max planck institut
informatik

Schnellste Wege
Wie funktioniert ein Navi?

Kurt Mehlhorn und Adrian Neumann
Max-Planck-Institut für Informatik

Vorlesung Ideen der Informatik

Schnellste Wege
Routenfinden im Navi

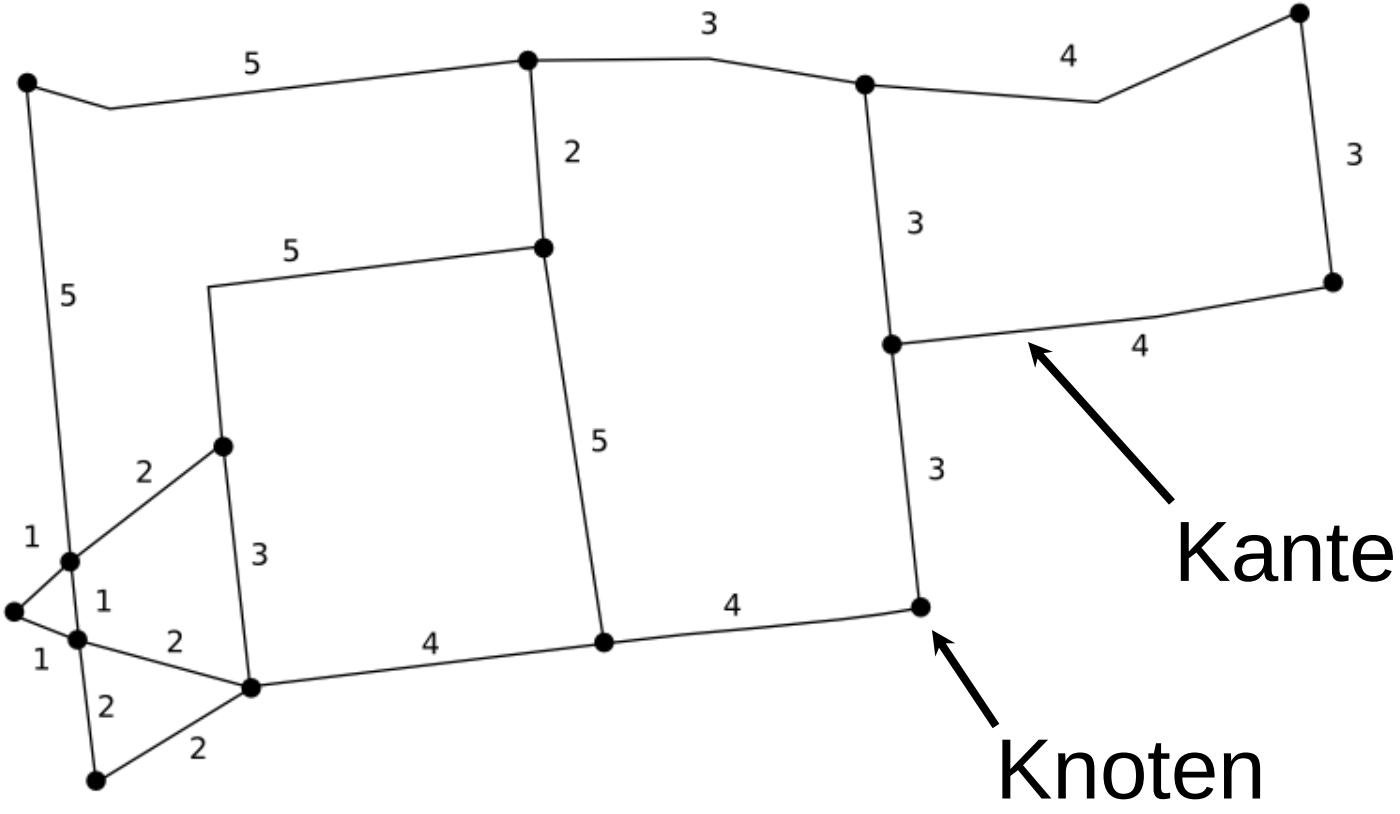
- Karten und Graphen
- Algorithmen für schnellste Wege
  - Erster Versuch
  - Dijkstras Algorithmus
- Schnellste Wege in Straßengraphen

Landkarten

Landkarten enthalten sehr viel Information; nur Straßengraph ist wichtig

Graphen als Abstraktion

Graphen bestehen aus Knoten und Kanten. Jede Kante hat eine Fahrzeit.



G = (V,E), E Teilmenge V x V

Straßennetzwerke

- Europa: 24 Millionen Knoten, 58 Millionen Kanten
- Schnellste Wege kann man trotzdem in Sekunden berechnen
- Oder in Millisekunden nachschlagen
- Algorithmen arbeiten auf dem Graphen und zeigen Ergebnis auf der Karte
- Fahrzeit über eine Kante
  - Länge / angenommene Durchschnittsgeschwindigkeit
  - Tatsächlich bekannt durch Beobachtung

Algs für schnellste Wege: Grundidee

Wenn ich vom Startknoten s in 30 Minuten nach A komme

und

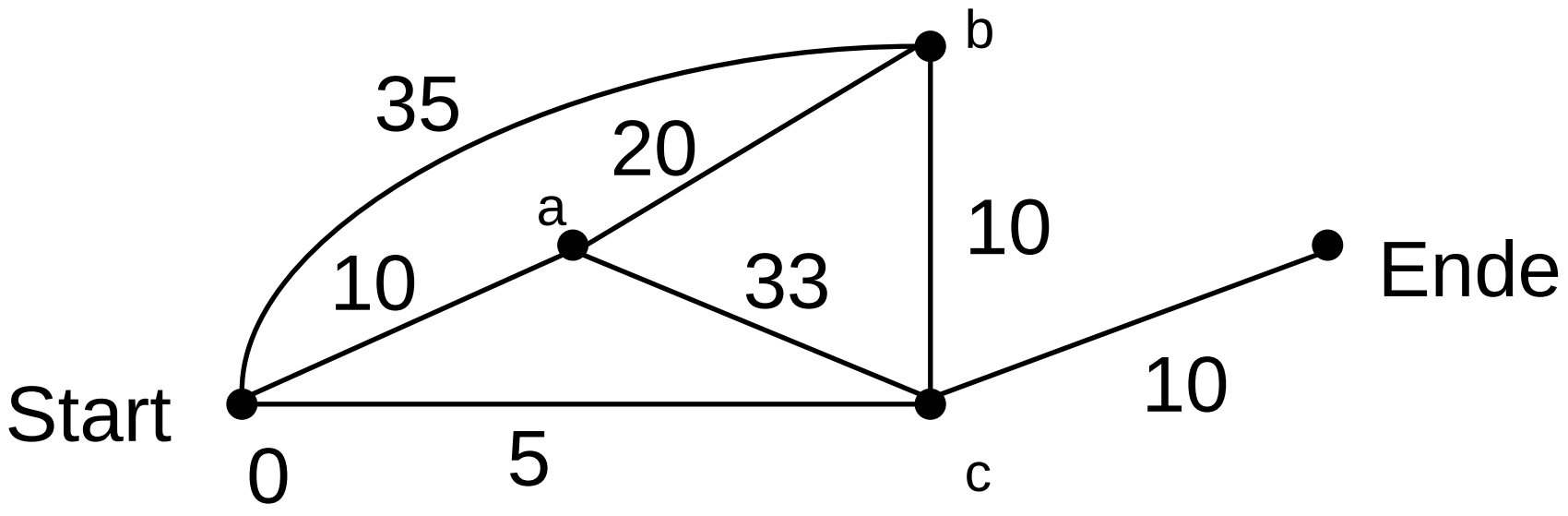
von A nach B in 5 Minuten,

dann komme ich in 35 Minuten vom Startknoten s nach B.

Ein erster Algorithmus

1. Fahrzeit von Start nach Start = 0 Minut.
2. Für alle anderen Knoten, weiß ich noch keinen Weg: also Fahrzeit = unendlich
3. Falls Fahrzeit nach A = X Min und Straße A→B braucht Y Min, dann Fahrzeit nach B = min(X+Y,schon bekannte Fahrzeit nach B).
4. Wiederhole 3. solange noch eine Fahrzeit verbessert werden kann

Beispiel



Ein erster Algorithmus (Pseudocode)

Setze dist[s] $\leftarrow$ 0 und dist[v] $\leftarrow \infty$ für v $\neq$ s

Solange es eine Kante (u,v) gibt mit

dist[u] + zeit(u,v) $<$ dist[v]

setze dist[v] auf dist[u] + zeit(u,v)

Dabei zeit(u,v) = Fahrzeit über die Kante von u nach v.

u, v sind Namen für Knoten

dist[v] = beste bekannte Fahrzeit von s nach v

Fragen

- Finden wir so immer den schnellsten Weg (von s zu allen anderen Knoten)?
- Wie lange dauert das?

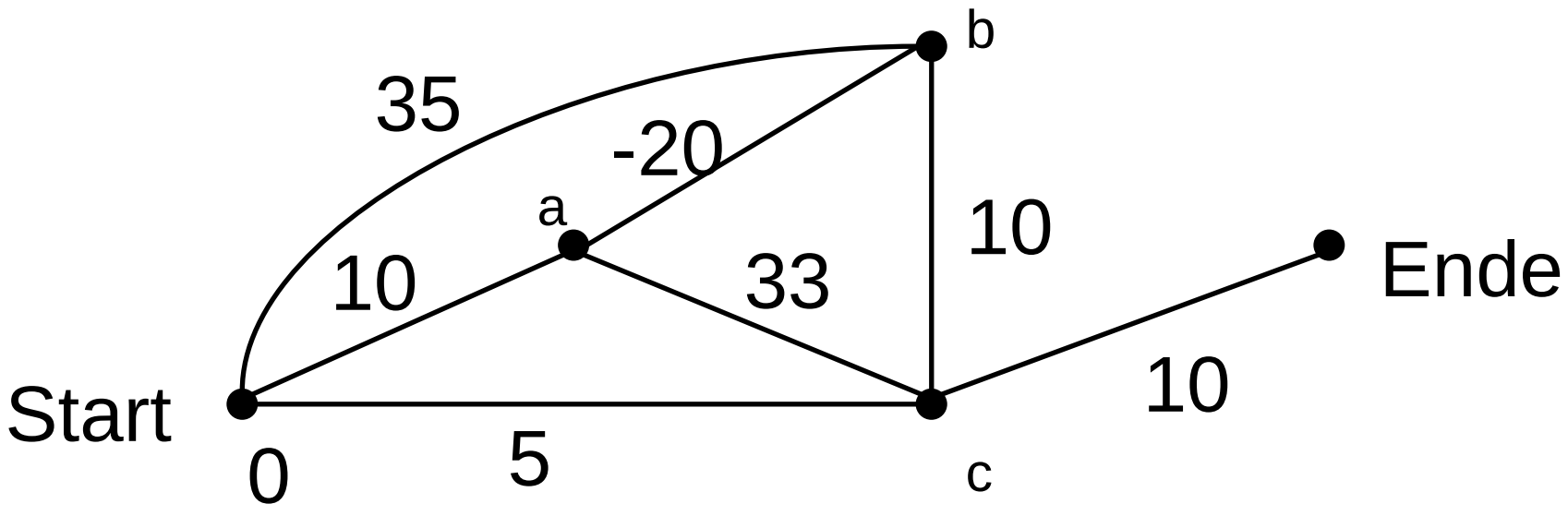
Korrektheit

- Immer gilt für alle Knoten v:

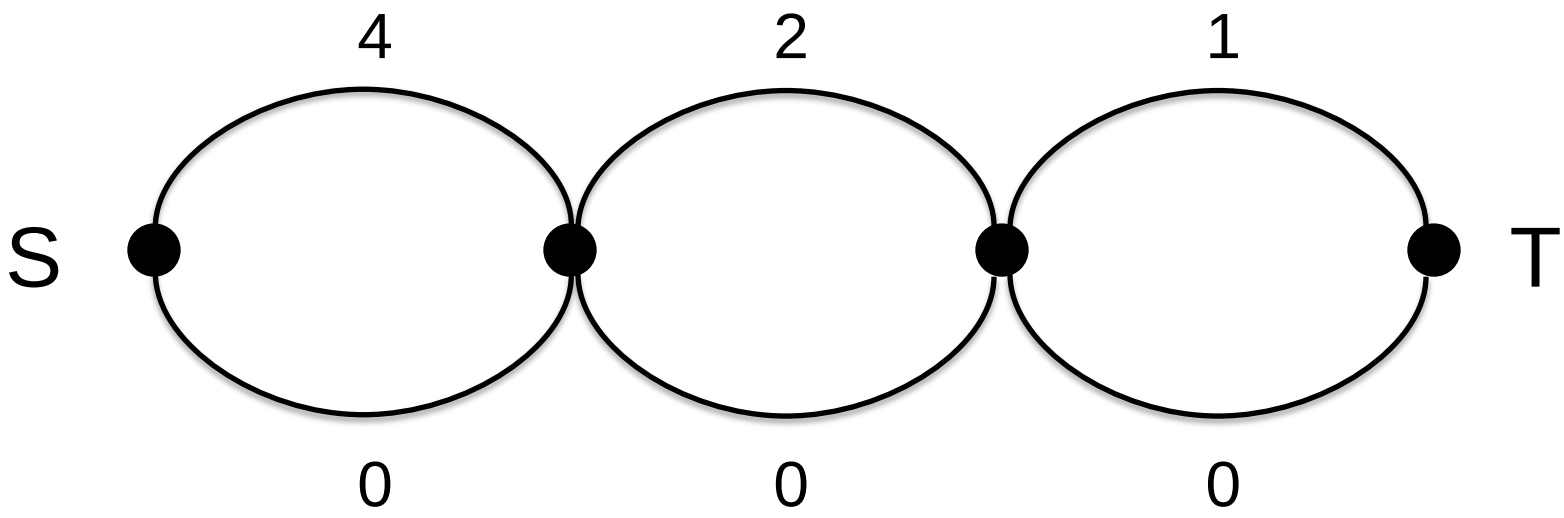
  dist[v] ≥ schnellste Fahrzeit von s nach v
- Beh: wenn Alg anhält, dann dist[v] = … für alle v
- Betrachte schnellsten Weg von s nach v:

$$s \rightarrow a \rightarrow b \rightarrow c \rightarrow v$$

Aufgabe: Fahrzeiten dürfen nun auch negativ sein. Die Fahrzeit des Weges Start, a, b, c, Ende ist 10 – 20 + 10 + 10 = 10. Funktioniert der Algorithmus noch?



Effizienz



- Fahrzeit nach T wird 8 mal geändert
- Ein Ort mehr: Laufzeit verdoppelt sich

Exponentielles Wachstum ist ein Killer

- 4 Orte: 8 Änderungen
- 5 Orte: 16 Änderungen
- 6 Orte: 32
- 7 Orte: 64
- 21 Orte: mehr als 1.000.000
- 41 Orte: mehr als 1.000.000.000.000

Mein Rechner kann 10^9 Operationen/sec.

Geschicktes Auswählen

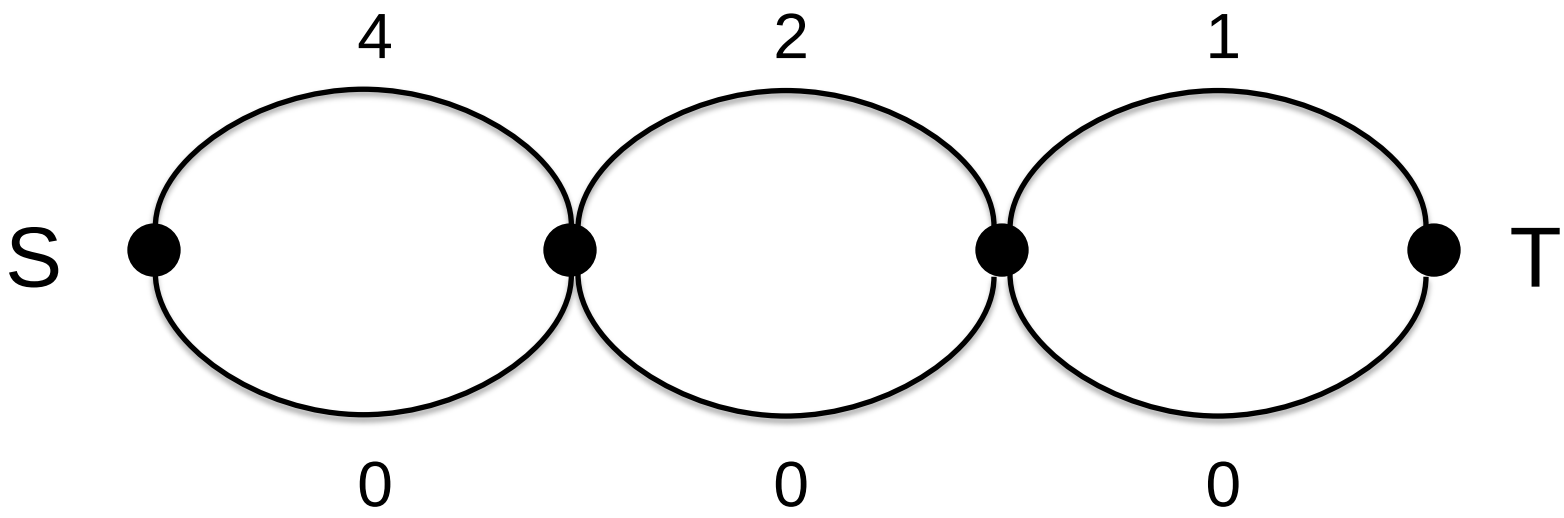
1. Nach S in 0
2. Wenn A in X und A→B braucht Y, dann B in X+Y Min
3. Wiederhole 2. solange nicht stabil

Wende 2. immer auf alle Kanten A→B aus dem Knoten A mit der kleinsten Fahrzeit an
(auf den wir das nicht schon vorher angewendet haben)

Dijkstras Algorithmus (1959)

Turing Award (1972)

Beispiel



Fahrzeit wird über jede Kante
nur einmal propagiert

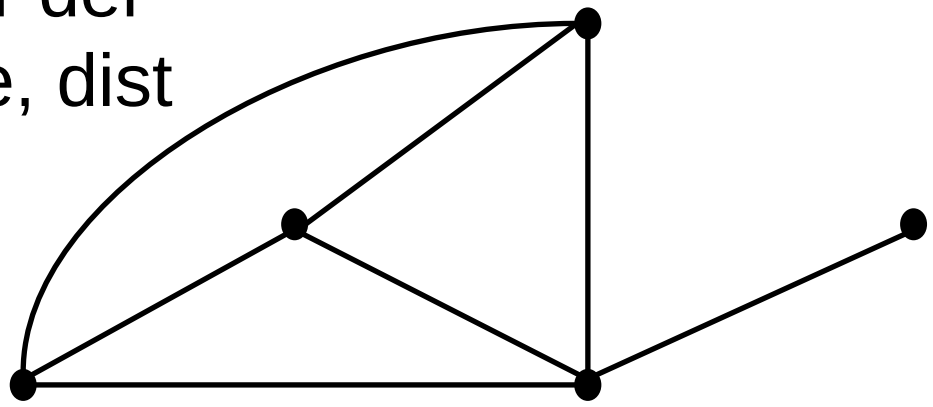
Pseudocode

Dijkstra(s): **dist[v] = beste bekannte Fahrzeit nach v**

```
dist[s] ← 0 und markiere s als aktiv
für alle v ≠ s:
    dist[v] ← unendlich und markiere v als aktiv
solange es aktiven Knoten gibt:
    u ← der aktive Knoten mit kleinstem Wert dist[u]
    markiere u als nicht aktiv
    fuer alle Kanten (u,v) tue:
        falls dist[u] + Zeit(u,v) < dist[v]:
            dist[v] ← dist[u] + Zeit(u,v)
```

Speicherlayout

- Knoten = nummeriert von 0 bis n-1
- Kanten = nummeriert von 0 bis m-1; zuerst die Kanten aus 0 heraus, dann aus 1, …
- Für jede Kante: Ziel und Fahrzeit
- Für jeden Knoten: Nummer der ersten ausgehenden Kante, dist und ist-aktiv



Anmerkungen

- u ← der aktive Knoten mit kleinstem Wert dist[u]

ist ausführlicher

mindist ← unendlich
Für i = 0 bis n – 1 tue
    falls (aktiv[i] und dist[i] < mindist)
    dann mindist ← dist[i]; u ← i;

- Pfade mitberechnen, Wellenausbreitung

Korrektheit

Ähnlich zum Grundalgorithmus, aber etwas komplexer.

Siehe nächste Folie.

Wenn sie eine mathematische Ader haben, sollten sie versuchen, dass Argument nachzuvollziehen, sonst überspringen sie die Folie.

Korrektheit

- Immer: dist[v] ≥ kürzeste Fahrzeit von s nach v
- Beh: dist[v] = …, wenn v deaktiviert wird
- Bew durch Induktion über Anzahl der Kanten im schnellsten Weg nach v
- Anzahl = 0, dann v = s, also Beh. richtig
- Anzahl > 0, sei uv die letzte Kante auf schnellstem Weg nach v. Dann stimmt Beh. für u. Wenn u deaktiviert wird, gilt

  dist[u] = Fahrzeit nach u < Fahrzeit nach v ≤ dist[v].
- Also ist v noch aktiv, wenn u deaktiviert wird. Wenn u deaktiviert wird, setzen wir

  dist[v] ← dist[u] + Fahrzeit(u,v) = Fahrzeit(v)

Aufgabe

Fahrzeiten dürfen nun auch negativ sein. Geben sie ein Beispiel an, dass der Alg von Dijkstra nun nicht mehr funktioniert.

Laufzeit

n = #Knoten, m = #Kanten,

m_u = Kanten aus u heraus

Dijkstra(s):

dist[s] ← 0, markiere s als aktiv

für alle v ≠ s:

dist[v] ← unendlich

markiere v als aktiv

} ~n

solange es aktiven Knoten gibt:

~n {

u ← der aktive Knoten mit kleinstem Wert dist[u]

markiere u als nicht aktiv

~m_u {

fuer alle Kanten (u,v) tue:

falls dist[u] + Zeit(u,v) < dist[v]:

dist[v] ← dist[u] + Zeit(u,v)

Laufzeit

- n mal Minimum finden: n mal n = n^2
- Alle Kanten verfolgen: m

Insgesamt: $m + n^2$

Kann verbessert werden auf m + n log n

und braucht dann etwa 10sec für Europa

Navigationssysteme

- Navigationsysteme im Auto benutzen Dijkstra + Straßenhierarchie

- Auskunftssysteme (etwa Google Maps) vorberechnen Antworten zum Teil

Nachschlagen statt Rechnen

- Idee: Alle Wege vorberechnen
  - Europa: $24*10^6$ Knoten
  - 24 Millionen mal Dijkstra:

  24 mal 10^6 mal 10 Sekunden; das ist etwas unter 8 Jahren

- So einfach geht es nicht

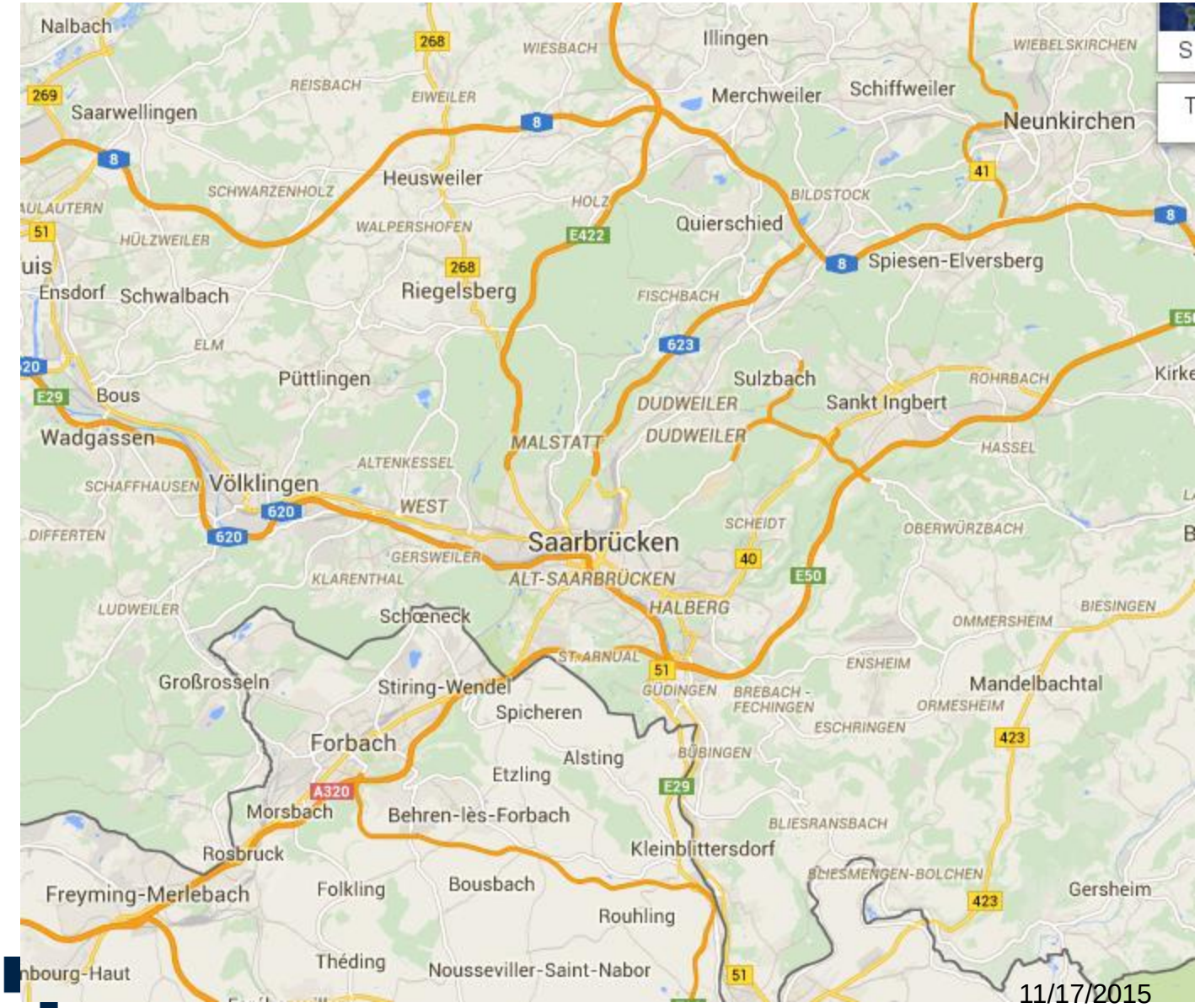
Transitknoten (Bast-Funke)

- Kurze Wege: on-the-fly mit Dijkstra
- Alle weiten Wege passieren eine kleine Menge von Transitknoten
  - Gesamtmenge der Transitknoten ist klein
  - Für jeden festen Startpunkt gibt es nur sehr wenige Transitknoten

Hannah Bast, Stefan Funke

Saar LB Preis

Transitknoten

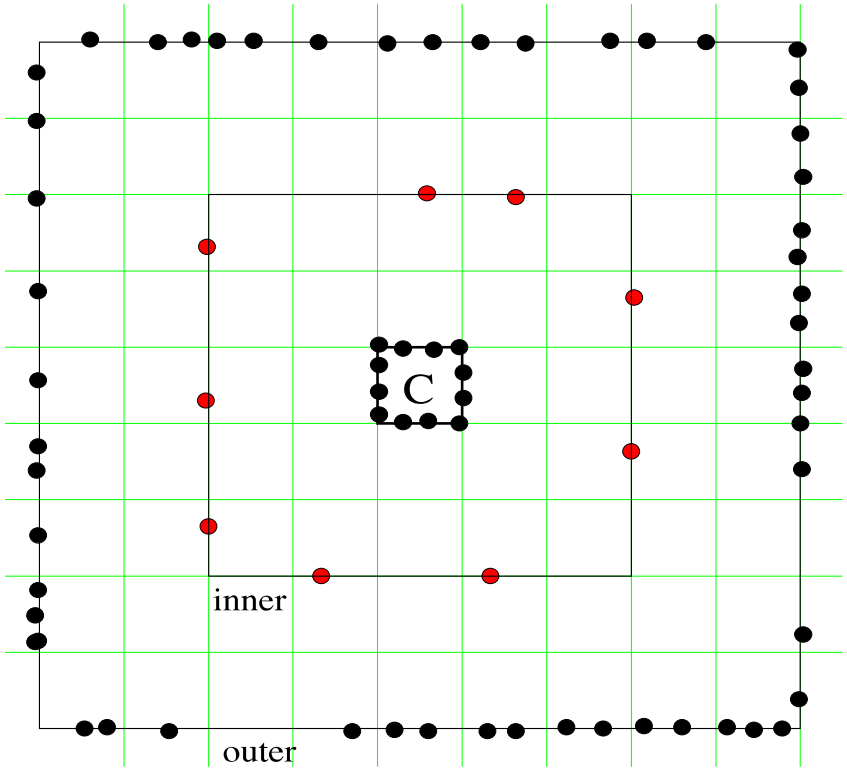


Transitknoten

- KM wohnt in Scheidt
- Meine Transitknoten
  - nach Osten: Autobahnauffahrt St. Ingbert
  - Nach Süden: Kleinblittersdorf
  - Nach Westen: goldene Bremm
  - Nach Nord-Westen: Stadtautobahn
  - Nach Norden und Nordosten: Autobahn Sulzbach
- Alle Bewohner von Scheidt benutzen die gleichen Transitknoten

Transitknoten

- Gitter über die Welt legen, Punkt = Kreuzung von Gitter und Straße
- Für jede Zelle C Wege nach “outer” ausrechnen
- Die Knoten aus “inner”, die benutzt werden sind Transitknoten für C



Vorberechnen

- Europa: 24 Mio Knoten, 10000 Transitknoten
- Für jeden Knoten zu seinen Transitknoten
  - Ungefähr 10/Knoten → $24*10^7$ Wege
- Zwischen allen Transitknoten paarweise
  - Ungefähr 10000 Knoten → 10^8 Wege
- Passt auf einen Server!

Wege Finden

- Kurze Wege: Dijkstra
- Lange Wege: A nach B
  - Zerlegen in A – T_1 – T_2 – B wobei T_1 Transitknoten für A und T_2 für B
  - Probiere alle Möglichkeiten für T1 und T2 (jeweils etwa 10) und bestimme den minimalen Wert von

    $dist(A,T_1) + dist(T_1,T_2) + dist(T_2,B)$
  - Das sind 10 + 10 + 100 Speicherzugriffe

Zusammenfassung

- Dijkstra ist ein sehr schneller Alg für kürzeste Wege (wenige Sekunden in Graph mit 10 Mio Knoten und 30 Mio Kanten.
- Straßengraphen haben Struktur (Hierarchie der Straßen, Fast-Planarität)
- Vorberechnen ist eine gute Idee, wenn man viele Anfragen zu beantworten hat.

Aufgabe

- Die Karte von Europa hat 24 Mio Knoten. Nehmen sie an, dass sie 1 KiloByte brauchen, um einen Weg zwischen zwei Knoten abzuspeichern. Wieviel Speicherplatz brauchen sie, um alle Wege abzuspeichern?
- Was sind die Transitknoten für ihren Wohnort?

Aufgabe: Zugverbindungen

Beim Auskunftssystem der Bahn gibt man Start, Ziel und gewünschte Abfahrtszeit ein und bekommt die früheste Ankunftszeit. Wie könnte das gehen?

Idee: modellieren sie einen Bahnhof durch 24 x 60 Knoten, die den Minuten des Tages entsprechen. Ein Zug der Bahnhof A um 2:54 verlässt und Bahnhof B um 3:23 erreicht, modellieren sie durch eine Kante von A2:54 nach B3:23. Wie verbinden sie die Knoten, die zum gleichen Bahnhof gehören? Wie beantworten sie eine Anfrage?