Karlsruher Institut für Technologie

Praktikum Routenplanung

Vorbesprechung, Wintersemester 2014/2015

Moritz Baum, Julian Dibbelt, Ben Strasser | 22. Oktober 2014

Institut für Theoretische Informatik · Algorithmik · Prof. Dr. Dorothea Wagner

Organisatorisches

Praktikum

- Erste Phase: 3 Übungsblätter einzeln lösen
- Zweite Phase: Große Aufgabe in 3er-Gruppen
- Betreuer: Moritz Baum, Julian Dibbelt, Ben Strasser
- Credits: 6 ETCS (4 SWS)
- Email: {moritz.baum, dibbelt, strasser}@kit.edu
- Sprechstunde in Raum 318 und 322
- Bei Fragen kommt einfach vorbei

Homepage: `http://i11www.iti.kit.edu/teaching/winter2014/algorithmengineeringpraktikum/`
Google: uni karlsruhe wagner -> Lehre -> Praktikum

Organisatorisches

Voraussetzungen

- Ihr seid im Master Informatik[1]
- Ihr habt Interesse an algorithmischen Fragestellungen
- Ihr mögt Algorithmen implementieren
- Ihr könnt C++

Anmeldung

- Verbindliche Anmeldung sobald Gruppenarbeit beginnt

[1] Alle anderen bitte melden.

Problemstellung

Gesucht:

- finde die beste Verbindung in einem Transportnetzwerk

Idee:

- Netzwerk als Graphen $G = (V, E)$
- Pfad durch Graph entspricht Route
- klassisches Problem (Dijkstra)

Probleme:

- Transportnetzwerke sind groß
- Dijkstra zu langsam ($>$ 1 Sekunde)

Problemstellung

Gesucht:

- finde die beste Verbindung in einem Transportnetzwerk

Idee:

- Netzwerk als Graphen $G = (V, E)$
- Pfad durch Graph entspricht Route
- klassisches Problem (Dijkstra)

Probleme:

- Transportnetzwerke sind groß
- Dijkstra zu langsam ($>$ 1 Sekunde)

Beschleunigungstechniken

Beobachtungen:

- viele Anfragen in (statischem) Netzwerk
- manche Berechnungen scheinen unnötig

Idee:

- Zwei-Phasen Algorithmus:
  - offline: berechne Zusatzinformation während Vorberechnung
  - online: beschleunige Berechung mit diesen Zusatzinformationen
- drei Kriterien:
  - wenig Zusatzinformation
  - kurze Vorberechnung (im Bereich Stunden/Minuten)
  - hohe Beschleunigung

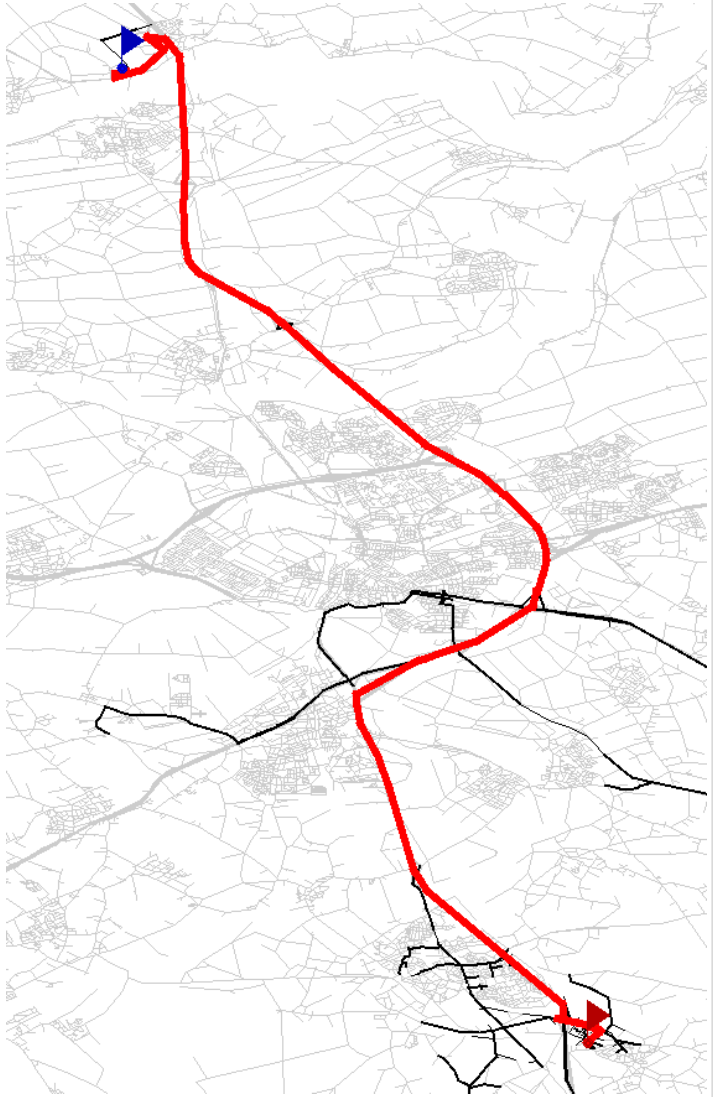
Beschleunigungstechniken

Beobachtungen:

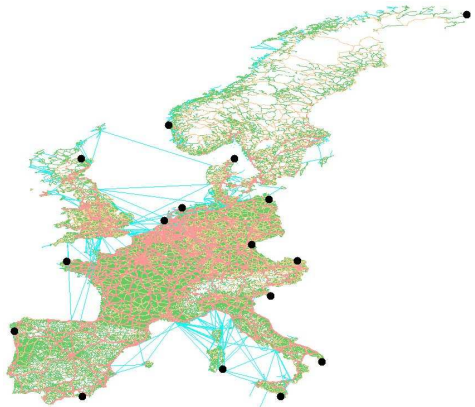
- viele Anfragen in (statischem) Netzwerk
- manche Berechnungen scheinen unnötig

Idee:

- Zwei-Phasen Algorithmus:
  - offline: berechne Zusatzinformation während Vorberechnung
  - online: beschleunige Berechung mit diesen Zusatzinformationen
- drei Kriterien:
  - wenig Zusatzinformation
  - kurze Vorberechnung (im Bereich Stunden/Minuten)
  - hohe Beschleunigung



Ergebnisse

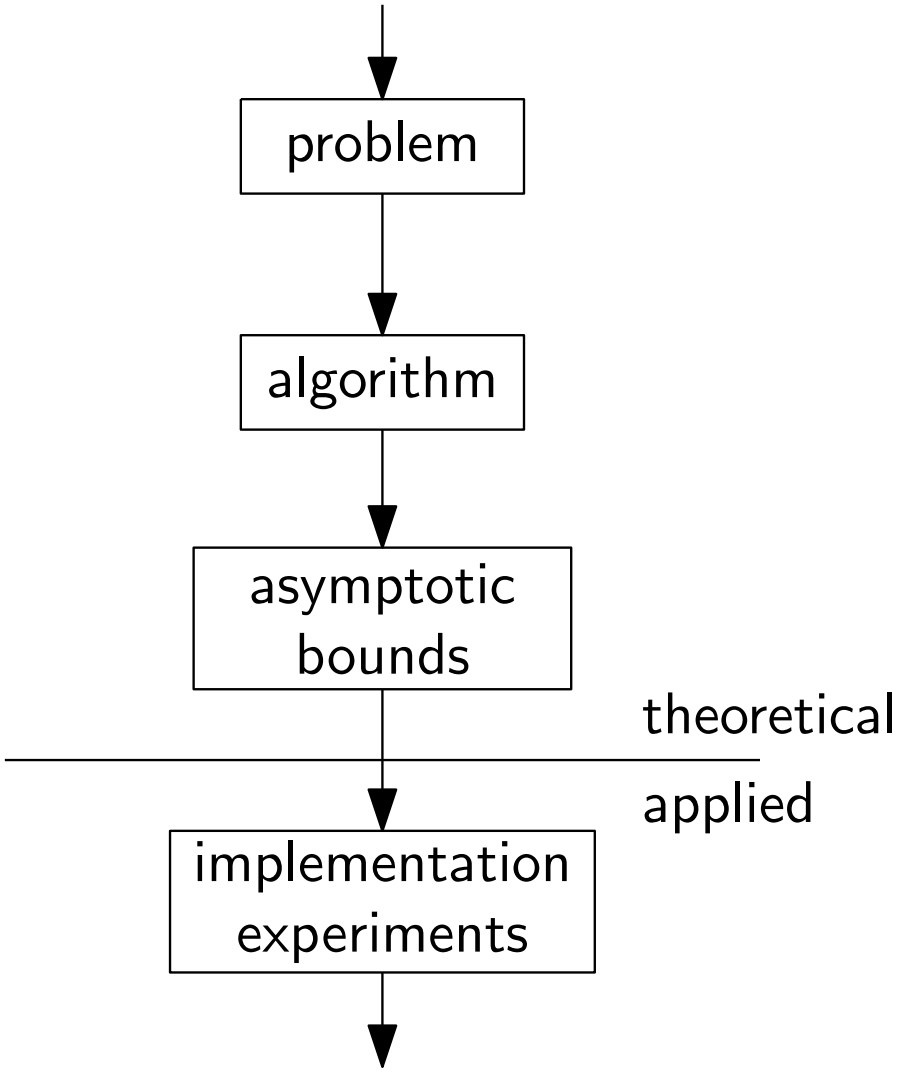


Eingabe: Straßennetzwerk von Westeuropa

- 18 Mio. Knoten
- 42 Mio. Kanten

		Vorberechnung		Anfrage	
	Jahr	Zeit [h:m]	Platz [byte/n]	Zeit [ms]	Beschl.
Dijkstra	1959	0:00	0	5 153.0	0
Arc-Flags	2004	17:08	19	1.6	3 221
Transit-Node Routing	2006	1:15	226	0.0043	1.2 Mio.
Contraction Hier.	2008	0:29	≤ 4	0.19	27 121
Hub Labels	2011	$\approx$ 4:30	1 241	$\approx$ 0.0005	ca. 11 Mio.

Klassischer Ansatz



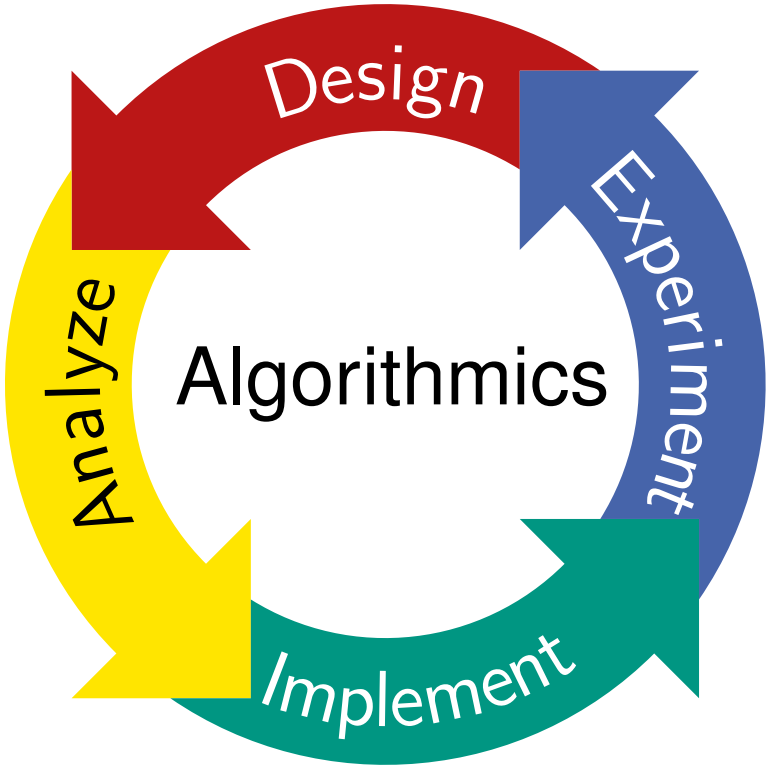
Lücke Theorie vs. Praxis

Theorie	vs.	Praxis
einfach	Problem-Modell	komplex
einfach	Maschinenmodell	komplex
komplex	Algorithmen	einfach
fortgeschritten	Datenstrukturen	einfach
worst-case	Komplexitäts-Messung	typische Eingaben
asymptotisch	Effizienz	konstante Faktoren

hier:

- sehr anwendungsnahes Gebiet
- Eingaben sind echte Daten
  - Straßengraphen
  - Eisenbahn (Fahrpläne)
  - Flugpläne

Algorithm Engineering

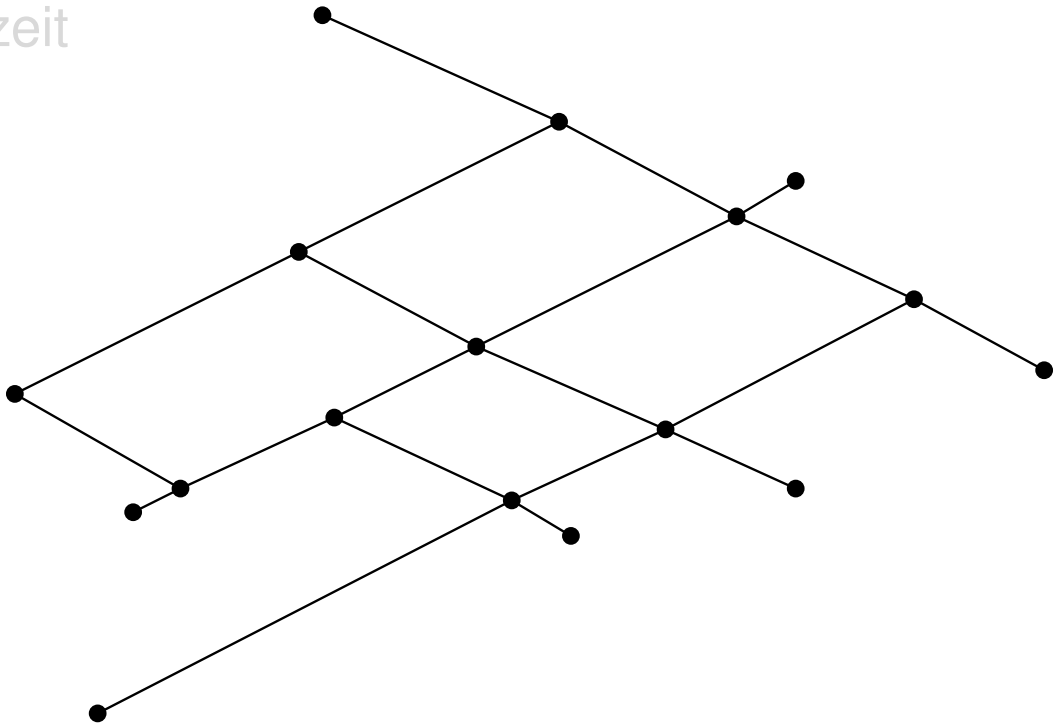


Modellierung (Straßengraphen)

Modellierung (Straßengraphen)

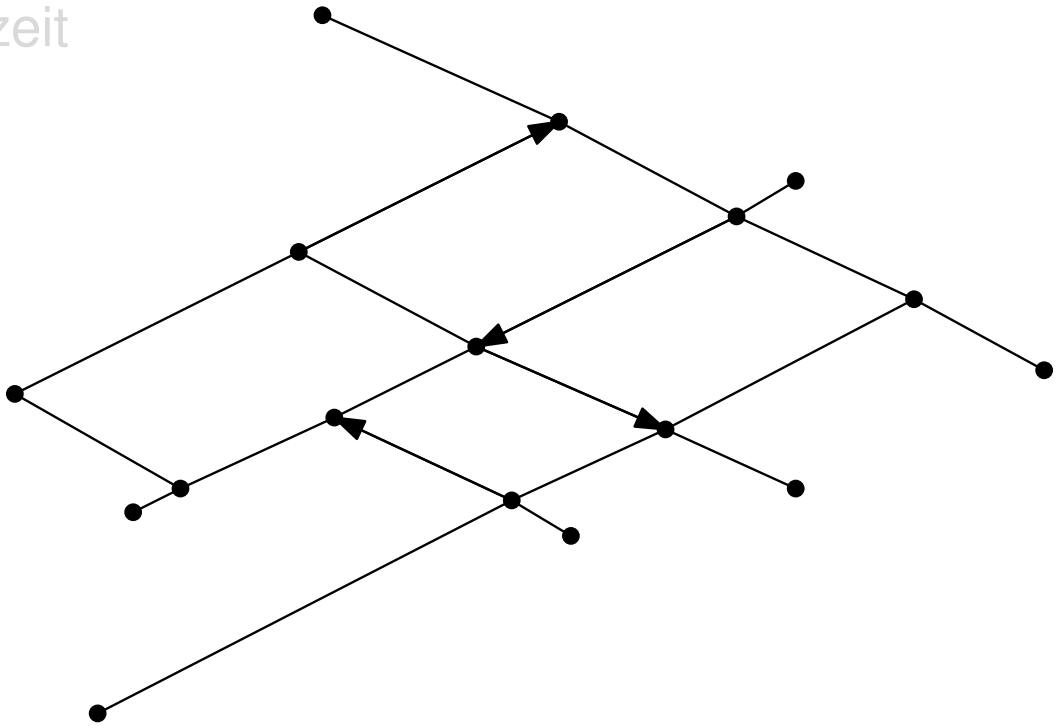
Modellierung (Straßengraphen)

- Knoten sind Kreuzungen
- Kanten sind Straßen
- Einbahnstraßen
- Metrik ist Reisezeit



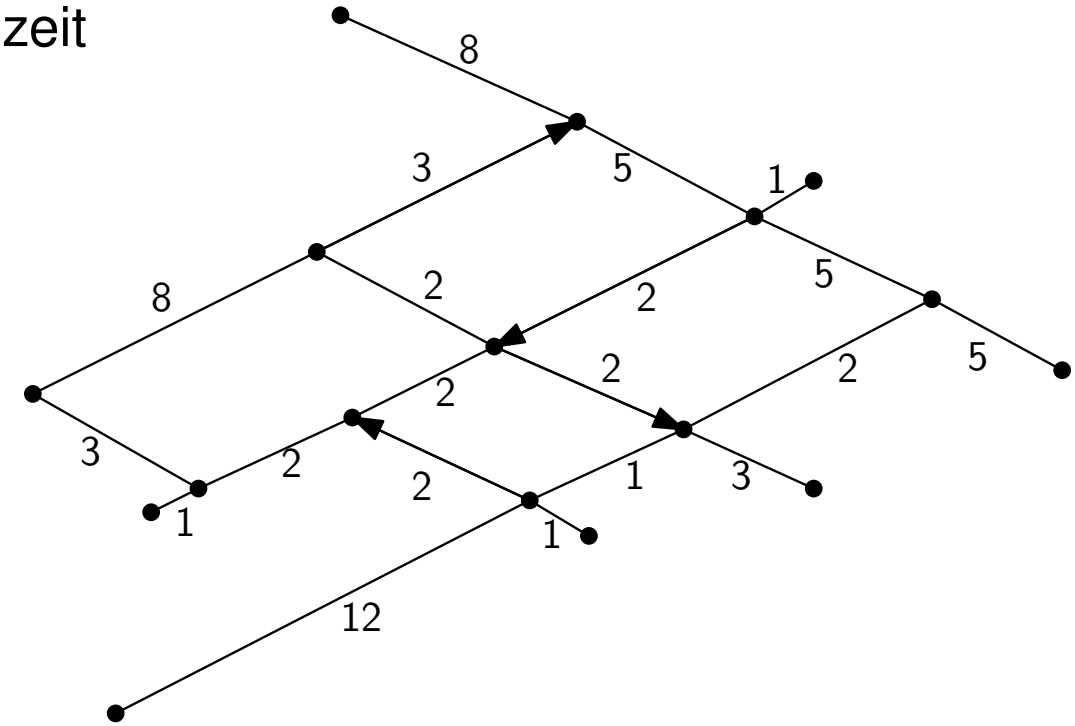
Modellierung (Straßengraphen)

- Knoten sind Kreuzungen
- Kanten sind Straßen
- Einbahnstraßen
- Metrik ist Reisezeit



Modellierung (Straßengraphen)

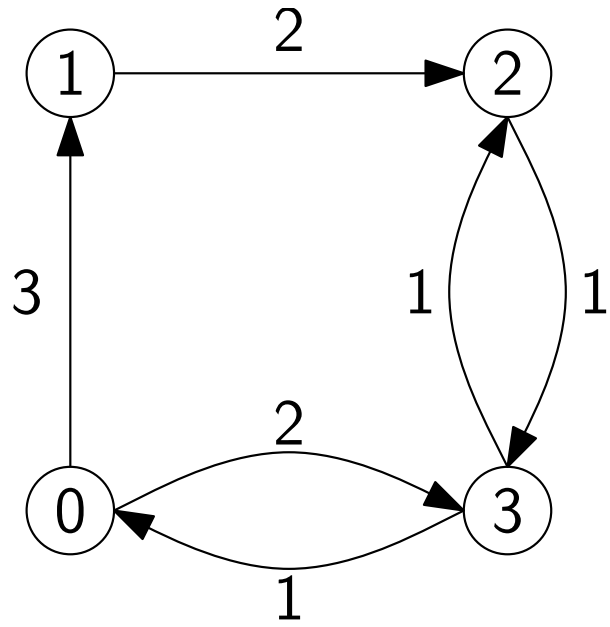
- Knoten sind Kreuzungen
- Kanten sind Straßen
- Einbahnstraßen
- Metrik ist Reisezeit



Graph-Repräsentationen

Vier klassische Ansätze:

- Adjazenzmatrix
- Adjazenzlisten
- Adjazenzarray
- Kantenarray

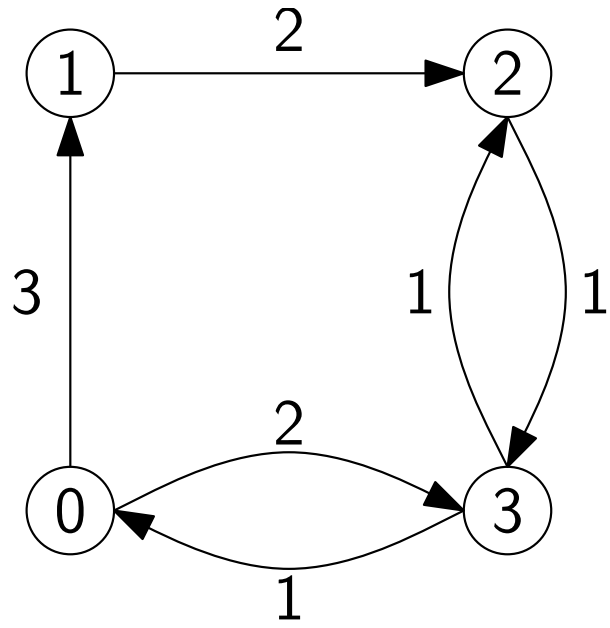


Graph-Repräsentationen

Vier klassische Ansätze:

- Adjazenzmatrix
- Adjazenzlisten
- Adjazenzarray
- Kantenarray

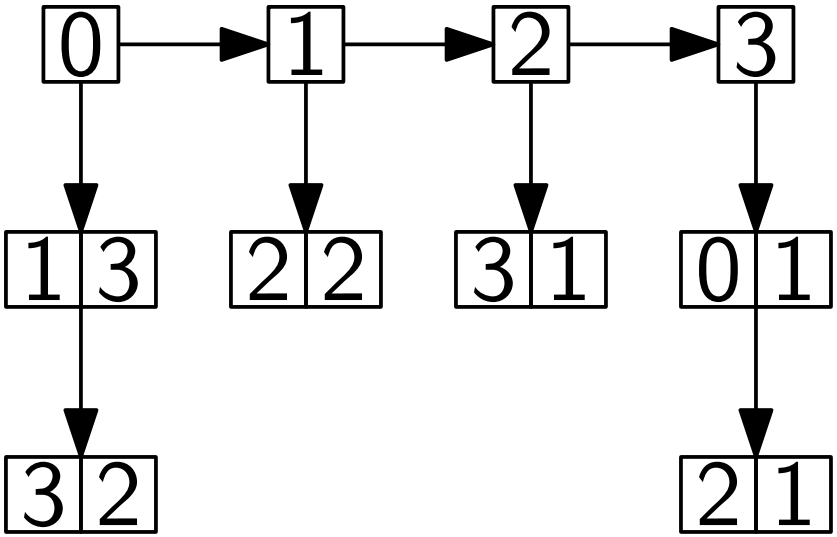
	0	1	2	3
0	–	3	–	2
1	–	–	2	–
2	–	–	–	1
3	1	–	1	–

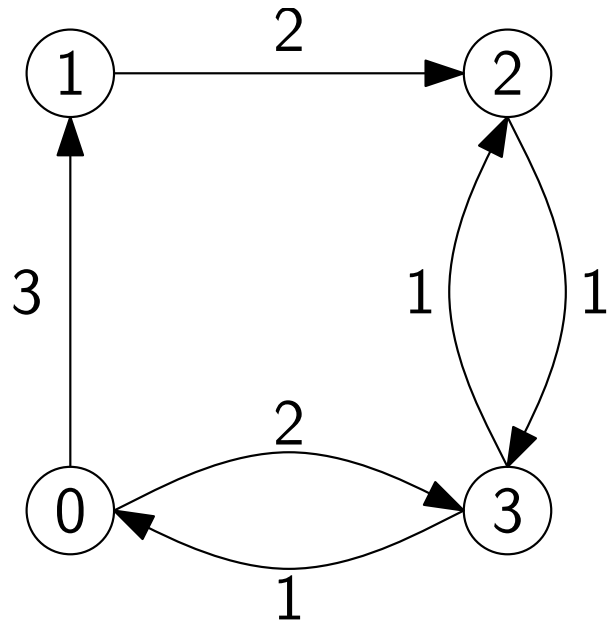


Graph-Repräsentationen

Vier klassische Ansätze:

- Adjazenzmatrix
- Adjazenzlisten
- Adjazenzarray
- Kantenarray

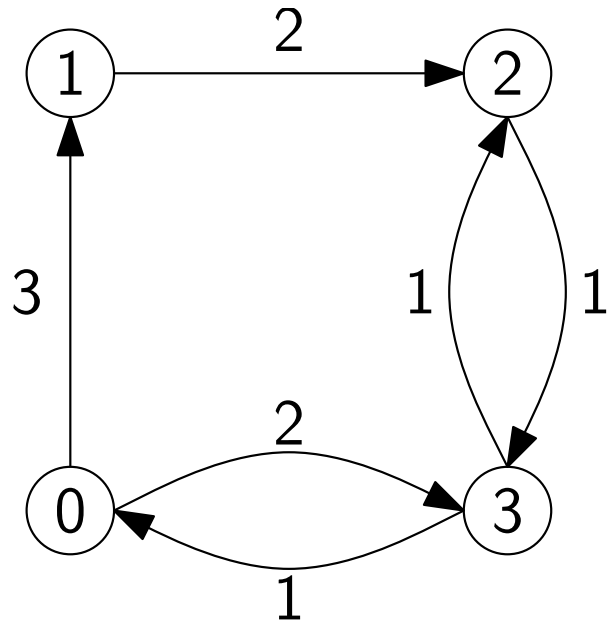


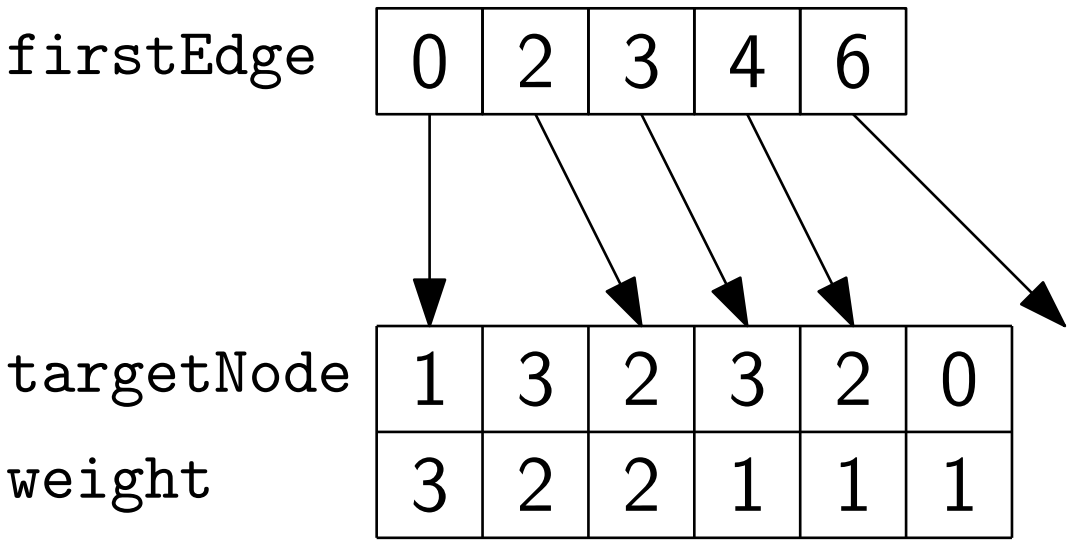


Graph-Repräsentationen

Vier klassische Ansätze:

- Adjazenzmatrix
- Adjazenzlisten
- Adjazenzarray
- Kantenarray



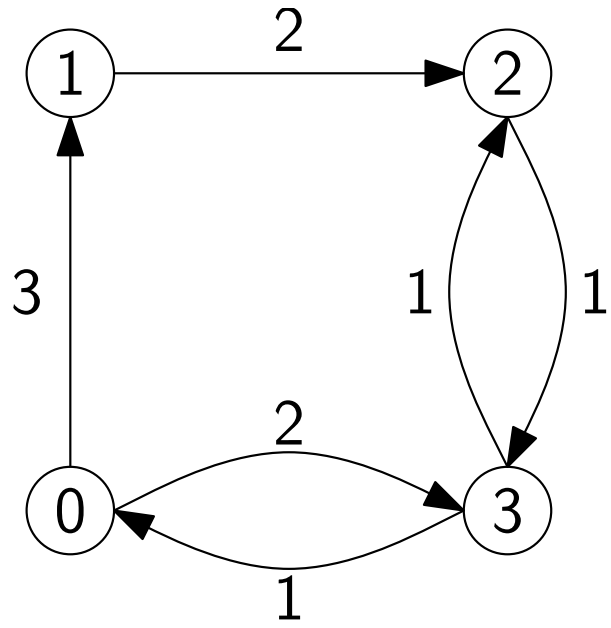


Graph-Repräsentationen

Vier klassische Ansätze:

- Adjazenzmatrix
- Adjazenzlisten
- Adjazenzarray
- Kantenarray

source	1	0	3	0	2	3
target	2	1	2	3	3	0
weight	2	3	1	2	1	1



Was benutzen wir?

Adjazenzmatrix:

- Braucht $O(n^2)$ Speicher
- $n = 18 \cdot 10^6$
- Speicher $\geq 1/4$ Terrabyte
- Impraktikabel

Kantenarray:

- Perfekt für einfache Transformationen (z.B. Graph umdrehen)
- Traversierung (i.e. Pfadsuche) geht nur schlecht

Adjazenzlisten vs Adjazenzarray

- Liste hat nur Vorteil wenn der Graph verändert wird
- Ist bei Pfadsuche nicht der Fall

Wir nehmen ein Kantenarray in der Ein- und Ausgabe und ein Adjazenzarray während der Suche.

Was benutzen wir?

Adjazenzmatrix:

- Braucht $O(n^2)$ Speicher
- $n = 18 \cdot 10^6$
- Speicher $\geq 1/4$ Terrabyte
- Impraktikabel

Kantenarray:

- Perfekt für einfache Transformationen (z.B. Graph umdrehen)
- Traversierung (i.e. Pfadsuche) geht nur schlecht

Adjazenzlisten vs Adjazenzarray

- Liste hat nur Vorteil wenn der Graph verändert wird
- Ist bei Pfadsuche nicht der Fall

Wir nehmen ein Kantenarray in der Ein- und Ausgabe und ein Adjazenzarray während der Suche.

Was benutzen wir?

Adjazenzmatrix:

- Braucht $O(n^2)$ Speicher
- $n = 18 \cdot 10^6$
- Speicher $\geq 1/4$ Terrabyte
- Impraktikabel

Kantenarray:

- Perfekt für einfache Transformationen (z.B. Graph umdrehen)
- Traversierung (i.e. Pfadsuche) geht nur schlecht

Adjazenzlisten vs Adjazenzarray

- Liste hat nur Vorteil wenn der Graph verändert wird
- Ist bei Pfadsuche nicht der Fall

Wir nehmen ein Kantenarray in der Ein- und Ausgabe und ein Adjazenzarray während der Suche.

Was benutzen wir?

Adjazenzmatrix:

- Braucht $O(n^2)$ Speicher
- $n = 18 \cdot 10^6$
- Speicher $\geq 1/4$ Terrabyte
- Impraktikabel

Kantenarray:

- Perfekt für einfache Transformationen (z.B. Graph umdrehen)
- Traversierung (i.e. Pfadsuche) geht nur schlecht

Adjazenzlisten vs Adjazenzarray

- Liste hat nur Vorteil wenn der Graph verändert wird
- Ist bei Pfadsuche nicht der Fall

Wir nehmen ein Kantenarray in der Ein- und Ausgabe und ein Adjazenzarray während der Suche.

Ablauf des Praktikums

1. Überblick der Themen (heute)
2. Einlesen und Bearbeiten der 3 Aufgabenblätter
   - jeder einzeln
3. Gruppenbildung und Zuteilung der Themen
4. Kurzvorträge
   - Vorstellung des gewählten Themas
   - Teilaufgaben, Zwischenschritte
   - ca. 5–10 min
5. Implementierungsphase und experimentelle Auswertung
6. Endvorträge, ca. 20 min
7. Abgabe der Ausarbeitung
   - Knappe Beschreibung Algorithmen
   - Implementierungsdetails und Experimente
   - ca. 10 Seiten (oder soviel, wie ihr braucht, um verständlich zu sein)

Zeitplan

Woche	Aufgabe bis hierhin
Heute 22.10.	Vorbesprechung
ca. 29.10.	Abgabe Blatt 1 / Einlesen
ca. 5.11.	Abgabe Blatt 2 / Einteilung Gruppen
ca. 12.11.	Abgabe Blatt 3 / Vorbereitung Vorträge
ca. 19.11.	Zwischenvorträge (je 5–10 Minuten)
02.02.–06.02.	Endvorträge + Demo (je 20 Minuten)
Ende Februar	Draft-Version von Ausarbeitung
Ende März	Abgabe Ausarbeitung

Ort: Im Poolraum 305 (falls nicht anders per E-Mail angekündigt)

Es gilt Anwesenheitspflicht.

Wir melden innerhalb der ersten Wochen an.

Themen

- Jedes Thema besteht aus einem kleinen Kern und mehreren optionalen Teilen.
- Wie viele optionale Teile umgesetzt werden müssen, hängt von der Gruppengröße ab.

Customization

Problemstellung

Schnelles Berechnen von kürzesten Wegen in Straßennetzwerken mit beliebigen Metriken.

Motivation

- Speed-Up Technik die mit beliebigen Metriken umgehen kann
  Zeit, Fußgänger, keine Autobahnen, Höhenbeschränkungen, etc
- Vorberechnung pro Metrik soll sehr schnell sein
  Ein paar Sekunden für den gesamten Graphen
- Extrem schnelle lokale Updates
- Echtzeit Staudaten
- Schnelle Queryzeiten ($\leq 10\,\text{ms}$)

Customizable Route Planning

Kern

- Vorberechnung
- Customization
- Distanzanfragen

Wahlpflicht

- Visualisierung Suchraum
- Pfadentpackung
- Parallelisierung
- Untersuchung verschiedener Partitionen (+ Visualisierung)
- Point-of-interest Anfragen

Gegeben

- Multi-Level Zellenpartition

D. Delling, A. V. Goldberg, T. Pajor, R. F. Werneck:
Customizable Route Planning in Road Networks.
In: http://research.microsoft.com/apps/pubs/?id=198358

Customizable Route Planning

Kern

- Vorberechnung
- Customization
- Distanzanfragen

Wahlpflicht

- Visualisierung Suchraum
- Pfadentpackung
- Parallelisierung
- Untersuchung verschiedener Partitionen (+ Visualisierung)
- Point-of-interest Anfragen

Gegeben

- Multi-Level Zellenpartition

D. Delling, A. V. Goldberg, T. Pajor, R. F. Werneck:
Customizable Route Planning in Road Networks.
In: `http://research.microsoft.com/apps/pubs/?id=198358`

Customizable Contract. Hierarchies

Kern

- Vorberechnung
- Customization
- Distanzanfragen

Wahlpflicht

- Visualisierung Suchraum
- Pfadentpackung
- Parallelisierung
- Detailierte Untersuchung Nested-Dissection (+Visualisierung)
- Contraction Graph Datenstruktur

Gegeben

- Nested-Dissection Ordnung

J. Dibbelt, B. Strasser, D. Wagner:
Customizable Contraction Hierarchies.
In: http://arxiv.org/abs/1402.0402

Customizable Contract. Hierarchies

Kern

- Vorberechnung
- Customization
- Distanzanfragen

Wahlpflicht

- Visualisierung Suchraum
- Pfadentpackung
- Parallelisierung
- Detailierte Untersuchung Nested-Dissection (+Visualisierung)
- Contraction Graph Datenstruktur

Gegeben

- Nested-Dissection Ordnung

J. Dibbelt, B. Strasser, D. Wagner:
Customizable Contraction Hierarchies.
In: `http://arxiv.org/abs/1402.0402`

PHAST

Problemstellung

Schnelle Berechnung von one-to-all kürzeste-Wege Bäumen auf Straßennetzwerken.

Motivation

- Es gibt n^2 viele kürzeste Wege
- Bei $n = 18 \cdot 10^6$ dauert das
- Viele Vorberechnungen nur optimal, wenn alle aufgezählt werden.
- PHAST nutzt die Hardware geschickt aus
- Geht in unter einem Tag

PHAST

Kern

- Basis-PHAST
- Graphdiameter bestimmen

Wahlpflicht

- Parallelisierung
  - Threads und SSE oder GPU
- Anwendung: Arc-Flags
  - Single-Level Partition mit METIS
  - Arc-Flags berechnen & visualisieren
  - Flaggenkomprimierung
- One-to-many (RPHAST)

Gegeben

- Knotenordnung

D. Delling, A. V. Goldberg, A. Nowatzyk, R. F. Werneck:
PHAST: Hardware-Accelerated Shortest Path Trees.
In: *Journal of Parallel and Distributed Computing.*

PHAST

Kern

- Basis-PHAST
- Graphdiameter bestimmen

Wahlpflicht

- Parallelisierung
  - Threads und SSE oder GPU
- Anwendung: Arc-Flags
  - Single-Level Partition mit METIS
  - Arc-Flags berechnen & visualisieren
  - Flaggenkomprimierung
- One-to-many (RPHAST)

Gegeben

- Knotenordnung

D. Delling, A. V. Goldberg, A. Nowatzyk, R. F. Werneck:
PHAST: Hardware-Accelerated Shortest Path Trees.
In: *Journal of Parallel and Distributed Computing.*

Hub-Labeling

Problemstellung

Sehr schnelle Routenplanung.

Motivation

- Wie schnell ist eine Distanzanfragen möglich?
- Erweiterte Anfragen, wie z. B. POI-Anfragen.

Hub-Labels, Pruned Labeling

Kern

- Vorberechnung, geg. Ordnung
- Distanzanfrage

Wahlpflicht

- Ordnungsbestimmung
- Visualisierung der Labels
- Pfadentpacken
- Komplexe Anfrage wie POI (in C++, kein SQL)

Gegeben

- (Anfangs-)Knotenordnung

D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck.
Robust Exact Distance Queries on Massive Networks.
In: `http://research.microsoft.com/apps/pubs/default.aspx?id=208867`

Hub-Labels, Pruned Labeling

Kern

- Vorberechnung, geg. Ordnung
- Distanzanfrage

Wahlpflicht

- Ordnungsbestimmung
- Visualisierung der Labels
- Pfadentpacken
- Komplexe Anfrage wie POI (in C++, kein SQL)

Gegeben

- (Anfangs-)Knotenordnung

D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck.
Robust Exact Distance Queries on Massive Networks.
In: `http://research.microsoft.com/apps/pubs/default.aspx?id=208867`

RAPTOR

Problemstellung

Schnelle Berechnung von multi-kriteriellen (Reisezeit, Anzahl Umstiege, etc.) Anfragen in öffentlichen Verkehrsnetzen.

Motivation

- Netzwerke sind *zeitabhängig*
- Bestehen aus Stops, Routen, Trips, . . .
- Modellierung als Graphen zu kompliziert/langsam
- Optimierung von Ankunftszeit alleine nicht ausreichend
- Dynamische Aspekte: Verspätungen, Ausfälle, . . .

RAPTOR

Kern

- Implementierung von RAPTOR für Ankunftszeit und Umstiege

Wahlpflicht

- Erweiterung von RAPTOR auf Kriterium *Tarifzonen*
- Parallelisierung von RAPTOR auf Multi-Core
- Ausgabe der Routen und Visualisierung

Gegeben

- Fahrplan als Datenstruktur

D. Delling, T. Pajor, R. F. Werneck:
Round-Based Public Transit Routing.
In: *Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12)*, pages 130-140. SIAM, 2012.

RAPTOR

Kern

- Implementierung von RAPTOR für Ankunftszeit und Umstiege

Wahlpflicht

- Erweiterung von RAPTOR auf Kriterium *Tarifzonen*
- Parallelisierung von RAPTOR auf Multi-Core
- Ausgabe der Routen und Visualisierung

Gegeben

- Fahrplan als Datenstruktur

D. Delling, T. Pajor, R. F. Werneck:
Round-Based Public Transit Routing.
In: *Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12)*, pages 130-140. SIAM, 2012.

Themenübersicht

Themen

- Customizable Route Planning / Contraction Hierachies
  Routenplanung in Straßennetzwerken mit beliebigen Metriken
- PHAST
  Schnelle Berechnung von one-to-all kürzesten Wegen
- Hub-Labels
  Sehr schnelle Distanzanfragen
- RAPTOR
  Multi-kriterielle Fahrplanauskunft

Gruppen- und Themenzuteilung in zwei Wochen (Abgabe 2. ÜB)

Accounts

Rechner-Login

- Ausfüllen von Antragsblatt
- Wir geben per Email bescheid, sobald angelegt

Git-Zugang

- Wir benutzen Git zur Versionskontrolle
  (siehe erstes Blatt)
- Zugangsdaten per Email