Graphenalgorithmen: Blatt 7

Lukas Kalbertodt, Elena Resch, Mirko Wagner

5. Juni 2015

Aufgabe 14:

Im Algorithmus von Dijkstra wird für jede ausgehende Kante eines Knoten folgendes ausgeführt (vgl. Script Abbildung 4.2, Zeile 7):

$$d[j] = \min\{d[j], d[i] + c_{ij}\}$$

Um die Wege maximaler Kapazitäten herauszufinden, wird dieses Update geändert in:

$$d[j] = \max\{d[j], \min\{d[i], c_{ij}\}\}$$

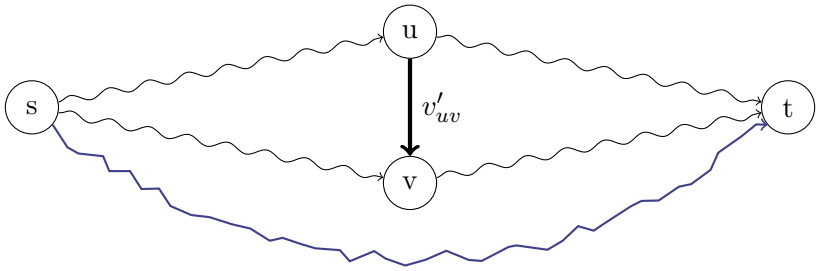
Außerdem müssen wir bei Bestimmung des nächsten Knotens (vgl. Script Abbildung 4.2, Zeile 4) nicht den Knoten mit minimalem $d[i]$ wählen, sondern mit dem maximalen $d[i]$. So garantieren wir, dass wenn ein Knoten gewählt wird, sein Weg-Wert schon maximal ist und sich nicht mehr verändern wird.

Aufgabe 15:

Annahme: Der Graph enthält keine negativen Kreise.

Die Laufzeit beider Update-Algorithmen ist mindestens $\Omega(n^2)$, weil im Worst-Case $\Theta(n^2)$ Werte in d geändert werden. Ein Worst-Case Beispiel: Sei G' ein Graph mit Kantenzusammenhangszahl $\lambda(G') = 1$. Die beiden Teilkomponenten S und $\overline{S}$ von G' seien – für sich genommen – stark zusammenhängend und enthalten beide $\Theta(n)$ Knoten. Sei e die kritische Kante, die von S nach $\overline{S}$ führt. Es gibt nun Wege von allen Knoten in S zu allen Knoten in $\overline{S}$, also $(n^2/4) \in \Theta(n^2)$ Einträge in d. Wenn sich die Kosten von e ändern, ändern sich auch die Kosten von allen $\Theta(n^2)$ Wegen.

Folgende Abbildung dient lediglich zur Veranschaulichung:



(a) Für jedes Knotenpaar (s, t) sind die neuen minimalen Kosten (d'):

$$d'[s,t] = \min\{d[s,t],\ d[s,u] + c'_{uv} + d[v,t]\} \tag{1}$$

Wir führen dieses Update für jedes Knotenpaar (n^2) in beliebiger Reihenfolge aus. Da ein Update konstante Laufzeit hat, liegt die Gesamtlaufzeit in $\mathcal{O}(n^2)$. Im Weiteren wird bewiesen, dass d' korrekt ist:

Es ist offensichtlich, dass die Gleichung $d'[s,t] = \min\{d[s,t],\ d'[s,u] + c'_{uv} + d'[v,t]\}$ gilt (der Unterschied zu Gleichung (1) ist, dass auf der rechten Seite d' statt d genutzt wird). Im Folgenden wird gezeigt, dass es ausreicht die alten kürzesten-Wege-Kosten d zu nutzen. *Beobachtung*: $d'[\cdot,\cdot] \leq d[\cdot,\cdot]$.
Angenommen es würde einen Unterschied machen, ob man d oder d' nutzt, d.h.

$$d'[s,u] + c'_{uv} + d'[v,t] \neq d[s,u] + c'_{uv} + d[v,t] \tag{2}$$
$$d'[s,u] + d'[v,t] \neq d[s,u] + d[v,t] \tag{3}$$

Damit der kürzeste Pfad P_a von s nach u günstiger wird ($d'[s,u] < d'[s,u]$), müsste er die Kante (u,v) enthalten. Wäre dies der Fall, wäre u schon vorher in P_a vorhanden, was der Annahme widerspricht, P_a sei der kürzeste Pfad von s nach u. Somit ist $d'[s,u] = d[s,u]$. Mit der selben Begründung kann man zeigen, dass $d'[v,t] = d[v,t]$.

Damit ist Gleichung (3) unwahr und die Annahme, die kürzeste-Wege Berechnung sei dadurch beeinflusst, ob man d oder d' nutzt, widerlegt. □

(b) In diesem Fall muss man alle kürzesten Weglängen $d[i,j]$ neu berechnen. Man könnte lediglich feststellen, ob die Kante c_{uv} in dem kürzesten Weg enthalten war. Falls die bei keinem Weg der Fall ist, brauch man nichts neu zu berechnen. Falls das aber der Fall ist, kann nicht (mit wenig Aufwand) den „nächstbesten“ Weg finden.

Programmieraufgabe P5

	Laufzeit (sec)	
n	Floyd	Bellmann-Ford
10	0.001	0.050
50	0.028	2.159
100	0.008	90.457
150	0.039	836.231
250	0.092	abgebrochen (nach sehr langem Warten)

Durch das Erstellen größerer Graphen mit n Knoten und $m = \binom{n}{2} = \frac{n(n-1)}{2} = O(n^2)$ Kanten werden die Laufzeitunterschiede deutlich.
Floyds Algorithmus in seiner Laufzeit von $O(n^3)$ zeigt eine bessere Performance als Bellmann-Ford-Algorithmus mit $O(n^2 m) = O(n^4)$.