Der Algorithmus von Floyd & Warshall

<u>Idee:</u> Dynamische Programmierung

$F_k[i, j] :=$ Länge eines kürzesten i-j-Pfades an, der als **innere Knoten** (das sind die Knoten des Pfades außer den Endknoten i und j) nur Knoten aus $\{1, \ \ldots, \ k\}$ verwendet.

Initialisierung:

$$\forall\, i,\ j\ \in\ \mathrm{V}: \qquad F_0[i, j] := \begin{cases} l(\{i, j\}) & \text{falls } \{i, j\} \in \mathrm{E} \\ 0 & \text{falls } i = j \\ \infty & \text{sonst} \end{cases}$$

Rekursion:

for $(k := 1; k \leq n; k{+}{+})$

$$\forall\, i,\ j\ \in\ \mathrm{V}: F_k[i, j] := \min\{F_{k-1}[i, j], F_{k-1}[i, k] + F_{k-1}[k, j]\};$$

Ausgabe: $F_n[i, j]$ gibt die Länge eines kürzesten i-j-Pfades an.

Der Algorithmus von Bellman & Ford

<u>Idee:</u> Dynamische Programmierung

$B_k[i]$:= Länge eines kürzesten s-i-Pfades an, der aus höchstens k Kanten besteht.

Initialisierung:

$$\forall\, i \;\in\; \mathrm{V}: \qquad B_1[i] := \begin{cases} l(\{s,i\}) & \text{falls } \{s,i\} \in \mathrm{E} \\ 0 & \text{falls } i = s \\ \infty & \text{sonst} \end{cases} \qquad . \tag{1}$$

Rekursion:

for $(k := 2; k \leq n-1; k++)$

$\qquad \forall\, i \;\in\; \mathrm{V} \setminus \{s\}: \quad B_k[i] := \min_{v \in \Gamma(i)} \{B_{k-1}[v] + l(\{v,i\})\};$

$\qquad B_{k-1}[s] := 0;$

Ausgabe: $B_{n-1}[i]$ gibt die Länge eines kürzesten s-i-Pfades an.

Modifikation des Algorithmus von Floyd & Warshall

... für gerichtete Graphen $D = (V, A)$ mit beliebige Gewichtsfunktion $\ell : A \to \mathbb{R}$.

Lemma: Enthält D einen negativen Kreis, so gibt es mindestens ein $i \in V$, für das $F_n[i, i] < 0$ gilt.

Man kann also in den Algorithmus von Floyd-Warshall einen Test auf negative Kreise einbauen, indem man direkt vor der Ausgabe den folgenden Befehl einfügt:

$\forall\, i \ \in \ \mathrm{V}$: if $F_n[i, i] < 0$ then STOP: Es gibt einen negativen Kreis;

Modifikation des Algorithmus von Bellman & Ford

Idee: $B_k[i]$ enthält nun die Länge eines kürzesten *gerichteten* s-i-Kantenzuges, der aus höchstens k Kanten besteht.

Beobachtungen:

- Jeder beliebige Kantenzug, der keinen Kreis enthält, besteht aus maximal $n-1$ Kanten.
- In einem Graphen ohne negative Kreise gilt daher $\forall\, i \in \mathrm{V} : B_n[i] \geq B_{n-1}[i]$.
- Gibt es dagegen einen (von s aus erreichbaren) negativen Kreis in D, so gibt es einen Knoten $i \in \mathrm{V}$, bei dem ein Kantenzug aus n Kanten mit Länge $B_n[i]$ diesen Kreis häufiger durchläuft als jeder Kantenzug aus max. $n-1$ Kanten der Länge $B_{n-1}[i]$. Demnach gilt dann $B_n[i] < B_{n-1}[i]$.

Man kann also in den Algorithmus von Bellman-Ford einen Test auf negative Kreise einbauen, indem man in der Rekursion auch $\forall\, i \in \mathrm{V} : B_n[i]$ berechnet und direkt vor der Ausgabe den folgenden Befehl einfügt:

$\forall\, i \in \mathrm{V}$: **if** $B_n[i] < B_{n-1}[i]$ **then** STOP: Es gibt einen negativen Kreis;

Der Algorithmus von Johnson

Ziel: Lösung des “all pairs”–Problem mit einer beliebigen (negativen) Längenfunktion ℓ

Idee: Verwende eine so genannte *Rekalibrierung*:

$f : \mathrm{V} \mapsto \mathbb{R}$ heißt eine **Rekalibrierung** eines gerichteten Graphen $D = (V; A)$ mit Gewichtsfunktion $\ell : \mathrm{V} \mapsto \mathbb{R}$, falls $\forall\, (u, v) \in \mathrm{A} : f(u) + \ell(u, v) \geq f(v)$ gilt.

Beobachtungen:

Ist f eine Rekalibrierung, so ist $\ell_f : \mathrm{A} \mapsto \mathbb{R}$ mit
$\forall\, (u, v) \in \mathrm{A} : \ell_f(u, v) := \ell(u, v) + f(u) - f(v)$ eine nichtnegative Längenfunktion.

Für ℓ_f gilt zusätzlich: Ist $\mathrm{P} = (v_0, \ \ldots, \ v_k)$ ein Pfad, so ist
$\ell_f(\mathrm{P}) = \sum_{i=0}^{k-1} \ \ell_f(v_i, v_{i+1}) = \sum_{i=0}^{k-1} \Big(\ l(v_i, v_{i+1}) + f(v_i) - f(v_{i+1}) \ \Big) =$
$\sum_{i=0}^{k-1} \ \ell(v_i, v_{i+1}) + f(v_0) - f(v_k) = \ell(\mathrm{P}) + f(v_0) - f(v_k)$, d.h., die Pfadlängen unterscheiden sich nur durch eine additive Konstante, die nicht vom Pfad, sondern nur von Anfangs- und Endknoten abhängt.

Also ist P genau dann ein kürzester s-t-Pfad bzgl. ℓ, wenn P ein kürzester s-t-Pfad bzgl. ℓ_f ist.

Der Algorithmus von Johnson (2)

Ansatz für Algorithmus: Bestimme zuerst eine Rekalibrierung f und löse dann das "all pair shortest path"–Problem bzgl. ℓ_f (z.B. mit Dijkstra).

Bestimmung der Rekalibrierung:

Füge zume Graphen einen zusätzlichen Knoten ξ hinzu und verbinde ξ mit jedem anderen Knoten durch eine Kante der Länge 0 verbindet.

Verwende nun den Algorithmus von Bellman & Ford, um in dem neuen Graphen kürzeste ξ-v Pfade zu bestimmen, für alle $v \in V$.

Falls der Algorithmus von Bellman & Ford einen negativen Kreis findet: STOP

Ansonsten sei $d(v)$ die Länge eines kürzesten ξ-v-Pfades an (da negative Kanten auftreten können, ist d nicht zwangsläufig konstant Null!).

d ist eine Rekalibrierung, da $\forall\ (u, v)\ \in\ \mathrm{A} : d(u) + \ell(u, v) \geq d(v)$ nach Definition von d gilt.

Das Kürzeste-Wege-Problem

	$\ell \geq 0$	ℓ beliebig
Startknoten s	Dijkstra: $\mathcal{O}(m + n \log n)$	Bellman-Ford: $\mathcal{O}(nm)$
Alle Paare	Dijkstra: $\mathcal{O}(nm + n^2 \log n)$ Floyd-Warshall: $\mathcal{O}(n^3)$	Bellman-Ford & Dijkstra: $\mathcal{O}(nm + n^2 \log n)$ Floyd-Warshall: $\mathcal{O}(n^3)$