ALGORITHMEN UND DATENSTRUKTUREN

DER DIJKSTRA-ALGORITHMUS

Eric Kunze

eric.kunze@mailbox.tu-dresden.de

TU Dresden, 03.03.2020

Algorithmus von Dijkstra

SETTING [3]

gegeben:

- gerichteter, gewichteter Graph $G = (V, E, c)$ mit
  - $V = \{1, ..., n\}$
  - $c(v) \geq 0$ für alle $v \in V$ (nichtnegative Kantengewichte)
- Startknoten s (*Quelle*)

Ziel:

- kürzeste Entfernung von s nach v für alle $v \in V$

Idee:

- Wir kennen einen kürzesten Weg p von s nach v.
- Verlängerung von p um eine Kante (v, v')
- Wir erhalten einen kürzesten Weg von s nach v'.

OPTIMALITÄTSEIGENSCHAFT

Theorem

Für jeden kürzesten Weg $p = (v_0, v_1, \ldots, v_k)$ *von* v_0 *nach* v_k *ist jeder Teilweg* $(v_i, \ldots, v_j)$ *mit* $1 \leq i < j \leq k$ *auch ein kürzester Weg von* v_i *nach* v_j*.*

Beweis (nach [1])**.** Sei p wie oben ein kürzester Weg. Angenommen es gäbe einen Teilweg $(u, \ldots, v)$, der kein kürzester Weg ist. Dann gibt es also einen kürzeren Weg $(u, w_1, \ldots, w_\ell, v)$ von u nach v. Dann wäre aber auch der Weg $p' = (v_1, \ldots, u, w_1, \ldots, w_\ell, v, \ldots, v_k)$ von v_1 nach v_k kürzer als p im Widerspruch zur Optimalität von p. Also muss auch der Weg $(u, \ldots, v)$ optimal sein. $\square$

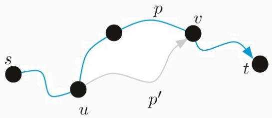

Abbildung 1: Teilwege von optimalen Wegen sind wieder optimal. [1]

DER ALGORITHMUS [2]

Notation.

- M ... Menge der Knoten, zu der ein kürzester Weg bekannt ist
- $p(v_k)$ ... Vorgänger von v_k auf dem kürzesten Weg nach v_k
- $d(v_k)$ ... Länge des (bisher) kürzesten Weges zu v_k

Initialisierung.

- $M = \{s\}$
- $d(s) = 0$
- für $v \neq s$ setze

$$p(v) := \begin{cases} s & \text{für } (s,v) \in E \\ 0 & \text{für } (s,v) \notin E \end{cases} \qquad d(v) := \begin{cases} c(s,v) & \text{für } (s,v) \in E \\ +\infty & \text{für } (s,v) \notin E \end{cases}$$

DER ALGORITHMUS [2]

1. Bestimme $u \notin M$ mit $d(u) = \min\{d(v) : v \notin M\}$.
   - ▷ Falls $d(u) = +\infty$, dann STOP (kein neuer Weg möglich)
   - ▷ Andernfalls setze $M := M \cup \{u\}$
2. Für alle $v \notin M$ mit $(u, v) \in E$:
   falls $d(v) > d(u) + c(u, v)$ (also ein kürzerer Weg ist gefunden),
   dann setze $d(v) = d(u) + c(u, v)$ und $p(v) = u$
3. Falls $M \neq V$, gehe zu Schritt 1. Sonst STOP.

BEISPIEL

Wir notieren die notwendigen Informationen als Tripel

$$\big(\text{Knotennummer}, \text{Entfernung von der Quelle}, \text{Vorgängerknoten}\big)$$
$$= \big(\quad v_k \quad , \quad d(v_k) \quad , \quad p(v_k) \quad\big)$$

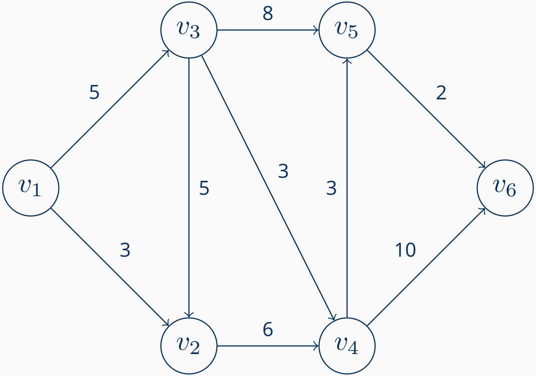

Abbildung 2: Graph $G = (V, E, c)$

BEISPIEL

gewählt	Menge der Randknoten
$(v_1, 0, -)$	$\{\underline{(v_2, \mathbf{3}, v_1)}, (v_3, 5, v_1)\}$
$(v_2, 3, v_1)$	$\{\underline{(v_3, \mathbf{5}, v_1)}, (v_4, 9, v_2)\}$
$(v_3, 5, v_1)$	$\{\underline{(v_4, \mathbf{8}, v_3)}, (v_5, 13, v_3)\}$
$(v_4, 8, v_3)$	$\{\underline{(v_5, \mathbf{11}, v_4)}, (v_6, 18, v_4)\}$
$(v_5, 11, v_4)$	$\{\underline{(v_6, \mathbf{13}, v_5)}\}$
$(v_6, 13, v_5)$	$\emptyset$

REKONSTRUKTION DES KÜRZESTEN WEGES

Wir betrachten beispielhaft den Weg von v_1 nach v_5.

- Es ist $d(v_5) = 11$, d.h. der kürzeste Weg von v_1 zu v_5 ist 11 Einheiten lang.
- Es gilt $p(v_5) = v_4$. Wir betrachten weiter die Vorgänger-Funktion:

$$p(v_5) = v_4 \quad \leftarrow \quad p(v_4) = v_3 \quad \leftarrow \quad p(v_3) = v_1$$

Somit ist der kürzeste Weg von v_1 zu v_5 also gegeben durch

$$v_1 \to v_3 \to v_4 \to v_5$$

EINE ANDERE ART DER DOKUMENTATION

$p =$	0	v_1	v_1	$\cancel{v_2}$ v_3	$\cancel{v_3}$ v_4	$\cancel{v_4}$ v_5
	v_1	v_2	v_3	v_4	v_5	v_6
$d =$	$\underline{0}$	∞	∞	∞	∞	∞
		$\underline{3}$	5	∞	∞	∞
			$\underline{5}$	9	∞	∞
				$\underline{8}$	13	∞
					$\underline{11}$	18
						$\underline{13}$

Algorithmus von Ford-Moore

SETTING [2]

gegeben:

- gerichteter, gewichteter Graph $G = (V, E, c)$ mit
  - $V = \{1, \ldots, n\}$
  - *beliebiger* Funktion $c\colon E \to \mathbb{R}$ (auch negative Gewichte)
- Startknoten $s \in V$

gesucht:

- *längste* Wege von $s \in V$ zu allen anderen Knoten

naive Idee:

- Dijkstra-Algorithmus für $G' = (V, E, -c)$
- Problem: negative Kantengewichte nicht zulässig
- Ausweg: Algorithmus von Ford & Moore

DER ALGORITHMUS

Algorithmus von Ford/Moore:

1. Wähle $s \in V$ als Startknoten und setze $d(s) = 0$, sowie
$$p(v) := \begin{cases} s & (s,v) \in E \\ 0 & \text{sonst} \end{cases} \qquad d(v) := \begin{cases} c(s,v) & (s,v) \in E \\ -\infty & \text{sonst} \end{cases}$$
für alle $s \neq v$. Definiere außerdem
$A := \{s\} \cup \{v \in V : (s,v) \in E\}$, $B := \emptyset$ und $k := 1$.
2. Falls $A = \emptyset$ oder $k = |V|$, dann STOP.
3. Für alle $u \in A$ und alle $v \in V$ mit $(u,v) \in E$: falls $d(v) < d(u) + c(u,v)$, dann setze $d(v) = d(u) + c(u,v)$ und $p(v) = u$ sowie $B := B \cup \{v\}$.
4. Setze $A = B$, $B = \emptyset$, $k := k + 1$ und gehe zu Schritt 1.

LITERATUR

BÜSING, C. :
***Graphen- und Netzwerkoptimierung*.**
Heidelberg : Spektrum Akademischer Verlag, 2010. –
ISBN 9783827424228

MARTINOVIC, J. :
***Optimierung*.**
Vorlesungsmitschrift, Januar 2020

VOGLER, H. :
***Algorithmen, Datenstrukturen und Programmierung*.**
Vorlesungsskript, September 2018