

Kürzeste-Wege-Algorithmen und Datenstrukturen

Institut für Informatik
Universität zu Köln
SS 2009

Teil 1

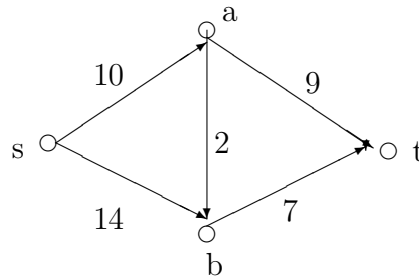
Inhaltsverzeichnis

1	Kürzeste Wege	2
1.1	Voraussetzungen	2
1.2	Dijkstra-Algorithmus	2
1.3	Korrektheit des Algorithmus	3
1.4	Abschätzung der Laufzeit	4

1 Kürzeste Wege

1.1 Voraussetzungen

Gegeben sei ein gerichteter Graph $G = (V, E)$ mit Knotenmenge V und Kantenmenge E und Entfernungen $c(v, w) \geq 0$ für $(v, w) \in E$. Zusätzlich sei ein Knoten $s \in V$ ausgezeichnet.



Definition 1.1. Ein **Weg** P von $v \in V$ nach $w \in V$ ist eine Folge v_0, v_1, \dots, v_k von Knoten, so dass $v_0 = v, v_k = w$ und $(v_i, v_{i+1}) \in E$ für $0 \leq i \leq k - 1$. Der Weg hat die **Länge** $c(P) := \sum_{i=0}^{k-1} c(v_i, v_{i+1})$.

Gesucht: Kürzeste Wege von s zu allen anderen Knoten.

Idee des Verfahrens: Wir vergrößern schrittweise eine Menge M von Knoten, für deren Elemente $v \in M$ wir bereits einen kürzesten Weg von s nach v gefunden haben. Allen anderen Knoten $v \notin M$ ordnen wir die Länge des bisher gefundenen kürzesten Weges zu.

1.2 Dijkstra-Algorithmus

Wir bezeichnen mit $Dist(v)$ die Länge des (bisher gefundenen) kürzesten Weges von s nach v und mit $Vor(v)$ den Vorgänger von v auf einem solchen Weg.

- (1) Setze $Dist(s) := 0, M := \{s\}$
- (2) Für $v \in V$ mit $(s, v) \in E$ setze $Dist(v) := c(s, v)$ und $Vor(v) := s$
- (3) Für $v \in V$ mit $(s, v) \notin E$ setze $Dist(v) := +\infty$ und $Vor(v) := \emptyset$
- (4) Bestimme $u \notin M$ mit $Dist(u) = \min\{Dist(v) : v \notin M\}$.
- (5) Falls $Dist(u) = \infty$, dann **STOP**. Andernfalls setze $M = M \cup \{u\}$.
- (6) Für alle $v \notin M$ mit $(u, v) \in E$
Falls $Dist(v) > Dist(u) + c(u, v)$ dann setze:
 $Dist(v) = Dist(u) + c(u, v)$
 $Vor(v) = u$
- (7) Falls $M \neq V$, dann gehe zu 4, sonst **STOP**.

Die Menge M scheint sich „kreisförmig“ um den Startknoten s auszubreiten. Sei $d(u)$ der Wert von $Dist(u)$ zu dem Zeitpunkt, zu dem u in die Menge M aufgenommen wird.

1.3 Korrektheit des Algorithmus

Lemma 1.2. *Wird u vor v in M aufgenommen, so gilt $d(u) \leq d(v)$.*

Beweis. Angenommen, es existieren Knoten u und v , so dass u vor v aufgenommen wird, aber $d(u) > d(v)$ gilt. Unter allen diesen Knoten v wähle den als ersten in M aufgenommenen. In dem Moment, in dem u aufgenommen wird, gilt $d(u) = \text{Dist}(u) \leq \text{Dist}(v)$. Da per Definition $d(u)$ nicht mehr verändert wird, muss später $\text{Dist}(v)$ verringert worden sein. Dies kann nur passieren, wenn ein Knoten w in M aufgenommen und $\text{Dist}(v) = \text{Dist}(w) + c(w, v)$ gesetzt wird.

Sei w der letzte solche Knoten, d.h. es gilt $d(v) = d(w) + c(w, v)$. Nach Wahl von v gilt aber $d(u) \leq d(w)$. Da $c(w, v) \geq 0$, folgt $d(v) \geq d(u)$, im Widerspruch zur Voraussetzung. \square

Wenn das Verfahren korrekt ist, muss am Schluss gelten, dass $\text{Dist}(v) \leq \text{Dist}(u) + c(u, v)$, denn ansonsten wäre es kürzer, über u und die Kante (u, v) nach v zu gehen. Das folgende Lemma zeigt, dass unser Verfahren zumindest diese notwendige Bedingung erfüllt.

Lemma 1.3. *Nach Beendigung des Verfahrens gilt für alle Kanten $(u, v) \in E$:*

$$\text{Dist}(v) \leq \text{Dist}(u) + c(u, v).$$

Beweis. Die Aussage gilt sicherlich direkt nach der Iteration, in der u aufgenommen wird. Wenn sie zu einem späteren Zeitpunkt nicht mehr gilt, muss $\text{Dist}(u)$ verringert worden sein, da $\text{Dist}(v)$ nicht wächst. D.h. es gilt $\text{Dist}(u) < d(u)$. Dies kann wiederum nur passiert sein, als ein Knoten w in M aufgenommen und $\text{Dist}(u) = d(w) + c(w, u)$ gesetzt wurde. Nach dem ersten Lemma: $d(w) \geq d(u)$ und $c(w, u) \geq 0 \leadsto \text{Dist}(u) \geq d(u)$, im Widerspruch zur Voraussetzung. \square

Daraus folgt unmittelbar die Korrektheit des Verfahrens:

Satz 1.4. *Der Dijkstra-Algorithmus berechnet kürzeste Wege von s zu allen anderen erreichbaren Knoten.*

Beweis. Wir zeigen per Induktion über die Anzahl der Knoten auf einem kürzesten Weg, dass die notwendige Bedingung auch hinreichend ist. Die Aussage ist sicherlich richtig für s . Sei jetzt v ein Knoten und u ein Vorgänger auf einem kürzesten Weg. Per Induktion können wir annehmen, dass $\text{Dist}(u)$ die Länge eines kürzesten Weges von s nach u ist. Da $\text{Dist}(v) \leq \text{Dist}(u) + c(u, v)$, ist dann $\text{Dist}(v)$ die Länge eines kürzesten Weges nach v . \square

Aus dem zweiten Lemma folgt:

- Sobald u in M aufgenommen wird, ist ein kürzester Weg gefunden worden.
- Der Wert $\text{Dist}(u)$ ändert sich nicht mehr.
- Es reicht daher, eine im allgemeinen kleinere Menge U von Knoten dynamisch zu verwalten.
- U besteht aus allen Knoten, für die bereits ein Weg gefunden wurde, der jedoch noch nicht der kürzeste sein muss.

- U wird auch als **Front** bezeichnet.

Satz 1.5. *Der Dijkstra-Algorithmus endet, nachdem für alle von s aus erreichbaren Knoten die kürzesten Wege gefunden worden sind.*

Beweis. Nach Satz 1.4 berechnet der Dijkstra-Algorithmus einen kürzesten Weg für alle von s erreichbaren Knoten. Nachdem auch für den letzten dieser Knoten ein solcher Weg berechnet wurde, befindet sich der Algorithmus in Schritt (7).

1.Fall: In Schritt (7) gilt $M = V$. Dann stoppt der Algorithmus nach Schritt (7).

2.Fall: In Schritt (7) gilt $M \neq V$. Da bereits für alle Knoten ein kürzester Weg berechnet worden ist, befinden sich in $Q := V \setminus M$ nur noch Knoten, die von s und damit insbesondere von M aus nicht erreichbar sind. Für alle $v \in Q$ gilt also: $Dist(v) = \infty$. Bei der Bestimmung des Minimums in Schritt (4) erhält man folglich einen Knoten u mit $Dist(u) = \infty$. Nach der Vorschrift aus (5) hält demnach der Algorithmus. \square

1.4 Abschätzung der Laufzeit

Dijkstra-Algorithmus

- (1) $Dist(s) := 0, U = \{s\}$
- (2) for $v \in V \setminus \{s\}$ do
- (3) $Dist(v) = +\infty, Vor(v) = Null$
- (4) endfor
- (5) while $U \neq \emptyset$
- (6) choose $u \in U$ with $Dist(u)$ minimal (**finde Min**)
- (7) $U := U \setminus \{u\}$ (**lösche Min**)
- (8) for all $(u, v) \in E$
- (9) if $Dist(u) + c(u, v) < Dist(v)$ then
- (10) $Dist(v) = Dist(u) + c(u, v), Vor(v) = u$, (**verringere**)
- (11) $U := U \cup \{v\}$ (**füge ein**)
- (12) endif
- (13) endfor
- (14) endwhile

Lemma 1.6. *Die Laufzeit des Dijkstra-Algorithmus ist*

$$\mathcal{O}(n \cdot \max\{ \mathcal{O}(\text{finde Min}), \mathcal{O}(\text{lösche Min}), \mathcal{O}(\text{füge ein}) \}) + m \cdot \mathcal{O}(\text{verringere}).$$

Beweis: Die while-Schleife kann höchstens $n = |V|$ mal ausgeführt werden, da jeder Knoten höchstens einmal als Minimum auftreten kann. Die darin enthaltene for-Schleife kann aber auch nur einmal für jede Kante durchlaufen werden. Somit:

n Ausführungen von *finde Min*, *lösche Min*, *füge ein*,

m Ausführungen von *verringere*.

\square