

# Algorithmen II

## Vorlesung am 16.10.2012

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER



## Vorlesung:

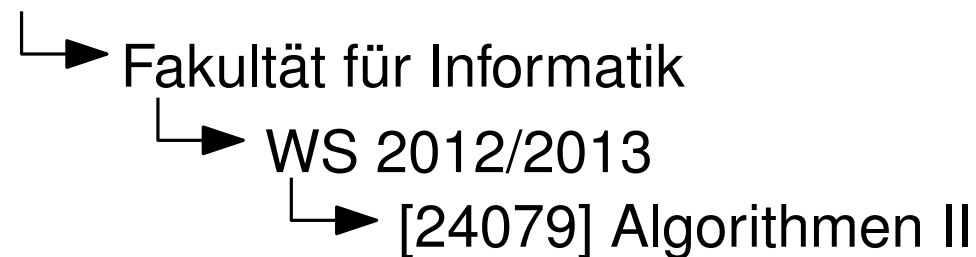
Prof. Dr. Dorothea Wagner

## Übung:

- Thomas Bläsius (`thomas.blaesius@kit.edu`)
- Benjamin Niedermann (`benjamin.niedermann@kit.edu`)
- Sprechzeiten: Termin nach Vereinbarung

# Homepage und Forum

- <http://i11www.iti.kit.edu>
- Aktuelle Informationen/Termine
- Skripte, Folien, Übungsblätter
- Literaturempfehlungen
- Forum
  - Für Fragen an die Übungsleiter.
  - Für Fragen untereinander.
  - erreichbar unter: <https://ilias.studium.kit.edu>



Anmeldung mit Rechenzentrum-account erforderlich

- In der Regel findet jede zweite Woche eine Übung statt.
- Besprechung von Übungsblättern.
- Übungsblätter werden auf der Homepage rechtzeitig online gestellt.
- Übungsblätter können/**sollten** bearbeitet werden.
  - ↳ Helfen den Stoff zu vertiefen + gute Vorbereitung für Klausur

1. Übungsblatt bereits online,  
Besprechung in der 1. Übung am 25. Oktober

- Umfasst zwei Stunden.
- Orientierung: Vorlesung Algorithmen II des Vorjahres + Vorlesung Algorithmentechnik behandelte ähnlichen Stoff.
- Hauptklausur: voraussichtlich am 01.03.2013
- Nachklausur: Termin noch nicht bekannt.

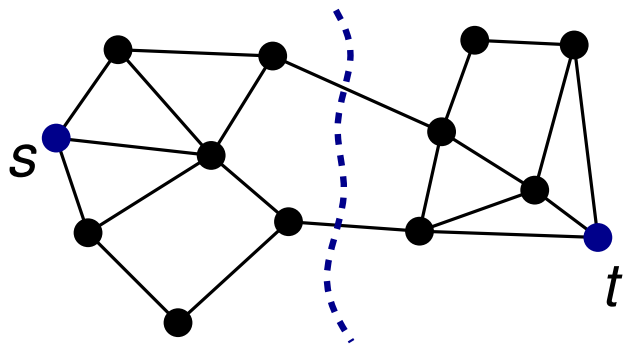
Genauere Klausurtermine werden rechtzeitig bekannt gegeben.

# Ziele der Vorlesung

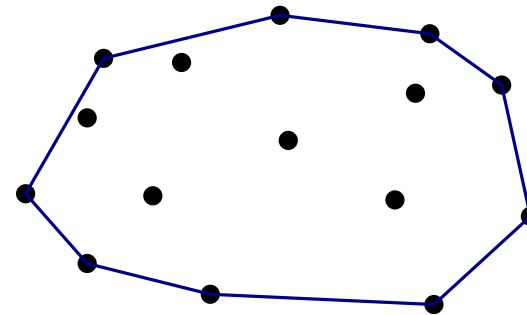
Die Vorlesung soll ein vertieftes Verständnis von Algorithmen vermitteln:

1. Es werden verschiedene Arten von Algorithmen betrachtet, u.a.:

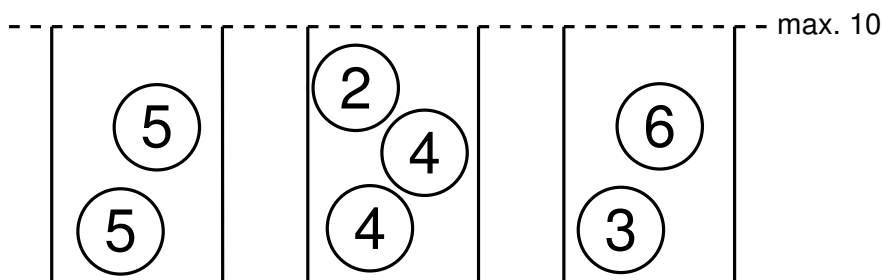
Graphenalgorithmen



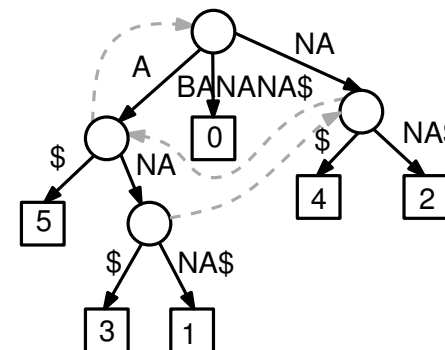
Algorithmische Geometrie



Kombinatorische Optimierung



Algorithmen für fortgeschrittene Datenstrukturen

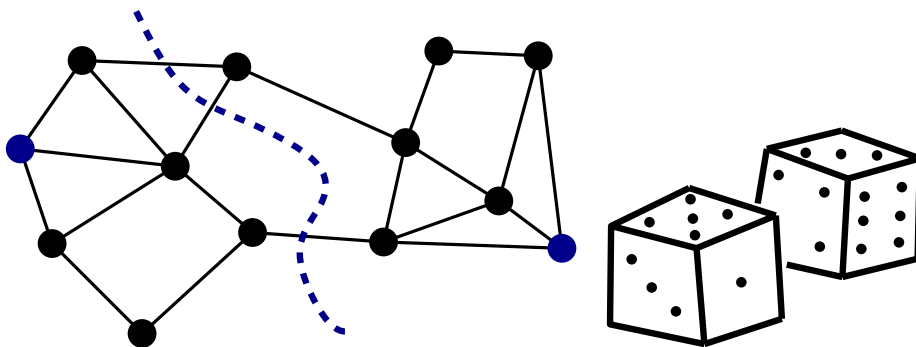


# Ziele der Vorlesung

Die Vorlesung soll ein vertieftes Verständnis von Algorithmen vermitteln:

2. Es werden verschiedene Methodiken betrachtet, u.a.:

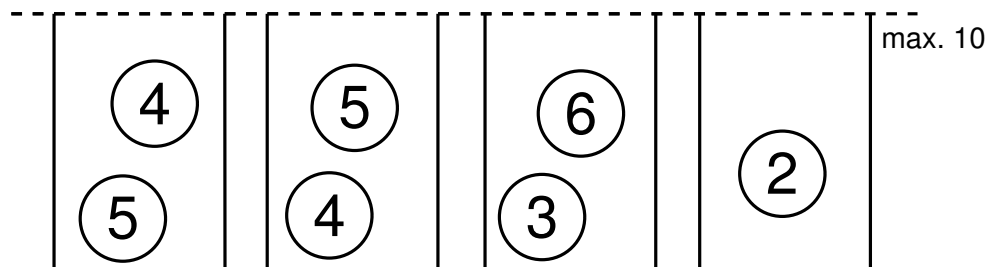
Randomisierte Algorithmen



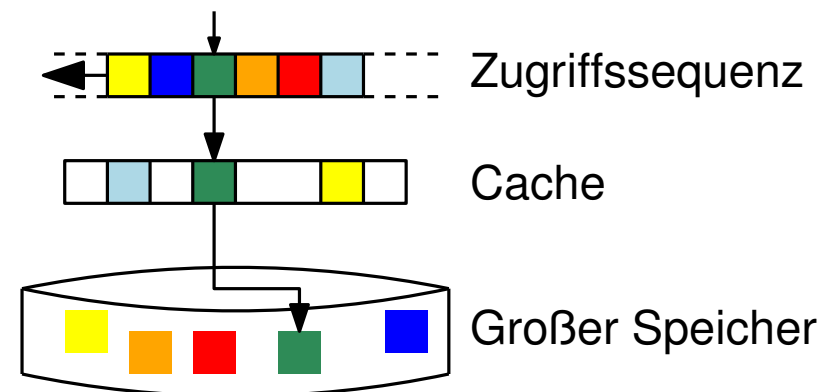
Parallele Algorithmen



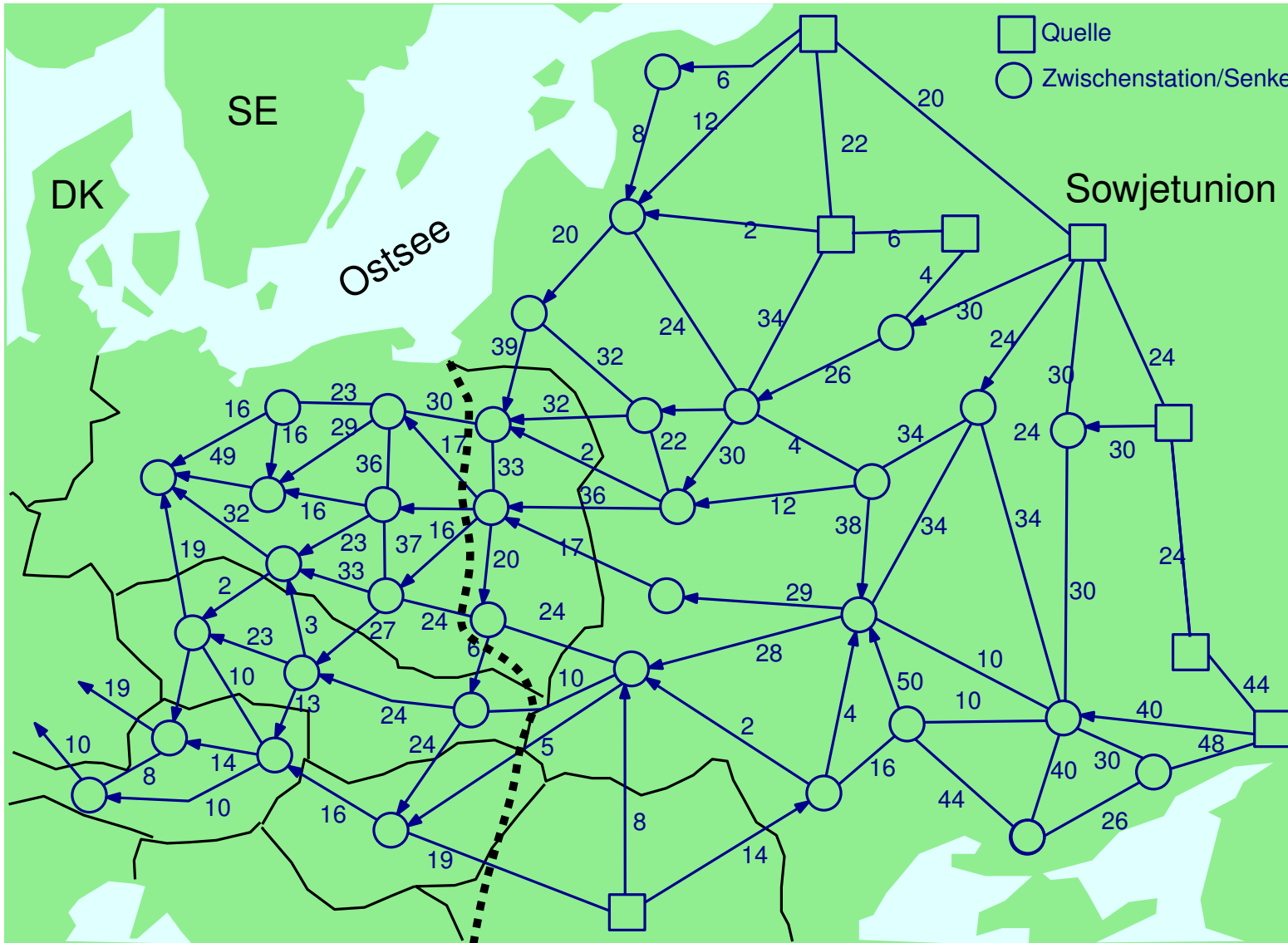
Approximierende Algorithmen



Online Algorithmen



# 1. Beispiel: Transportnetzwerke



Waren optimal durch Netzwerk schicken?

Flaschenhals: Wie berechnen?

Basierend auf "On the history of the transportation and maximum flow problems" von Alexander Schrijver

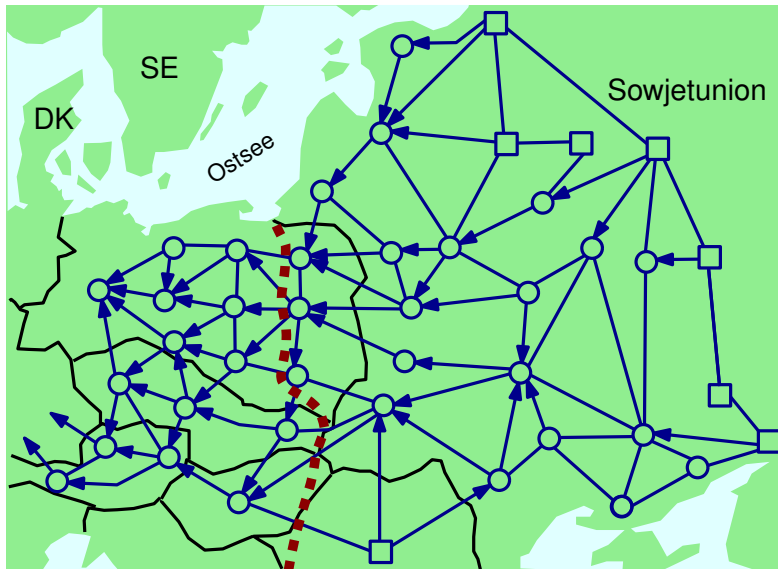


# Wie Waren optimal durch das Netz schicken?

	Arkhangelsk	Yaroslavl'	Murom	Balakhonikha	Dzerzhinsk	Kishert'	Sverdlovsk	Artemovsk	Iledzhk	Dekonskaya	demand:
Agryz				709	1064	693					2
Aleksandrov					397			1180			4
Almaznaya								81		65	1.5
Alchevskaya								106		114	4
Baku								1554		1563	10
Barybino								985		968	2
Berendeevo		135			430						10
Bilimbai						200	59				1
Bobrinskaya								655		663	10
Bologoe		389						1398			1
Verkhov'e								675		661	1

- 1930: Sowjetischer Wissenschaftler Tolstoï versucht Transportproblem auf konkretem Schienennetz zu lösen.
- Erster Ansatz: Ausschließlich zwei Quellen.
- Zweiter Ansatz: Produzenten und Konsumenten liegen entlang einer ringförmigen Bahnstrecke.
- Ergebnis: Findet optimale Lösung, beweist aber nicht, dass die Lösung optimal ist.

# Flaschenhals: Wie berechnen?

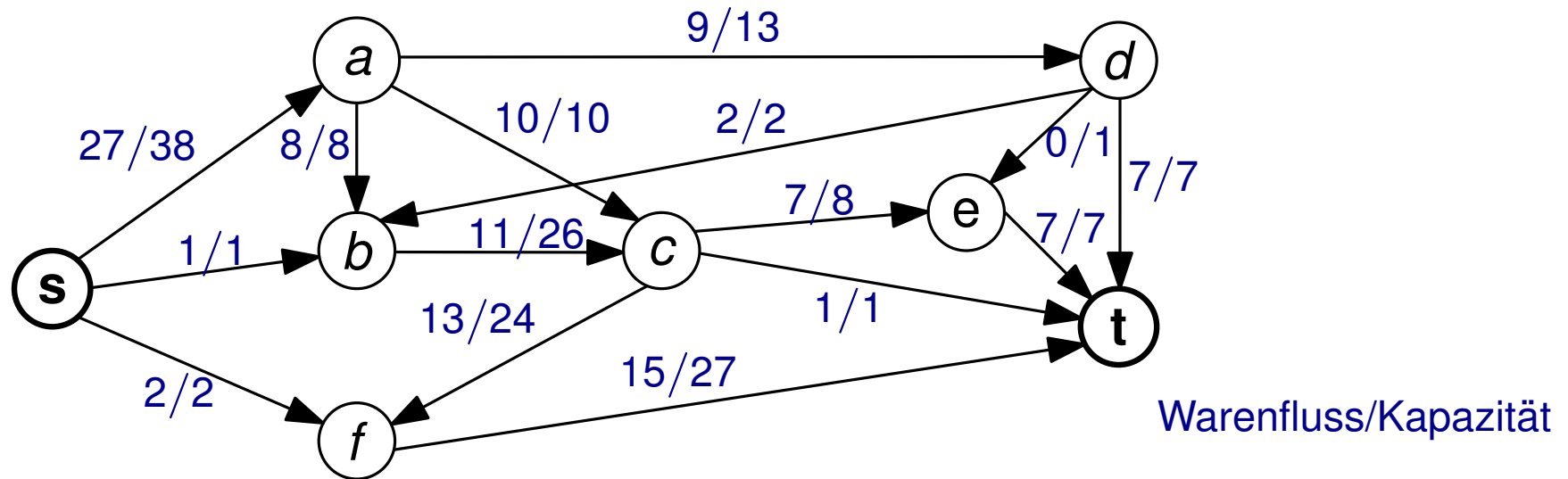


- 1954–1955: Harris and Ross formulieren in *Fundamentals of a Method for Evaluating Rail Net Capacities* Flussproblem in Transportnetzwerken.
- Motivation ist das sowjetische Schienennetz: Finde minimalen Schnitt.

## Problemdefinition von Harris und Ross:

*Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given city to the other.*

1955: Ford und Fulkerson präsentieren allgemeine Lösung für Flussproblem und minimalen Schnitt.



- Vereinfachende Annahme: Ein Produzent (Quelle  $s$ ) und ein Konsument (Senke  $t$ ).
- Jeder Weg zwischen zwei Stationen besitzt eine Kapazität, die der Warenfluss nicht überschreiten darf.
- Was an eine Zwischenstation transportiert wird, muss auch von dort abtransportiert werden.
- Es darf von einer Zwischenstation nicht mehr abtransportiert werden, als dort ankommt.

Modelliere Transportnetzwerk als ein Tupel  $(D, s, t, c)$ , so dass:

- $D = (V, E)$  ist ein einfacher gerichteter Graph.
- $s$  und  $t$  sind Knoten in  $V$  (Quelle und Senke).
- $c : E \rightarrow \mathbb{R}_0^+$  ist Kantengewichtsfunktion.

Modelliere den Warenfluss in  $(D, s, t, c)$  als eine Funktion  $f : E \rightarrow \mathbb{R}_0^+$ , so dass folgende zwei Bedingungen gelten:

## Kapazitätsbedingung:

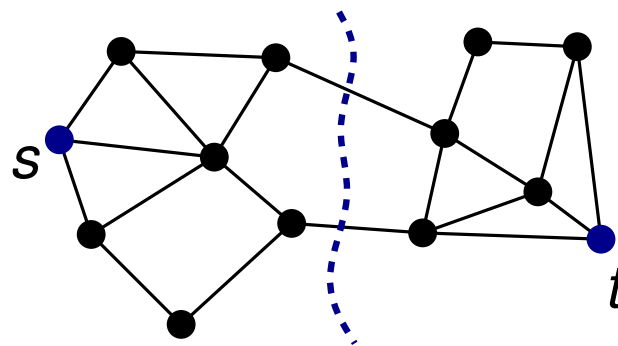
Für alle  $(i, j) \in E$  gilt:  $0 \leq f(i, j) \leq c(i, j)$

## Flusserhaltung:

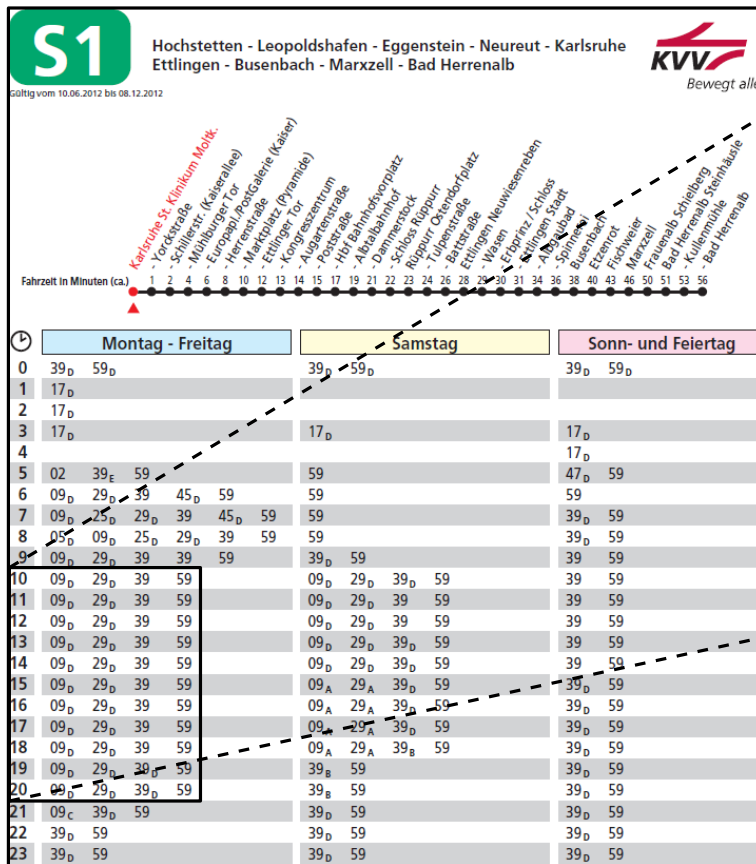
Für alle  $i \in V \setminus \{s, t\}$  gilt: 
$$\sum_{(i,j) \in E} f(i, j) - \sum_{(j,i) \in E} f(j, i) = 0$$

# In der Vorlesung behandelte Fragestellungen

- Wie können maximale Flüsse in einem Netzwerk effizient berechnet werden?
  - Ford-Fulkerson-Algorithmus
  - Algorithmus von Edmonds und Karp
  - Algorithmus von Goldberg und Tarjan
- Wie hängen Flüsse und Schnitte in einem Netzwerk (Graph) zusammen?
- Kann man Flussnetzwerke verwenden, um andere algorithmische Probleme zu lösen?



# 2. Beispiel: Periodische Fahrpläne



Aushangfahrplan der Linie S1 an der Station Städtisches Klinikum

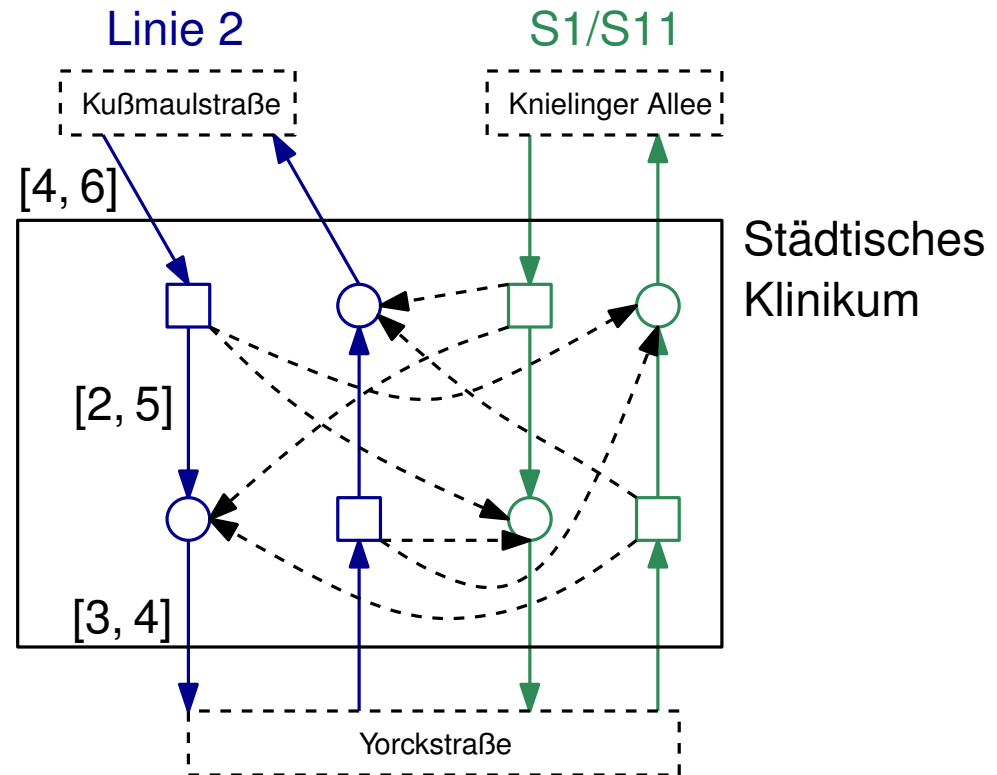
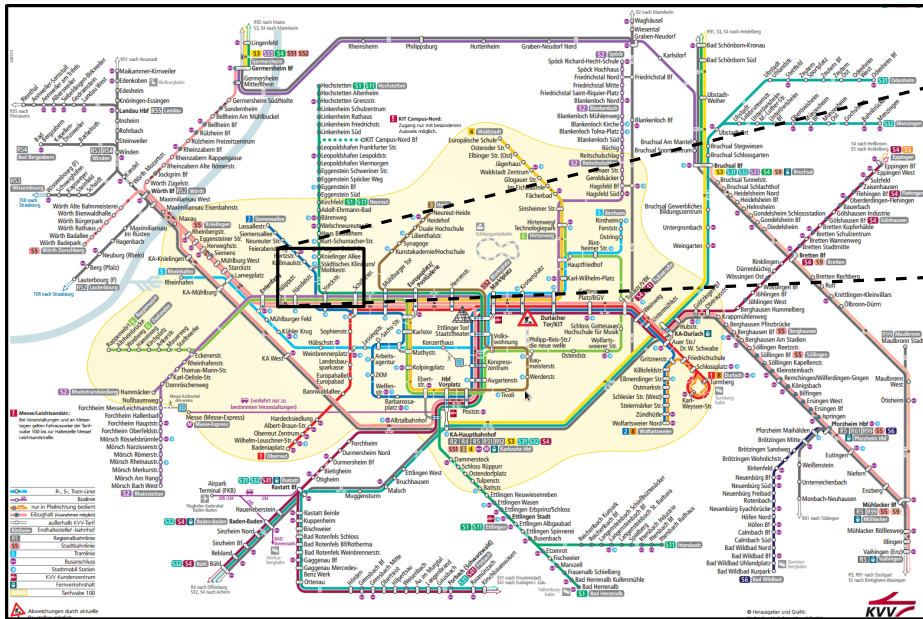
10	09 <sub>D</sub>	29 <sub>D</sub>	39	59
11	09 <sub>D</sub>	29 <sub>D</sub>	39	59
12	09 <sub>D</sub>	29 <sub>D</sub>	39	59
13	09 <sub>D</sub>	29 <sub>D</sub>	39	59
14	09 <sub>D</sub>	29 <sub>D</sub>	39	59
15	09 <sub>D</sub>	29 <sub>D</sub>	39	59
16	09 <sub>D</sub>	29 <sub>D</sub>	39	59
17	09 <sub>D</sub>	29 <sub>D</sub>	39	59
18	09 <sub>D</sub>	29 <sub>D</sub>	39	59
19	09 <sub>D</sub>	29 <sub>D</sub>	39 <sub>D</sub>	59
20	09 <sub>D</sub>	29 <sub>D</sub>	39 <sub>D</sub>	59

Fahrplan ist periodisch

Fragestellung: Wie Fahrplan aller Linien berechnen, so dass

- Aushangfahrpläne periodisch sind,
- kurze Umstiegszeiten zu anderen Linien entstehen,
- Bahnen eine gewisse Aufenthaltszeit pro Station haben,
- und viele weitere Bedingungen.

# Modellierung als Graph



- Für jede Linie führe Kanten und Knoten für Hin- und Rückrichtung ein.
- Für jede Richtung führe Knoten für **Ankunft** und **Abfahrt** ein.
- Führe für mögliche Umstiege Kanten ein.
- Annotiere jede Kante mit Zeitintervall [minimale Dauer, maximale Dauer]

## T-PERIODIC EVENT SCHEDULING PROBLEM

gegeben: gerichteter Graph  $G = (V, A)$  und Vektoren  $l, u \in \mathbb{Q}^{|A|}$ .

gesucht: Vektor  $\pi \in [0, T)^V$ , so dass für jede Kante  $a = (u, v) \in A$  gilt:

$$(\pi_v - \pi_u - l_a) \bmod T \leq u_a - l_a \text{ (oder } \pi_v - \pi_u \in [l_a, u_a]_T)$$

### Erklärung:

- $G$  ist aus Liniennetz extrahierter Graph.
- $T$  ist gewünschte Periode (im Beispiel 10 min).
- $l$  gibt minimale Dauer und  $u$  gibt maximale Dauer der einzelnen Kanten an.
- $\pi$  enthält für jeden Knoten  $v \in V$  einen Zeitpunkt wann Ereignis von  $v$  auftritt (bezüglich der Periode  $T$ ).

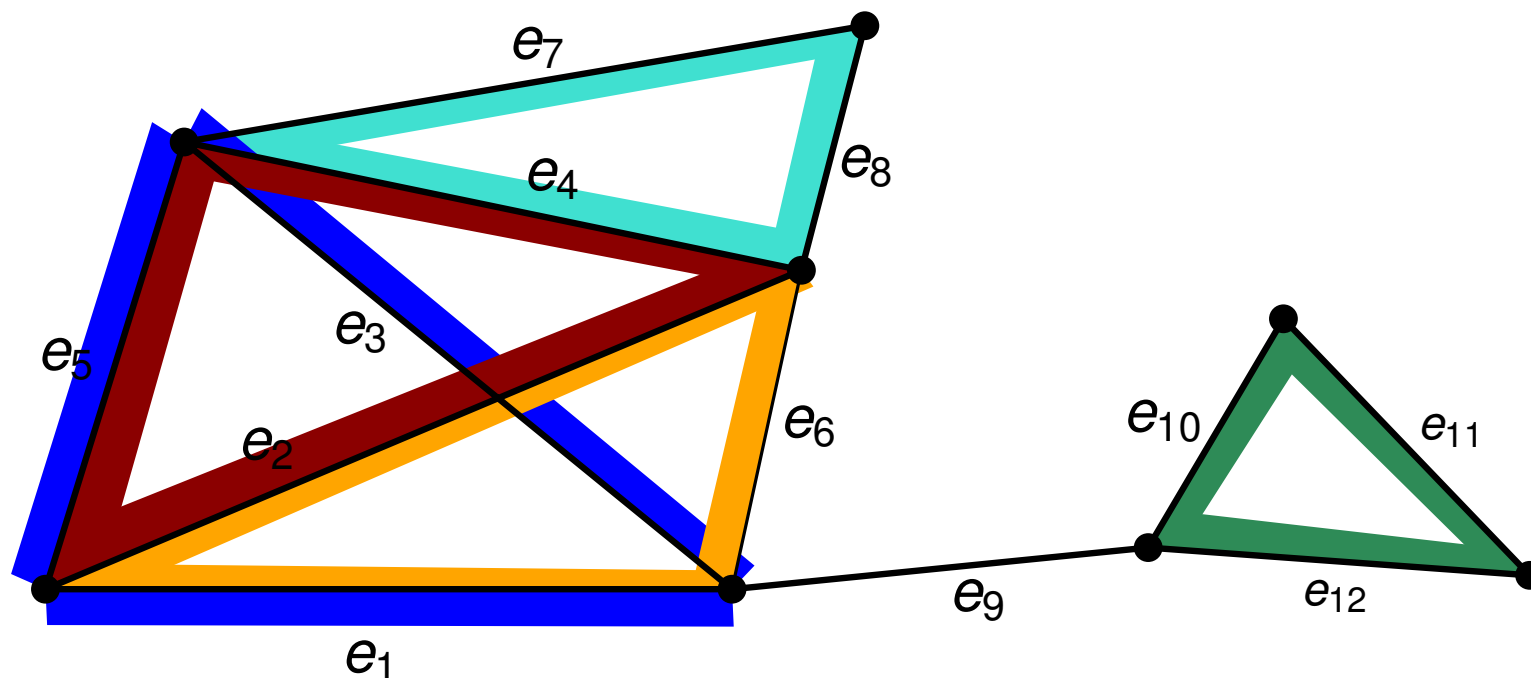


# In der Vorlesung behandeltes Problem

Lösungsansätze des Periodic Event Scheduling Problem werden nicht direkt behandelt, sondern nur ein Teilproblem:

Ein Teilgraph  $C = (V_C, E_C)$  von  $G = (V, E)$  heißt **Kreis** in  $G$ , falls alle Knoten aus  $V_C$  in  $C$  geraden Grad haben.

**Problemstellung:** Finde minimal große Menge  $\mathcal{C}$  an Kreisen in  $G$ , so dass sich alle Kreise in  $G$  aus  $\mathcal{C}$  zusammensetzen lassen.



### 3. Beispiel: Volltextsuche

B	l	a	u	k	r	a	u	t		b	l	e	i	b	t		B	l	a	u	k	r	a	u	t		u	n	d
B	r	a	u	t	k	l	e	i	d		b	l	e	i	b	t		B	r	a	u	t	k	l	e	i	d		

Gegeben: Zwei Folgen  $T$  und  $P$  von Zeichen mit Länge  $n$  und  $m$  ( $m \leq n$ ).

Gesucht: Alle Vorkommen von  $P$  in  $T$ .

1. Beobachtung: Ohne Vorwissen über  $P$  und  $T$  benötigt jeder Algorithmus  $\Omega(m + n)$  Zeit:

- Alle Zeichen von  $T$  müssen mindestens einmal betrachtet werden  $\rightarrow \Omega(n)$ .
- Alle Zeichen von  $P$  müssen mindestens einmal betrachtet werden  $\rightarrow \Omega(m)$ .

## Matching(Text $T$ , Muster $P$ )

1. Setze  $n =$  Länge von  $T$  und setze  $m =$  Länge von  $P$ .
2. Für  $i = 0$  bis  $n - m$  führe aus
  - (a) Falls  $P[1 \dots m] = T[i + 1 \dots i + m]$ , dann  
gebe aus: Muster  $P$  taucht mit Verschiebung  $i$  in  $T$  auf.

## Analyse:

- Schleife benötigt  $n - m$  Schritte.
- Vergleich in Schleife benötigt  $m$  Schritte

Algorithmus benötigt  $\Theta((n - m + 1) \cdot m)$

Hinweis: Trotz schlechter theoretischer Laufzeit, in der Praxis nahezu lineare Laufzeit: In natürlichsprachlichen Texten kann der Vergleich  $P[1 \dots m] = T[i + 1 \dots i + m]$  häufig bereits nach den ersten zwei Zeichen abgebrochen werden. (In der englischen Sprache im Durchschnitt nach 1.07 Zeichen.)

# In der Vorlesung behandelte Algorithmen

Idee: Investiere Zeit in Vorberchnungsschritt, um bei Anfrage Zeit zu sparen.

	Vorbereitungszeit	Suchzeit	typische Anwendung
Naiver Ansatz	—	$\Theta((n - m + 1) \cdot m)$	A B
Rabin-Karp-Algorithmus	$\Theta(m)$	average $\Theta(n + m)$ worst $\Theta((n - m + 1) \cdot m)$	A
Endlicher Automat	$\mathcal{O}(m \cdot  \Sigma )$	$\Theta(n)$	
Knuth-Morris-Pratt	$\Theta(m)$	$\Theta(n)$	
Suffix Trees	$\Theta(n)$	$\Theta(m)$	B

A: Ein Muster, verschiedene Texte.

B: Ein Text, verschiedene Muster.

$\Sigma$  = Alphabet der Zeichenfolgen

## 4. Beispiel: Knotenüberdeckung

### **Definition:** Knotenüberdeckung (Vertex Cover)

Gegeben ein Graph  $G = (V, E)$ . Eine *Knotenüberdeckung*  $S \subseteq V$  ist eine Teilmenge von  $V$ , so dass

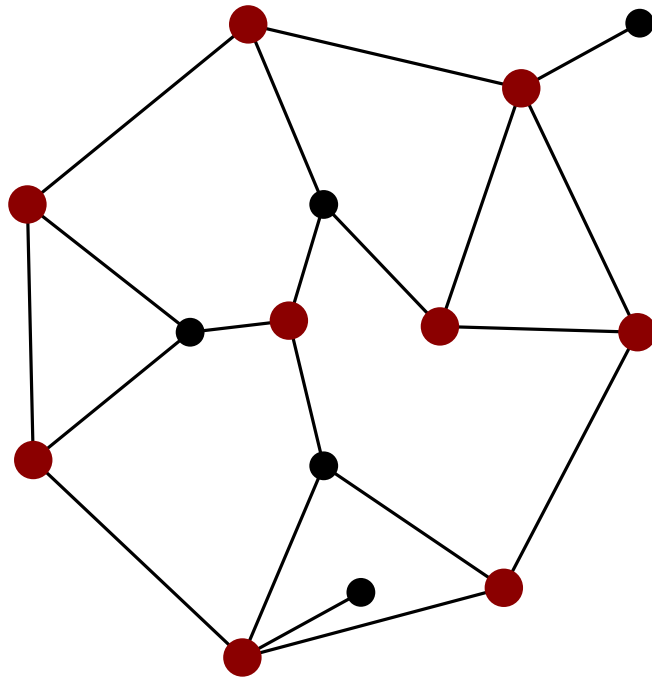
für alle  $\{u, v\} \in E$  gilt  $u \in S$  oder  $v \in S$ .

### VERTEX COVER PROBLEM

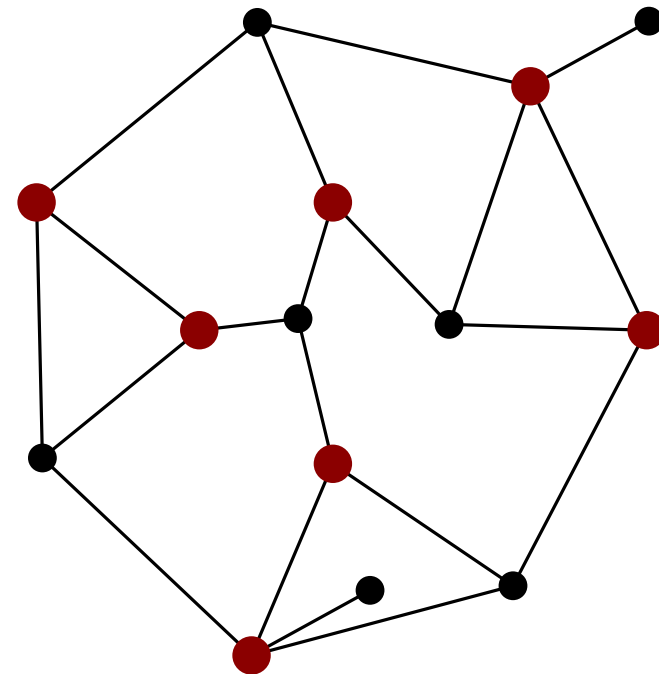
Gegeben ein Graph  $G = (V, E)$ . Gibt es eine Knotenüberdeckung  $S$  mit maximal  $k$  Knoten?

Problem ist  $\mathcal{NP}$ -vollständig.

# Knotenüberdeckung



nicht minimal



minimal