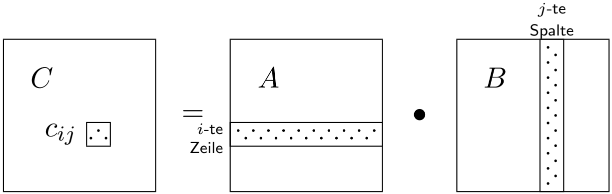
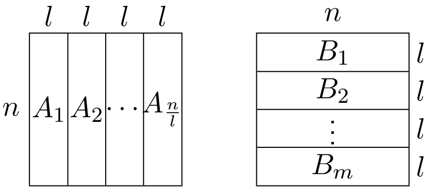
6.3 Der 4-Russen-Algorithmus für boolesche Matrixmultiplikation

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

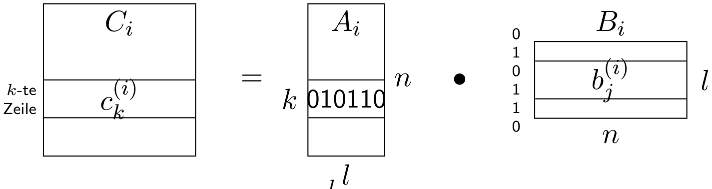
$$C = \bigvee_{i=1}^{m} C_i, \text{ da}$$

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

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:



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

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$[0] := [0, \ldots, 0]$

for $j := 1$ **to** $2^{\lfloor \log n \rfloor} - 1$ **do**

$p := \lfloor \log j \rfloor$ **co** p Index der vordersten 1 von j **oc**

comb$[j]$:=comb$[j - 2^p] \vee b_{(i-1)\lfloor \log n \rfloor + 1 + p}$

od

Zeitbedarf:

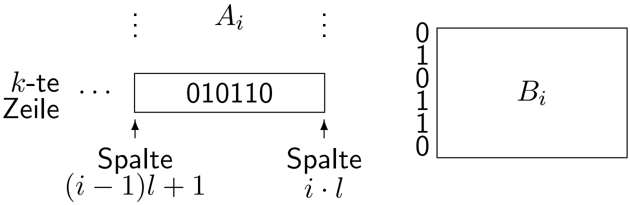
(a) sequentiell: $\boxed{\mathcal{O}(n^2)}$

(b) Vektoroperationen der Breite n: $\boxed{\mathcal{O}(n)}$

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 comb$[0..2^{l-1}]$ **of** boolean-vector; **int** nc
for $i := 1$ **to** n **do** $c[i] := [0, \ldots, 0]$ **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
        $c[j] := c[j] \vee$comb$[nc]$
    od
od

Beispiel 116



Zeitbedarf:

(a) sequentiell:

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

(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)