AD-Übung zum 15. Januar 2014

Arne Beer, MN 6489196
Merve Yilmaz, MN 6414978
Sascha Schulz, MN 6434677

14. Januar 2014

1. Dijkstra-Aldorithmus

(a) Tabellarische Darstellung der Entwicklung:

Schritt	Knoten	$v.dist$	$v.\pi$
0	1	0	
	2	∞	NIL
	3	∞	NIL
	4	∞	NIL
	5	∞	NIL
	6	∞	NIL
	7	∞	NIL
1	1	0	
	2	4	1
	3	∞	NIL
	4	∞	NIL
	5	∞	NIL
	6	5	1
	7	∞	NIL
2	1	0	
	2	4	1
	3	10	2
	4	∞	NIL
	5	∞	NIL
	6	5	1
	7	3	2
3	1	0	
	2	4	1
	3	14	2
	4	∞	NIL
	5	14	6
	6	5	1
	7	7	2

Schritt	Knoten	$v.dist$	$v.\pi$
4	1	0	
	2	4	1
	3	13	7
	4	∞	NIL
	5	10	7
	6	5	1
	7	7	2
5	1	0	
	2	4	1
	3	12	5
	4	15	5
	5	10	7
	6	5	1
	7	7	2
6	1	0	
	2	4	1
	3	12	5
	4	14	3
	5	10	7
	6	5	1
	7	7	2

Kürzester Pfad für $1 \to 4$:
$1 \to 2 \to 7 \to 5 \to 3 \to 4$ mit Distanz 14.

(b) Der Dijkstra-Algorithmus führt bei Graphen mit negativen Kanten, wie hier mit $w(5,1) = -5$ zu falschen Ergebnissen.

Berechnen wir mit dem Naiven Dijkstra-Algorithmus alle kürzesten Pfade zu der Source (3), so gelangt der Algorithmus für den Zielknoten 1 zu dem Ergebnis, dass der kürzeste Pfad $3 \to 4 \to 1$ mit einer Distanz von 5 ist. Der tatsächlich kürzeste Pfad ist allerdings $3 \to 4 \to 5 \to 1$ mit

einer Distanz von 1. Ursache ist, dass stets nur der nächst dichteste Knoten in die Menge der bestimmten Knoten aufgenommen wird, und somit der Knoten 1 vor dem Knoten 5.

2. Dijkstra-Modifikation

Da die Distanz des Pfades nicht weiter von belang ist, sondern ausschließlich der maximal-schwere Pfadabschnitt, merken wir uns diesen und nutzen dies als Kriteritum zu Vorgänger-Auswahl:

```
DijkstraMinWeightPaths(G,w,s) {
    S := {s} # S set of explored vertices
    maxw(s) := 0
    while not S = V {
        U := {u not in S | u neigh. of a vertex in S} # candidates
        for all u in U {
            for all pre(u) in S that are predecessors of u {
                maxw'(u, pre(u)) := max{ maxw(pre(u)), w(pre(u),u) }
            }
        }
        u* := argmin{maxw'(u, pre(u)) | u in U} # choose best candidate
        maxw(u*) = maxw'(u*)
        S = S U {u*}
    }
}
```

Die verwendete Funktion $maxw(v)$ gibt das größte Kantengewicht für den Knoten v an, welches sich auf der Pfadstrecke von v zur Source befindet.

Der Algorithmus terminiert weiterhin, da mit jedem Schritt ein Knoten aus der Menge V\S der Menge S hinzugefügt wird. Da die Menge V endlich ist, sind somit nach endlich vielen Schritten alle Knoten S hinzugefügt und der Algorithmus terminiert.

Der Algorithmus liefert das korrekte Ergebnis. Pro Schritt erzeugt er einen (Teil-)Pfad, welcher das bisher minimale Gewicht an der schwersten Kante hat. Somit werden Pfade in der aufsteigenden Reihenfolge ihrer maximalen Gewichte erzeugt.

Es können keine Zyklen erzeugt werden, da weiterhin ausschließlich adjazente Knoten betrachtet werden, welche noch nicht der Ergebnismenge hinzugefügt wurden. Es ist dadurch ausschließlich möglich bisherige Pfade um Kanten zu verlängern, die einen bereits besuchten Knoten mit einem unbesuchten Knoten verbinden.

3. Adjazenzmatrix und Pfade

(a) *Behauptung:*
Bei einem beliebigen ungewichteten Graphen G sei A die zugehörige $n \times n$-Adjazenzmatrix. Dann entspricht der Wert für alle $k \in \mathbb{N}_0$ an der Stelle $A^k[i,j]$ der Anzahl verschiedener Pfade der länge k von i nach j in G.

Induktionsanfang:
Der Fall k=0 ist trivial (Einheitsmatrix). Es wereden 0 Kanten besucht, Knoten sind mit sich selbst verbunden.

Der Fall k=1 ist die Adjazenzmatrix. Sie gibt an, welche Knoten mi einer Kante erreichbar sind. Da es sich um einen ungerichteten Graphen handelt ist diese stets symmetrisch.

Der Fall k=2 ist der erste betrachtenswerte. Bei der Matritzenmultiplikation werden Zeile und Spalte der Adjazenzmatrix miteinander in Bezug gesetzt. Enthalten beide Werte ungleich 0, ergibt dies einen Möglichen Pfad der Länge 2.

Wir betrachten die Operationen für ein beliebiges Feld $A_{[i,j]}$ der Ergebnismatrix A^2 im Detail:
$A^2_{[i,j]} = A_{[i,1]} \cdot A_{[1,j]} + A_{[i,2]} \cdot A_{[2,j]} + \ldots + A_{[i,n]} \cdot A_{[n,j]} = \sum_{k=1}^{k=n} (A_{[i,k]} \cdot A_{[k,j]})$.
Der Fall $i = k = j$ repräsentiert dabei eine Schleife (der Knoten ist adjazent zu sich selbst).

Andernfalls repräsentiert k einen Knoten und die Multiplikation ‘prüft‘ ob sowohl i als auch j zu diesem adjazent sind.

Das Ergebnis der Matritzenmultiplikation bietet folglich für das Feld $A^2_{[i,j]}$ die Summe aus Schleife und Anzahl der gemeinsamen Adjazenten, und somit die korrekte Anzahl aller möglichen Pfade der Länge 2 vom Knoten i zum Knoten k.

Induktionsannahme:
Die Behauptung gilt für einen beliebig gewählten Graphen G, wir verfügen somit über ein gültige A^n mit den korrekten Anzahlen für mögliche Pfade der Länge n.

Induktionsschritt:
Für die Korrektheit von A^{n+1} betrachtetn wir die Matritzenmultiplikation $A^n \cdot A$. Wieder betrachten wir das Erzeugen eines beliebigen Wertes in A^{n+1} : $A^{n+1}_{[i,j]}$:

$A^{n+1}_{[i,j]} = \sum_{k=1}^{k=n}(A^n_{[i,k]} \cdot A_{[k,j]})$

Sei k für die folgende Betrachtung nun fest gewählt (wir befinden uns also innerhalb der Summenoperation). Die Matrix A^n gibt im Feld $A^n_{[i,k]}$ an, wie viele Pfade der länge n es vom Knoten der i-ten Zeile zum Knoten der k-ten Spalte gibt. Wir multiplizieren dies mit der einfachen Adjazenzmatrix, welche Auskunft gibt, ob für $A^n_{[k,j]}$ der Knoten der k-ten Zeile mit der j-ten Spalte adjazent ist. Ist dies der Fall, steht dort eine 1. Dann aber gibt es für dieses k genau $A^n_{[i,k]}$ viele Pfade, welche sich aus einem Pfad der länge n von i nach k und anschließender Kante von k nach j zusammen setzen und somit eine Gesamtlänge von n+1 haben.

Die Summe dieser Pfade stellt schließlich alle Möglichkeiten dar, einen beliebig langen Pfad der länge n zu gehen und anschließend mittels einer Adjazenz diesen auf n+1 zu erweitern.

Folglich ist die Behauptung korrekt.

(b)
```
existingPath(matrix A, lenght k){
A = A^k
    foreach(i,j){
        if(A[i,j] > 0){
            //Possible Path from i to j
        }
    }
}
```

(c) TODO

4. Kürzeste Pfade in Routingnetzwerken

(a) Bei einer absoluten Gleichverteilung, wuerde jede Kante die selbe Last besitzen.

Die Gesamtzahl aller verwendeten Kanten lässt sich also wie folgt berrechnen: $\sum_{p \in W} l(p)$. In einem optimalen Pfadesystem verteilt sich die Gesamtzahl auf alle Kanten. Jede Kante hat folglich die Last $\frac{1}{|E|} \sum_{p \in W} l(p)$, was die untere Grenze für $c(W)$ angibt In jedem nicht-optimalen Pfadsystem hingegen hat mindestens eine Kante eine höhere Last, damit kann $c(W)$ nur höher werden.