

Lemma 3.2

Sei G ein Digraph mit konservativen Kantengewichten $c: E(G) \rightarrow \mathbb{R}$.
Seien s und w zwei Knoten.

Ist $e = (v, w)$ die letzte Kante eines kürzesten ~~Weges~~ Pfades P von s nach w , dann ist

- $P_{[s,v]}$: P ohne die Kante e , d.h. der Pfad bis v .
ein kürzester Pfad von s nach v .

Beweis:

Angenommen Q ist ein kürzerer s - v -Pfad als $P_{[s,v]}$.

Dann ist $c(Q) + c(e) < c(P)$.

Falls Q nicht w enthält, dann ist $Q+e$ kürzer als P .

Falls Q w enthält, dann ist $Q_{[w,v]} + e$ ein Kreis und $Q_{[s,w]}$ hat Länge

$$\begin{aligned}
 c(Q_{[s,w]}) &= c(Q) + c(e) - c(Q_{[w,v]} + e) \\
 &< c(P) - c(Q_{[w,v]} + e) \\
 &\leq c(P)
 \end{aligned}$$

da c konservativ.



Lemma 3.2 ist die Basis für die meisten Algorithmen zur Berechnung kürzester Wege:

Wege werden schrittweise aus kürzeren Wegen aufgebaut.

(Was heißt "kürzer" ?

- geringeres Kantengewicht \rightarrow Dijkstra
- geringere Kantenzahl \rightarrow Moore-Bellman-Ford)

Gleichzeitig liefert Lemma 3.2 eine Erklärung, warum gleich die kürzesten Wege von einer Quelle zu allen $v \in V(G)$ berechnet werden: Da man nicht weiß, welches der „vorletzte“ Knoten, also der Vorgänger von t sein wird, überprüft man i.d.R. alle Abstände.

3.2. Dijkstras Algorithmus

Für nichtnegative ^{ganzzahlige} Kantengewichte könnte man

das Problem durch Breitensuche lösen, indem man

Kanten durch Einheitskanten ersetzt:



- und dann Breitensuche anwendet.

Nachteil: Bläst Input um einen exponentiellen Faktor auf!

(Statt $\Theta(n \log m + m \log n + \sum_{e \in E(G)} \log c(e))$)

bekommen wir $\Theta(n \log m + m \log n + \sum_{e \in E(G)} c(e))$

$\Theta(n' \log m' + m' \log n')$

mit $m' = \sum_{e \in E(G)} c(e)$

$n' = n + \sum_{e \in E(G)} (c(e) - 1)$

Besser ist

Algorithmus 3.3 (Dijkstra 1959)

Eingabe: Digraph G , Gewichte $c: E(G) \rightarrow \mathbb{R}_+$,
Knoten $s \in V(G)$.

Ausgabe: Für jeden Knoten $v \in V(G)$ die Angaben
 $l(v)$: Länge eines kürzesten $s-v$ -Pfad
 $p(v)$: Vorgänger von v in einem kürzesten $s-v$ -Pfad.

(Falls v nicht erreichbar ist, dann gilt $l(v) = \infty, p(v) = NIL$.)

1 Setze: $l(s) := 0$
 $l(v) := \infty$ für alle $v \in V(G) \setminus \{s\}$
 $R := \emptyset$

2 Finde einen Knoten $v \in V(G) \setminus R$ mit
 $l(v) = \min_{w \in V(G) \setminus R} l(w)$

3 Setze $R := R \cup \{v\}$

4 FOR ~~ALL~~ $(w \in V(G) \setminus R \text{ mit } (v,w) \in E(G))$ DO
IF $l(w) > l(v) + c((v,w))$
THEN $l(w) := l(v) + c((v,w))$,
 $p(w) := v$.

5 IF $R \neq V(G)$ THEN GOTO (2)

Satz 3.4

Dijkstra's Algorithmus ist korrekt. Die Laufzeit ist $O(n^2)$.

Beweis:

wir zeigen, dass bei jeder Ausführung von (2) folgendes gilt:

- (a) Für alle $v \in R$ und alle $w \in V(G) \setminus R$: $l(v) \leq l(w)$
- (b) Für alle $v \in R$ gilt:
 - (i) $l(v)$ ist die Länge eines kürzesten $s-v$ -Pfad's in G .
 - (ii) Falls $l(v) < \infty$, dann gibt es einen $s-v$ -Pfad der Länge $l(v)$, für den alle Knoten in R liegen und (für $v \neq s$) dessen letzte Kante $(p(v), v)$ ist.
- (c) Für alle $w \in V(G) \setminus R$ gilt:
 - (i) $l(w)$ ist die Länge eines kürzesten $s-w$ -Pfad's in $G[R \cup \{w\}]$.
 - (ii) Für $l(w) < \infty$ ist $p(w) \in R$ und

$$l(w) = l(p(w)) + c(p(w), w).$$