

Algorithmen und Datenstrukturen

Sommersemester 2021

Stefan Näher

Universität Trier

naeher@uni-trier.de

Vorlesung 18

17. Juni 2021

Kürzeste oder billigste Pfade

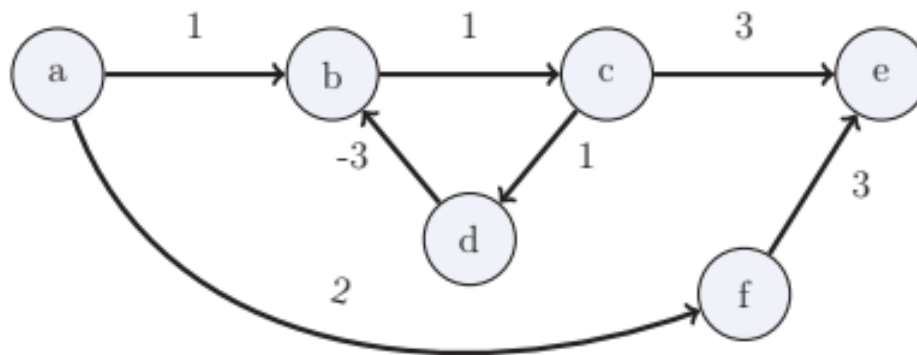
Gerichteter Graph: $G = (V, E)$

Kostenfunktion: $cost : E \rightarrow \mathbb{R}$

$cost(e)$ = Kosten der Kante e

Anstatt $cost((v, w))$ schreiben wir nur $cost(v, w)$.

Beispiel 4.1



Ein Pfad von $s \in V$ nach $w \in V$ in einem gerichteten Graphen $G = (V, E)$ ist eine Folge $P = v_0, \dots, v_k$ mit $v_0 = s$ und $v_k = w$.

Definition über Kanten

Ein Pfad von $s \in V$ nach $w \in V$ in einem gerichteten Graphen $G = (V, E)$ ist eine Folge $P = (e_1, e_2, \dots, e_n)$ von Kanten $e_i \in E$.

Länge eines Pfades

Die Länge eines Pfades wird oft über die Anzahl der enthaltenen Kanten definiert. Hier allerdings **nicht**!

Trotzdem sprechen wir von kürzesten Pfaden, da das Problem im Englischen „shortest path“ genannt wird und es oft mit „kürzestem Weg“ übersetzt wird. Da oft von billigsten Pfaden gesprochen wird, wenn es um die Kosten eines Pfades geht, werden in diesem Skript die Bezeichnungen „kürzeste Wege“ und „billigste Pfade“ (oder Mischformen daraus) synonym verwendet.

Die Kosten eines Pfades ist die Summe der Kosten der Kanten entlang des Pfades.

$$\textit{cost}(P) := \sum_{e \in P} \textit{cost}(e)$$

Achtung: Eine Kante kann mehrfach vorkommen.

Im Beispiel 4.1:

$$\textit{cost}(a \rightarrow f \rightarrow e) = 5$$

$$\textit{cost}(a \rightarrow b \rightarrow c \rightarrow e) = 5$$

$$\textit{cost}(a \rightarrow b \rightarrow c \rightarrow d \rightarrow b \rightarrow c \rightarrow e) = 4$$

Die Distanz zweier Knoten entspricht dem kürzesten Weg zwischen diesen beiden Knoten.

Da es durch negative Zyklen unendlich viele Wege geben kann, wird hier nicht das Minimum, sondern das Infimum (s. 1.2.2) genutzt.

$$\textit{dist}(v, w) := \inf \{ \textit{cost}(P) \mid P \text{ ist Pfad von } v \text{ nach } w \}$$

dh. $\textit{dist}(v, w) = -\infty$, falls negativer Zyklus auf P existiert.

$\textit{dist}(v, w) = +\infty$, falls kein Pfad von v nach w existiert.

Im Beispiel 4.1:

$$\textit{dist}(a, f) = 2$$

$$\textit{dist}(e, a) = +\infty$$

$$\textit{dist}(a, e) = -\infty$$

Pfad: $a \rightarrow b \rightarrow (c \rightarrow d \rightarrow b \rightarrow c)^i \rightarrow e$

Kosten: $4 - i$ dh. beliebig klein.

Eingabe: $G = (V, E)$, $s \in V$, $cost : E \rightarrow \mathbb{R}$

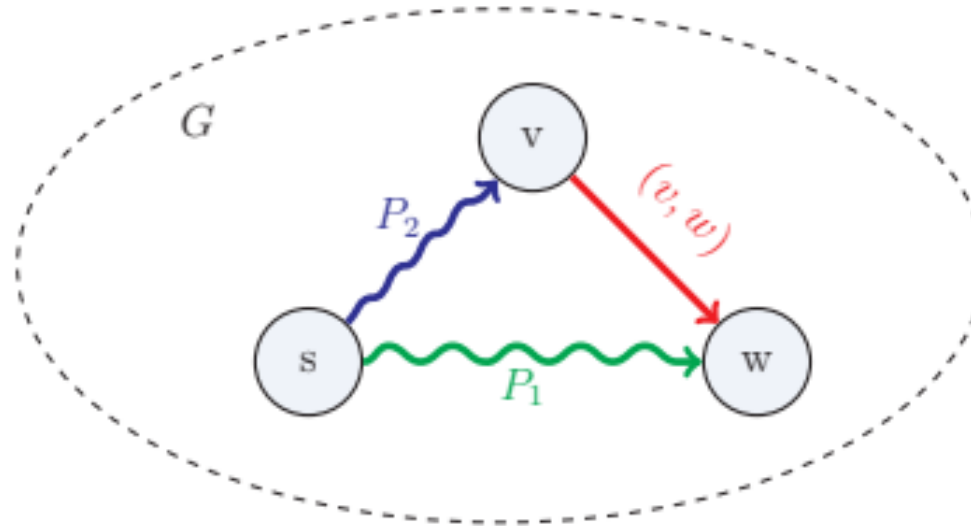
Ausgabe: $\forall v \in V \quad dist(s, v)$ (und entsprechende Pfade)

Beobachtung:

Die $dist$ -Funktion erfüllt die Δ -Ungleichung.

Für jede Kante $(v, w) \in E$ gilt:

$$dist(s, w) \leq dist(s, v) + cost(v, w)$$



P_1 ist ein kürzester Pfad von s nach w .

P_2 ist ein kürzester Pfad von s nach v .

Ein erster Algorithmus

Ideen:

- verwende (temporäre) DIST - Labels
- überschätze die dist-Werte: $\text{DIST}[s] = 0$ und $\text{DIST}[v] = \infty$ für $v \neq s$
- solange eine Kante (u, v) die Δ -Ungleichung verletzt:
korrigiere $\text{DIST}[v]$ auf $\text{DIST}[u] + \text{cost}(u, v)$

1. **forall** $v \in V$ **do** $\text{DIST}[v] \leftarrow \infty$; **od**
2. $\text{DIST}[s] \leftarrow 0$;
3. **while** $\exists (u, v) \in E$ mit $\text{DIST}[v] > \text{DIST}[u] + \text{cost}(u, v)$ **do**
4. $\text{DIST}[v] \leftarrow \text{DIST}[u] + \text{cost}(u, v)$;
5. **od**

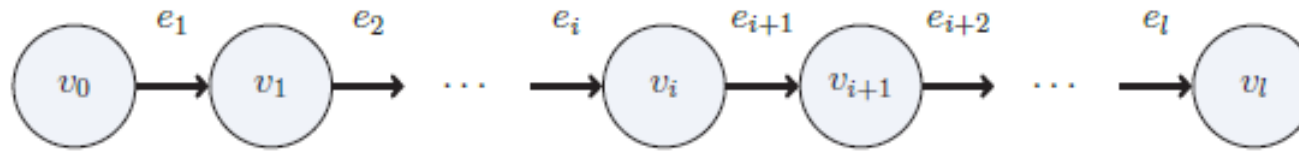
Eigenschaften des Algorithmus

- a) Es gilt stets $\text{DIST}[v] \geq \text{dist}(s, v)$.
- b) Wenn $\text{DIST}[v] < \infty$, dann existiert ein Pfad von s nach v mit Kosten $\text{DIST}[v]$.
- c) Die kürzesten Pfade bilden einen Baum T mit Wurzel s .
- d) Für jede Kante (v, w) auf einem kürzesten Pfad (Kante in T) gilt:
 $\text{dist}(s, w) = \text{dist}(s, v) + \text{cost}(v, w)$, d.h. Δ -Ungleichung mit Gleichheit.

Beweis

- a) $dist(s, v)$ sind die Kosten des kürzesten Pfades von s nach v . $DIST[v]$ ist zu Beginn $+\infty$ und wird im Laufe des Algorithmus, falls mindestens ein Pfad von s nach v existiert, auf die Kosten dieses Pfades verringert. Somit kann der Wert von $DIST[v]$ auch nie kleiner als die Kosten des kürzesten Pfades ($dist(s, v)$) werden. Ab dem Zeitpunkt, an dem der Algorithmus den kürzesten Pfad findet, gilt: $DIST[v] = dist(s, v)$
- b) Sobald der Algorithmus einen Pfad von s nach v findet, berechnet er die Kosten dieses Pfades und setzt dieses Ergebnis als neuen $DIST$ -Wert von v . Da der Pfad und die Kosten der einzelnen Kanten endlich sind, ist das Ergebnis und somit $DIST[v]$ echt kleiner als $+\infty$.
- c) In jedem Knoten mit $DIST[v] < \infty$ mündet genau ein kürzester Pfad. Bei Korrektur des $DIST$ -Wertes wird die eingehende Kante von T definiert.

d) Die Δ -Ungleichung mit Gleichheit ergibt sich aus der Definition von $dist(s, w)$.



$$dist(s, v_i) = \sum_{k=1}^i e_k$$

$$dist(s, v_{i+1}) = \sum_{k=1}^i e_k + cost(v_i, v_{i+1})$$

$$dist(s, v_{i+1}) = \sum_{k=1}^i e_k + e_{i+1}$$

$$dist(s, v_{i+1}) = \sum_{k=1}^{i+1} e_k$$

(v_i, v_{i+1}) verbindet den Knoten v_{i+1} mit dem auf dem kürzesten Weg davor liegenden Knoten.