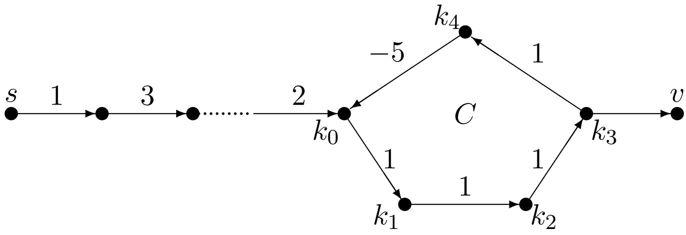
8.4 Digraphen mit negativen Kantengewichten

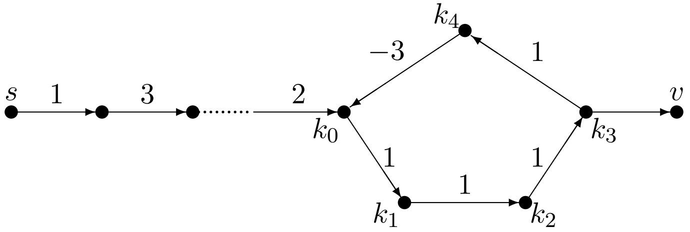
8.4.1 Grundsätzliches

Betrachte Startknoten s und einen Kreis C mit Gesamtlänge < 0.



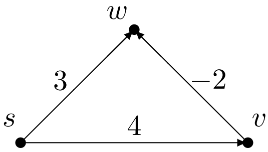
Sollte ein Pfad von s nach C und von C nach v existieren, so ist ein kürzester Pfad von s nach v nicht definiert.

Falls aber die Gesamtlänge des Kreises $C \geq 0$ ist,



dann ist der kürzeste Pfad (der dann o.B.d.A. als kreisfrei genommen werden kann) wohldefiniert. Probleme gibt es also nur dann, wenn G einen Zyklus negativer Länge enthält.

Dijkstra's Algorithmus funktioniert bei negativen Kantenlängen nicht:



Bei diesem Beispielgraphen (der nicht einmal einen negativen Kreis enthält) berechnet der Dijkstra-Algorithmus die minimale Entfernung von s nach w fälschlicherweise als 3 (statt 2).

8.4.2 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:

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

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:

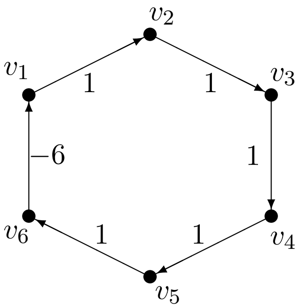
$$B_n[i] < B_{n-1}[i]$$

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**

8.4.3 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**

8.4.4 Der Algorithmus von Johnson

Definition 197

Sei $\mathrm{d} : A \to \mathbb{R} \cup \{+\infty\}$ 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:

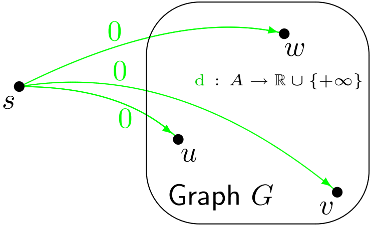
$$\mathrm{d}\text{-Länge} := \sum_{i=0}^{k-1} \mathrm{d}(v_i, v_{i+1})$$

Demnach ist:

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

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:

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

8.5 Zusammenfassung

	$d \geq 0$	d allgemein
sssp	D (Fibonacci): $\mathcal{O}(m + n \cdot \log n)$	B-F: $\mathcal{O}(n \cdot m)$
apsp	D: $\mathcal{O}(nm + n^2 \log n)$ 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.