

## 6.2 Boolesche Matrixmultiplikation und Transitiv 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:

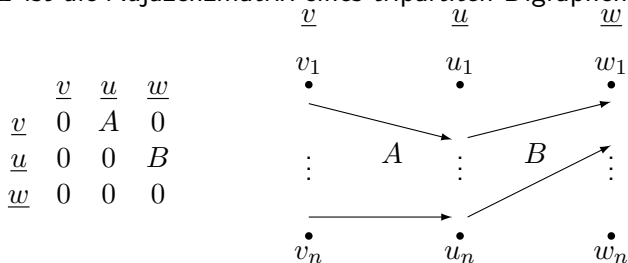
Seien boolesche Matrizen  $A$ ,  $B$  gegeben und ihr boolesches Produkt  $C = A \cdot B$  gesucht.

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  $T(n) \leq cM(n)$ :

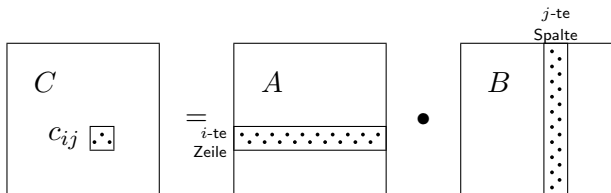
$$\begin{aligned} T(n) &\leq 2T\left(\frac{n}{2}\right) + 6M\left(\frac{n}{2}\right) + c'n^2 \\ &\leq 2cM\left(\frac{n}{2}\right) + 6M\left(\frac{n}{2}\right) + c'n^2 \quad \left| \text{Vor.: } M(2n) \geq 4M(n) \right. \\ &\quad \left| \text{da } M(n) \geq n^2 \right. \\ &\leq \frac{1}{4}(2c + 6 + 4c')M(n)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

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

Also  $T(n) = \mathcal{O}(M(n))$ . □

## 6.3 Der 4-Russen-Algorithmus für boolesche Matrixmultiplikation

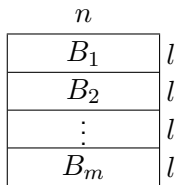
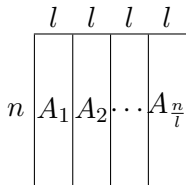
Gegeben zwei boolesche  $n \times n$  Matrizen  $A, B$ ; gesucht  $C = A \cdot B$ .



Sei  $l := \lfloor \log n \rfloor$ , o.B.d.A. gelte  $l|n$  ( $l$  teilt  $n$ ).



Teile  $A$  auf (setze  $m := \frac{n}{l}$ ):



Sei  $A = A'_1 \vee A'_2 \vee \dots \vee A'_m$ ,  $B = B'_1 \vee B'_2 \vee \dots \vee B'_m$ ,  
 $C_i := A'_i \cdot B'_i$  für  $i = 1, \dots, m$ . Dann gilt

$$C = \bigvee_{i=1}^m C_i, \text{ da}$$

$$C = AB = \left( \bigvee_{i=1}^m A'_i \right) \left( \bigvee_{i=1}^m B'_i \right) = \bigvee_{\substack{1 \leq i \leq m \\ 1 \leq j \leq m}} A'_i B'_j = \bigvee_{i=1}^m A_i B_i,$$

da  $A'_i B'_j = 0$  für  $i \neq j$  ( $A'_i$  und  $B'_j$  sind ja  $n \times n$  Matrizen mit 0 außerhalb des jeweiligen Streifens).

Gegeben die  $C_i$ 's, benötigen wir Zeit  $\mathcal{O}(mn^2)$ .

Betrachte eine Zeile von  $C_i$ :

$$\begin{array}{|c|} \hline C_i \\ \hline k\text{-te} \\ \text{Zeile} \\ \hline c_k^{(i)} \\ \hline \end{array} = k \begin{array}{|c|} \hline A_i \\ \hline 010110 \\ \hline \end{array} \cdot \begin{array}{|c|} \hline B_i \\ \hline 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ \hline b_j^{(i)} \\ \hline n \\ \end{array}$$

$$c_k^{(i)} = \bigvee_{j=1}^l a_k^{(i)} \cdot b_j^{(i)}$$

Der Algorithmus berechnet einfach zunächst alle booleschen Linearkombinationen der Zeilen von  $B_i$  (Prozedur bcomb) und damit  $c_k^{(i)}$  für alle überhaupt möglichen  $a_k^{(i)}$ .

Betrachte  $A$ ,  $B$  und  $C$  als Matrizen von Zeilenvektoren:

$$A = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}, \quad B = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}, \quad C = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$$

```

proc bcomb(int i) =
  comb[0] := [0, ..., 0]
  for j := 1 to 2[log n] - 1 do
    p := [log j]    co p Index der vordersten 1 von j oc
    comb[j] := comb[j - 2p] ∨ b(i-1)[log n]+1+p
  od

```

**Zeitbedarf:**

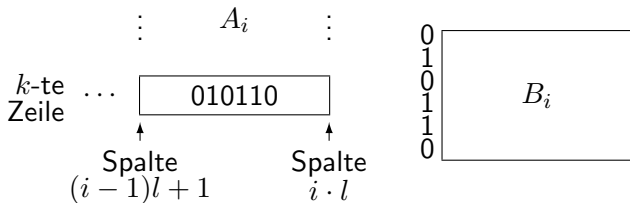
- (a) sequentiell:  $\mathcal{O}(n^2)$
- (b) Vektoroperationen der Breite  $n$ :  $\mathcal{O}(n)$

```

algorithm four-russians(array  $a, b, c$ ) =
  co  $a, b, c$  als Vektoren von  $n$  Zeilenvektoren organisiert oc
  const  $l = \lfloor \log n \rfloor$  co wir nehmen an  $l | n$  oc
  array  $\text{comb}[0..2^{l-1}]$  of boolean-vector; int  $nc$ 
  for  $i := 1$  to  $n$  do  $c[i] := [0, \dots, 0]$  od
  for  $i := 1$  to  $\frac{n}{l}$  do   co berechne die  $C_i$ 's oc
    bcomb( $i$ )
    for  $j := 1$  to  $n$  do
      co Bitmuster in Binärzahl wandeln oc
       $nc := 0$ 
      for  $k := i \cdot l$  downto  $(i - 1) \cdot l + 1$  do
         $nc := nc + nc +$  if  $a[j, k]$  then 1 else 0 fi
      od
       $c[j] := c[j] \vee \text{comb}[nc]$ 
    od
  od

```

## Beispiel 116



## Zeitbedarf:

(a) sequentiell:

$$\mathcal{O}\left(\frac{n}{l} \cdot (n^2 + n(l + n))\right) = \mathcal{O}\left(\frac{n^3}{l}\right) = \boxed{\mathcal{O}\left(\frac{n^3}{\log n}\right)}$$

(b) Vektorrechner der Breite  $n$  (Interpretation eines Bitintervalls als Zahl in  $\mathcal{O}(1)$  Zeit):

$$\mathcal{O}\left(\frac{n}{l} \cdot (n + n(1 + 1))\right) = \boxed{\mathcal{O}\left(\frac{n^2}{\log n}\right) \text{ (Vektoroperationen)}}$$

## Satz 117

Der 4-Russen-Algorithmus berechnet das Produkt zweier boolescher Matrizen sequentiell in Zeit  $\mathcal{O}\left(\frac{n^3}{\log n}\right)$  bzw. mit  $\mathcal{O}\left(\frac{n^2}{\log n}\right)$  Bitvektoroperationen der Breite  $n$ .

Beweis:

s.o.







V.L. Arlazarov, E.A. Dinic, M.A. Kronrod, I.A. Faradzev:  
*On economical construction of the transitive closure of an  
oriented graph*  
Soviet Math. Dokl. **11**, pp. 1209–1210 (1970)