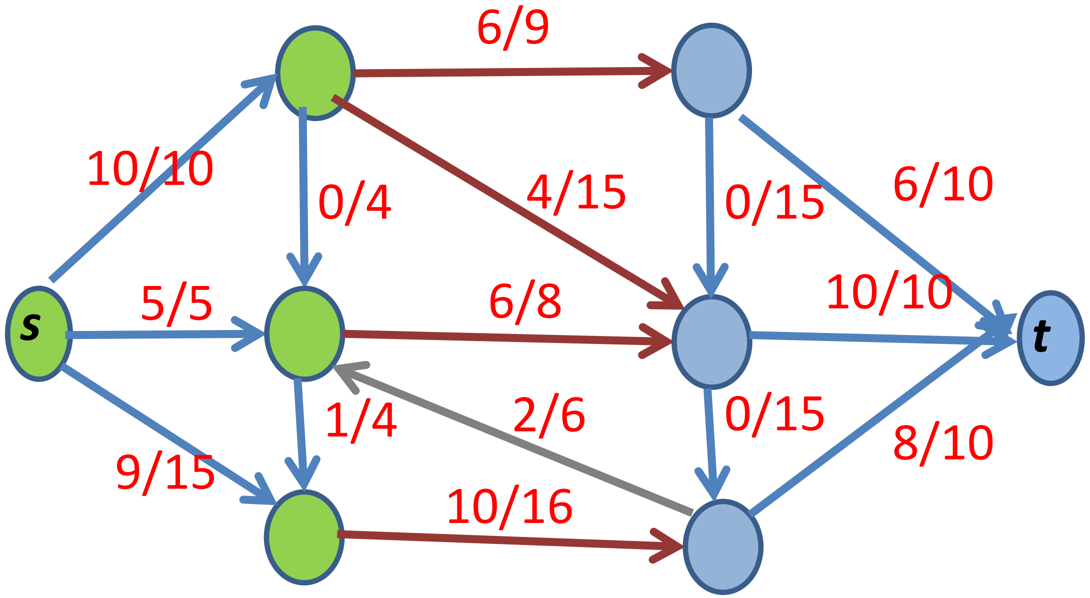
Schnitte in Flussnetzwerken

- **Schnitt (*cut*)** in einem Flussnetzwerk: Partition der Knoten in zwei Teilmengen, von denen eine *s* und die andere *t* enthält
- **Schnittkante (*crossing edge*)** verbindet einen Knoten in der einen Teilmenge mit einem Knoten in der anderen Teilmenge



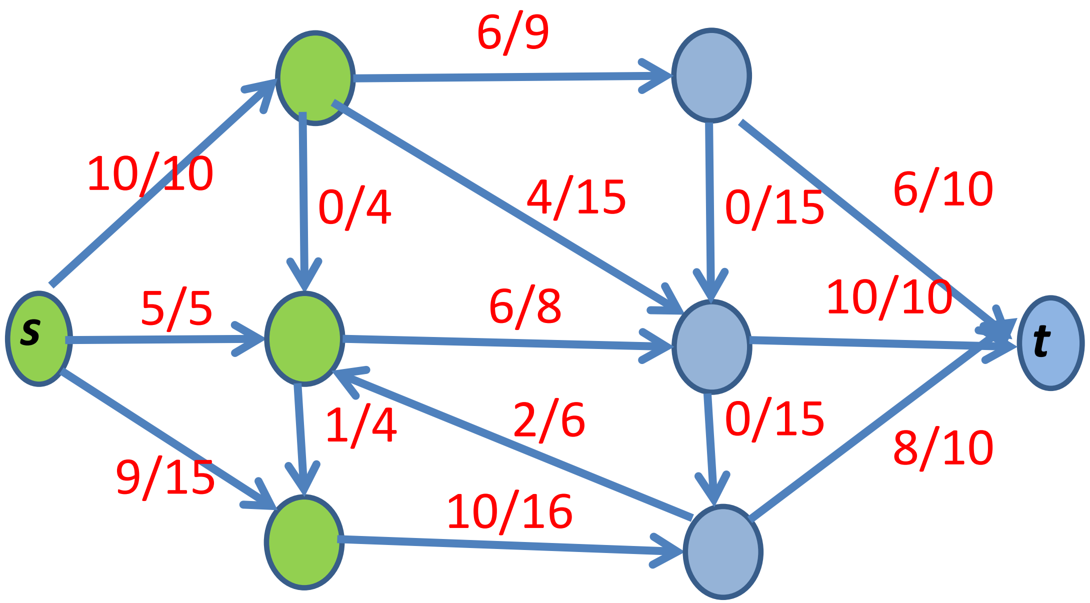
Nettofluss und Kapazität in einem Schnitt

- **Nettofluss** über den Schnitt
  = Summe **Vorwärtsflüsse** – Summe **Rückwärtsflüsse**
- **Kapazität** des Schnitts
  = Summe der Kapazitäten der Vorwärts-Schnittkanten



Zusammenhang zwischen Flusswert und Nettofluss

- **Flusswert-Lemma** *(flow value lemma)* :
  Für jeden Fluss und jeden Schnitt in einem Flussnetzwerk gilt:
  Wert des Flusses = Nettofluss über den Schnitt $\leq$ Kapazität des Schnitts



Zusammenhang zwischen Flüssen und Schnitten

Minimaler Schnitt (*mincut*): Schnitt minimaler Kapazität unter allen Schnitten eines Flussnetzwerks.

- Offensichtlich gilt: $maxflow \leq mincut$
- Wir zeigen: $maxflow = mincut$

Angenommen, der Wert eines Flusses *f* ist *gleich* der Kapazität irgendeines Schnitts.

- *f* nutzt also die Kapazität des Schnitts voll aus

→ Dann ist *f* ein maximaler Fluss und der Schnitt ist minimal

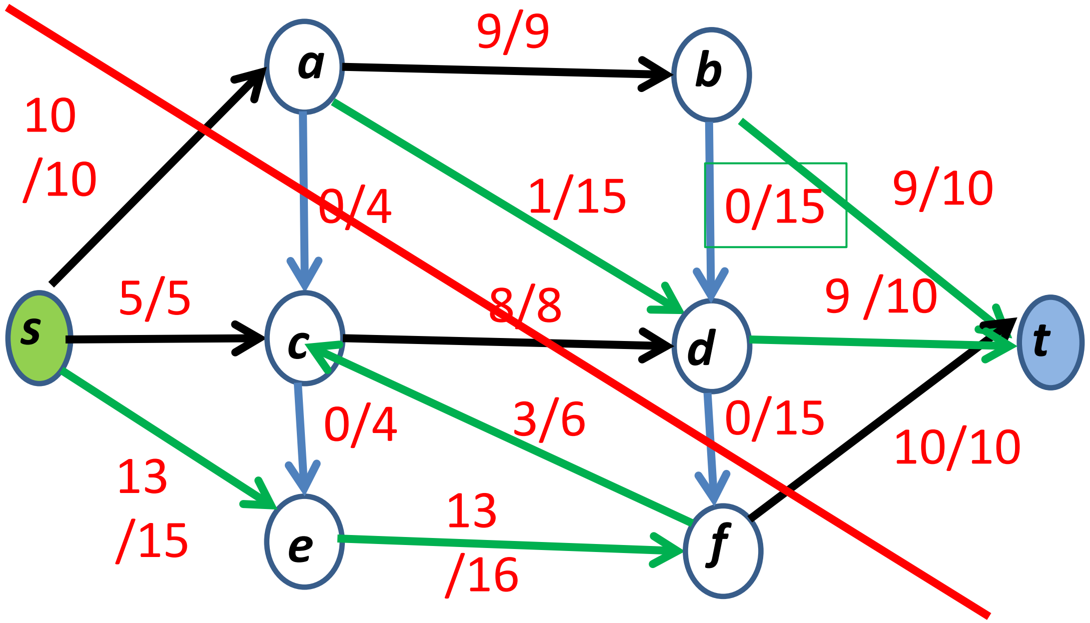
Maxflow-mincut-Theorem

Folgende 3 Bedingungen sind für jeden Fluss *f* in einem Netzwerk äquivalent:

1. *f* ist ein maximaler Fluss
2. Es gibt keinen Erweiterungspfad für *f*
3. Es gibt einen Schnitt, dessen Kapazität gleich dem Wert von f ist.

➔ ***Wenn FF-Algorithmus terminiert, ist maximaler Fluss gefunden***

Beispiel 1 Maximaler Fluss



Schnitt: {s-c-e-f} und {a-b-d-t}

Schnittkanten
- s-a 10/10
- c-d 8/8
- f-t 10/10

Alle voll ausgeschöpft,
➔ minimaler Schnitt und
Maximaler Fluss gefunden

Zusammenfassung Graphen

- Ungerichtete und gerichtete Graphen
- Tiefensuche für Strukturinformationen
- Breitensuche für kürzesten Pfad (Anzahl Kanten)
- Kantengewichtete Graphen
- Minimal Spannende Bäume
- Dijkstra-Algorithmus für kürzeste Wege
- Ford-Fulkerson-Algorithmus für maximalen Fluss