4.2 Modifikation des Bellman-Ford-Algorithmus

$B_k[i]$ gibt die Länge eines kürzesten gerichteten s-i-Pfades an, der aus höchstens k Kanten besteht. Jeder Pfad, der keinen Kreis enthält, besteht aus maximal $n-1$ Kanten. In einem Graphen ohne negative Kreise gilt daher:

$$\forall i \in V : B_n[i] \geq B_{n-1}[i]$$

Gibt es hingegen einen (von s aus erreichbaren) Kreis negativer Länge, so gibt es einen Knoten $i \in V$, bei dem ein Pfad aus n Kanten mit der Länge $B_n[i]$ diesen Kreis häufiger durchläuft als jeder Pfad aus maximal $n-1$ Kanten der Länge $B_{n-1}[i]$. Demnach gilt in diesem Fall:

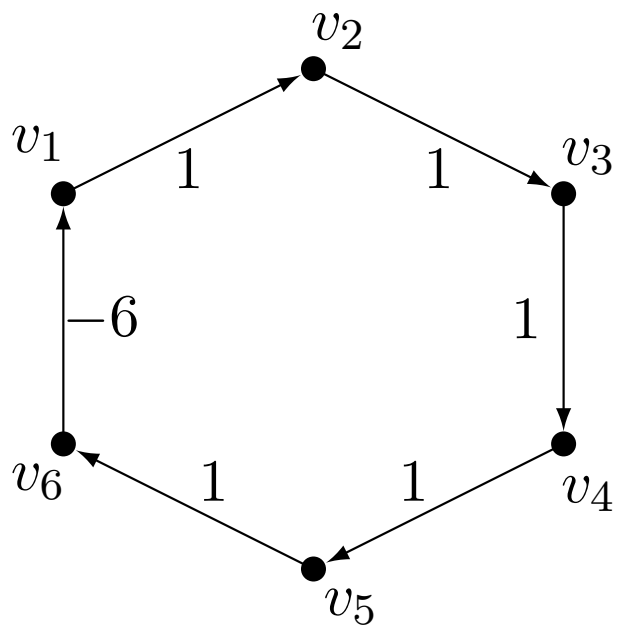
$$B_n[i] < B_{n-1}[i]$$

Man kann also in den Algorithmus von Bellman-Ford einen Test auf negative Kreise einbauen, indem man auch für alle $i \in V$ $B_n[i]$ berechnet und am Ende den folgenden Befehl einfügt:

for $i := 1$ **to** n **do**
  if $B_n[i] < B_{n-1}[i]$ **then** stop „Negativer Kreis“ **fi**

4.3 Modifikation des Floyd-Algorithmus

Falls kein negativer Kreis existiert, funktioniert der Algorithmus weiterhin korrekt.



$$c_{16}^6 = 5 = c_{16}^5$$

$$c_{61}^6 = -6 = c_{61}^5$$

$$c_{11}^6 = \min\{c_{11}^5, c_{16}^5 + c_{61}^5\} = -1$$

$\Rightarrow$ der Graph enthält einen negativen Kreis, gdw ein $c_{ii}^n < 0$ existiert.

Man kann also in den Algorithmus von Floyd einen Test auf negative Kreise einbauen, indem man am Ende den folgenden Befehl einfügt:

for $i := 1$ **to** n **do**
    if $c_{ii}^n < 0$ **then** stop „Negativer Kreis“ **fi**

4.4 Der Algorithmus von Johnson

Definition 113
Sei $\mathrm{d} : A \to \mathbb{R}$ eine Distanzfunktion. Eine Abbildung

$$r : V \to \mathbb{R}$$

heißt Rekalibrierung, falls gilt:

$$(\forall (u,v) \in A)[r(u) + \mathrm{d}(u,v) \geq r(v)]$$

Beobachtung: Sei r eine Rekalibrierung (für d). Setze $\mathrm{d}'(u,v) := \mathrm{d}(u,v) + r(u) - r(v)$. Dann gilt:

$$\mathrm{d}'(u,v) \geq 0$$

Sei $u = v_0 \to \cdots \to v_k = v$ ein Pfad. Dann ist:

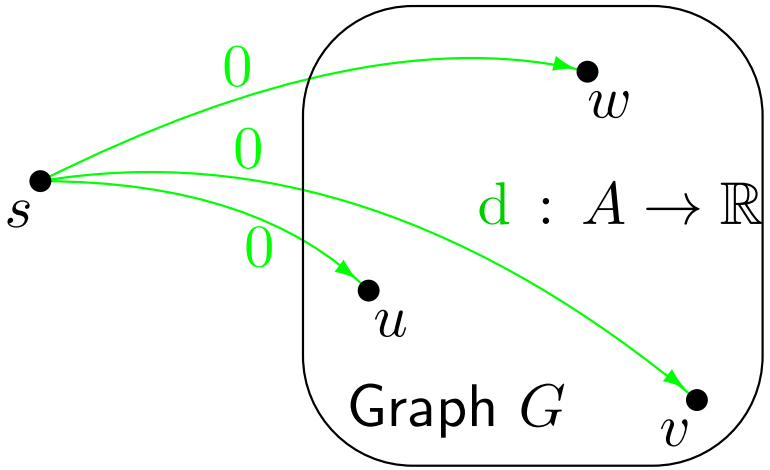
$$\text{d-Länge} := \sum_{i=0}^{k-1} \mathrm{d}(v_i, v_{i+1})$$

Demnach ist:

$$\begin{aligned}
\text{d}'\text{-Länge} &= \sum_{i=0}^{k-1} \mathrm{d}'(v_i, v_{i+1}) \\
&= \sum_{i=0}^{k-1} (\mathrm{d}(v_i, v_{i+1}) + r(v_i) - r(v_{i+1})) \\
&= \sum_{i=0}^{k-1} \mathrm{d}(v_i, v_{i+1}) + r(v_0) - r(v_k)
\end{aligned}$$

Also ist ein d-kürzester Pfad von u $(= v_0)$ nach v $(= v_k)$ auch ein d'-kürzester Pfad und umgekehrt. Nach einer Rekalibrierung kann man also auch die Algorithmen anwenden, die eine nichtnegative Distanzfunktion d voraussetzen (z.B. Dijkstra).

Berechnung einer Rekalibrierung:



Füge einen neuen Knoten s hinzu und verbinde s mit jedem anderen Knoten $v \in V$ durch eine Kante der Länge 0.

Berechne sssp von s nach allen anderen Knoten $v \in V$ (z.B. mit Bellman-Ford). Sei $r(v)$ die dadurch berechnete Entfernung von s zu $v \in V$. Dann ist r eine Rekalibrierung, denn es gilt:

$$r(u) + \mathrm{d}(u, v) \geq r(v) \,.$$

5. Zusammenfassung

	$d \geq 0$	d allgemein
sssp	D (Fibonacci): $\mathcal{O}(m + n \cdot \log n)$ D (Radix): $\mathcal{O}(m + n\sqrt{\log C})$	B-F: $\mathcal{O}(n \cdot m)$
apsp	D: $\mathcal{O}(n \cdot m + n^2 \min\{\log n, \sqrt{\log C}\})$ F: $\mathcal{O}(n^3)^{(*)}$	J: $\mathcal{O}(n \cdot m + n^2 \log n)$ F: $\mathcal{O}(n^3)$

Bemerkung$^{(*)}$: In der Praxis ist der Floyd-Algorithmus für kleine n besser als Dijkstra's Algorithmus.

6. Transitive Hülle

6.1 Min-Plus-Matrix-Produkt und Min-Plus-Transitive Hülle

Ring $\mathbb{Z}(+, \times)$ — Gruppe, Halbgruppe

Semiring $\mathbb{N}(+, \times)$ — Halbgruppe, Halbgruppe

Wir betrachten den (kommutativen) Semiring über $\mathbb{R} \cup \{\infty\}$ mit den Operationen $\min$ und $+$. Für jede der beiden Operationen haben wir ein Monoid. Es gilt das Distributivgesetz $a + \min\{b, c\} = \min\{a + b, a + c\}$.

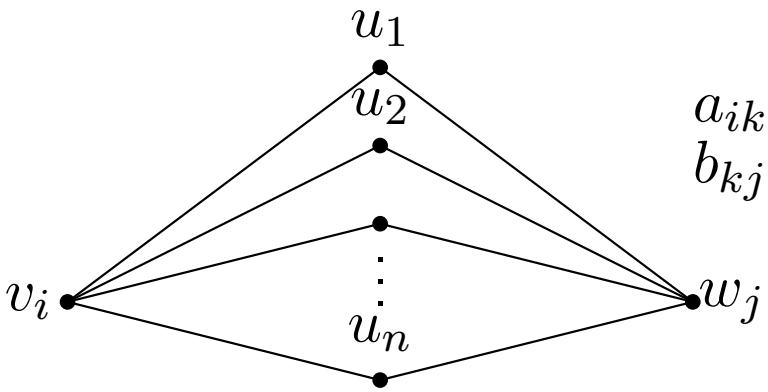
Normale Matrixmultiplikation:

$$A = (a_{ij})_{1 \le i,j \le n}, \quad B = (b_{ij})_{1 \le i,j \le n}, \quad I = (\delta_{ij})_{1 \le i,j \le n}$$

$$C = A \cdot B = (c_{ij})_{1 \le i,j \le n}, \quad c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Entsprechend für Min-Plus:

$$c_{ij} = \min\{a_{ik} + b_{kj};\ 1 \le k \le n\}$$



$$a_{ik} = \mathrm{d}(v_i, u_k)$$
$$b_{kj} = \mathrm{d}(u_k, w_j)$$

Anwendung:
kürzeste Wege von v_i nach w_j in einem Graph ($A = B$); dabei ist

$$I_{\min,+} = \begin{pmatrix} 0 & & \infty \\ & \ddots & \\ \infty & & 0 \end{pmatrix}$$

Sei A Entfernungsmatrix, $A = (a_{ij})_{1 \le i,j \le n} = (\mathrm{d}(v_i, v_j))_{1 \le i,j \le n}$.
Setze $a_{ii} = 0$ für $i = 1, \ldots, n$.

Betrachte A^2 mit dem Min-Plus-Produkt, $A^2 =: (a_{ij}^{(2)})_{1 \le i,j \le n}$.

Dann ist $a_{ij}^{(2)}$ die Länge eines kürzesten Pfades von v_i nach v_j, der höchstens zwei Kanten enthält. Induktion ergibt: $a_{ij}^{(k)}$ ist die Länge eines kürzesten Pfades von v_i nach v_j mit höchstens k Kanten.
Falls die $\mathrm{d}(v_i, v_j)$ alle ≥ 0 sind, gibt es immer kürzeste Pfade, die höchstens $n - 1$ Kanten enthalten.

Damit ergibt sich folgende alternative Lösung des all-pairs-shortest-path-Problems:

Berechne A^{n-1} (Min-Plus)!

Es genügt auch, $A^{2^{\lceil \log(n-1) \rceil}}$ durch wiederholtes Quadrieren zu berechnen (nicht $A^2, A^3, A^4, \ldots$).

Definition 114
$A^* := \min_{i \geq 0}\{A^i\}$ heißt Min-Plus-Transitive Hülle.

Bemerkung: $\min$ wird komponentenweise gebildet. Wenn $\mathrm{d} \geq 0$, dann $A^* = A^{n-1}$.