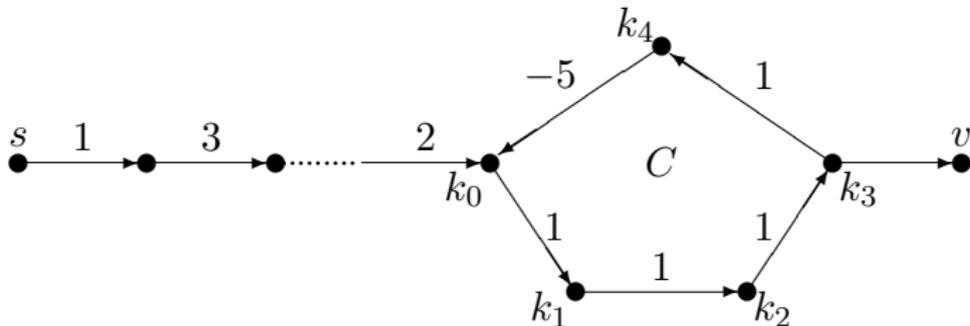


4. Digraphen mit negativen Kantengewichten

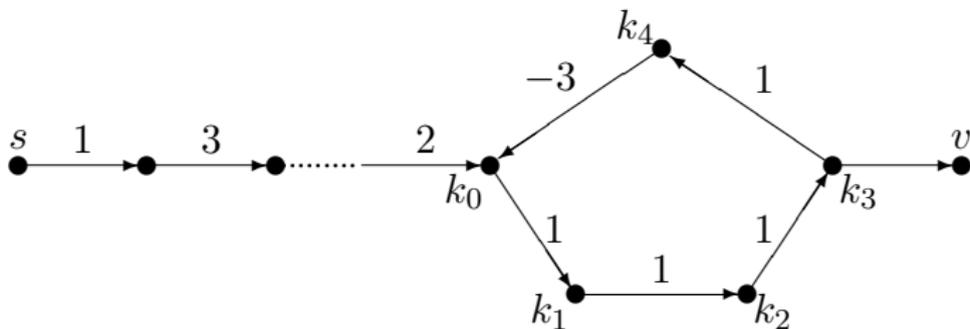
4.1 Grundsätzliches

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



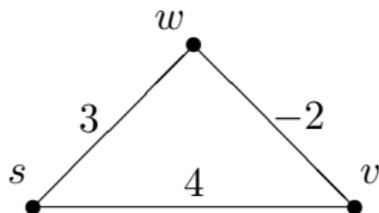
Sollte ein Pfad von s nach C und von C nach v existieren, so ist ein kürzester Pfad von s nach v **nicht definiert**.

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



dann ist der kürzeste Pfad 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).

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:

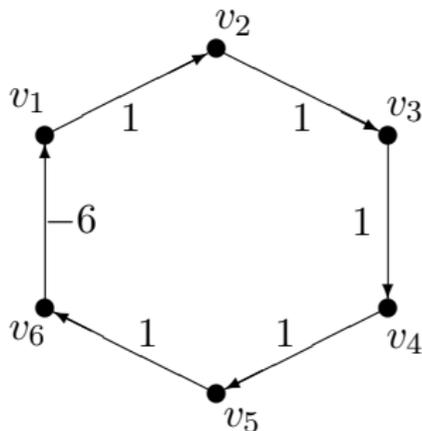
$$B_n[i] < B_{n-1}[i]$$

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 $d : A \rightarrow \mathbb{R}$ eine Distanzfunktion. Eine Abbildung

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

heißt **Rekalibrierung**, falls gilt:

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

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

$$d'(u, v) \geq 0$$

Sei $u = v_0 \rightarrow \dots \rightarrow v_k = v$ ein Pfad. Dann ist:

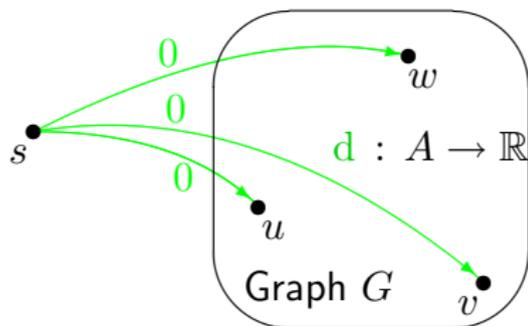
$$\mathbf{d}\text{-Länge} := \sum_{i=0}^{k-1} \mathbf{d}(v_i, v_{i+1})$$

Demnach ist:

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

Also ist ein \mathbf{d} -kürzester Pfad von $u (= v_0)$ nach $v (= v_k)$ auch ein \mathbf{d}' -kürzester Pfad und umgekehrt. Nach einer Rekalibrierung kann man also auch die Algorithmen anwenden, die eine nichtnegative Distanzfunktion \mathbf{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:

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

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: $\mathcal{O}(n \cdot m)$
apssp	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.

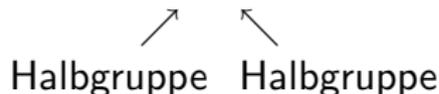
6. Transitive Hülle

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

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



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



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**
 $a + \min\{b, c\} = \min\{a + b, a + c\}$.

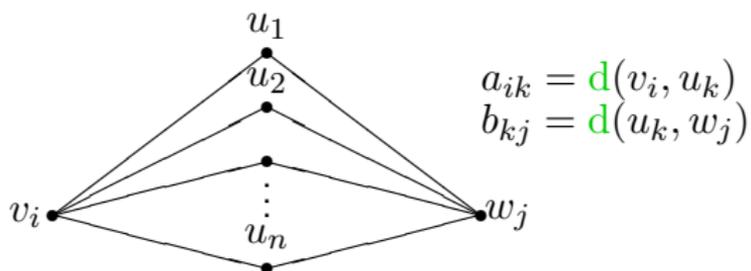
Normale Matrixmultiplikation:

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

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

Entsprechend für Min-Plus:

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



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, $A = (a_{ij})_{1 \leq i, j \leq n} = (d(v_i, v_j))_{1 \leq i, j \leq n}$.
Setze $a_{ii} = 0$ für $i = 1, \dots, n$.

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

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 $d(v_i, v_j)$ 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, \dots).

Definition 114

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

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

6.2 Boolesche Matrixmultiplikation und Transitive Hülle

Wir ersetzen nun im vorhergehenden Abschnitt die Distanzmatrix durch die (boolesche) Adjazenzmatrix und $(\min, +)$ durch (\vee, \wedge) , d.h.:

$$C = A \cdot B; \quad c_{ij} = \bigvee_{k=1}^n a_{ik} \wedge b_{kj}$$

Wenn wir zudem $a_{ii} = 1$ für $1 \leq i \leq n$ setzen, dann gilt für A^k (boolesches Produkt, $A^0 = I$)

$$a_{ij} = \begin{cases} 1 & \text{falls es im Graphen einen Pfad von } v_i \text{ nach } v_j, \\ & \text{bestehend aus } \leq k \text{ Kanten gibt} \\ 0 & \text{falls es im Graphen keinen Pfad von } v_i \text{ nach } v_j, \\ & \text{bestehend aus } \leq k \text{ Kanten gibt} \end{cases}$$

Transitive Hülle:

$$A^* := \bigvee_{i \geq 0} A^i \quad (= A^{n-1})$$

ist damit die Adjazenzmatrix der transitiven Hülle des zugrunde liegenden Digraphen.

Satz 115

Sei $M(n)$ die Zeitkomplexität für das boolesche Produkt zweier $n \times n$ -Matrizen, $T(n)$ die Zeitkomplexität für die transitive Hülle einer $n \times n$ booleschen Matrix.

Falls $\underbrace{T(3n) \leq cT(n)}_{\substack{\text{sicher erfüllt, falls} \\ T \text{ polynomiell}}}$ und $\underbrace{M(2n) \geq 4M(n)}_{\substack{\text{sicher erfüllt, falls} \\ M(n) \geq n^2}}$, dann gilt:

$$T(n) = \Theta(M(n)).$$

Beweis:

(1) Matrixmultiplikation \prec transitive Hülle:

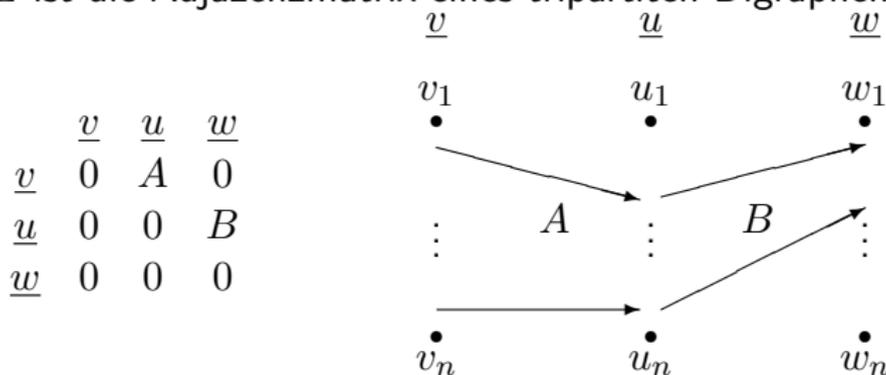
Seien boolesche Matrizen A , B gegeben und ihr boolesches Produkt $C = A \cdot B$ gesucht.

Setze:

$$L = \underbrace{\begin{pmatrix} 0 & A & 0 \\ 0 & 0 & B \\ 0 & 0 & 0 \end{pmatrix}}_{3n} \Bigg\} 3n$$

Beweis (Forts.):

L ist die Adjazenzmatrix eines tripartiten Digraphen, denn:



Daher kann L^* leicht bestimmt werden:

$$L^* = \begin{pmatrix} I & A & AB \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix} \quad (= I \vee L \vee L^2)$$

Also gilt: $M(n) \leq T(3n) = \mathcal{O}(T(n))$.

Beweis (Forts.):

(2) Transitiv Hülle \prec Matrixmultiplikation:

Gegeben: $n \times n$ boolesche Matrix L ; gesucht: L^* ; Annahme: n ist Zweierpotenz. Teile auf:

$$L = \left(\begin{array}{cc} A & B \\ C & D \end{array} \right) \begin{array}{l} \} \frac{n}{2} \\ \} \frac{n}{2} \end{array}; \quad L^* = \left(\begin{array}{cc} E & F \\ G & H \end{array} \right)$$

$\underbrace{\hspace{1.5cm}}_{\frac{n}{2}} \quad \underbrace{\hspace{1.5cm}}_{\frac{n}{2}}$

Es gilt also:

$$\begin{array}{ll} E = (A \vee BD^*C)^* & \text{betrachte alle Pfade von der ersten} \\ & \text{Hälfte der Knoten zur ersten Hälfte} \\ F = EBD^* & \text{analog} \\ G = D^*CE & \text{analog} \\ H = D^* \vee GF & \text{analog} \end{array}$$

Beweis (Forts.):

Um L^* zu berechnen, benötigen wir **zwei** Transitive-Hülle-Berechnungen und **sechs** Matrixprodukte für Matrizen der Dimension $\frac{n}{2} \times \frac{n}{2}$ (nämlich $M_1 = D^*C$, $M_2 = BM_1$, $M_3 = EB$, $M_4 = M_3D^*$, $M_5 = M_1E$, $M_6 = GF$), plus den Aufwand für \vee , der $\leq c'n^2$ ist. Wir zeigen nun durch Induktion ($n = 1\sqrt$), dass $T(n) \leq cM(n)$:

$$\begin{aligned} T(n) &\leq 2T\left(\frac{n}{2}\right) + 6M\left(\frac{n}{2}\right) + c'n^2 \\ &\leq 2cM\left(\frac{n}{2}\right) + 6M\left(\frac{n}{2}\right) + c'n^2 \quad \left| \text{Vor.: } M(2n) \geq 4M(n) \right. \\ &\quad \left| \text{da } M(n) \geq n^2 \right. \\ &\leq \frac{1}{4}(2c + 6 + 4c')M(n) \\ &\leq cM(n) \end{aligned}$$

falls $c \geq \frac{1}{4}(2c + 6 + 4c')$, also falls $c \geq 3 + 2c'$.

Also $T(n) = \mathcal{O}(M(n))$. □