Maximaler Fluß und minimaler Schnitt

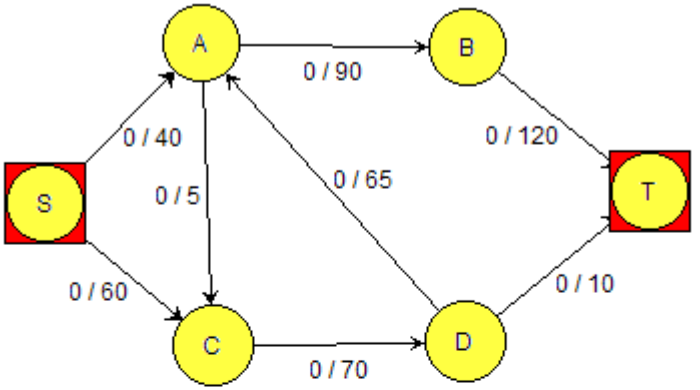
Von Sebastian Thurm
sebastian.thurm@student.uni-magedburg.de

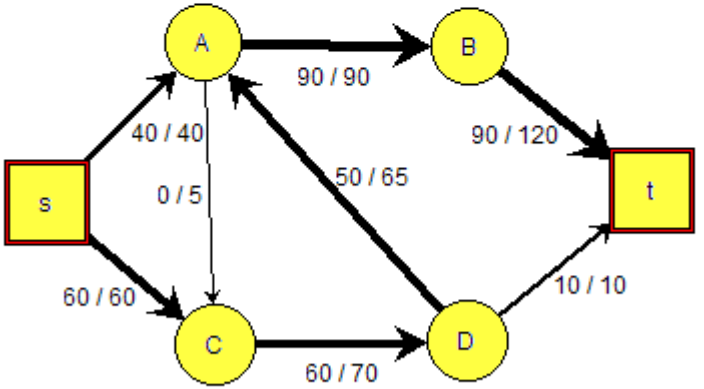
Maximaler Fluß und minimaler Schnitt

- Was$_{\text{ist das}}$?
  - Maximaler Fluss
  - Minimaler Schnitt
- Warum$_{\text{tut man das}}$?
  - Logistische Probleme: Post / Netzwerk / Pipelines
  - Finden von Schwachstellen in Netzwerken
- Wie$_{\text{tut man das}}$?
  - Ford-Fulkerson-Algorithmus
  - MinCut-Algorithmus, Recursive-Contract-Algrithmus
- Zusammenfassung

Was ist der Maximale Fluss ?

- Gewichteter Graph
- Kosten entsprechen maximaler Kapazität
- Gesucht:
  - Gesamtkapazität des Graphen zwischen Start- und Zielpunkt (s und t)

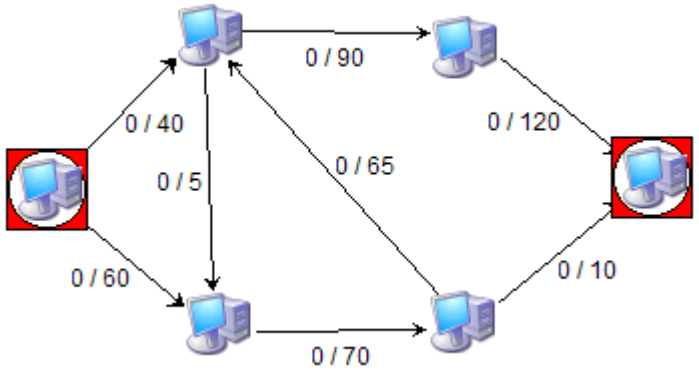


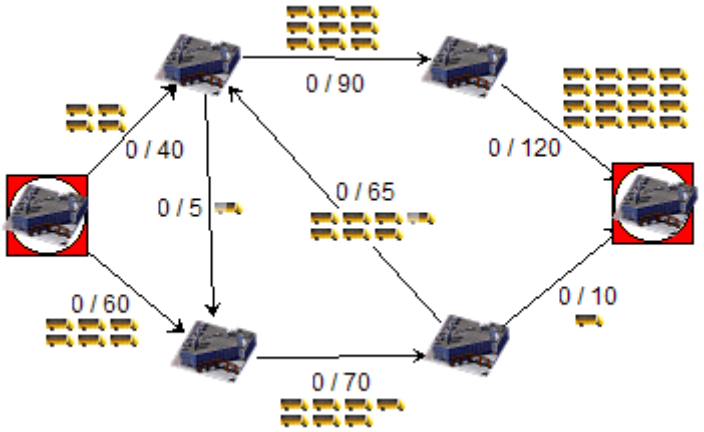


Maximaler Fluss = 40 + 50 +10 = 100

Warum sucht man den Maximalen Fluss ?

- Optimierung von Computernetzwerken
- Logistische Probleme (z.B. Post)





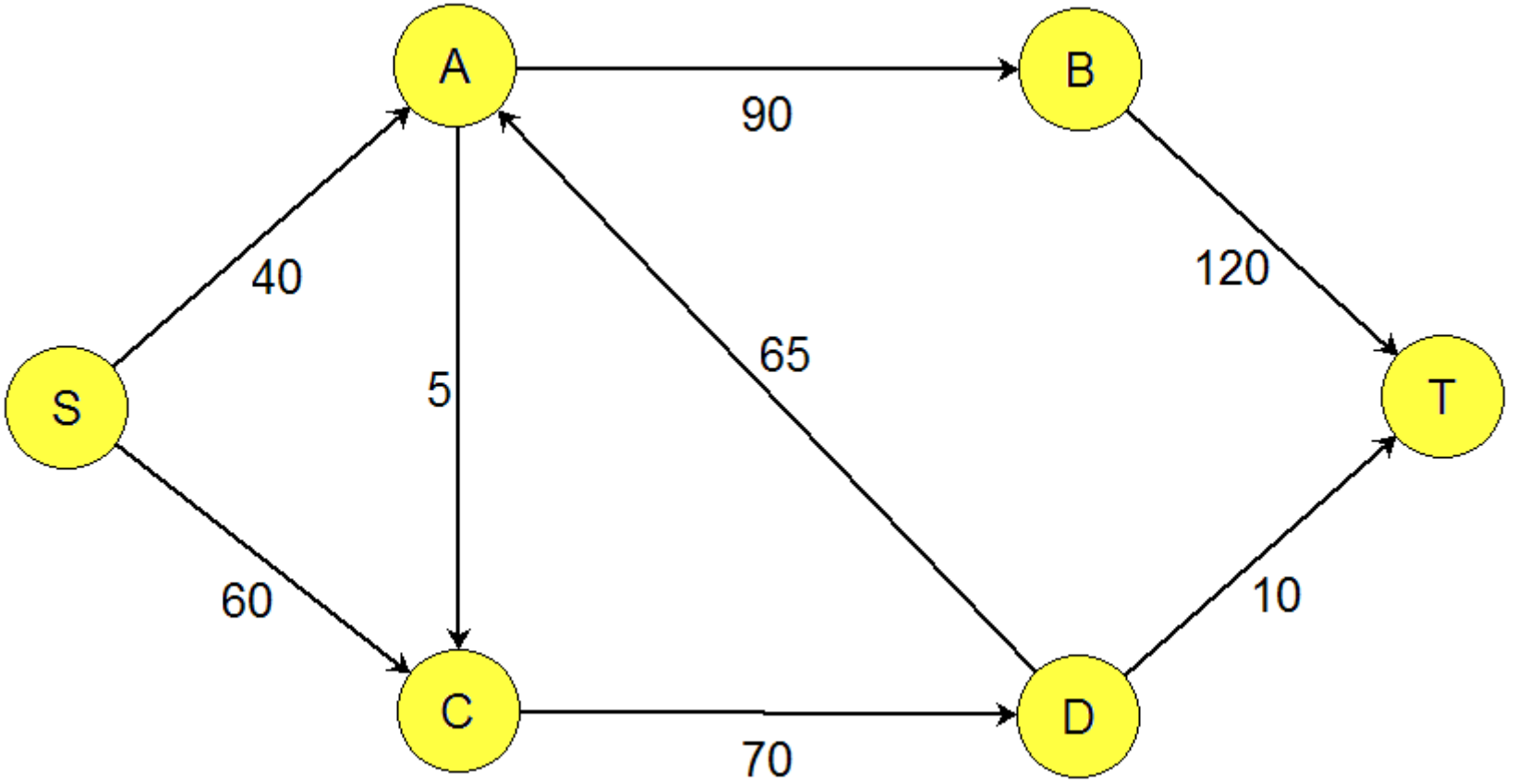
- Schaltungen in der E-Technik
- Röhrensysteme in der Industrie

Wie ermittelt man den Maximalen Fluss ?

- Ford-Fulkerson-Algorithmus (L. R. Ford, D. R. Fulkerson 1962)
  - Eingabe: Gewichteter Graph, Start- und Zielknoten
  - Ausgabe: maximaler Fluss zwischen Start- und Zielknoten
  - do
    - Pfad zwischen Start- und Zielknoten suchen (Breitensuche)
    - Fluss des gefundenen Pfades zum Gesamtfluss hinzufügen
  - bis keine neuen Pfade mehr gefunden werden
  - Ausgabe des Gesamtflusses
- Aufwand: O(n*m) wobei n=Kantenzahl und m=Knotenzahl

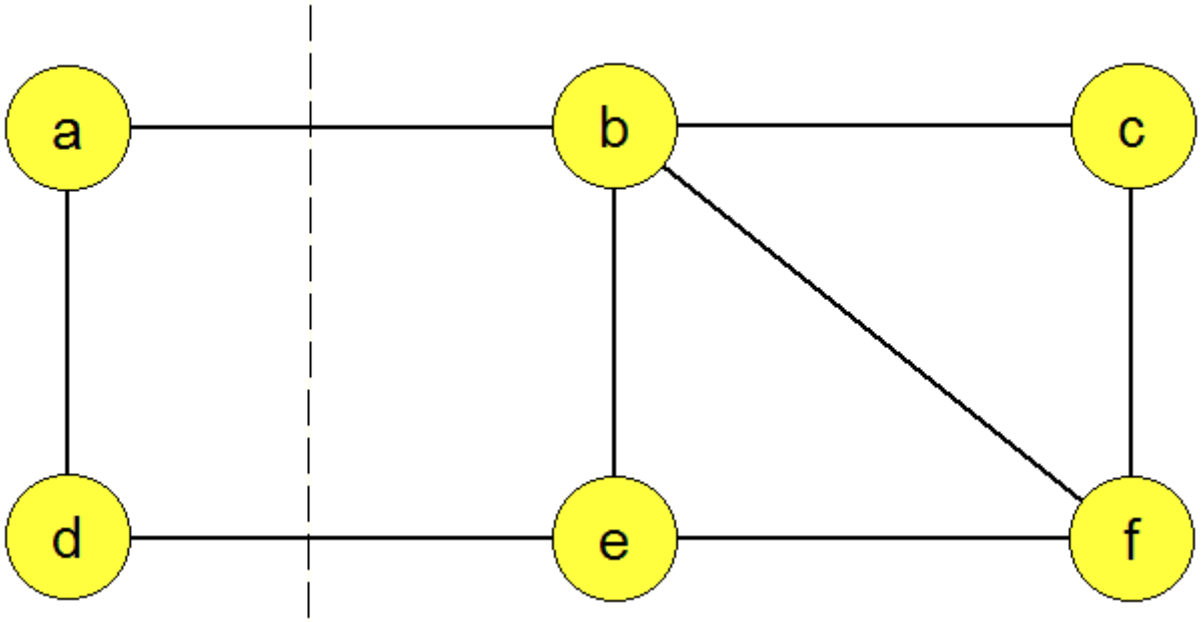
Ford-Fulkerson-Algorithmus am Beispiel

- Eingabe eines Graphen



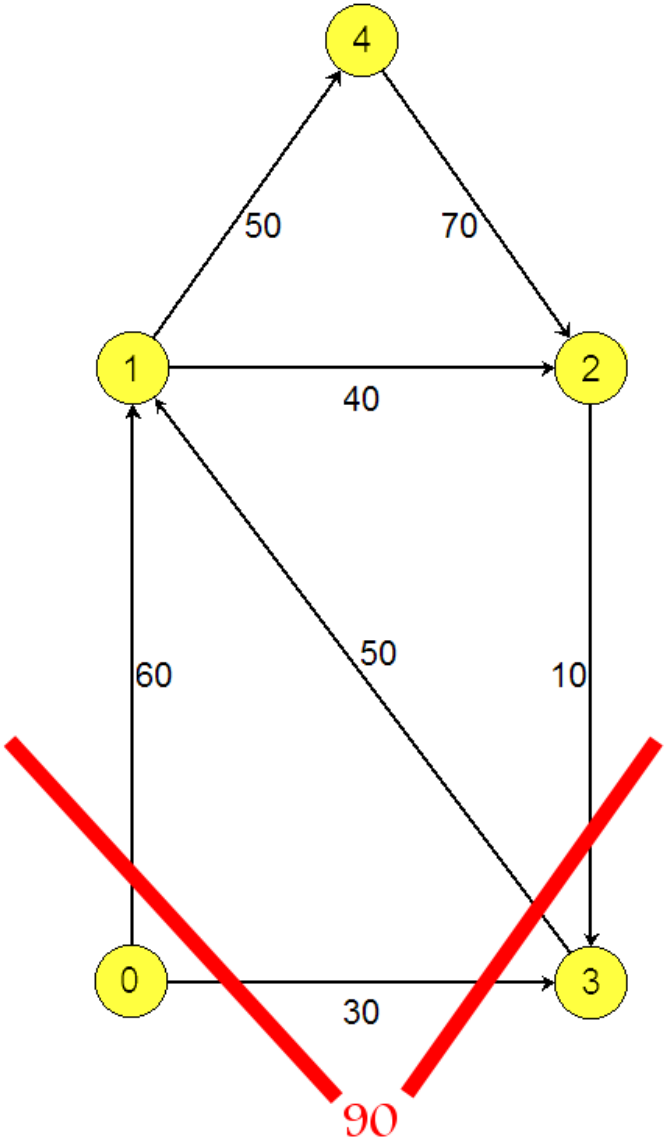
Was ist ein Schnitt ?

- Ein Schnitt teilt die Knotenmenge eines Graphen in zwei nichtleere Partitionen
- Beispielnotation:
  1. Partition {a, d}
  2. Partition {b, c, e, f}
  3. Schnittkanten
     (a, b) und (d, e)
- Finden von beliebigen Schnitten ist trivial



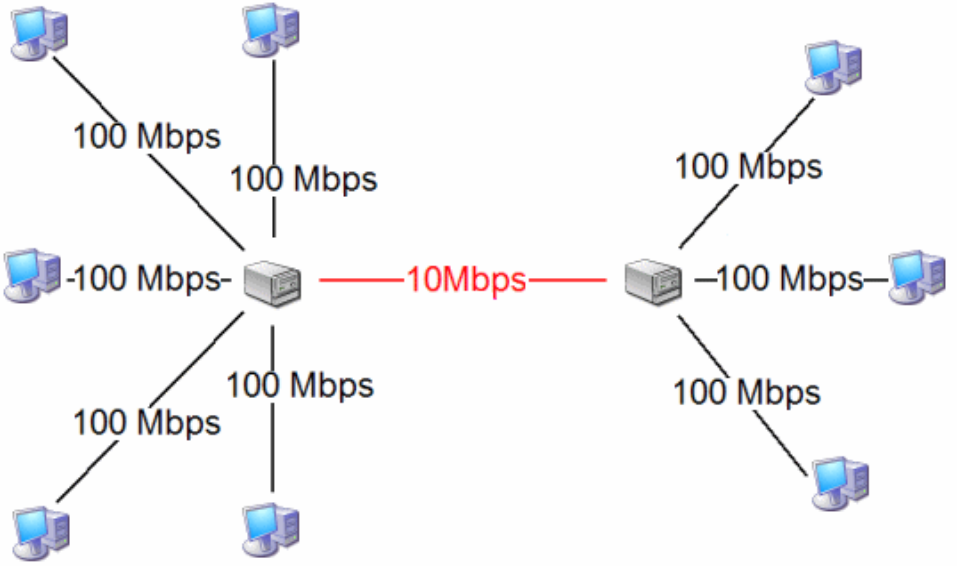
Was ist ein minimaler Schnitt ?

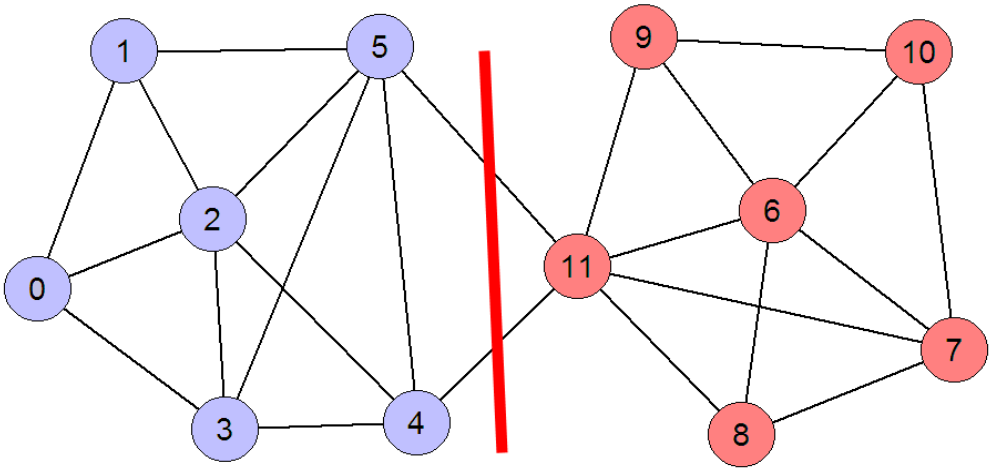
- Schnitt mit dem geringsten Kantengewicht zwischen den Partitionen
- Nicht immer eindeutig



Warum sucht man den minimalen Schnitt ?

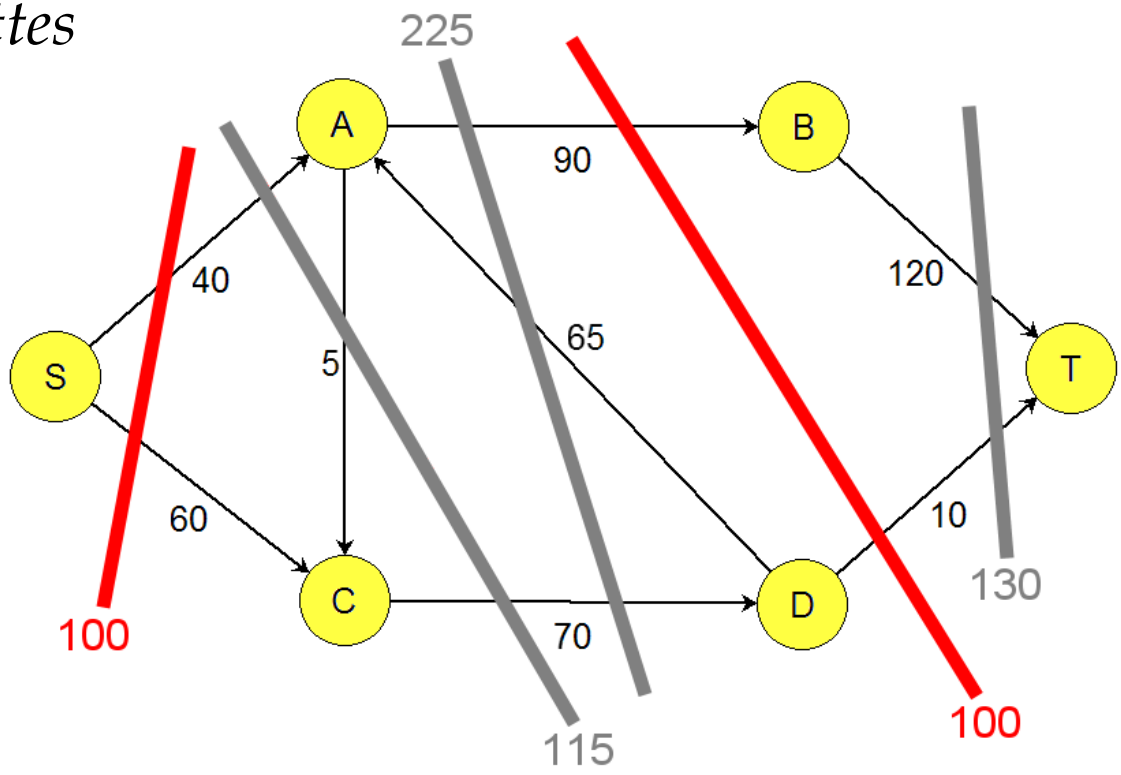
- Anwendungsbeispiele:
  - Auffinden von Schwachstellen in Netzwerken
  - Analyse von Webseiten: kleine Schnitte (wenige Links) -> geringe Themenverwandschaft





Wie findet man den minimalen Schnitt ?

- Ein (brutaler) Ansatz:
  - **Max-Flow-Min-Cut-Theorem** von Ford und Folkerson:
    - Wert eines maximalen Flusses entspricht dem Wert eines minimalen *s-t-Schnittes*
  - Zur Ermittlung eines generellen minimalen Schnittes „einfach“ alle „maximalen Flüsse“ bestimmen
  - Laufzeit quadratisch zur Anzahl der Knoten



Wie findet man den minimalen Schnitt richtig?

- **Algorithmus MinimumCut / MinCut**
  - Deterministisch
  - Laufzeit $O(nm + n^2 \log n)$
  - Findet *sicher* einen minimalen Schnitt
- **Algorithmus RecursiveContract**
  - Randomisiert
  - Laufzeit $O(n^2 \log^3 n)$
  - Findet *alle* minimalen Schnitte mit *sehr hoher Wahrscheinlichkeit*

Der MinimumCut-Algorithmus

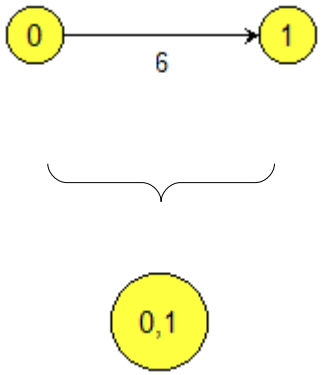
- Eingabe: Gewichteter Graph G, beliebiger Startknoten a
- Ausgabe: minimaler Schnitt des Graphen
- do
  - Prozedur MinimumCutPhase(G, a)
    (gibt CutOfThePhase zurück)
  - MinimumCut = max(MinimumCut, CutOfThePhase)
- bis G nur noch einen Knoten enthält
- Ausgabe des MinimumCut

Die Prozedur MinimumCutPhase

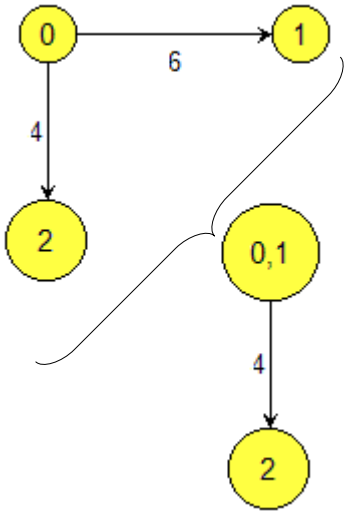
- Eingabe: Gewichteter Graph G, beliebiger Startknoten a
- Ausgabe: CutOfThePhase (möglicher minimaler Schnitt)
- A <- {a}
- do
  - Füge zu A den am engsten verbundenen Knoten aus G hinzu (bzgl. dessen Kantengewichtes)
- bis alle Knoten aus G in A sind
- Kantengewicht zwischen zuletzt zu A hinzugefügtem Knoten und dem „Restgraph“ entspricht CutOfThePhase
- Zusammenfassung der beiden zuletzt zu A hinzugefügten Knoten zu einem neuen Knoten

Zusammenfassen von Knoten

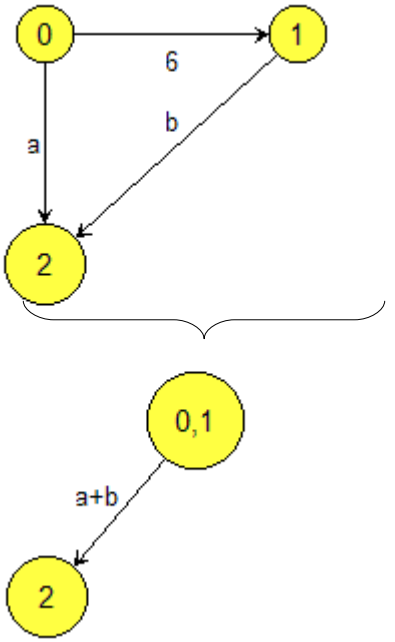
Hier werden die Knoten 0 und 1 zusammengefasst



Kanten zwischen den Knoten werden entfernt

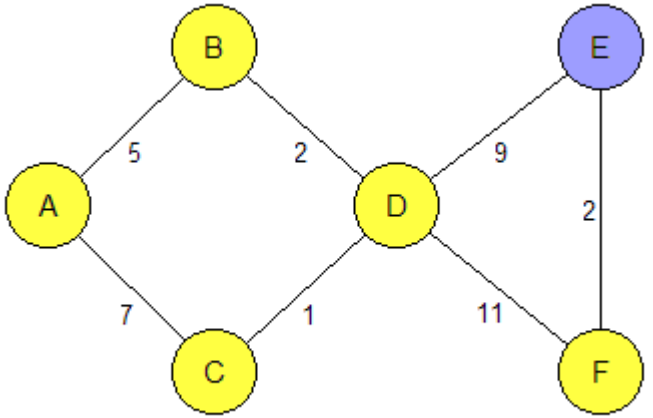


„unbeteiligte“ Kanten bleiben unverändert



„Doppelkanten“ werden zuammengefasst

Ein Beispiel zum MinCut-Algorithmus

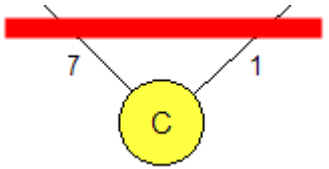


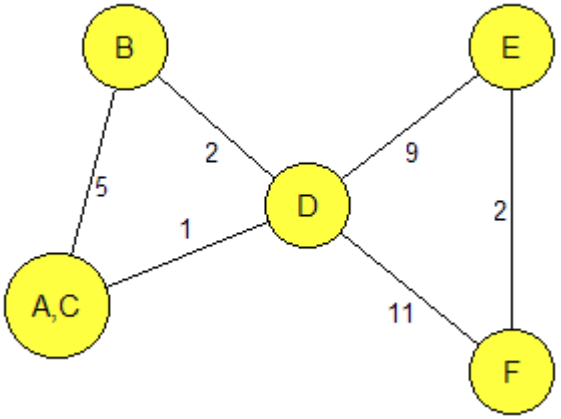
A={E},{D},{F},{B},{A},{C}

CutOfThePhase = 7+1 = 8

merge({A},{C})

MinimumCut = 8



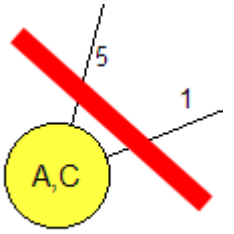


A={E},{D},{F},{B},{A,C}

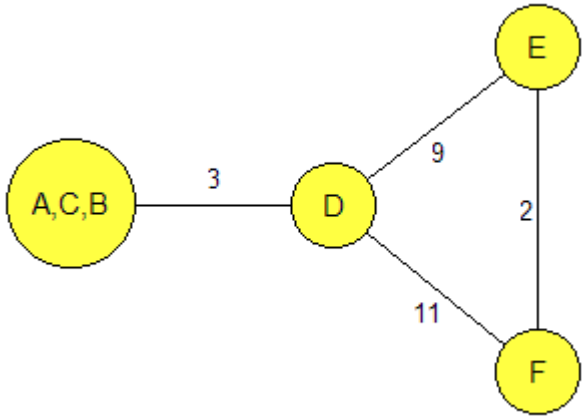
CutOfThePhase = 5+1 = 6

merge({B},{A,C})

MinimumCut = 6



Ein Beispiel zum MinCut-Algorithmus

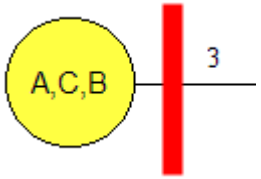


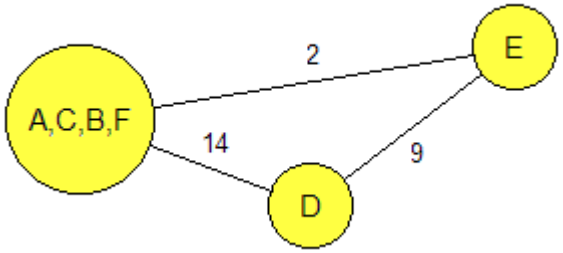
A={E},{D},{F},{A,C,B}

CutOfThePhase = 3

merge({F},{A,C,B})

MinimumCut = 3



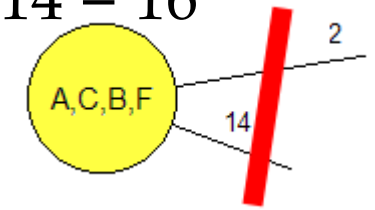


A={E},{D},{A,C,B,F}

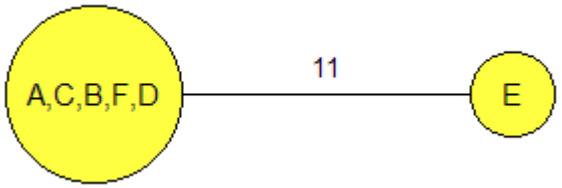
CutOfThePhase = 2 + 14 = 16

merge({D},{A,C,B,F})

MinimumCut bleibt 3



Ein Beispiel zum MinCut-Algorithmus

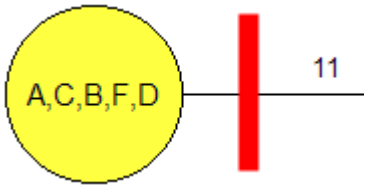


A={E},{A,C,B,F,D}

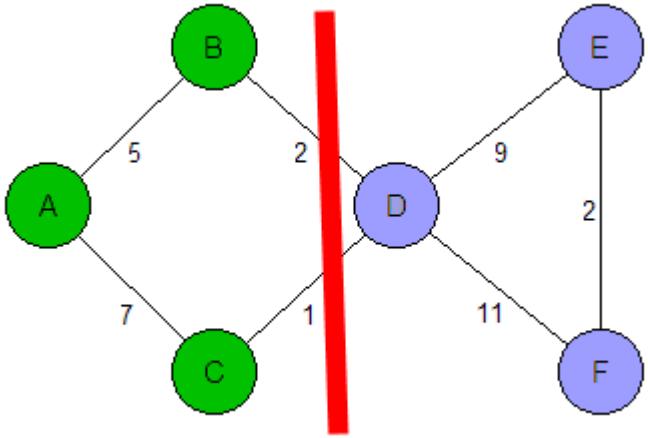
CutOfThePhase = 11

merge({E},{A,C,B,F,D})

MinimumCut bleibt 3



- minimaler Schnitt ist der kleinste aller CutOfThePhase
- also: {A,B,C},{D,E,F} mit dem Betrag 3



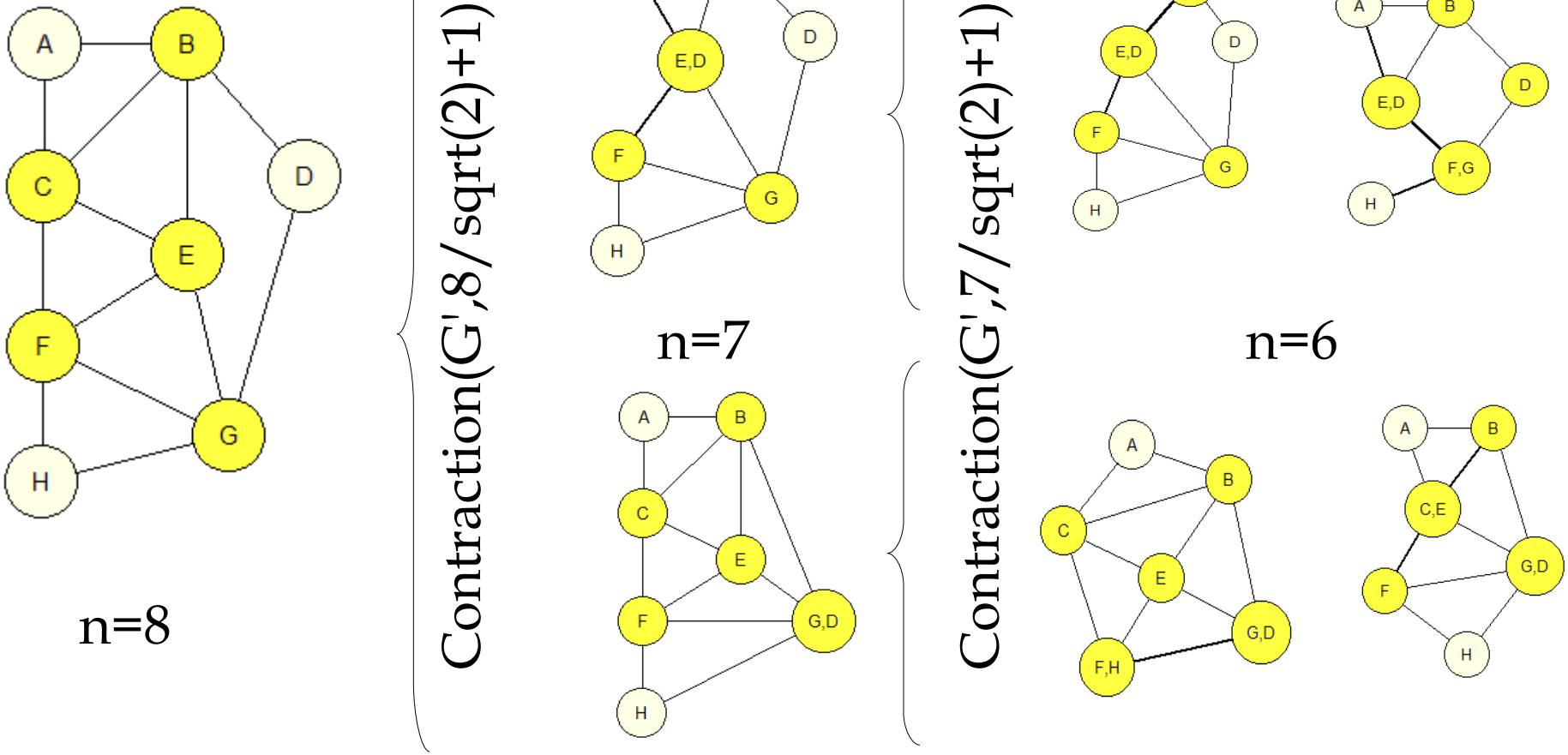
Recursive-Contract: Prozedur Contract(G, n)

- Contract(G, n)
  - Eingabe: Graph G und Zielknotenzahl n
  - Ausgabe: G' mit n Knoten
  - do
    - Wähle zwei beliebige Knoten
    - Füge beide Knoten zu einem neuen zusammen
  - Solange |G| > n
  - Ausgabe: G'

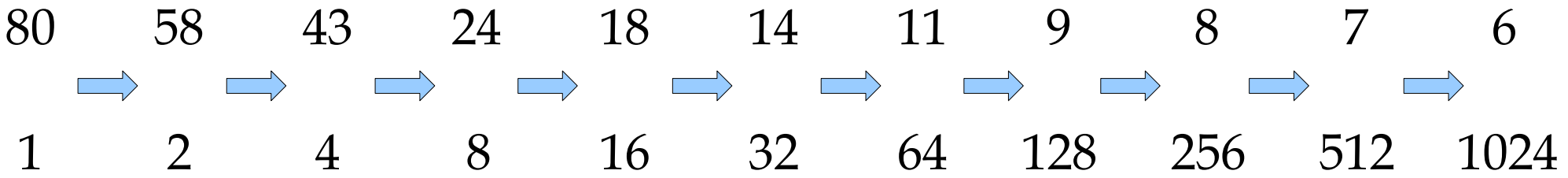
Ein alternativer Ansatz: Recursive-Contract

- Recursive-Contract(G, n)
  - Eingabe: Gewichteter Graph mit n Knoten
  - Falls |G| < 7
    - Contract(G, 2)
    - Ausgabe: Kantengewicht zwischen resultierenden Knoten
  - Sonst *zweimal*
    - G' = Contract(G, ⌈n/sqrt(2)+1⌉)
    - Recursive-Contract(G', |G'|)
  - Ausgabe: Minimum der beiden rekursiven Aufrufe

Ein alternativer Ansatz:
Recursive-Contract

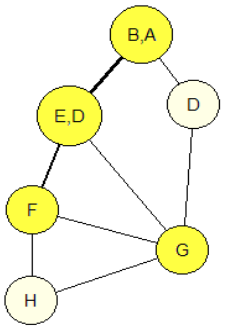


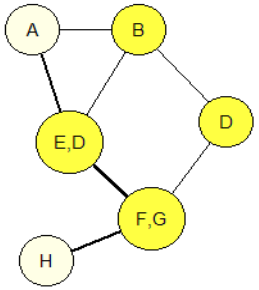
Ein alternativer Ansatz: Recursive-Contract

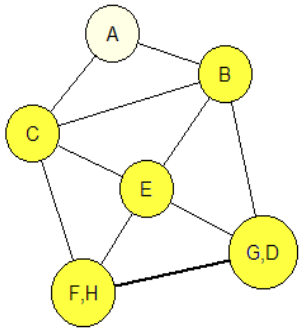


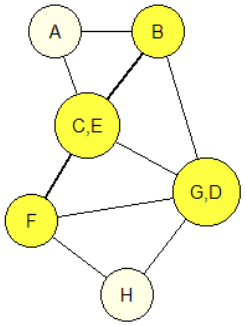
- Aus einem Graphen mit 80 Knoten entstehen 1024 verschiedenen Graphen die auf minimale Schnitte untersucht werden
- starke Chancensteigerung
- hoher Platzverbrauch

Ein alternativer Ansatz:
Recursive-Contract





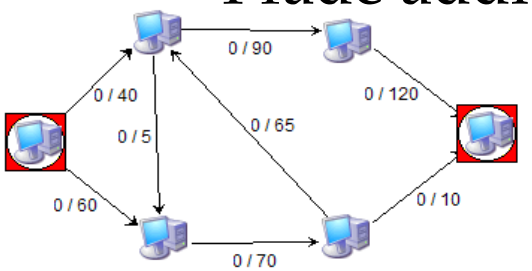




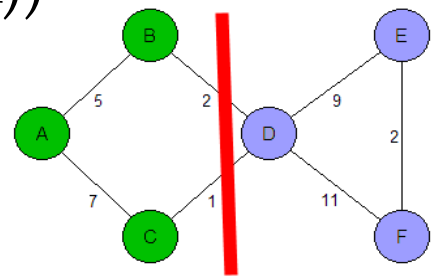
- Gefundene Graphen können nun mit Contract(G, 2) auf minimalen Schnitt untersucht werden
- Minimale Schnitte werden rekursiv verglichen
- Ausgabe des kleinsten gefundenen Schnittes

Zusammenfassung

- maximaler Fluss
  - Maximale Kapazität der Kanten zwischen zwei Knoten
  - Netzwerke, Logistik...
  - Ford-Fulkerson-Algorithmus (O(n*m))
    - Pfade suchen
    - Gesamtfluss aller Pfade addieren



- minimaler Schnitt
  - Zerlegung in zwei Teilgraphen mit minimalen „Schnittkosten“
  - Schwachstellen, Webseiten...
  - MinCut findet sicher einen Schnitt (O(nm + n^2logn))
  - Recursive-Contract findet wahrscheinlich alle Schnitte (O(n^2 $\log^3$ n))



Literatur

- Implementierung und Visualisierung des Algorithmus von Ford-Fulkerson zur Berechnung eines maximalen (s, t)-Flusses in einem Netzwerk
  - http://www.informatik.uni-trier.de/Algorithms/PROJECTS/WS05-06/
- Seminarvortrag Datenstrukturen und Algorithmen
  Professor Hagerup, Professur für Komplexität und Algorithmen
  - http://www.informatik.uni-frankfurt.de/~erps/theoseminarfolien.pdf
  - http://www.informatik.uni-frankfurt.de/~erps/theoseminar.pdf
- Netzwerk-Fluss-Algorithmen
  - http://www.leda-tutorial.org/de/offiziell/ch05s03s04.html