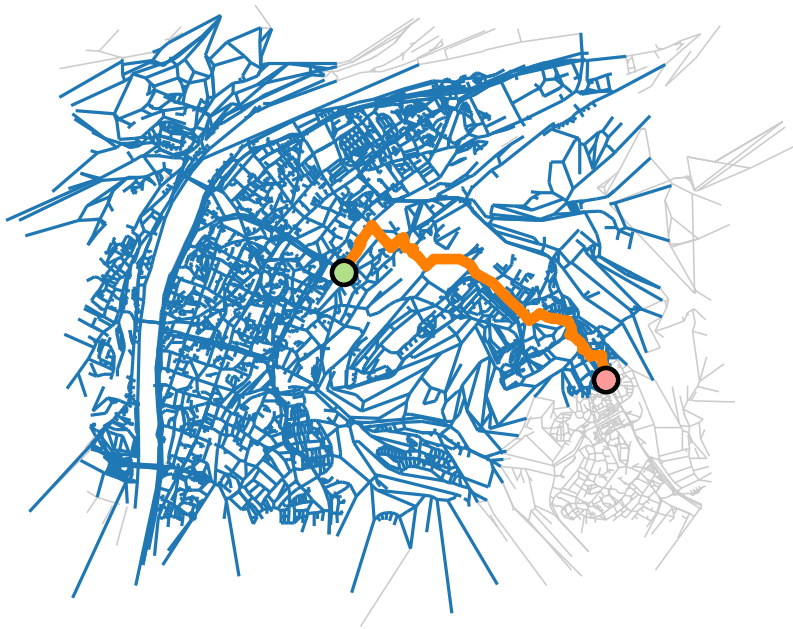


Algorithmen für geographische Informationssysteme

2. Vorlesung Routenplanung

Teil I: Problemstellung





Wiederholung – DIJKSTRA



DIJKSTRA(WeightedGraph G , Vertex s)

INITIALIZE(G, s)

$Q = \text{new PriorityQueue}(V, d)$

while not $Q.\text{EMPTY}()$ **do**

$u = Q.\text{EXTRACTMIN}()$

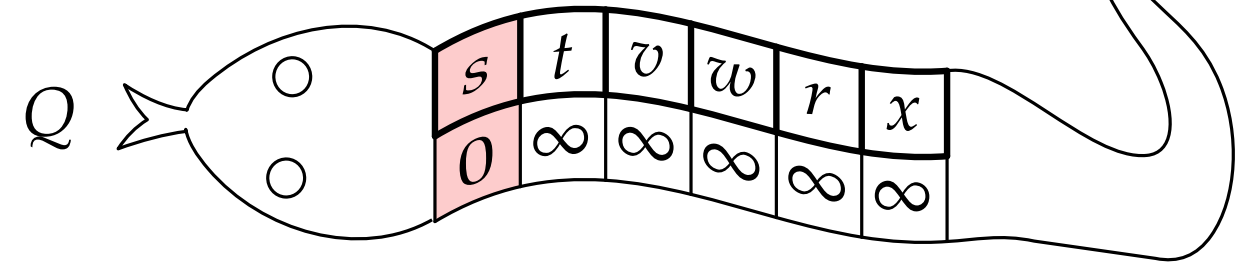
foreach $v \in \text{Adj}[u]$ **do**

if $v.d > u.d + w(u, v)$ **then**

$v.d = u.d + w(u, v)$

$v.\pi = u$

$Q.\text{DECREASEKEY}(v, v.d)$



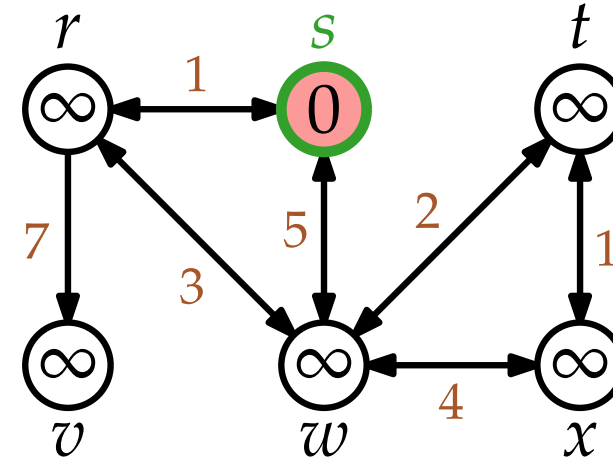
INITIALIZE(Graph G , Vertex s)

foreach $u \in V$ **do**

$u.d = \infty$

$u.\pi = \text{nil}$

$s.d = 0$



Wiederholung – DIJKSTRA



DIJKSTRA(WeightedGraph G , Vertex s)

INITIALIZE(G, s)

$Q = \text{new PriorityQueue}(V, d)$

while not $Q.\text{EMPTY}()$ **do**

$u = Q.\text{EXTRACTMIN}()$

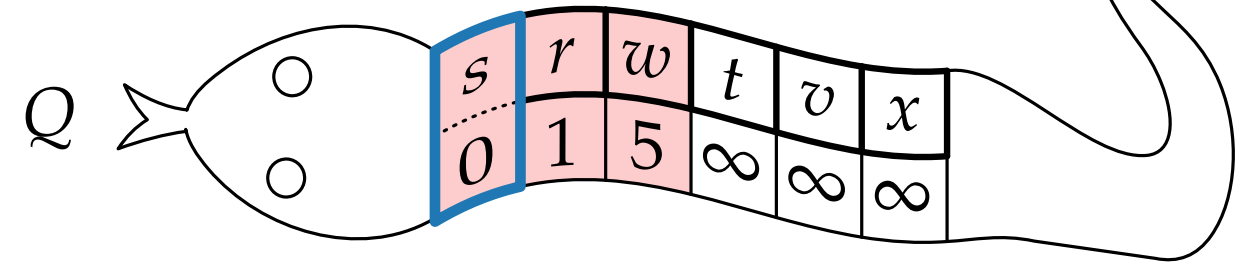
foreach $v \in \text{Adj}[u]$ **do**

if $v.d > u.d + w(u, v)$ **then**

$v.d = u.d + w(u, v)$

$v.\pi = u$

$Q.\text{DECREASEKEY}(v, v.d)$



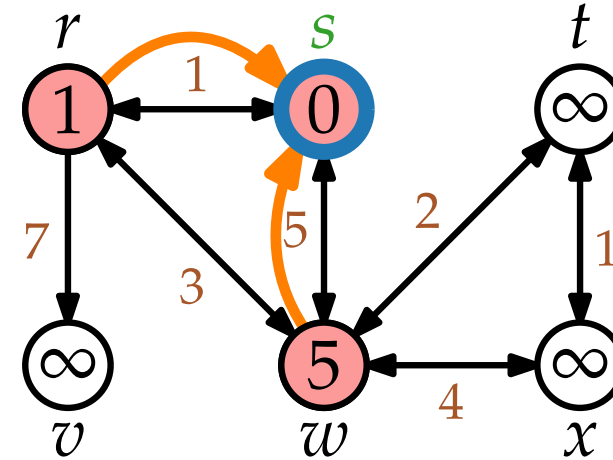
INITIALIZE(Graph G , Vertex s)

foreach $u \in V$ **do**

$u.d = \infty$

$u.\pi = \text{nil}$

$s.d = 0$



Wiederholung – DIJKSTRA



DIJKSTRA(WeightedGraph G , Vertex s)

INITIALIZE(G, s)

$Q = \text{new PriorityQueue}(V, d)$

while not $Q.\text{EMPTY}()$ **do**

$u = Q.\text{EXTRACTMIN}()$

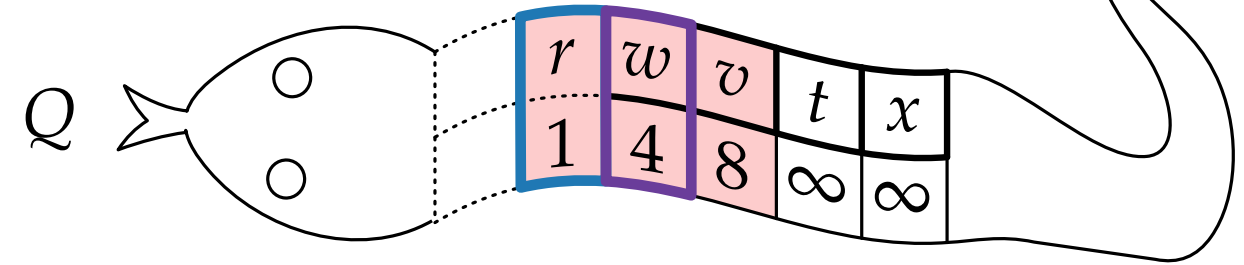
foreach $v \in \text{Adj}[u]$ **do**

if $v.d > u.d + w(u, v)$ **then**

$v.d = u.d + w(u, v)$

$v.\pi = u$

$Q.\text{DECREASEKEY}(v, v.d)$



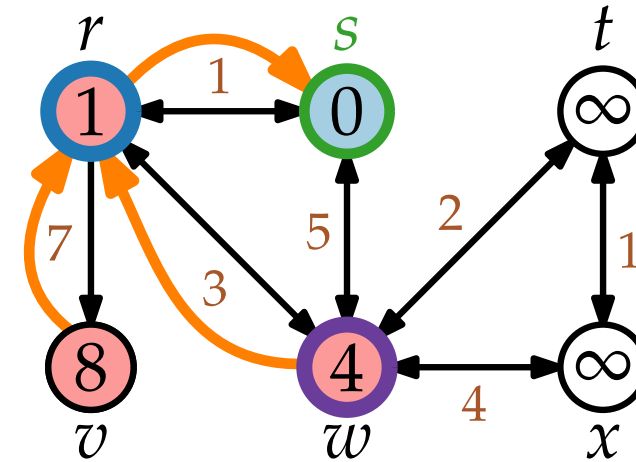
INITIALIZE(Graph G , Vertex s)

foreach $u \in V$ **do**

$u.d = \infty$

$u.\pi = \text{nil}$

$s.d = 0$



Wiederholung – DIJKSTRA

DIJKSTRA(WeightedGraph G , Vertex s)

INITIALIZE(G, s)

$Q = \text{new PriorityQueue}(V, d)$

while not $Q.\text{EMPTY}()$ **do**

$u = Q.\text{EXTRACTMIN}()$

foreach $v \in \text{Adj}[u]$ **do**

if $v.d > u.d + w(u, v)$ **then**

$v.d = u.d + w(u, v)$

$v.\pi = u$

$Q.\text{DECREASEKEY}(v, v.d)$

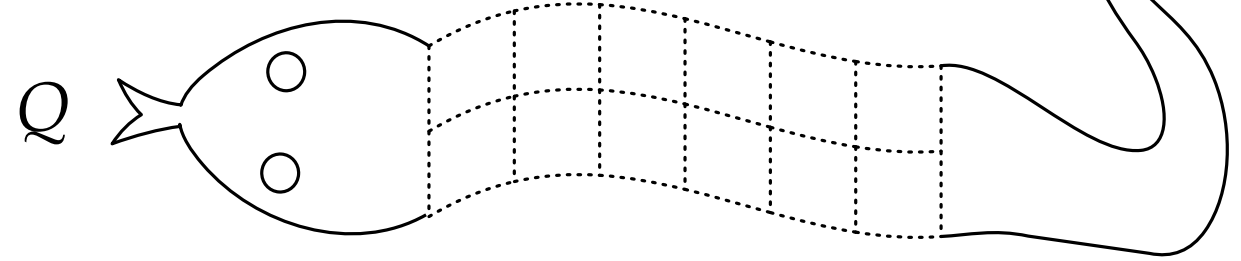
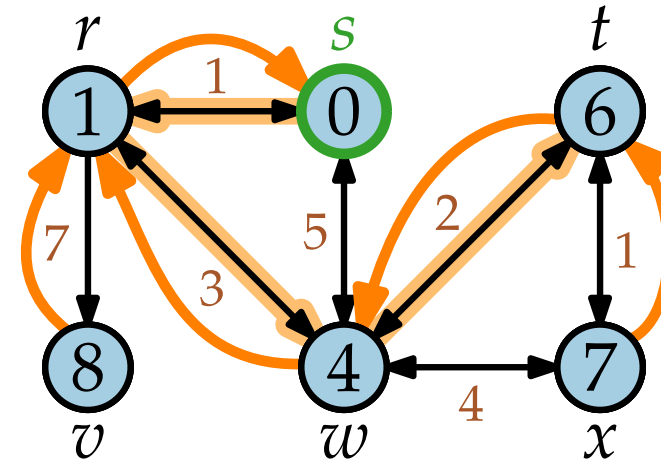
INITIALIZE(Graph G , Vertex s)

foreach $u \in V$ **do**

$u.d = \infty$

$u.\pi = \text{nil}$

$s.d = 0$

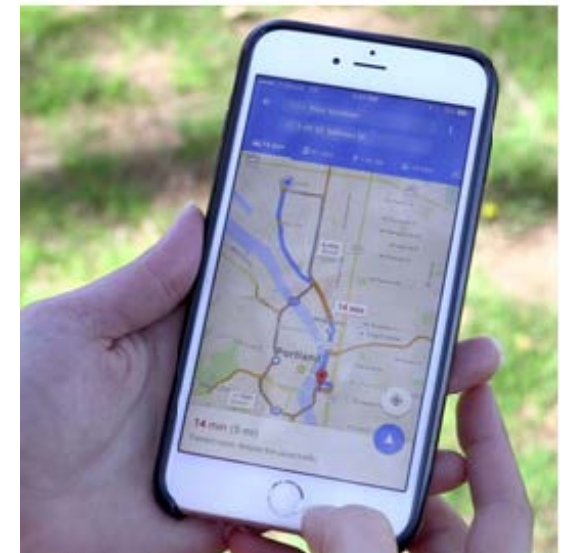
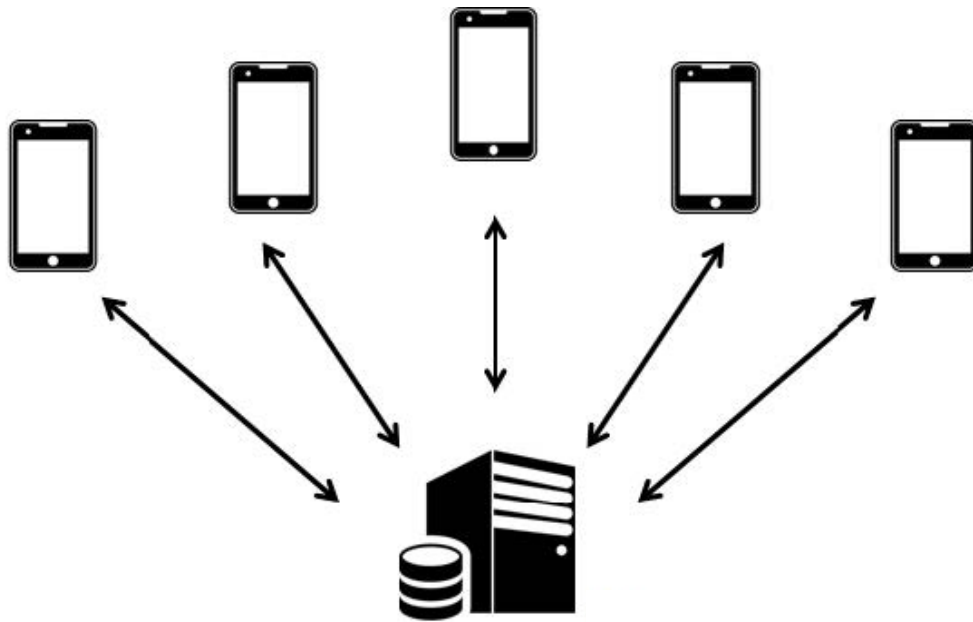


Motivation



„Jeden Tag werden mit den Google Maps 1 Milliarde Kilometer navigiert“

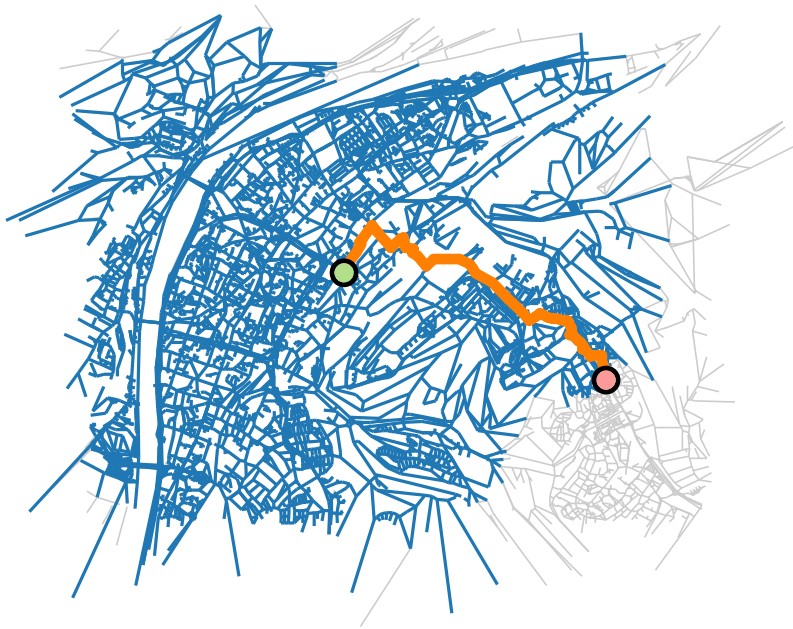
(<https://www.googlewatchblog.de/2017/05/offizielle-statistiken-google-nutzerzahlen/>)



Algorithmen für geographische Informationssysteme

2. Vorlesung Routenplanung

Teil II: Schätzfunktionen



GREEDYBESTFIRSTSEARCH



Idee:

Wir schätzen!

PHASE 1 PHASE 2 PHASE 3

Finde
gute
Schätz-
funktion

gehe immer zum
(geschätzt) nächsten
Knoten zu t

Demo.

<https://algo.uni-trier.de/demos/shortestpath.html>

GREEDY BEST-FIRST-SEARCH berechnet
schnell eine **gute** Lösung...

GBFS(WeightedGraph G , Vertices s, t)

INITIALIZE(G, s)

$Q = \text{new PriorityQueue}(V, d)$

while not $Q.\text{EMPTY}()$ do

$u = Q.\text{EXTRACTMIN}()$

 if $u == t$ then return

 foreach $v \in \text{Adj}[u]$ do

 if $v.d == \infty$ then

$v.d = \delta^*(v, t)$

$v.\pi = u$

$Q.\text{DECREASEKEY}(v, v.d)$

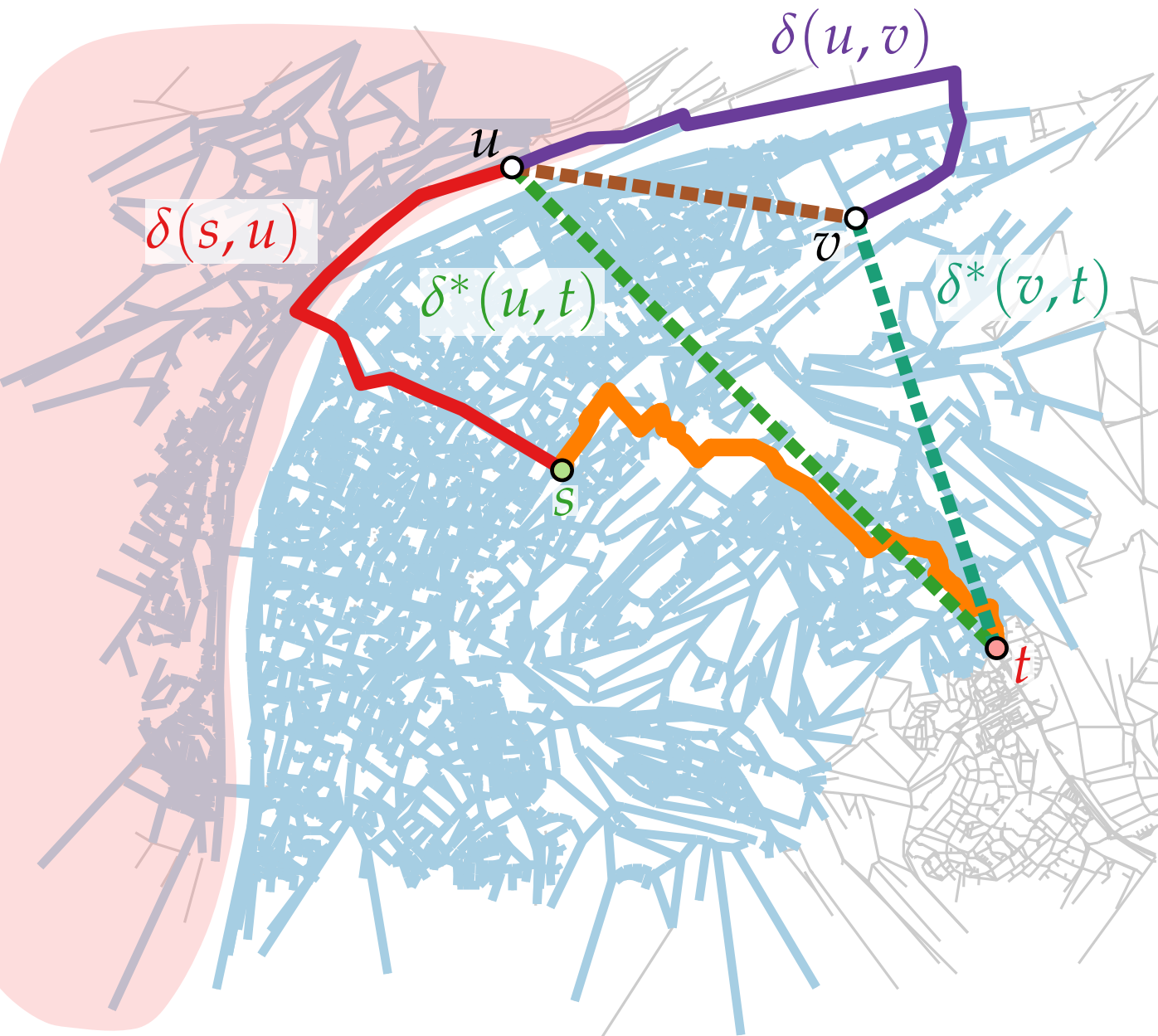
geschätzte Distanz

...die aber nicht immer **optimal** ist!



Schätzfunktionen

Ist die Suche hier sinnvoll?



$$\begin{aligned} & \cancel{\delta(s, u)} + \delta(u, t) \\ \geq & \cancel{\delta(s, u)} + \delta^*(u, t) \end{aligned}$$

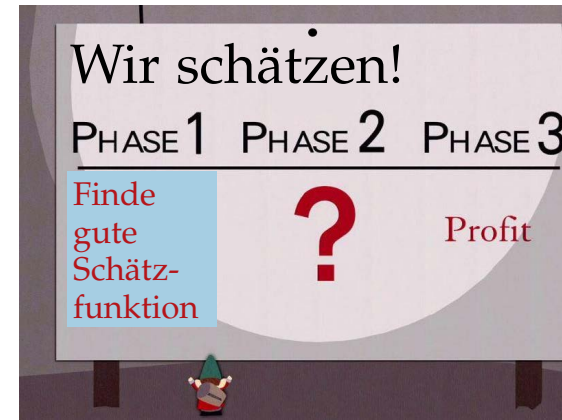
$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **optimistische** Schätzfunktion genau dann wenn $\delta(u, t) \geq \delta^*(u, t)$ für alle $u \in V$.

$$\begin{aligned} \delta(u, v) + \delta^*(v, t) & \underset{\triangle}{\geq} \delta^*(u, v) + \delta^*(v, t) \\ & \geq \delta^*(u, t) \end{aligned}$$

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **monotone** Schätzfunktion genau dann wenn $\delta(u, v) + \delta^*(v, t) \geq \delta^*(u, t)$ für alle $u, v \in V$.

Gute Schätzfunktion

Idee:



1. $\delta^*(u, t) = 0$

⊕ leicht zu berechnen

⊖ nicht aussagekräftig

2. $\delta^*(u, t) = \delta(u, t)$

⊕ sehr genau

⊖ zum Ausrechnen müssen wir das Kürzester-Pfad-Problem lösen ...

3. $\delta^*(u, t) = \delta_{\text{Euklid}}(u, t) = \sqrt{(x(u) - x(t))^2 + (y(u) - y(t))^2}$

⊕ relativ genau

⊖ funktioniert nur für Graphen mit geometrischen Informationen

4. $\delta^*(u, t) = \delta_{\text{Euklid}}^2(u, t) = (x(u) - x(t))^2 + (y(u) - y(t))^2$

⊕ relativ genau, keine Wurzel

⊖ nicht **optimistisch**

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **optimistische** Schätzfunktion genau dann wenn $\delta(u, t) \geq \delta^*(u, t)$ für alle $u \in V$.

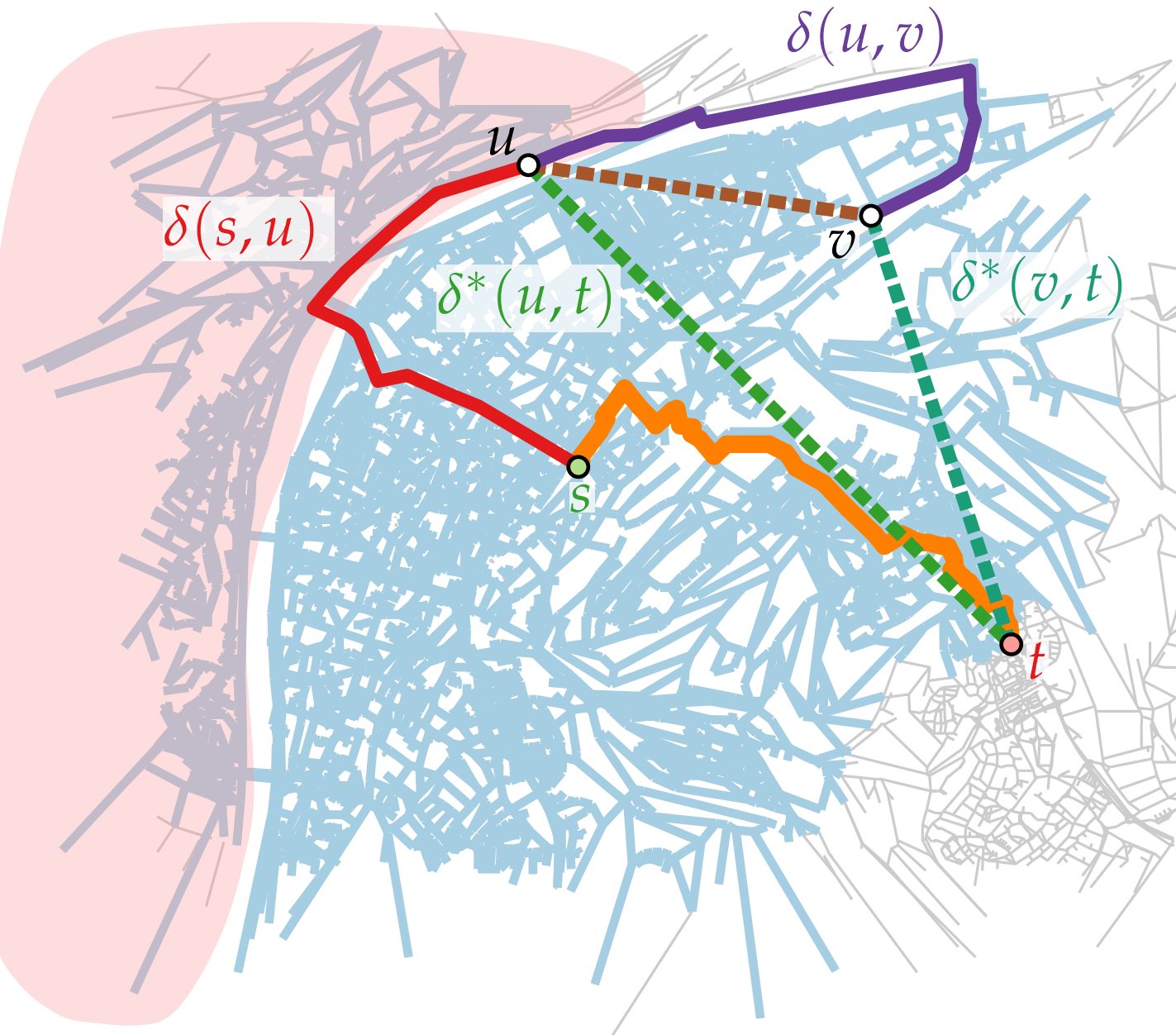
$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **monotone** Schätzfunktion genau dann wenn $\delta(u, v) + \delta^*(v, t) \geq \delta^*(u, t)$ für alle $u, v \in V$.

Neue Gewichte



$$\begin{aligned} & \cancel{\delta(s, u)} + \delta(u, t) \\ & \geq \cancel{\delta(s, u)} + \delta^*(u, t) \end{aligned}$$

Ist die Suche hier sinnvoll?



$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **optimistische** Schätzfunktion genau dann wenn $\delta(u, t) \geq \delta^*(u, t)$ für alle $u \in V$.

$$\begin{aligned} \delta(u, v) + \delta^*(v, t) & \underset{\triangle}{\geq} \delta^*(u, v) + \delta^*(v, t) \\ & \geq \delta^*(u, t) \end{aligned}$$

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **monotone** Schätzfunktion genau dann wenn $\delta(u, v) + \delta^*(v, t) \geq \delta^*(u, t)$ für alle $u, v \in V$.

Neue Kantengewichte:

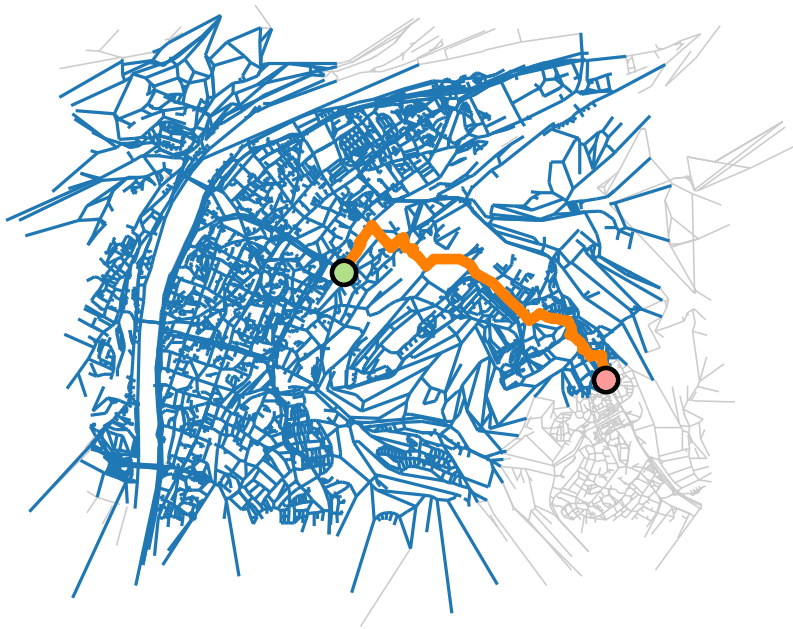
$$w'(u, v) = w(u, v) + \underbrace{\delta^*(v, t) - \delta^*(u, t)}$$

„wie viel näher kommen wir zu t ?“

Algorithmen für geographische Informationssysteme

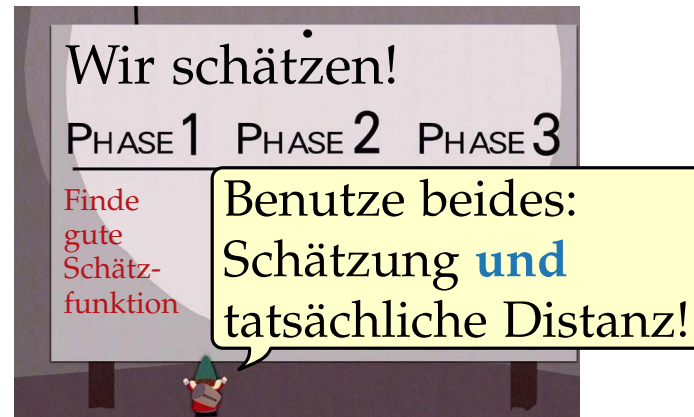
2. Vorlesung Routenplanung

Teil III: A*-Algorithmus



A*-Algorithmus

Idee:



$$w'(u, v) = w(u, v) + \delta^*(v, t) - \delta^*(u, t)$$

A*(WeightedGraph G, Vertices s, t)

INITIALIZE(G, s)

$Q = \text{new PriorityQueue}(V, d)$

while not Q.EMPT Y() **do**

$u = Q.\text{EXTRACTMIN}()$

if $u == t$ **then return**

foreach $v \in \text{Adj}[u]$ **do**

if $v.d > u.d + w'(u, v)$ **then**

$v.d = u.d + w'(u, v)$

$v.\pi = u$

 Q.DECREASEKEY($v, v.d$)

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **optimistische** Schätzfunktion genau dann wenn $\delta(u, t) \geq \delta^*(u, t)$ für alle $u \in V$.

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **monotone** Schätzfunktion genau dann wenn $\delta(u, v) + \delta^*(v, t) \geq \delta^*(u, t)$ für alle $u, v \in V$.

INITIALIZE(Graph G, Vertex s)

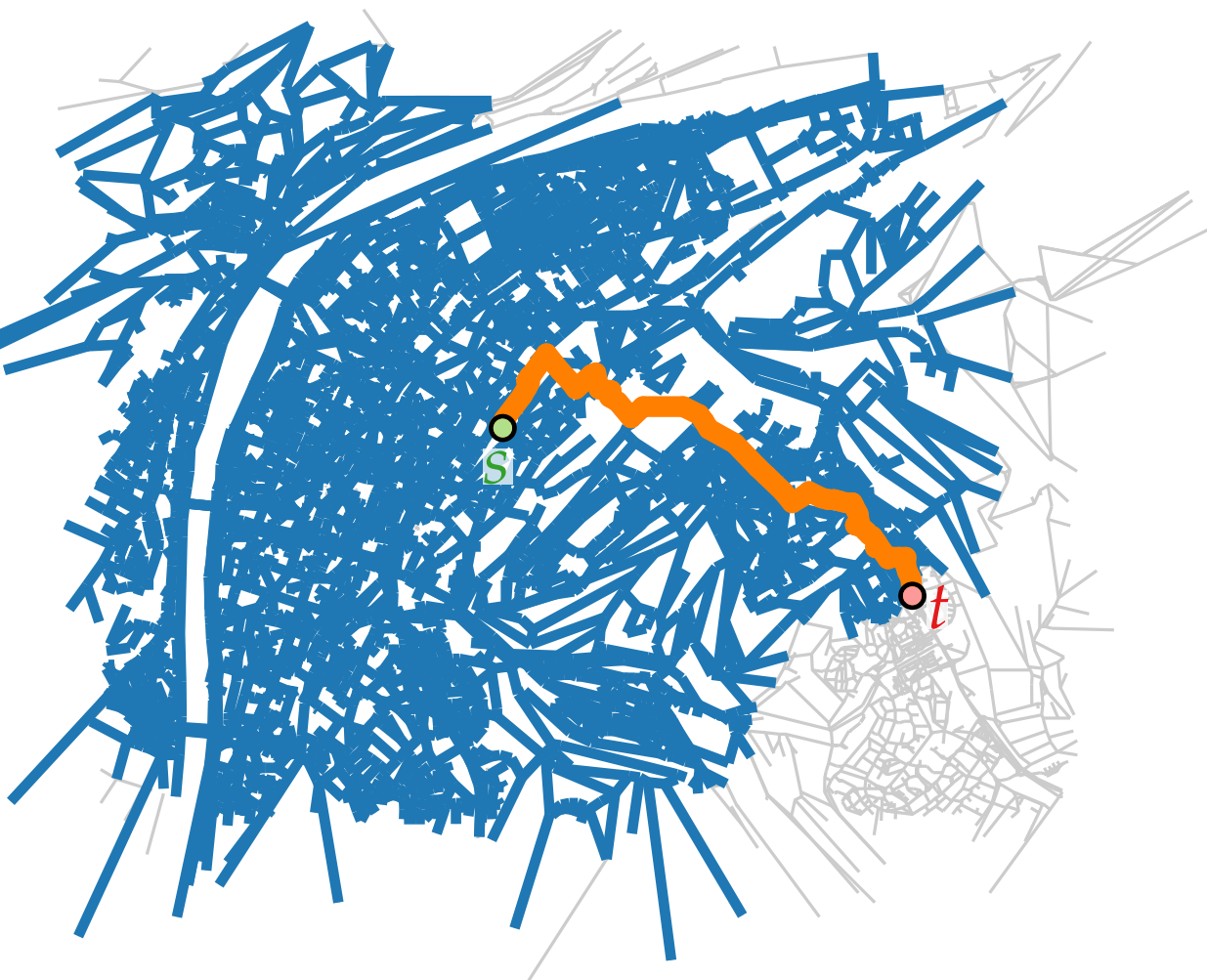
foreach $u \in V$ **do**

$u.d = \infty$

$u.\pi = \text{nil}$

$s.d = 0$

Vergleich DIJKSTRA – A^*



DIJKSTRA



A^*

Demo.

<https://algo.uni-trier.de/demos/shortestpath.html>



A^* – Korrektheit

$$w'(u, v) = w(u, v) + \delta^*(v, t) - \delta^*(u, t)$$

$$w'(P') = w(P') + \delta^*(t, t) - \delta^*(s, t)$$

$$w'(P) = w(P) + \delta^*(t, t) - \delta^*(s, t)$$

=

$$w'(v_3, t) = w(v_3, t) + \delta^*(t, t) - \cancel{\delta^*(v_3, t)}$$

+

$$w'(v_2, v_3) = w(v_2, v_3) - \cancel{\delta^*(v_3, t)} - \cancel{\delta^*(v_2, t)}$$

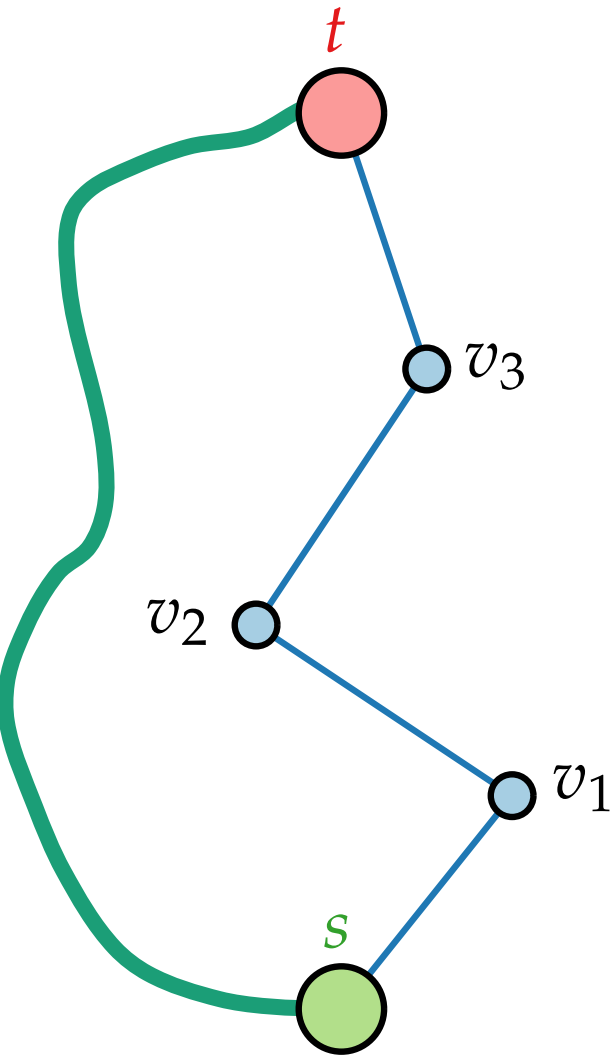
+

$$w'(v_1, v_2) = w(v_1, v_2) - \cancel{\delta^*(v_2, t)} - \cancel{\delta^*(v_1, t)}$$

+

Satz. DIJKSTRA berechnet in einem Graphen $G = (V, E; w)$ mit $w: E \rightarrow \mathbb{Q}_{\geq 0}$ in $\mathcal{O}(E + V \log V)$ Zeit den kürzesten Weg zwischen zwei Knoten.

Lemma. Ein Pfad P ist optimal bzgl. $w' \Leftrightarrow P$ ist optimal bzgl. w



kürzester s – t –Pfad P
 anderer s – t –Pfad P'



Monotone Schätzfunktion

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **optimistische** Schätzfunktion genau dann wenn $\delta(u, t) \geq \delta^*(u, t)$ für alle $u \in V$.

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **monotone** Schätzfunktion genau dann wenn $\delta(u, v) + \delta^*(v, t) \geq \delta^*(u, t)$ für alle $u, v \in V$.

Lemma. Jede **monotone** Schätzfunktion mit $\delta^*(t, t) = 0$ ist **optimistisch**.

Beweis. Betrachte beliebigen Knoten $u \in V$.

Wähle $v = t$.

Dann gilt $\delta(u, t) + \delta^*(t, t) \geq \delta^*(u, t)$
 $\Rightarrow \delta(u, t) \geq \delta^*(u, t)$ $\square$



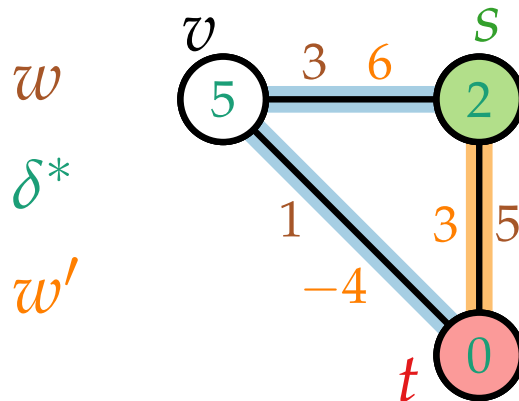
Monotone Schätzfunktion

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **optimistische** Schätzfunktion genau dann wenn $\delta(u, t) \geq \delta^*(u, t)$ für alle $u \in V$.

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **monotone** Schätzfunktion genau dann wenn $\delta(u, v) + \delta^*(v, t) \geq \delta^*(u, t)$ für alle $u, v \in V$.

Lemma. Wenn δ^* **nicht monoton** ist, dann kann es sein, dass A^* nicht den kürzesten Pfad findet.

Beweis.



$$\underset{1}{\delta(v, t)} + \underset{0}{\delta^*(t, t)} < \underset{5}{\delta^*(v, t)} \quad \text{also nicht monoton}$$

Lösung A^* : Länge 5

Kürzester Pfad: Länge 4



Monotone Schätzfunktion

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **optimistische** Schätzfunktion genau dann wenn $\delta(u, t) \geq \delta^*(u, t)$ für alle $u \in V$.

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **monotone** Schätzfunktion genau dann wenn $\delta(u, v) + \delta^*(v, t) \geq \delta^*(u, t)$ für alle $u, v \in V$.

Lemma. Wenn δ^* **monoton** ist, dann sind die Kantengewichte w' nicht-negativ.

Beweis. $w'(u, v) = w(u, v) + \delta^*(v, t) - \delta^*(u, t) \geq \delta(u, v) + \delta^*(v, t) - \delta^*(u, t) \geq 0$ □



Monotone Schätzfunktion

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **optimistische** Schätzfunktion genau dann wenn $\delta(u, t) \geq \delta^*(u, t)$ für alle $u \in V$.

$\delta^* : V \times V \rightarrow \mathbb{R}$ ist eine **monotone** Schätzfunktion genau dann wenn $\delta(u, v) + \delta^*(v, t) \geq \delta^*(u, t)$ für alle $u, v \in V$.

Lemma. Ein Pfad P ist optimal bzgl. w' $\Leftrightarrow P$ ist optimal bzgl. w

Lemma. Jede **monotone** Schätzfunktion mit $\delta^*(t, t) = 0$ ist **optimistisch**.

Lemma. Wenn δ^* **nicht monoton** ist, dann kann es sein, dass A^* nicht den kürzesten Pfad findet.

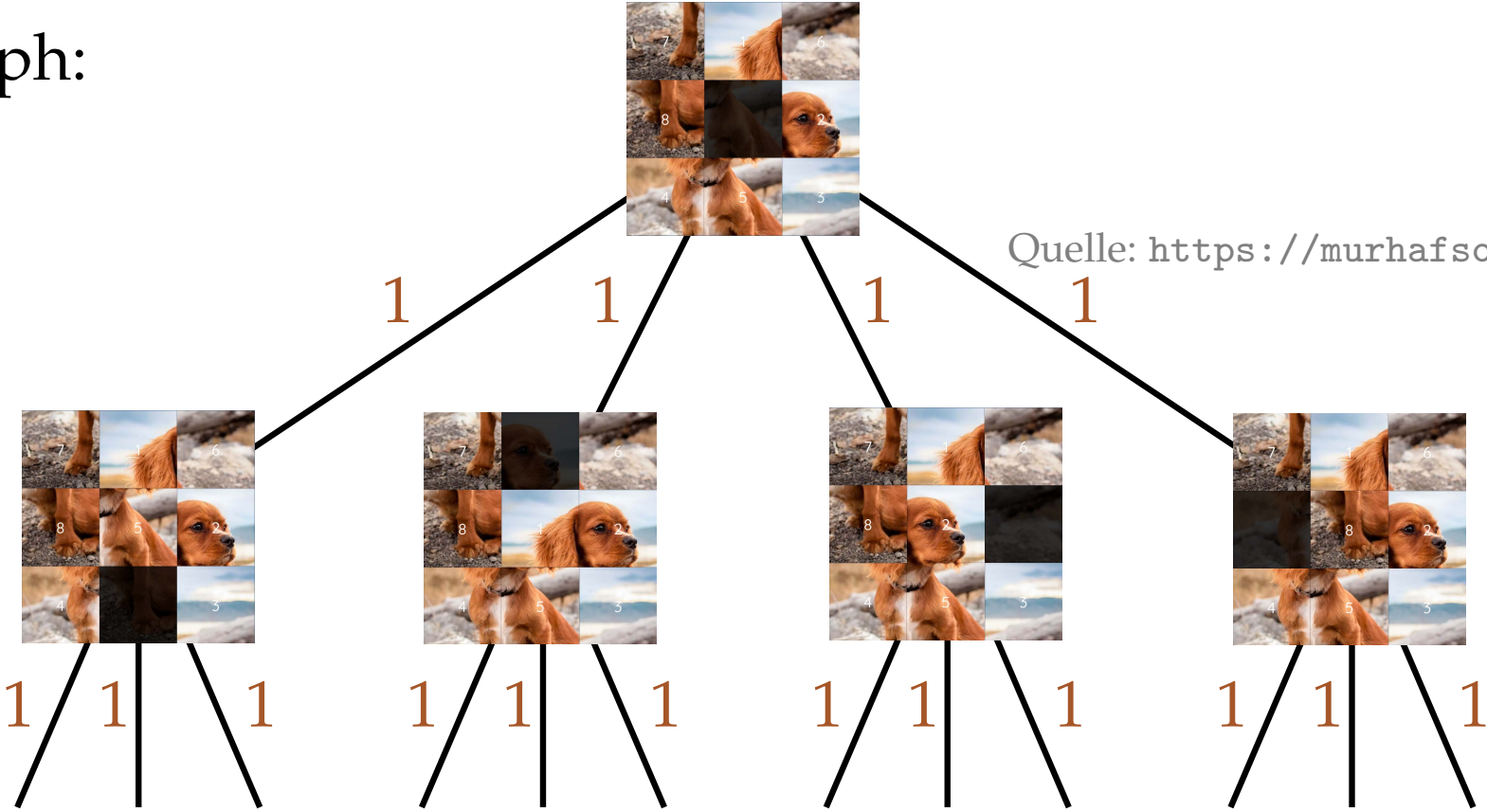
Lemma. Wenn δ^* **monoton** ist, dann sind die Kantengewichte w' nicht-negativ.

Satz. Wenn δ^* **monoton** ist, dann berechnet A^* in einem Graphen $G = (V, E; w)$ mit $w: E \rightarrow \mathbb{Q}_{\geq 0}$ in $\mathcal{O}(E + V \log V)$ Zeit den kürzesten Weg zwischen zwei Knoten. (gilt sogar, wenn δ^* „nur“ **optimistisch** ist.)

Nicht-geometrische Probleme: Schiebepuzzle



Modellierung als Graph:



Quelle: <https://murhafsousli.github.io/8puzzle>

Lösung: kürzester Weg von  nach 

Schätzfunktion?
#Kacheln an falscher Position

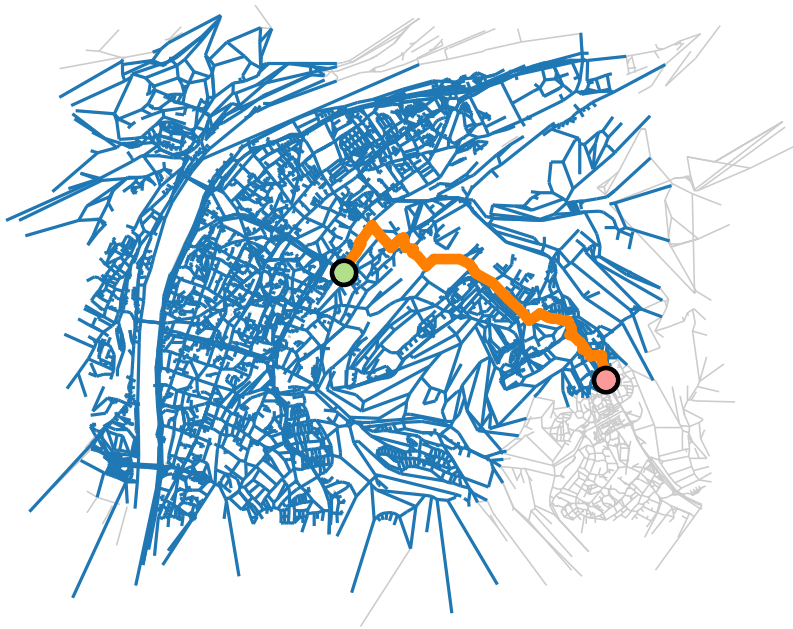
Durchschnittliche Anzahl besuchter Knoten wenn kürzester Pfad Länge x hat:

	$x = 4$	$x = 8$	$x = 12$
DIJKSTRA	112	6 300	3 600 000
A^*	13	39	227

Algorithmen für geographische Informationssysteme

2. Vorlesung Routenplanung

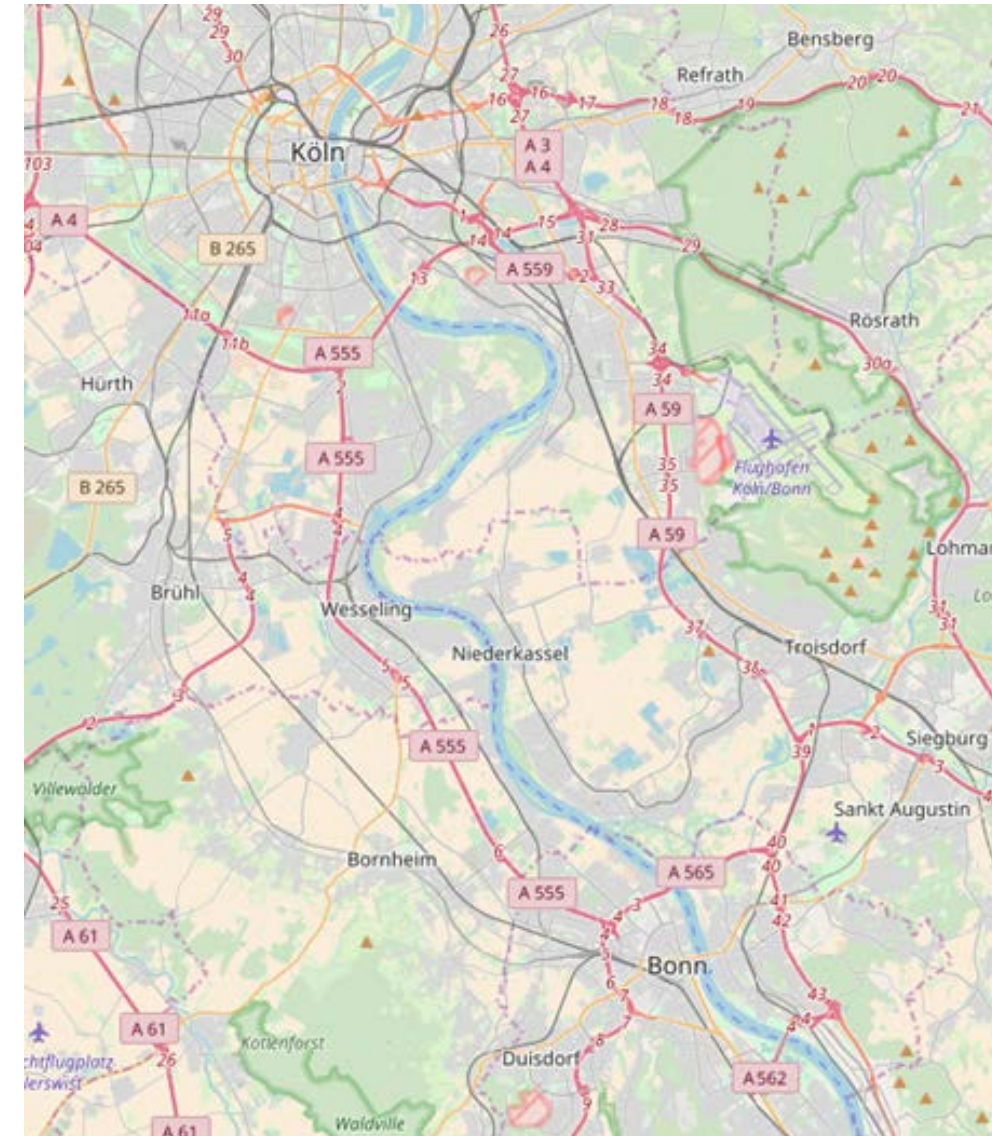
Teil IV: Contraction Hierarchies





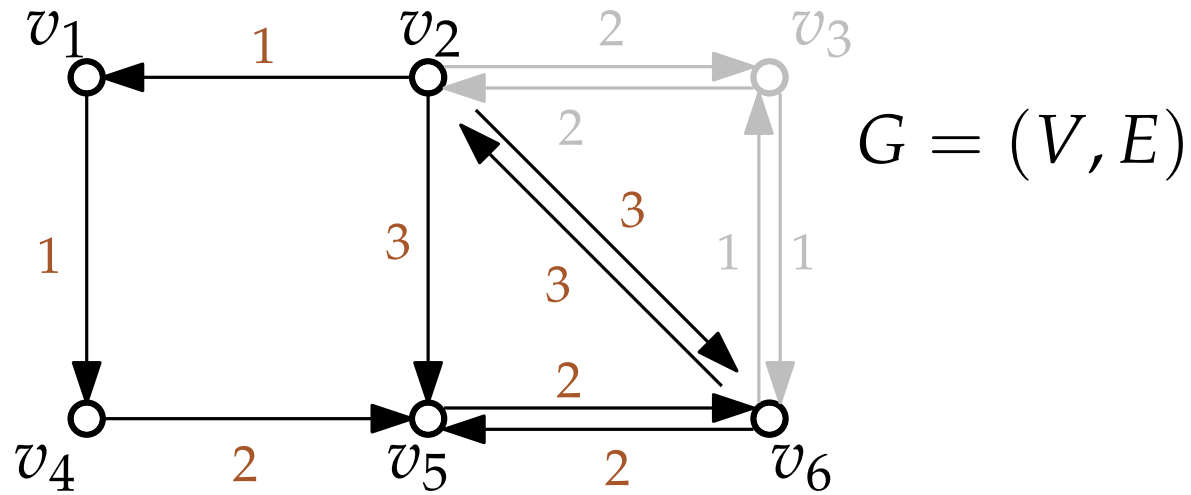
Contraction Hierarchies – Idee

- DIJKSTRA zu langsam für Server Queries
- A* funktioniert nicht für beliebige Kostenfunktionen
- **Idee:** Informationen vorberechnen um Queries zu beschleunigen
- Time–Space Tradeoff
- Es gibt eine Hierarchie:
Autobahn → Bundesstraße → Landstraße
→ Kreisstraße → Gemeindestraße → Feldwege
- Entferne nach und nach unwichtigste Knoten
- Füge Abkürzungen in das Straßennetz ein, um kürzeste Wege aufrechtzuerhalten





Kontraktion



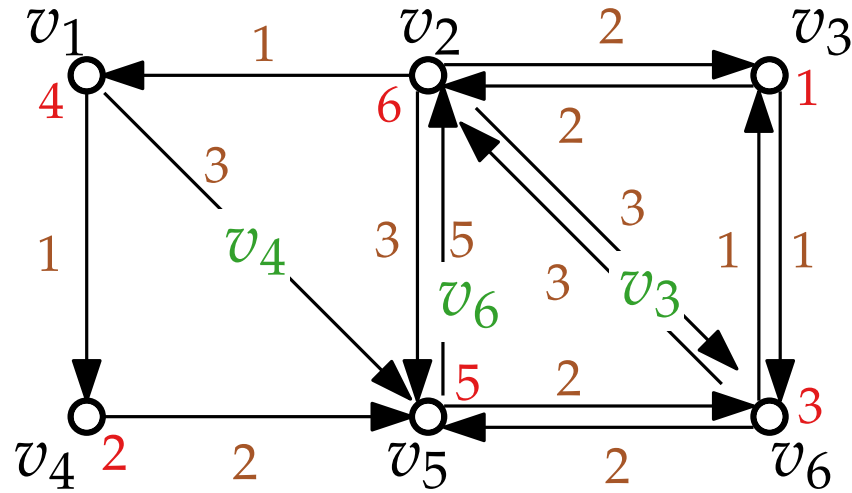
Kontraktion eines Knotens $v \in V$, z.B. v_3 :

- v und inzidente Kanten werden aus G entfernt (*deaktiviert*)
- Für jeden kürzesten Pfad uvw wird eine neue Kante uw mit entsprechendem Gewicht eingefügt.



Contraction Hierarchies – Definition

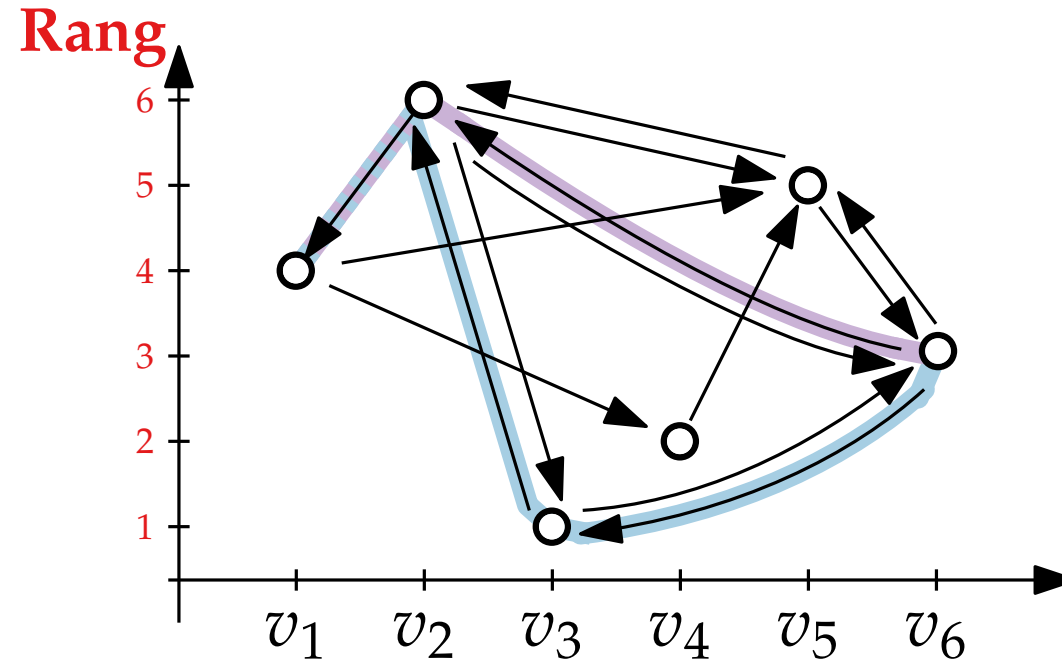
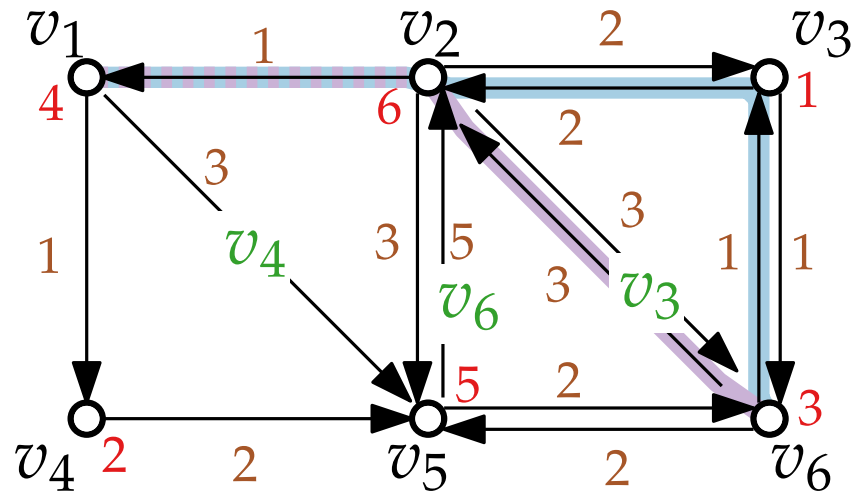
$$G' = (V, E')$$



- Knoten werden in bestimmter Reihenfolge kontrahiert,
z.B. $v_3, v_4, v_6, v_1, v_5, v_2$.
- Für jeden kontrahierten Knoten speichern wir den **Rang**,
d.h., **wann** der Knoten kontrahiert wurde.
- Für jede neue Kante speichern wir, **welche Knoten**
übersprungen wurden.

Contraction Hierarchies – Eigenschaften

$$G' = (V, E')$$



Für zwei Knoten $u, v \in V$, sei P_{uv} ein kürzester u - v -Pfad in G .

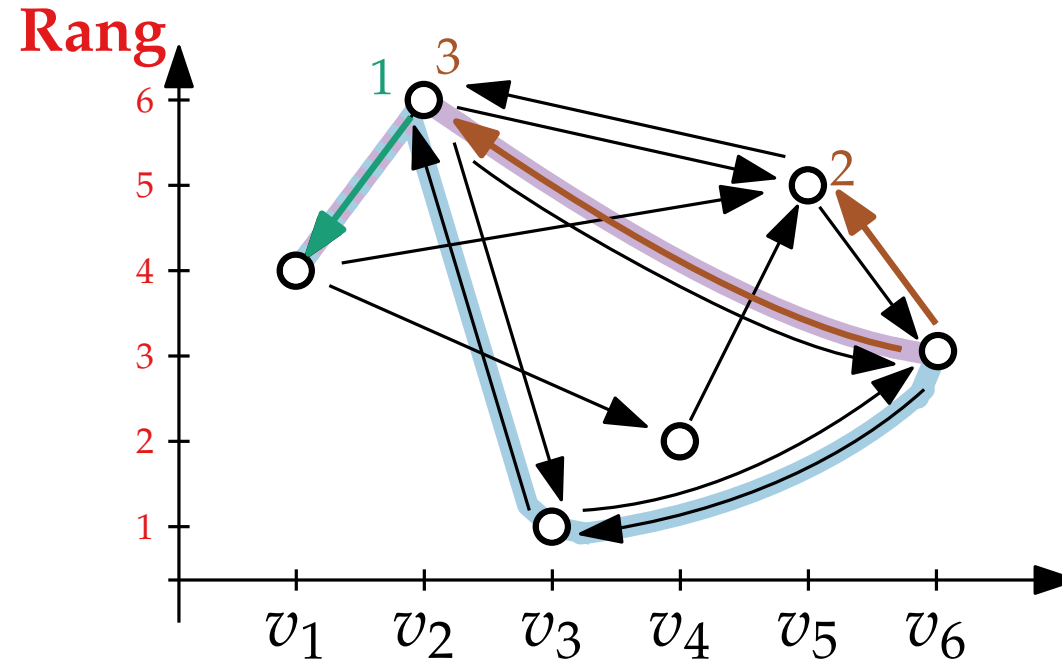
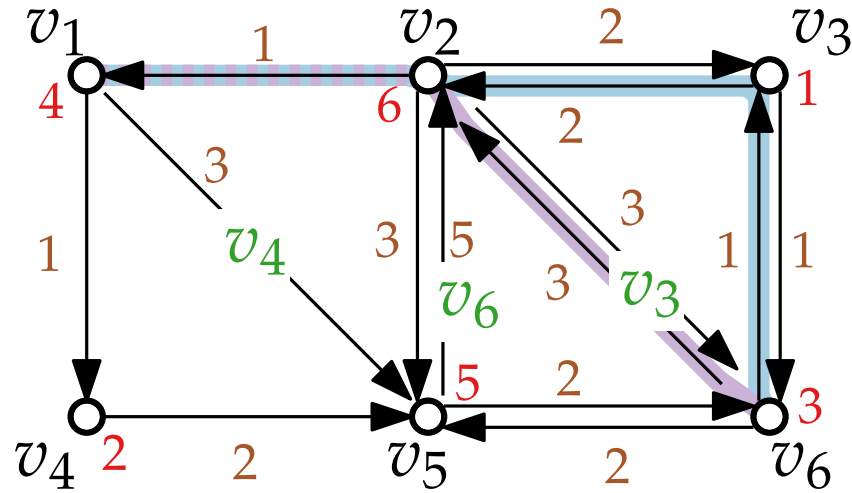
Es gibt einen kürzesten u - v Pfad P'_{uv} in G' , so dass:

- die **Ränge** der Knoten steigen zuerst und sinken dann,
- man erhält den Pfad P_{uv} aus P'_{uv} , indem man nach und nach die gespeicherten Abkürzungen auflöst.



Contraction Hierarchies – Benutzung

$$G' = (V, E')$$



P_{uv} berechnen:

1. Berechne P'_{uv} .
2. Transformiere P'_{uv} zu P_{uv} .

1. Benutze DIJKSTRA von u aus mit nur aufsteigenden Kanten
2. Benutze DIJKSTRA von v aus mit nur absteigenden Rückwärtskanten
3. „Wendepunkt“ hat niedrigste kumulative Distanz ($1 + 3 = 4$).

Contraction Hierarchies – Anmerkungen



- Die beiden DIJKSTRA-Durchläufe können gleichzeitig anstelle von nacheinander durchgeführt werden, indem eine *bidirektionale Suche* verwendet wird. Dadurch können noch mehr Knoten unerforscht bleiben.
- Die Methode funktioniert für jede Kontraktionssequenz, aber die Wahl der Knotenreihenfolge beeinflusst die Laufzeit erheblich. Es existieren mehrere Methoden zur Optimierung der Reihenfolge.