



Julius-Maximilians-  
**UNIVERSITÄT**  
**WÜRZBURG**

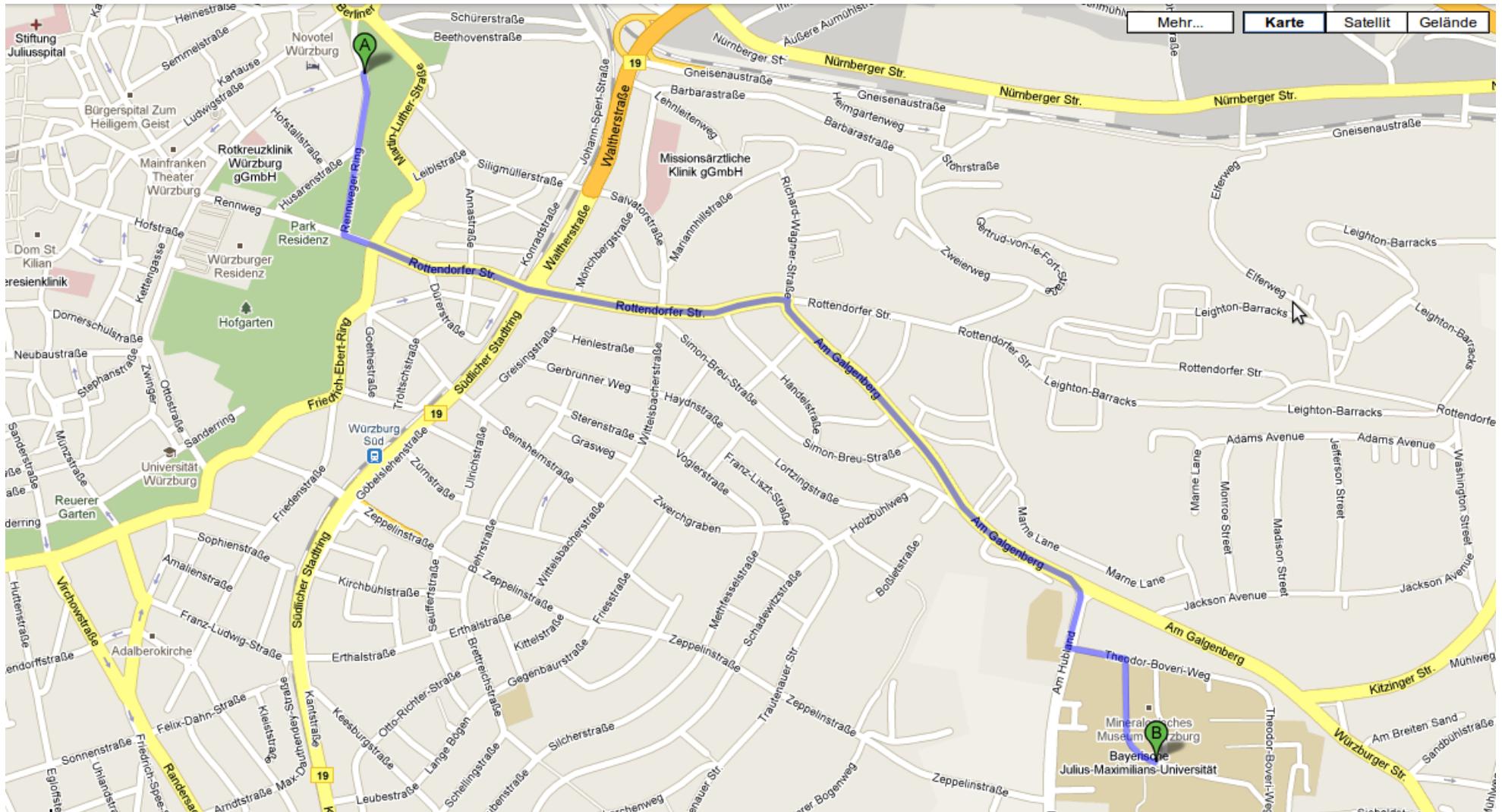


# Algorithmen und Datenstrukturen

Wintersemester 2020/21  
19. Vorlesung

Kürzeste Wege & Dijkstras Algorithmus

# Wozu kürzeste Wege?



# Modellierung des Problems Routenplanung

Straßenkreuzung → Knoten

Straßenabschnitt → zwei entgegengerichtete Kanten

Einbahnstraßenabschnitt → in Fahrtrichtung gerichtete Kante

Fahrtzeit für Abschnitt  $e$  → Kantengewicht  $w(e) \geq 0$

Straßennetz → gerichteter, gewichteter und zusammenhängender Graph  $G = (V, E)$

Start → Knoten  $s \in V$

Ziel → Knoten  $t \in V$

Start-Ziel-Route →  $s$ - $t$ -Weg, d.h. Folge von Kanten  $(s, v_1), (v_1, v_2), \dots, (v_k, t)$  in  $G$

# Wozu kürzeste Wege?



# Wozu kürzeste Wege? (II)

**DB BAHN**

Kontakt | Hilfe | Sitemap a a+ a++ Frage oder Suchbegriff eingeben ... Suchen

Startseite | Angebotsberatung | **Fahrplan & Buchung** | Services | BahnCard | Urlaub Meine Bahn

Suche  Auswahl  Ticket&Reservierung  Zahlung  Buchung  Bestätigung → Neue Anfrage

**Reisedaten** 1 Erwachsener, 2. Klasse

**Hinfahrt** von Würzburg Hbf Di, 12.01.10 10:23 Abfahrt  Ankunft → weitere Angaben ändern Aktualisieren

nach Mathematisches Institut, Würzburg

**Kurzfristige Fahrplanänderungen**  
Informieren Sie sich hier über aktuelle Verkehrsmeldungen.  
→ Weitere Informationen

Ihre Hinfahrtmöglichkeiten - sortiert nach **Abfahrt** Druckansicht

Bahnhof/Haltestelle	Datum	Zeit	Dauer	Umst.	Produkte	Normalpreis
		<b>↑ Früher</b>				
Würzburg Busbahnhof Mathematisches Institut, Würzburg	Di, 12.01.10	ab 10:29	0:16	0	Bus	Preisauskunft nicht möglich
	Di, 12.01.10	an 10:45				<span style="float: right;">→ Rückfahrt hinzufügen</span>
Würzburg Busbahnhof Mathematisches Institut, Würzburg	Di, 12.01.10	ab 10:49	0:16	0	Bus	Preisauskunft nicht möglich
	Di, 12.01.10	an 11:05				<span style="float: right;">→ Rückfahrt hinzufügen</span>
Würzburg Busbahnhof Mathematisches Institut, Würzburg	Di, 12.01.10	ab 11:09	0:16	0	Bus	Preisauskunft nicht möglich
	Di, 12.01.10	an 11:25				<span style="float: right;">→ Rückfahrt hinzufügen</span>
Details für alle anzeigen		<b>↓ Später</b>				

MobilCheck UmweltMobilCheck

**Zurück**

# Was ist das Problem?

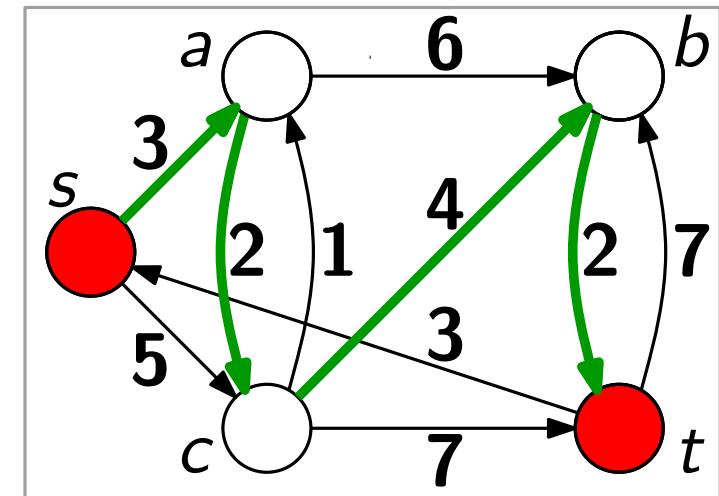
## Eingabe:

- gerichteter, zusammenhängender Graph  $G = (V, E)$  mit nicht-negativen **Kantengewichten**  $w: E \rightarrow \mathbb{Q}_0^+$ ,
- Knoten  $s$  und  $t$

## Ausgabe:

- kürzester  $s-t$ -Weg  $W$  in  $G$ , d.h.  $\sum_{e \in W} w(e)$  minimal.

Darstellung durch Vorgänger-Zeiger  $\pi$ : für jeden Knoten  $v$  sei  $\pi(v) \in V \cup \{nil\}$  Vorgänger von  $v$  auf kürzestem  $s-v$ -Weg.



# Was ist das Problem?

## Eingabe:

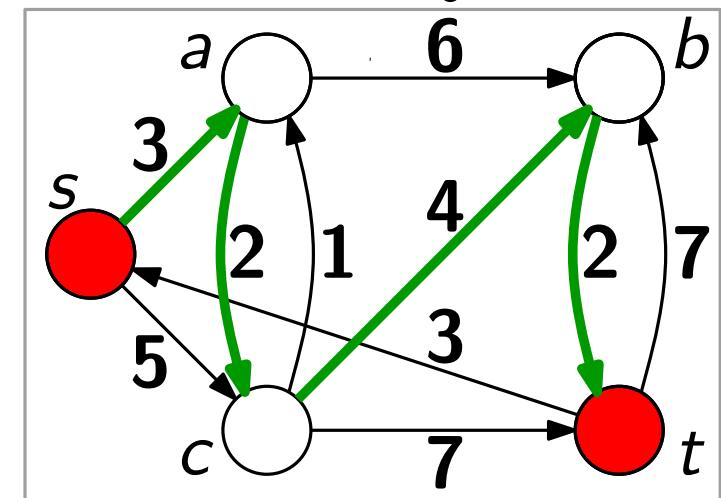
- gerichteter, zusammenhängender Graph  $G = (V, E)$  mit nicht-negativen **Kantengewichten**  $w: E \rightarrow \mathbb{Q}_0^+$ ,
- Knoten  $s$  und  $t$

## Ausgabe:

- kürzester  $\cancel{s-t}$ -Weg  $W_t$  in  $G$ ,  $\underbrace{\text{für alle } t \in V}_{\text{d.h. } \sum_{e \in W} w(e) \text{ minimal.}}$

Darstellung durch Vorgänger-Zeiger  $\pi$ : für jeden Knoten  $v$  sei  $\pi(v) \in V \cup \{nil\}$  Vorgänger von  $v$  auf kürzestem  $s-v$ -Weg.

**Nebenbemerkung:** Analoges Berechnungsverfahren?



# Dijkstra – BFS mit Gewichten

Dijkstra(WeightedGraph  $G = (V, E; w)$ , Vertex  $s$ )

Initialize( $G, s$ ) // Gewichtung

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

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

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

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

    Relax( $u, v; w$ )

$u.\text{color} = \text{black}$

Relax( $u, v; w$ )

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

$v.\text{color} = \text{gray}$

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

$v.\pi = u$

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

BFS(Graph  $G$ , Vertex  $s$ )

Initialize( $G, s$ )

$Q = \text{new Queue}()$

$Q.\text{Enqueue}(s)$

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

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

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

**if**  $v.\text{color} == \text{white}$  **then**

$v.\text{color} = \text{gray}$

$v.d = u.d + 1$

$v.\pi = u$

$Q.\text{Enqueue}(v)$

$u.\text{color} = \text{black}$

# Dijkstra – ein Beispiel

Dijkstra(WeightedGraph  $G$ , Vertex  $s$ )

Initialize( $G, s$ )

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

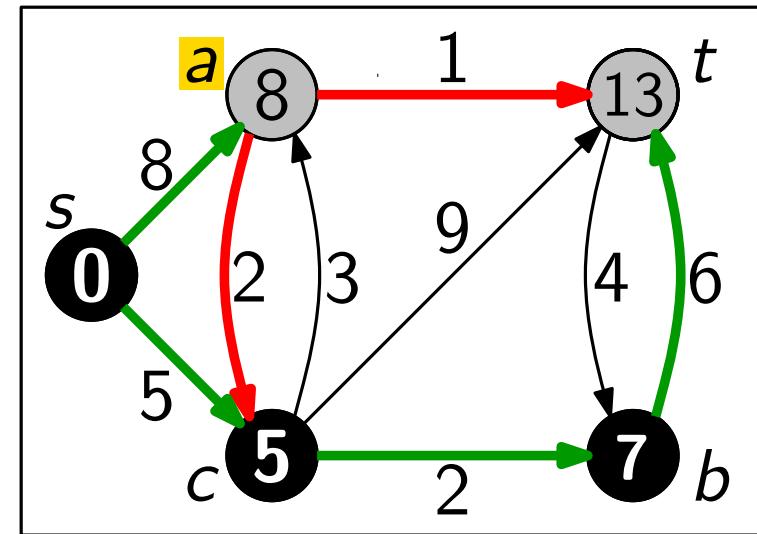
**while** not  $Q.\text{Empty}()$  **do**

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

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

      └ Relax( $u, v; w$ )

    └  $u.\text{color} = \text{black}$



Relax( $u, v; w$ )

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

$v.\text{color} = \text{gray}$

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

$v.\pi = u$

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

Initialize(Graph  $G$ , Vertex  $s$ )

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

$u.\text{color} = \text{white}$

$u.d = \infty$

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

$s.\text{color} = \text{gray}$

$s.d = 0$

# Dijkstra – ein Beispiel

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**

      └ Relax( $u, v; w$ )

    └  $u.\text{color} = \text{black}$

Relax( $u, v; w$ )

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

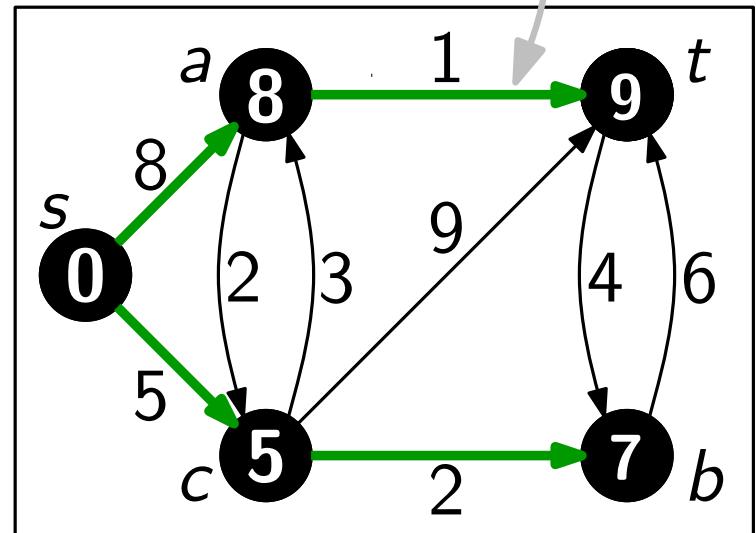
$v.\text{color} = \text{gray}$

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

$v.\pi = u$

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

Kürzester-Wege-Baum  
mit Wurzel  $s$



Initialize(Graph  $G$ , Vertex  $s$ )

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

$u.\text{color} = \text{white}$

$u.d = \infty$

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

$s.\text{color} = \text{gray}$

$s.d = 0$

# Dijkstra – die Laufzeit

Dijkstra(WeightedGraph  $G$ , Vertex  $s$ )

Abk. für  $O(|V|)$

Initialize( $G, s$ )

$O(V)$  Zeit

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

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

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

genau  $|V|$  mal

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

      Relax( $u, v; w$ )

Wie oft wird Relax aufgerufen?

$u.\text{color} = \text{black}$

Relax( $u, v; w$ )

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

Für jeden Knoten  $u \in V$  genau  $|\text{Adj}[u]| = \deg u^{(\text{out}-)}$  mal,

$v.\text{color} = \text{gray}$

also insg.  $\Theta(E)$  mal.

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

$v.\pi = u$

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

Also wird DecreaseKey  $O(E)$  mal aufgerufen.

# Dijkstra – die Laufzeit

**Satz.** Gegeben ein Graph  $G = (V, E)$ , läuft Dijkstras Alg. in  $O(V \cdot T_{\text{ExtractMin}}(|V|) + E \cdot T_{\text{DecreaseKey}}(|V|))$  Zeit.

Implementierung einer PriorityQueue	$T_{\text{ExtractMin}}(n)$	$T_{\text{DecreaseKey}}(n)$	$T_{\text{Dijkstra}}( V ,  E )$
als unsortiertes Feld	$O(n)$	$O(1)^*$	$O(V^2 + E)$
als Heap	$O(\log n)$	$O(\log n)^{**}$	$O((E + V) \log V)$
als Fibonacci-Heap	$O(\log n)$ <i>amortisiert</i>	$O(1)$ <i>amortisiert</i>	$O(E + V \log V)$ <i>im Worst-Case!</i>

\* ) Das geht, weil wir bei ExtractMin Lücken im Feld lassen; daher bleiben die Schlüssel an ihrem Platz ( $\rightarrow$  Direktzugriff)

\*\*) Das geht, obwohl wir im Heap nicht suchen können (!). Wir merken uns ständig für jeden Knoten, wo er im Heap steht.

**Korollar.** In einem Graphen  $G = (V, E; w)$  mit  $w: E \rightarrow \mathbb{Q}_{\geq 0}$  kann man in  $O(E + V \log V)$  Zeit die kürzesten Wege von einem zu allen Knoten berechnen (SSSP-Problem).

# Dijkstra – die Korrektheit

siehe [CLRS], Kapitel 24.3., Satz 24.6:  
Korrektheitsbeweis mittels Schleifeninvariante.

oder

MIT-Vorlesungsmitschnitt von Erik Demaine:  
[http://videolectures.net/mit6046jf05\\_demaine\\_lec17](http://videolectures.net/mit6046jf05_demaine_lec17)

# Wozu kürzeste Wege? (III) – SMSen

G**H**I

M**N**O

D**E**F

M**N**O

P**Q**R**S**

M**N**O

A**B**C

T**U**V

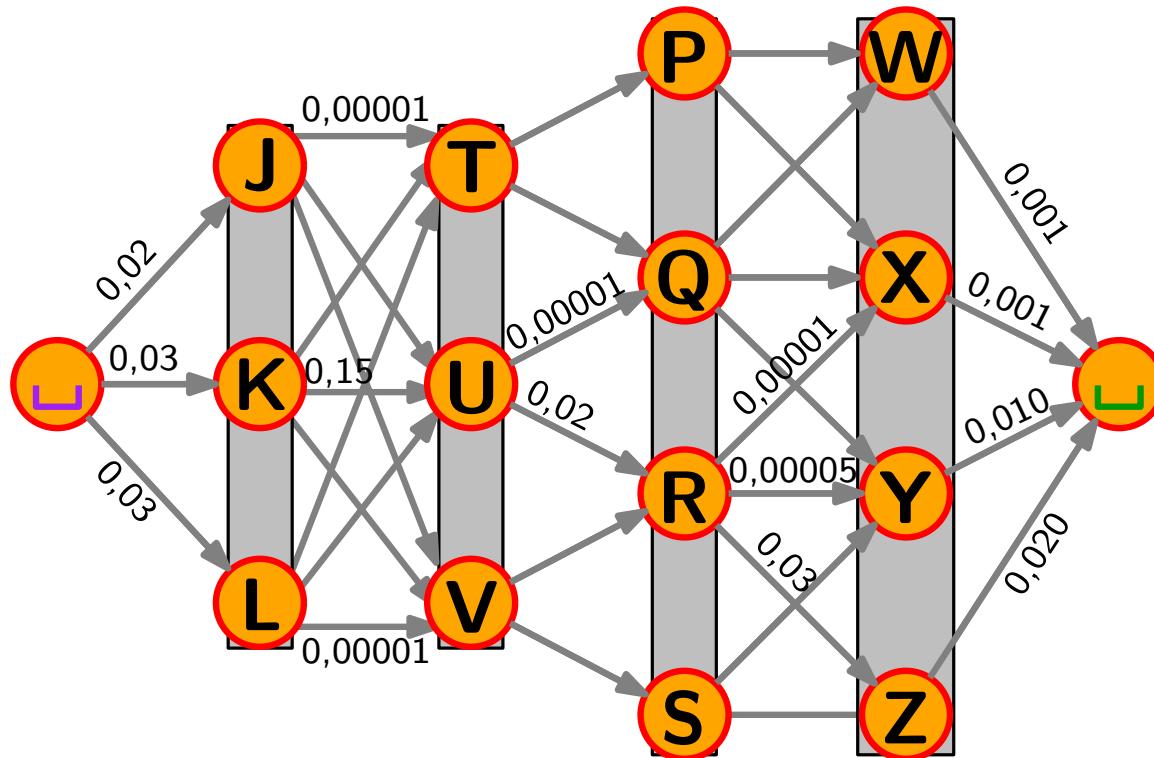
G**H**I

J**K**L

10:21 für T9



# Modellierung – SMSen



- Graph:** Knoten  $\hat{=}$  Buchstaben  
 Kanten  $\hat{=}$  aufeinanderfolgende Buchst.  
 Gewichte  $\hat{=}$  Wahrscheinlichkeiten  $w$  / Häufigkeiten
- Gesucht:** Weg  $P$  von  $\textcolor{violet}{\square}$  nach  $\textcolor{green}{\square}$  mit *größter WK* ( $= \prod_{e \in P} w(e)$ )
- Lösung:** *dynamisches Programmieren... [kommt noch!]*

# Literatur

- **A note on two problems in connexion with graphs.**

Edsger Wybe Dijkstra:

*Numerische Mathematik*,

Band 1, S. 269–271, 1959.

Lesen Sie  
mal rein!

- **Das Geheimnis des kürzesten Weges.**

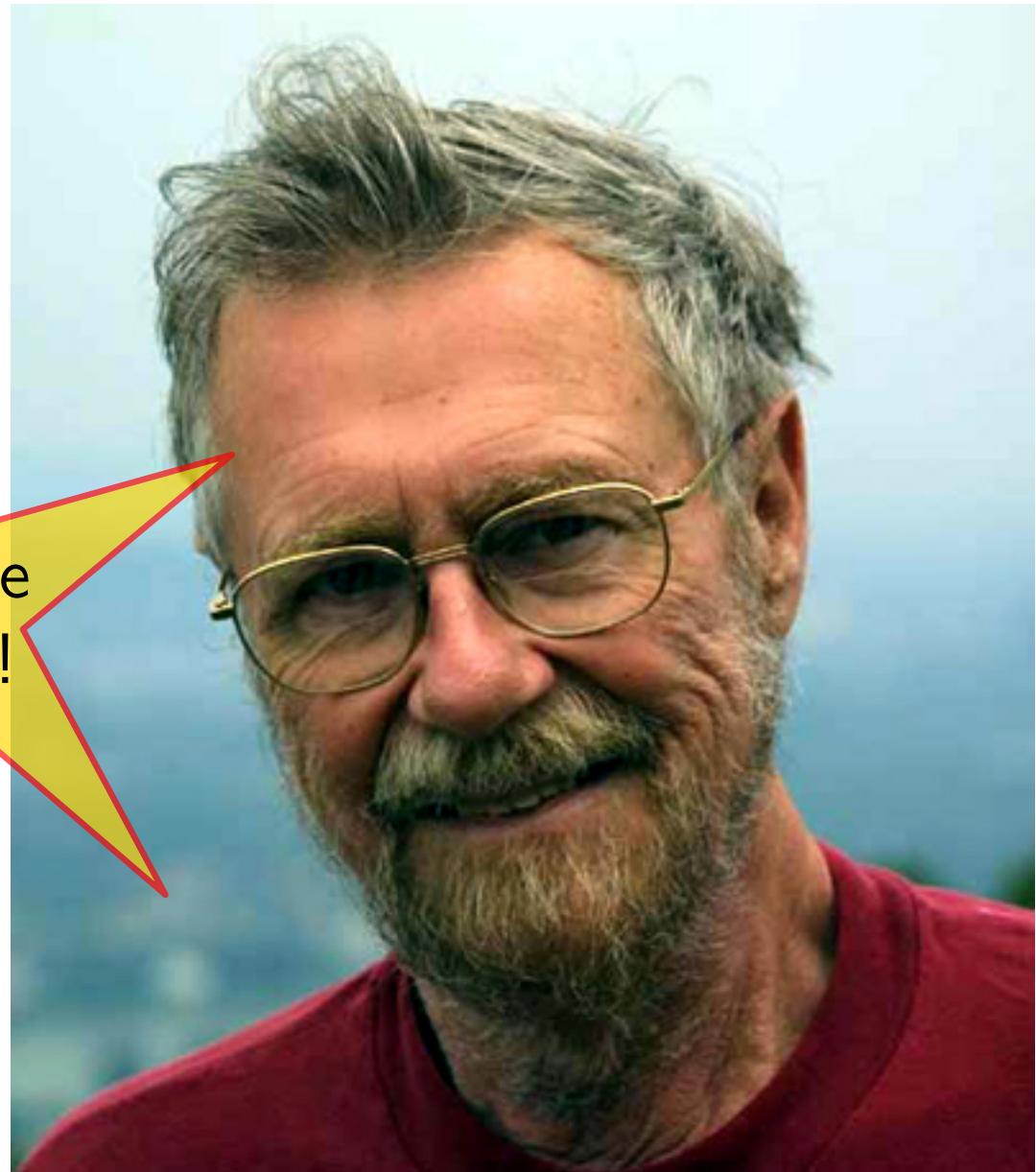
Ein mathematisches Abenteuer.

Peter Gritzmann und

René Brandenberg:

*Springer-Verlag*, 3. Aufl., 2005.

Beide Werke sind über die UB  
*frei zugänglich* und über unsere  
WueCampus-Seite verlinkt!



Edsger Wybe Dijkstra  
\* 1930 in Rotterdam  
† 2002 in Nuenen, Niederlande

# Kürzeste Wege nach Dijkstra

<i>Eingabe</i>	<i>Algorithmus</i>	<i>Laufzeit</i>
ungewichteter Graph	Breitensuche	$O(E + V)$ <i>letztes Mal</i> ✓
nicht-neg. Kantengew.	Dijkstra	$O(E + V \log V)$ ✓
azyklischer Graph	topol. Sortieren	$O(E + V)$ <i>nächstes Mal!</i> ✓
negative Kantengew.	Bellman-Ford	$O(EV)$ Vorlesung Adv. Algorithms (M.Sc.)
für alle Knotenpaare	$ V  \times$ Dijkstra	$O(V(E + V \log V))$ ✓
+ negative Kantengew.	Floyd-Warshall	$O(V^3)$
	Johnson	$O(V(E + V \log V))$
$k$ kürzeste $s-t$ -Wege	Eppstein	$O(k + E + V \log V)$