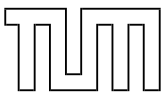
Fortgeschrittene Netzwerk- und Graph-Algorithmen

Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Wintersemester 2009/10



Shortcut-Werte

- geg.: gerichteter Graph $G = (V, E)$
- Ziel: Berechnung der shortcut-Werte aller Kanten in einem gerichteten Graph
- Maximale Erhöhung der Länge eines kürzesten Pfades durch Entfernen einer Kante $e = (u, v) \in E$
- Berechne für alle $e = (u, v) \in E$
  die Distanz von u zu v in $G_e = (V, E \setminus \{e\})$,
  denn die maximale Erhöhung betrifft immer (auch) die Endknoten der Kante

- einfache Lösung: $m = |E|$ SSSP Aufrufe
- besser: nur n äquivalente Aufrufe

Shortcut-Werte

- Annahme: keine negativen Kreise
  ($\Rightarrow$ $d(i,j)$ ist definiert für alle Knotenpaare (i,j)),
  keine parallelen Kanten
- Idee:

  Ein Aufruf für Knoten u berechnet shortcut-Werte für alle ausgehenden Kanten

Shortcut-Werte

Vorgehen:

- Fixiere einen Knoten u
- $\alpha_i = d(u, i)$: Distanz von u nach i.
- τ_i: zweiter Knoten der kürzesten Pfade von u zu i, falls dieser Knoten eindeutig ist.
  Ansonsten $\tau_i = \perp$ (impliziert zwei kürzeste Pfade der Länge α_i mit unterschiedlichen Anfangskanten).
- β_i: Länge des kürzesten Pfades von u nach i, so dass τ_i nicht zweiter Knoten
  (∞ falls es keinen Pfad mehr gibt, $\beta_i = \alpha_i$ falls $\tau_i = \perp$)

Shortcut-Werte

- Betrachte α_v, τ_v und β_v für einen Nachbarn v von u, also $(u, v) \in E$.
- Dann ist die shortcut-Distanz für (u, v) gleich α_v falls $\tau_v \neq v$ (also Kante (u, v) ist nicht einziger kürzester Pfad)
- Ansonsten, falls $\tau_v = v$, ist die neue Distanz β_v
- $\alpha_u = 0, \quad \tau_u = \emptyset, \quad \beta_u = \infty$

$$\alpha_j = \min_{i:(i,j)\in E} (\alpha_i + \omega(i,j))$$

- Nachbarn bezüglich eingehender Kanten, die zu einem kürzesten Pfad gehören:

$$I_j = \{i : (i,j) \in E \text{ und } a_j = a_i + \omega(i,j)\}$$

Shortcut-Werte

$$\tau_j = \begin{cases} j & \text{if } I_j = \{u\}, \quad u \text{ ist Anfang, Kante } (u,j) \\ a & \text{if } \forall i \in I_j : \ a = \tau_i \\ & \text{(Alle Vorgänger haben erste Kante } (u,a)), \\ \perp & \text{sonst} \end{cases}$$

Im Fall $\tau_j = \perp$ gilt $\beta_j = \alpha_j$, sonst

$$\beta_j = \min \left\{ \min_{i:\ (i,j)\in E, \tau_i = \tau_j} \beta_i + \omega(i,j), \min_{i:\ (i,j)\in E, \tau_i \neq \tau_j} \alpha_i + \omega(i,j) \right\}$$

Shortcut-Werte

Betrachte den Pfad p der zu β_j führt, also ein kürzester Pfad p von u nach j, der nicht mit τ_j beginnt. Wenn für den letzten Knoten i vor j in p gilt $\tau_i = \tau_j$, dann startet der Pfad p bis zu i nicht mit τ_j, und dieser Pfad wird in β_i und damit in β_j berücksichtigt.

Wenn anderenfalls $\tau_i \neq \tau_j$ für den vorletzten Knoten i von Pfad p gilt, dann beginnt einer der kürzesten Pfade von u nach i nicht mit τ_j und die Länge von p ist $\alpha_i + \omega(i, j)$.

Shortcut-Werte

Mit den Rekursionen können die Werte α_i,τ_i und β_i effizient berechnet werden.
Im Fall positiver Gewichte hängt jeder Wert α_i nur von Werten α_j ab, die kleiner als α_i sind
$\Rightarrow$ Berechnung in monotoner Weise nach Dijkstra
Bei positiven Gewichten ist der kürzeste-Wege-DAG kreisfrei und die Werte τ_i können in der Reihenfolge einer topologischen Sortierung berechnet werden. (sonst stark zusammenhängende Komponenten kontrahieren)
Werte β_i hängen nur von Werten $\beta_j \leq \beta_i$ ab
$\Rightarrow$ Berechnung in monotoner Weise nach Dijkstra
Bei negativen Kantengewichten (aber keine negativen Kreise) Dijkstra durch Bellman-Ford Algorithmus und β_i durch Berechnung von $\beta'_i = \beta_i - \alpha_i$ ersetzen