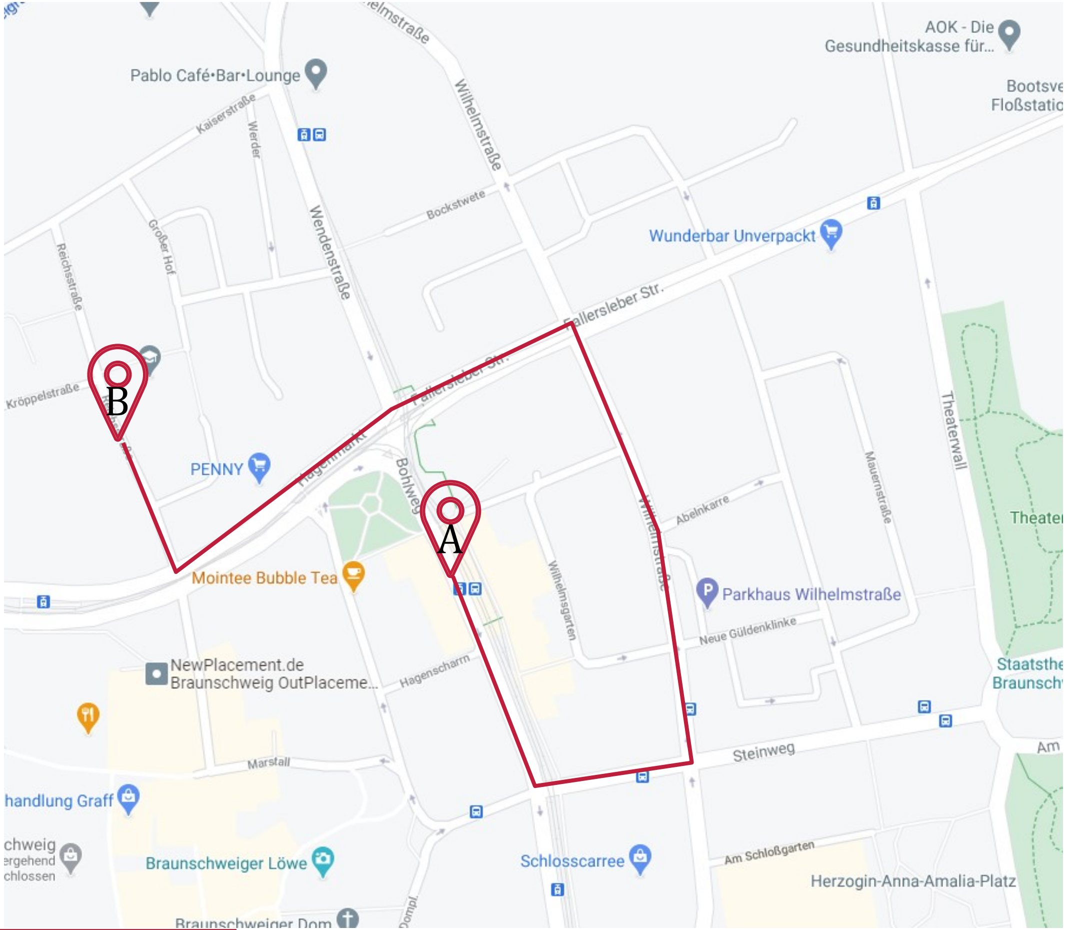
Netzwerkalgorithmen – Vorlesung #4

Arne Schmidt

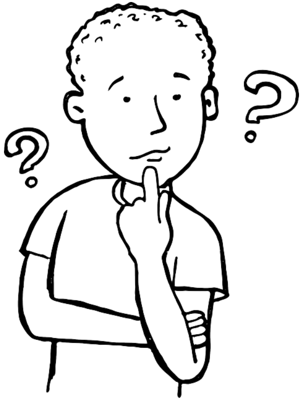
3 Kürzeste Wege



Wie gelang man von A nach B?

Problematisch: Einbahnstraßen!
⇒ Gerichtete Graphen

Gibt es einen kürzeren Weg?
Was heißt *kurz*?



Problem

Problem 3.1: Kürzester Weg Problem (Shortest Path, SP)

Gegeben:

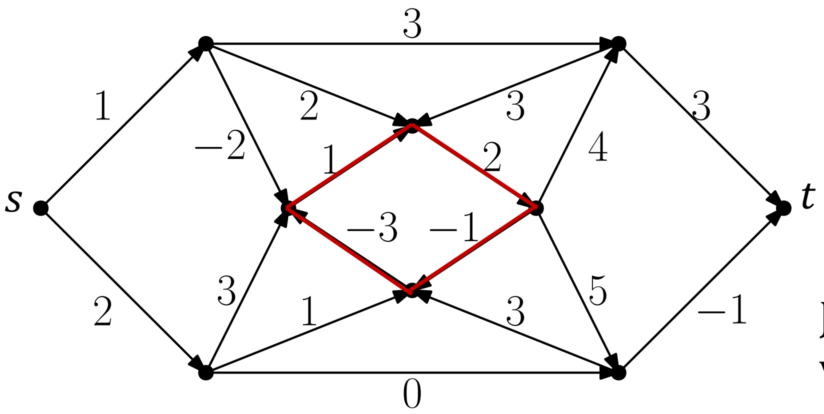
Digraph $D = (V, A)$

Kostenfunktion $c\colon\ A \to \mathbb{R}$

Knoten $s, t \in V$

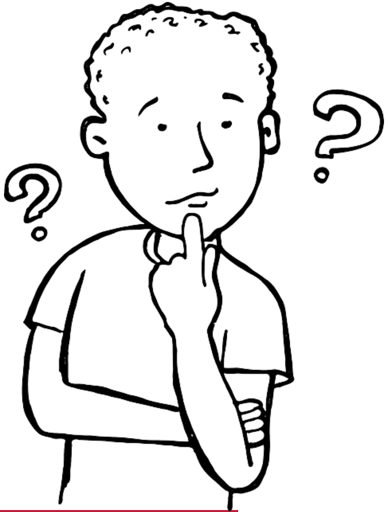
Gesucht:

Ein kostenminimaler st-Weg



Jede Runde im roten Kreis verringert die Gesamtkosten!

Kostenfunktionen



Schwer

Beliebige Gewichte:
Es können negative gerichtete Kreise auftreten.

Konservative Gewichte:
Jeder gerichtete Kreis besitzt nicht-negative Gesamtkosten.

Nicht-Negative Kosten:
Jede Kante besitzt ein Gewicht von mindestens Null.

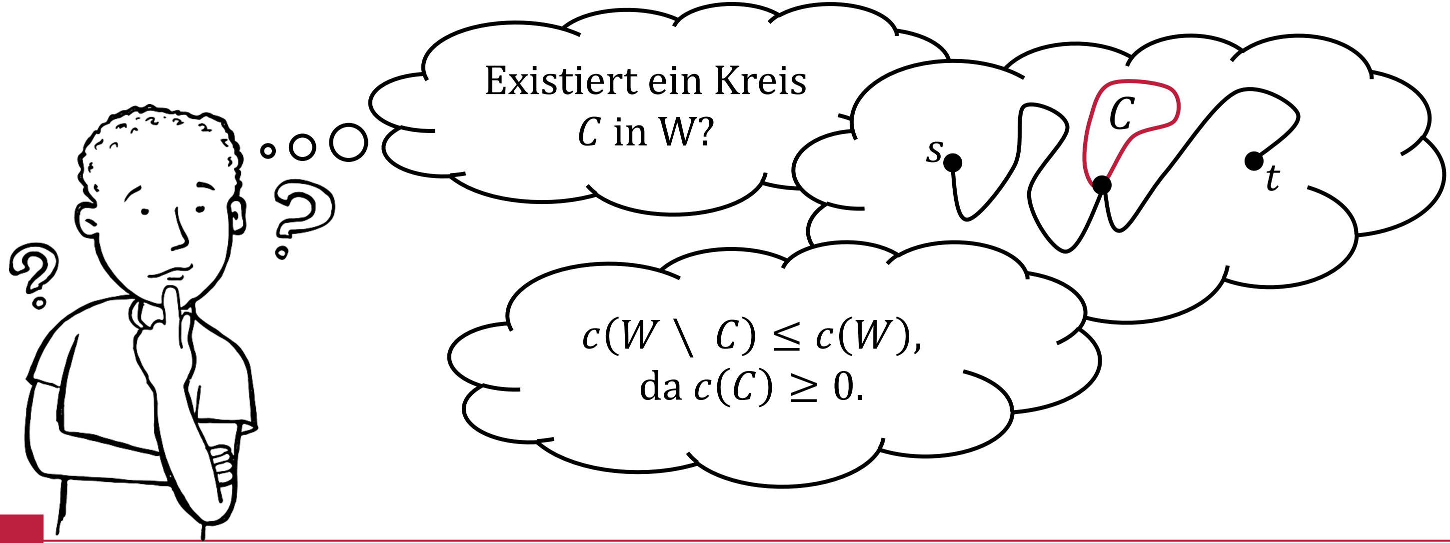
Konstante (positive) Kosten:
Entspricht der ungewichteten Variante. (siehe AuD)

Einfach

3.1 Eigenschaften kürzester Wege

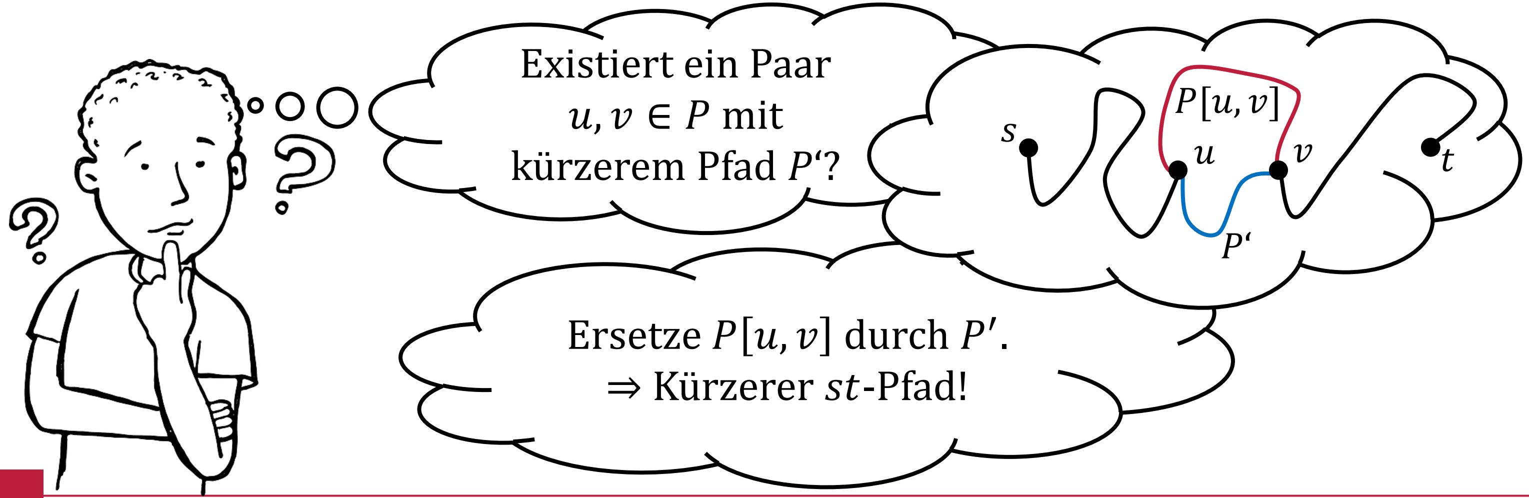
Eigenschaften kürzester Wege

Lemma 3.2
Sei $D = (V, A)$ ein gerichteter Graph und $c\colon\ A \to \mathbb{R}$ eine konservative Kostenfunktion. Existiert für zwei Knoten $s, t \in V$ ein kürzester st-Weg W, dann existiert auch ein kürzester st-Pfad.



Eigenschaften kürzester Wege

Lemma 3.3
Sei $D = (V, A)$ ein gerichteter Graph und $c\colon A \to \mathbb{R}$ eine konservative Kostenfunktion. Sei P ein kürzester st-Pfad mit $s, t \in V$.
Für jedes Paar $u, v \in P$, wobei u vor v liegt, ist $P[u, v]$ (Teilpfad von P zwischen u und v) ein kürzester uv-Pfad



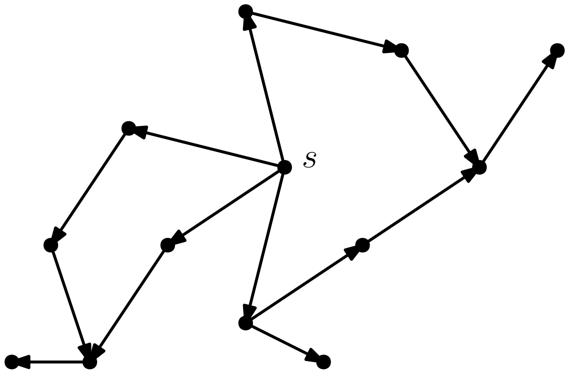
3.2 Single Source Shortest Path

Single Source Shortest Path

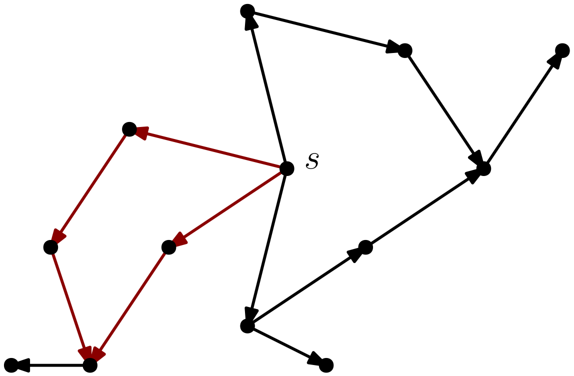
Problem 3.4: Kürzeste Wege Problem (Single Source Shortest Path, SSSP)
Gegeben:
Digraph $D = (V, A)$
Konservative Kostenfunktion $c\colon A \to \mathbb{R}$
Knoten $s \in V$
Gesucht:
Für jeden Knoten $v \in V$ einen kürzesten sv-Weg

Lemma 3.5
Seien $D = (V, A)$ ein gerichteter Graph, $c\colon A \to \mathbb{R}$ eine konservative Kostenfunktion und $s \in V$. Es existiert ein Teilgraph T von D, sodass
1. Der zugrunde liegende ungerichtete Graph von T ein Baum ist.
2. Der (eindeutige) Pfad zwischen s und $v \in V$ ein kürzester Pfad in D ist.

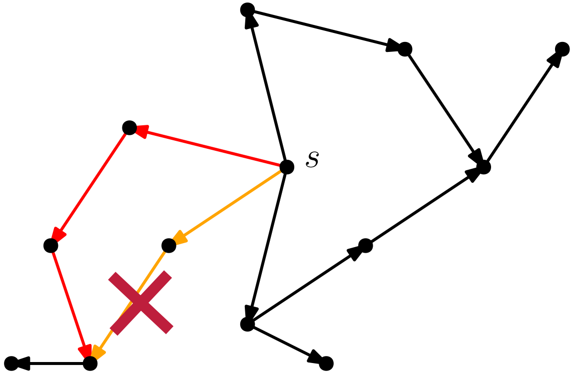
Beweisskizze Lemma 3.5



Überlappe gefundene kürzeste Wege



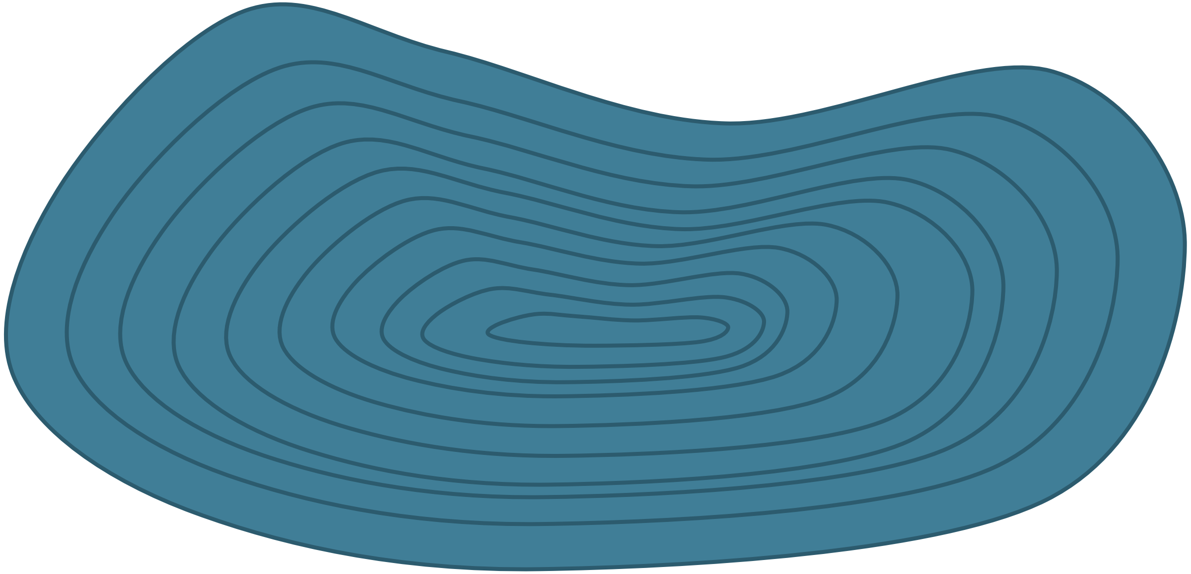
Identifiziere Kreis



Rot und gelb besitzen gleiches Gewicht ⇒ entferne letzte Kante auf einem der Wege

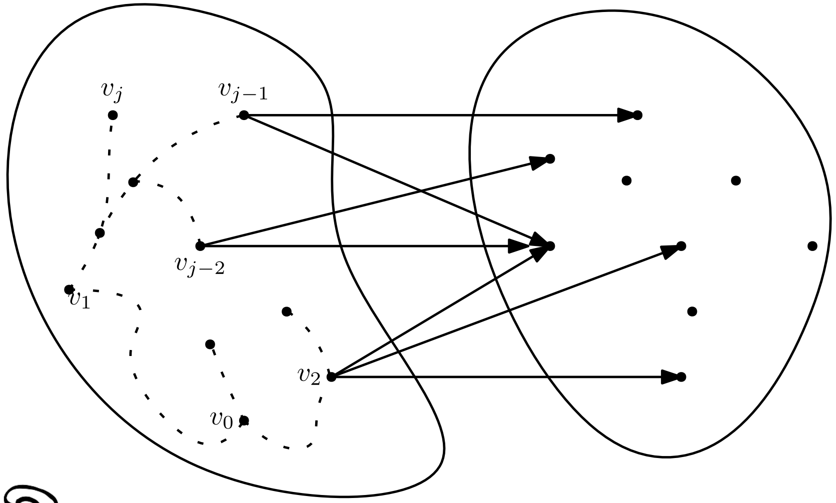
3.2.1 Nicht-negative Kantengewichte

Idee – Pfade wachsen lassen



Algorithmus von Dijkstra – Idee

Idee: Konstruiere kürzeste Pfade aus kürzesten Pfaden.



Angenommen: Für Knoten aus $R := \{s := v_0, v_1, \ldots, v_j\}$ sind kürzeste sv_i-Pfade bekannt und $c(e) \geq 0$.

Definiere:

$$c_{i,k} := \begin{cases} c((v_i, v_k)), & \text{falls } (v_i, v_k) \in A \\ \infty, & \text{sonst} \end{cases}$$

$\ell_k :=$ Länge eines kürzesten $v_0 v_k$-Pfades

Für alle $v_k \in V \setminus R$ ist nun:

$$\ell_k \leq \min_{v_i \in R} \left(\ell_i + c_{i,k} \right)$$

Existiert $v_k \in V \setminus R$, für den Gleichheit gilt?

Algorithmus von Dijkstra

Algorithmus 3.6

Eingabe:

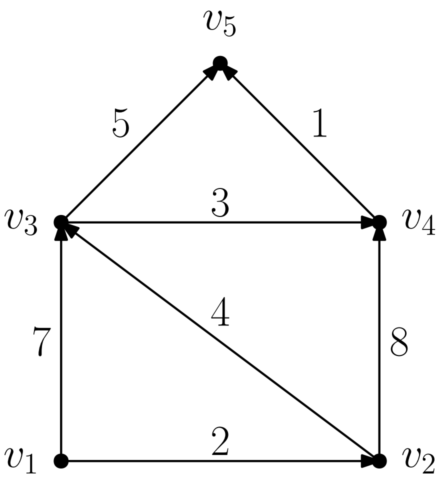
Digraph $D = (V, A)$, Kostenfunktion $c \colon A \to \mathbb{R}_{\geq 0}$, Startknoten $s \in V$

Ausgabe:

Länge $\ell(v)$ eines kürzesten sv-Pfades, Vorgänger $p(v)$ von v auf diesem Pfad

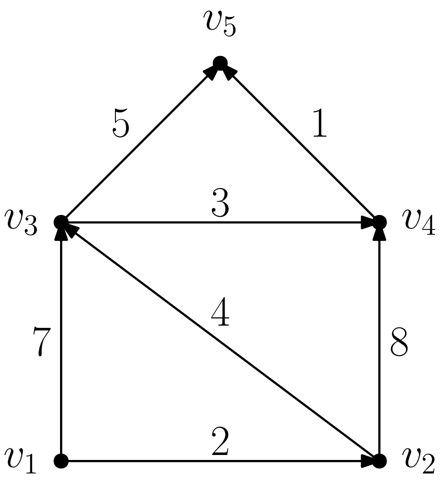
Function $\textsc{Dijkstra}(D, c, s)$
 $\ell(s) := 0$
 $\ell(v) := \infty,\ p(v) := nil \quad \forall v \in V \setminus \{s\}$
 $R := \emptyset$
 while $R \neq V$ do
  Wähle $v \in V \setminus R$ mit $\ell(v) = \min_{w \in V \setminus R} \ell(w)$
  $R := R \cup \{v\}$
  for all $w \in V \setminus R$ mit $(v, w) \in A$ do
   if $\ell(w) > \ell(v) + c\big((v, w)\big)$ then
    $\ell(w) := \ell(v) + c\big((v, w)\big)$
    $p(w) := v$

Beispiel



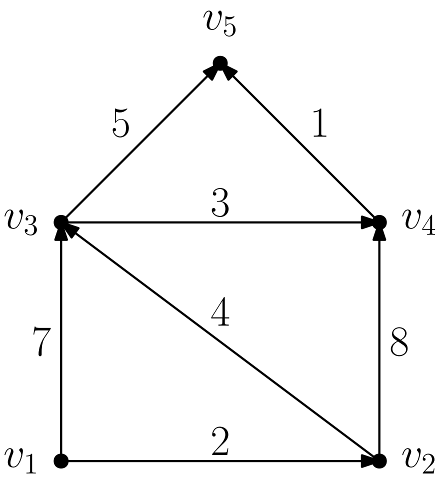
It.	v_1		v_2		v_3		v_4		v_5		R
	$\ell(v_1)$	$p(v_1)$	$\ell(v_2)$	$p(v_2)$	$\ell(v_3)$	$p(v_3)$	$\ell(v_4)$	$p(v_4)$	$\ell(v_5)$	$p(v_5)$	
0											
1											
2											
3											
4											
5											

Beispiel



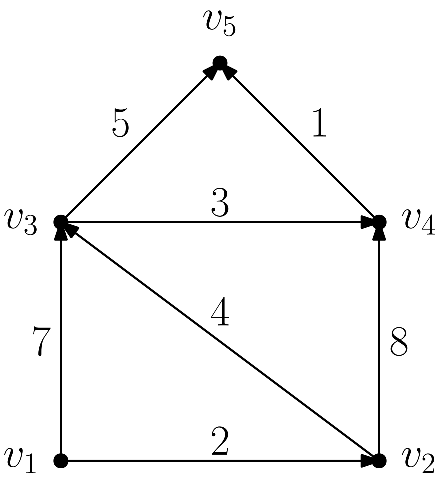
It.	v_1		v_2		v_3		v_4		v_5		R
	$\ell(v_1)$	$p(v_1)$	$\ell(v_2)$	$p(v_2)$	$\ell(v_3)$	$p(v_3)$	$\ell(v_4)$	$p(v_4)$	$\ell(v_5)$	$p(v_5)$	
0	0	–	∞	–	∞	–	∞	–	∞	–	$\emptyset$
1											
2											
3											
4											
5											

Beispiel



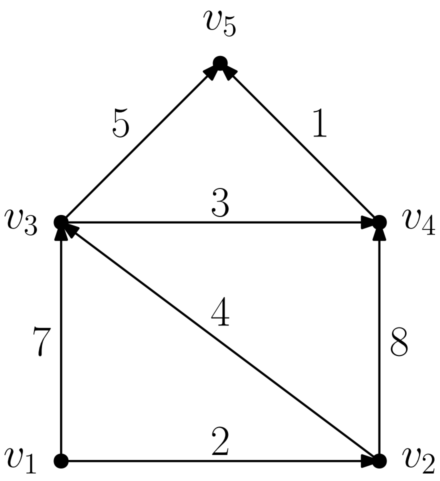
It.	v_1		v_2		v_3		v_4		v_5		R
	$\ell(v_1)$	$p(v_1)$	$\ell(v_2)$	$p(v_2)$	$\ell(v_3)$	$p(v_3)$	$\ell(v_4)$	$p(v_4)$	$\ell(v_5)$	$p(v_5)$	
0	0	–	∞	–	∞	–	∞	–	∞	–	$\emptyset$
1			2	v_1	7	v_1					v_1
2											
3											
4											
5											

Beispiel



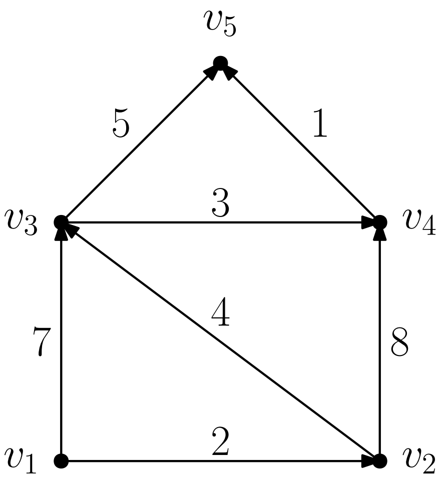
It.	v_1		v_2		v_3		v_4		v_5		R
	$\ell(v_1)$	$p(v_1)$	$\ell(v_2)$	$p(v_2)$	$\ell(v_3)$	$p(v_3)$	$\ell(v_4)$	$p(v_4)$	$\ell(v_5)$	$p(v_5)$	
0	0	–	∞	–	∞	–	∞	–	∞	–	$\emptyset$
1			2	v_1	7	v_1					v_1
2					6	v_2	10	v_2			v_1, v_2
3											
4											
5											

Beispiel



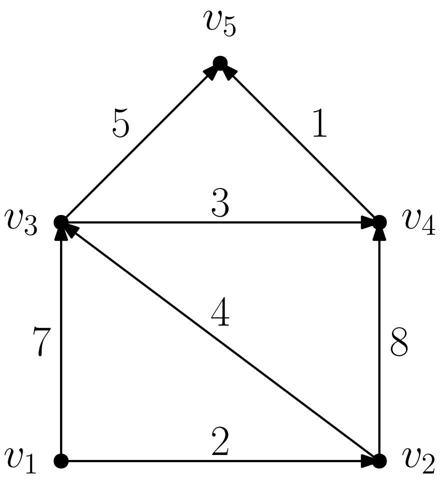
It.	v_1		v_2		v_3		v_4		v_5		R
	$\ell(v_1)$	$p(v_1)$	$\ell(v_2)$	$p(v_2)$	$\ell(v_3)$	$p(v_3)$	$\ell(v_4)$	$p(v_4)$	$\ell(v_5)$	$p(v_5)$	
0	0	–	∞	–	∞	–	∞	–	∞	–	$\emptyset$
1			2	v_1	7	v_1					v_1
2					6	v_2	10	v_2			v_1, v_2
3							9	v_3	11	v_3	v_1, v_2, v_3
4											
5											

Beispiel



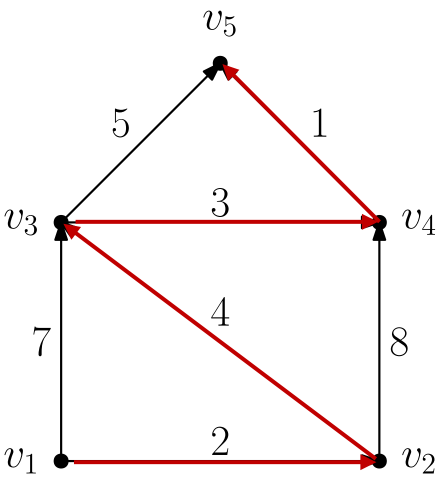
It.	v_1		v_2		v_3		v_4		v_5		R
	$\ell(v_1)$	$p(v_1)$	$\ell(v_2)$	$p(v_2)$	$\ell(v_3)$	$p(v_3)$	$\ell(v_4)$	$p(v_4)$	$\ell(v_5)$	$p(v_5)$	
0	0	–	∞	–	∞	–	∞	–	∞	–	$\emptyset$
1			2	v_1	7	v_1					v_1
2					6	v_2	10	v_2			v_1, v_2
3							9	v_3	11	v_3	v_1, v_2, v_3
4									10	v_4	v_1, v_2, v_3, v_4
5											

Beispiel



It.	v_1		v_2		v_3		v_4		v_5		R
	$\ell(v_1)$	$p(v_1)$	$\ell(v_2)$	$p(v_2)$	$\ell(v_3)$	$p(v_3)$	$\ell(v_4)$	$p(v_4)$	$\ell(v_5)$	$p(v_5)$	
0	0	–	∞	–	∞	–	∞	–	∞	–	$\emptyset$
1			2	v_1	7	v_1					v_1
2					6	v_2	10	v_2			v_1, v_2
3							9	v_3	11	v_3	v_1, v_2, v_3
4									10	v_4	v_1, v_2, v_3, v_4
5											v_1, v_2, v_3, v_4, v_5

Beispiel



It.	v_1		v_2		v_3		v_4		v_5		R
	$\ell(v_1)$	$p(v_1)$	$\ell(v_2)$	$p(v_2)$	$\ell(v_3)$	$p(v_3)$	$\ell(v_4)$	$p(v_4)$	$\ell(v_5)$	$p(v_5)$	
0	0	–	∞	–	∞	–	∞	–	∞	–	$\emptyset$
1			2	v_1	7	v_1					v_1
2					6	v_2	10	v_2			v_1, v_2
3							9	v_3	11	v_3	v_1, v_2, v_3
4									10	v_4	v_1, v_2, v_3, v_4
5											v_1, v_2, v_3, v_4, v_5

Algorithmus von Dijkstra

Idee Korrektheitsbeweis:

Zeige, dass folgende Invarianten (vor und) nach jeder Iteration gelten.

(a) Für alle $x \in R$ und $y \in V \setminus R$ gilt $\ell(x) \leq \ell(y)$

(b) Für alle $x \in R$ gilt

- $\ell(x)$ ist die Länge eines kürzesten sx-Weges in D.
- Ist $\ell(x) < \infty$, dann existiert sx-Pfad P in $D[R]$ mit Kosten $\ell(x)$ und $(p(x), x) \in P$.

(c) Für alle $w \in V \setminus R \setminus \{s\}$ gilt:

- $\ell(w)$ ist die Länge eines kürzesten sw-Pfades in $D[R \cup \{w\}]$
- Ist $\ell(w) < \infty$, dann ist $p(w) \in R$ und $\ell(w) = \ell\big(p(w)\big) + c\big((p(w), w)\big)$

Definition 3.7

Sei $D = (V, A)$ ein Digraph und $R \subseteq V$. $D[R] = (R, A')$ bezeichnet den von R **induzierten Teilgraphen** von D, d.h. $A' \coloneqq \{e = (v, w) \mid v, w \in R \text{ und e} \in A\}$

Algorithmus von Dijkstra

Idee Korrektheitsbeweis:

Zeige, dass folgende Invarianten (vor und) nach jeder Iteration gelten.

(a) Für alle $x \in R$ und $y \in V \setminus R$ gilt $\ell(x) \leq \ell(y)$

(b) Für alle $x \in R$ gilt

- $\ell(x)$ ist die Länge eines kürzesten sx-Weges in D.
- Ist $\ell(x) < \infty$, dann existiert sx-Pfad P in $D[R]$ mit Kosten $\ell(x)$ und $(p(x), x) \in P$.

(c) Für alle $w \in V \setminus R \setminus \{s\}$ gilt:

- $\ell(w)$ ist die Länge eines kürzesten sw-Pfades in $D[R \cup \{w\}]$
- Ist $\ell(w) < \infty$, dann ist $p(w) \in R$ und $\ell(w) = \ell\big(p(w)\big) + c\big((p(w), w)\big)$

Satz 3.8

Dijkstra‘s Algorithmus löst Problem 3.4 (Kürzeste Wege) korrekt für nicht-negative Kantenkosten in $O(n \log n + m)$ Zeit.

Algorithmus von Dijkstra

Satz 3.8
Dijkstra's Algorithmus löst Problem 3.4 (Kürzeste Wege) korrekt für nicht-negative Kantenkosten in $O(n \log n + m)$ Zeit.

Algorithmus 3.6
Eingabe:
 Digraph $D = (V, A)$, Kostenfunktion $c: A \to \mathbb{R}_{\geq 0}$, Startknoten $s \in V$
Ausgabe:
 Länge $\ell(v)$ eines kürzesten sv-Pfades, Vorgänger $p(v)$ von v auf diesem Pfad

```
Function DIJKSTRA(D, c, s)
    ℓ(s) ≔ 0
    ℓ(v) ≔ ∞  ∀v ∈ V \ {s}
    R ≔ ∅
    while R ≠ V do
        Wähle v ∈ V \ R mit l(v) = min_{w∈V\R} ℓ(w)
        R ≔ R ∪ {v}
        for all w ∈ V \ R mit (v, w) ∈ A do
            if ℓ(w) > ℓ(v) + c((v, w)) then
                ℓ(w) ≔ ℓ(v) + c((v, w))
                p(w) ≔ v
```

Zeilen 2–4: $O(n)$

Zeile 6: $O(\log n)$
Extract-Min Fibonacci-Heaps

Zeilen 8–11: $O\big(\delta^+(v)\big)$
Inkl. Decrease-Key im Fibonacci-Heap

Zeilen 5–11: $O(n \log n + m)$

Konservative Kantengewichte