Lösungen der Hausaufgaben von Übungsblatt 6

Algorithmen und Datenstrukturen (WS 2013, Ulrike von Luxburg)

Lösungen zu Aufgabe 1

(a) Jede Doppelzeile in folgender Tabelle entspricht den Werten für $v.dist$ und $v.\pi$, wobei bei letzterem NIL mittels '-' dargestellt ist.

Extracted	1	2	3	4	5	6	7
(init)	0	∞	∞	∞	∞	∞	∞
	-	-	-	-	-	-	-
1	0	4	∞	∞	∞	5	∞
	-	1	-	-	-	1	-
2	0	4	14	∞	∞	5	7
	-	1	2	-	-	1	2
6	0	4	14	∞	14	5	7
	-	1	2	-	6	1	2
7	0	4	13	∞	10	5	7
	-	1	7	-	7	1	2
5	0	4	12	15	10	5	7
	-	1	5	5	7	1	2
3	0	4	12	14	10	5	7
	-	1	5	3	7	1	2
4	0	4	12	14	10	5	7
	-	1	5	3	7	1	2

An der allerletzten Zeile kann man rückwärts die jeweiligen Vorgänger zurückverfolgen als $4 \leftarrow 3 \leftarrow 5 \leftarrow 7 \leftarrow 2 \leftarrow 1$, also ist $(1, 2, 7, 5, 3, 4)$ kürzester Pfad von 1 nach 4.

(b) Startet man den Dijkstra-Algorithmus an Knoten 4, so wird die Distanz von 4 zu 2 fälschlicherweise als 6 ausgegeben (entlang $(4, 2)$), anstatt Distanz 5 (entlang $(4, 5, 1, 2)$). Grund: Nach Knoten 4 wird als nächstes Knoten 1 extrahiert und bearbeitet. Die spätere Absenkung der Distanz auf -1 beim erneuten Erreichen von 1 über 5 kann daher für die aus 1 ausgehenden Pfade nicht mehr berücksichtigt werden.

Genauer: Der Dijkstra-Algorithmus gestartet an 4 liefert folgende Werte $v.dist$:

Extracted	1	2	3	4	5
(init)	∞	∞	∞	0	∞
4	3	6	∞	0	4
1	3	6	∞	0	4
5	-1	6	∞	0	4
2	-1	6	∞	0	4
3	-1	6	∞	0	4

Obwohl $d(4, 2) = 5$, liefert der Dijkstra-Algorithmus stattdessen Wert 6.

Alternativ kann man genauso zeigen, dass der Dijkstra-Algorithmus gestartet an 3 die Distanz zum Knoten 2 fälschlicherweise als 8 berechnet (entlang $(3, 4, 2)$). Allerdings ist die tatsächliche Distanz $d(3, 2) = 7$, wie der kürzeste Weg $(3, 4, 5, 1, 2)$ zeigt.

Lösungen zu Aufgabe 2

Die einzige Änderung muss in der Relax-Operation vorgenommen werden: Anstatt in $v.dist$ die Pfadlänge $\ell(p)$ als Aufsummierung der Kantengewichte zu speichern, verwenden wir nun das maximale Kantengewicht entlang des Pfads: $\max_{e \in p} w(e)$.

function $\textsc{Relax}(u, v)$
  if $v.dist > \max\{u.dist,\ w(u,v)\}$ **then**
    $v.dist \leftarrow \max\{u.dist,\ w(u,v)\}$
    $v.\pi \leftarrow u$
  end if
end function

Lösungen zu Aufgabe 3

(a) *Induktionsanfang* $k = 0$: Es ist $A^0 = I$ die Einheitsmatrix, welche korrekterweise keinen Pfad (der Länge 0) für $i \neq j$ als $A^0[i,j]$ liefert, sowie genau einen für $i = j$.

Induktionsvoraussetzung: $A^k[i,j]$ entspricht der Anzahl Pfade der Länge k von i nach j.

Induktionsschritt $k \mapsto k+1$: Es ist $A^{k+1}[i,j] = \sum_{v=1}^{n} A^k[i,v]A[v,j]$, worin nach IV $A^k[i,v]$ der Anzahl Pfade der Länge k von i nach v entspricht. Außerdem ist $A[v,j]$ gleich 1 genau dann, wenn eine Kante von v nach j existiert, sonst 0. Daher summiert obiger Ausdruck $A^{k+1}[i,j]$ über alle möglichen Pfade der Länge $k+1$ von i nach j auf und liefert somit deren Anzahl.

(b) Berechne A^k (effizienterweise etwa mittels "Square-And-Multiply") und liefere *true* als Ausgabe genau dann, wenn $A^k[i,j]$ ungleich 0 ist. In (a) wurde gezeigt, dass genau dann ein Pfad von i nach j der Länge k existiert.

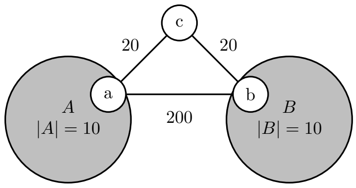
(c) Setze zunächst jeden Eintrag der Hauptdiagonalen von A auf 1 (füge also jedem bislang schleifenfreien Knoten eine Schleife hinzu). Offensichtlich (betrachte in obiger Summe den Fall $v = j$) gilt nun für alle $s \leq t$, dass $A^s[i,j] > 0 \Rightarrow A^t[i,j] > 0$. Daher ist nun $A^k[i,j]$ ungleich 0 genau dann, wenn ein Pfad von i nach j der Länge k oder kürzer existiert.

Lösungen zu Aufgabe 4

(a) Sei $L(W) := \sum_{p \in W} \ell(p)$ die Gesamtlänge aller Pfade ("Gesamtlast") in W. Man erhält denselben Wert über die alternative Aufsummierung $L(W) = \sum_{e \in E} c(e)$. Dies liefert, dass $|E| \cdot c(W) = \sum_{e \in E} c(W) \geq \sum_{e \in E} c(e) = \sum_{p \in W} \ell(p)$.

(Man könnte auch so argumentieren: Ganz allgemein gilt für n nicht-negative Zahlen $a_1, \ldots, a_n$, dass $\max_i\{a_i\} \geq \frac{1}{n} \sum_{i=1}^{n} a_i$, somit $c(W) \geq \frac{1}{|E|} \sum_{e \in E} c(e)$)

(b) Die Verwendung kürzester Pfade ist *nicht* hinreichend für eine optimale Kantenlast. Zum Beispiel habe der folgende Graph $G = (V, E)$ die Knotenmenge $V = A \mathbin{\dot\cup} B \mathbin{\dot\cup} \{c\}$ mit $|A| = |B| = 10$. Die Kantenmenge E sei darüber definiert, dass A und B jeweils vollständige Graphen sind, und $\{a, b, c\}$ für ein $a \in A$ und $b \in B$ ein Dreiecksgraph. G ist hier skizziert:



Sei nun W^* über die kürzesten Pfade in G definiert. Die Last der Kante (a, b) ist dann 200, da für jedes der hundert Paare $(x, y) \in A \times B$ der kürzeste Pfad (in beide Richtungen) über die Kante (a, b) führt. Hingegen ist die Last der Kante (c, a) nur 20, da hier nur die kürzesten Pfade zwischen c und den Knoten in A entlangführen (für die Kante (c, b) analog). Alle übrigen Routen verlaufen innerhalb von A (bzw. B) entlang der direkten Verbindungskante, jede demnach mit Last 2. Somit gilt $c(W^*) = 200$.

Offensichtlich lässt sich die maximale Kantenlast verringern, indem ein alternatives Pfadsystem W einige der Verbindungen zwischen A und B nicht entlang (a, b) routet, sondern entlang eines längeren Pfades über (a, c, b). Würde man zum Beispiel 90 Routen derart umleiten, so hätte jede der Kanten $(a, b), (b, c)$ und (c, a) die Last 110, und somit $c(W) < c(W^*)$.

Eine geringe Last wird also durch eine gute Verteilung aller Pfade im Netzwerk erreicht, nicht aber durch bloßes Verfolgen kürzester Pfade.