Mnożenie Macierzy

Klasyczny algorytm:

Matrix-Multiply(A,B)
if *columns*$[A] \neq$*rows*$[B]$ then
  error “niezgodne rozmiary”
else
  for $i \leftarrow 1$ to *rows*$[A]$ do
    for $j \leftarrow 1$ to *columns*$[B]$ do
      $C[i,j] \leftarrow 0$
      for $k \leftarrow 1$ to *columns*$[A]$ do
        $C[i,j] \leftarrow C[i,j] + A[i,k] \cdot B[k,j]$
  returnc C

Czas: $\Theta(p \cdot q \cdot r)$, gdzie $p \times q$, $q \times r$ – wymiary macierzy.

Mnożenie dwóch macierzy $n \times n$: $\Theta(n^3)$.

Dolne ograniczenie: $\Omega(n^2)$ (*wypisanie wyniku*).

Dziel i zwyciężaj

Liczymy $C = AB$, gdzie A i B macierze rozmiaru $n \times n$ (zakładamy, że $n = 2^k$). Każdą z A i B dzielimy na podmacierze wymiaru $n/2 \times n/2$.

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & g \\ f & h \end{pmatrix} \tag{1}$$

$$r = ae + bf \tag{2}$$

$$s = ag + bh \tag{3}$$

$$t = ce + df \tag{4}$$

$$u = cg + dh \tag{5}$$

8 mnożeń i 4 dodawania macierzy $n/2 \times n/2$.

Dziel i zwyciężaj

$$
\begin{aligned}
T(n) &= 8T(n/2) + cn^2 \\
&= 8(8T(n/2^2) + c(n/2)^2) + cn^2 \\
&= 8^0 cn^2 + 8^1 c(n/2)^2 + \ldots + 8^{\lg n} c \\
&= 8^{\lg n} c \sum_{i=0}^{\lg n} (2^i)^2/8^i \\
&= n^{\lg 8} c \sum_{i=0}^{\lg n} 2^{2i}/2^{3i} \\
&= n^{\lg 8} c \sum_{i=0}^{\lg n} 1/2^i \\
&= \Theta(n^3)
\end{aligned}
$$

Algorytm Strassena

Zamiast 8 mnożeń i 4 dodawań macierzy $n/2 \times n/2$ stosujemy 7 mnożeń i stałą liczbę dodawań. Wtedy:

$$
\begin{aligned}
T(n) &= 7T(n/2) + cn^2 \\
&= 7^{\lg n} c \sum_{i=0}^{\lg n} (2^{2-\lg 7})^i \\
& \quad \text{korzystamy z: } (2^{2-\lg 7}) < 1 \\
&= \Theta(n^{\lg 7}) \\
&= O(n^{2.81})
\end{aligned}
$$

Algorytm Strassena

1. Podziel A i B jak w równaniu (1)
2. Używając $\Theta(n^2)$ operacji dodawania i odejmowania wyznacz 14 macierzy $n/2 \times n/2$: A_1,$B_1 \ldots A_7$,B_7.
3. rekurencyjnie wylicz 7 iloczynów $P_i = A_i B_i$
4. oblicz podmacierze r, s, t, u używając $\Theta(n^2)$ operacji dodawania i odejmowania.

Dalej uzupełniamy szczegóły.
Przyjmujemy, że P_i może być zapisany jako:

$$\begin{aligned} P_i &= A_i B_i \\ &= (\alpha_{i1} a + \alpha_{i2} b + \alpha_{i3} c + \alpha_{i4} d) \cdot (\beta_{i1} e + \beta_{i2} f + \beta_{i3} g + \beta_{i4} h) \end{aligned}$$

gdzie $\alpha_{i,j}, \beta_{i,j} \in \{-1, 0, 1\}$.

Algorytm Strassena

$$
\begin{aligned}
r &= ae + bf \\
&= \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} +1 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}
\end{aligned}
$$

$$
= \begin{array}{cc} & \begin{array}{cccc} e & f & g & h \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{pmatrix} + & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \end{array}
$$

Algorytm Strassena

$$s = ag + bh = \begin{pmatrix} \cdot & \cdot & + & \cdot \\ \cdot & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

$$t = ce + df = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & \cdot \end{pmatrix}$$

Algorytm Strassena

$$u = cg + dh = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & + & \cdot \\ \cdot & \cdot & \cdot & + \end{pmatrix}$$

P_1, P_2

$$P_1 = A_1 B_1 = a(g-h) = ag - ah = \begin{pmatrix} . & . & + & - \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix}$$

$$P_2 = A_2 B_2 = (a+b)h = ah + bh = \begin{pmatrix} . & . & . & + \\ . & . & . & + \\ . & . & . & . \\ . & . & . & . \end{pmatrix}$$

Wtedy $s = P_1 + P_2$.

P_3, P_4

$$P_3 = A_3 B_3 = (c+d)e = ce + de = \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ + & . & . & . \\ + & . & . & . \end{pmatrix}$$

$$P_4 = A_4 B_4 = d(f-e) = df - de = \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ - & + & . & . \end{pmatrix}$$

Wtedy $t = P_3 + P_4$.

P_5, P_6

$$P_5 = A_5 B_5 = (a+d)(e+h) = ae+ah+de+dh = \begin{pmatrix} + & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & + \end{pmatrix}$$

$$P_6 = A_6 B_6 = (b-d)(f+h) = bf+bh-df-dh = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & - & \cdot & - \end{pmatrix}$$

$$r = P_5 + P_4 - P_2 + P_6$$

$$\begin{aligned}
r &= P_5 + P_4 - P_2 + P_6 \\
&= \begin{pmatrix} + & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & + \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ - & + & \cdot & \cdot \end{pmatrix} - \begin{pmatrix} \cdot & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} + P_6 \\
&= \begin{pmatrix} + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & - \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & + \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & - & \cdot & - \end{pmatrix} = \begin{pmatrix} + & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}
\end{aligned}$$

P_7

$$P_7 = A_7 B_7 = (a - c)(e + g) = ae + a\,g - ce - cg = \begin{pmatrix} + & . & + & . \\ . & . & . & . \\ - & . & - & . \\ . & . & . & . \end{pmatrix}$$

$u = P_5 + P_1 - P_3 - P_7$

$$
\begin{aligned}
u &= P_5 + P_1 - P_3 - P_7 \\
&= \begin{pmatrix} + & . & . & + \\ . & . & . & . \\ . & . & . & . \\ + & . & . & + \end{pmatrix} + \begin{pmatrix} . & . & + & - \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix} - \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ + & . & . & . \\ + & . & . & . \end{pmatrix} - P_7 \\
&= \begin{pmatrix} + & . & + & . \\ . & . & . & . \\ - & . & . & . \\ . & . & . & + \end{pmatrix} - \begin{pmatrix} + & . & + & . \\ . & . & . & . \\ - & . & - & . \\ . & . & . & . \end{pmatrix} = \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & + & . \\ . & . & . & + \end{pmatrix}
\end{aligned}
$$

Podsumowanie

$$
\begin{aligned}
P_1 &= a(g-h) \\
P_2 &= (a+b)h \\
P_3 &= (c+d)e \\
P_4 &= d(f-e) \\
P_5 &= (a+d)(e+h) \\
P_6 &= (b-d)(f+h) \\
P_7 &= (a-c)(e+g) \\
r &= P_5+P_4-P_2+P_6 \\
s &= P_1+P_2 \\
t &= P_3+P_4 \\
u &= P_5+P_1-P_3-P_7
\end{aligned}
$$

monoid

$(S, \oplus, \bar{0})$ jest *monoidem*, jeśli:

1. S jest *zamknięty* ze względu na $\oplus$: $\forall_{a,b \in S}\ a \oplus b \in S$.
2. $\oplus$ jest *łączny*: $\forall_{a,b,c \in S}\ a \oplus (b \oplus c) = (a \oplus b) \oplus c$.
3. $\bar{0}$ jest elementem *neutralnym*: $\forall_{a \in S} a \oplus \bar{0} = \bar{0} \oplus a = a$.

Quasi-pierścienie

Niech $(S, \oplus, \odot, \bar{0}, \bar{1})$ – struktura algebraiczna, gdzie $\oplus$, $\odot$ – binarne operacje na zbiorze S, a $\bar{0}$, $\bar{1}$ – wyróżnione elementy ($\bar{0} \neq \bar{1}$). Taką strukturę nazywamy *quasi-pierścieniem*) jeśli:

1. $(S, \oplus, \bar{0})$ jest *monoidem*.
2. $(S, \odot, \bar{1})$ jest *monoidem*.
3. $\bar{0}$ jest *anihilatorem*: $\forall_{a \in S}\ a \odot \bar{0} = \bar{0} \odot a = \bar{0}$.
4. $\oplus$ jest *przemienny*: $\forall_{a,b \in S}\ a \oplus b = b \oplus a$.
5. $\odot$ jest *rozdzielny* względem $\oplus$: $\forall_{a,b,c \in S}\ a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c) \wedge (b \oplus c) \odot a = (b \odot a) \oplus (c \odot a)$.

Quasi-pierścienie

Przykłady:

- *quasi-pierścień boolowski*: $(\{0,1\}, \vee, \wedge, 0, 1)$.
- liczby naturalne z dodawaniem i mnożeniem: $(N, +, \cdot, 0, 1)$

Zauważmy, że jeśli elementy macierzy są z quasi-pierścienia, to mnożenie macierzy jest dobrze zdefiniowane. Mnożenie klasyczne działa poprawnie, natomiast brakuje operacji przeciwnej do $\oplus$ aby zastosować algorytm Strassena.

Ponadto:

Tw. Jeśli $(S, \oplus, \odot, \bar{0}, \bar{1})$ jest quasi-pierścieniem, to $(S^{n\times n}, \oplus, \odot, \bar{0}, \bar{I}_n)$ też jest quasi-pierścieniem.

D-d Ćwiczenie $\Box$.

Pierścienie

$(S, \oplus, \odot, \bar{0}, \bar{1})$ jest *pierścieniem* jeśli jest quasi-pierścieniem oraz:

- Każdy element w S ma *element odwrotny względem dodawnia* w S: $\forall_{a \in S} \exists_{b \in S}\ a \oplus b = b \oplus a = \bar{0}$ (oznaczenie elementu odwrotnego do a: $-a$)

W pierścieniu można zdefiniować odejmowanie:
$a - b = a + (-b)$.
Przykłady pierścieni:

- Liczby całkowite $(Z, +, \cdot, 0, 1)$
- Liczby całkowite modulo n, dla całkowitego $n > 1$ $(Z_n, +, \cdot, 0, 1)$
- Zbiór $R[x]$ wielomianów skończonego stopnia zmiennej x z rzeczywistymi współczynnikami $(R[x], +, \cdot, 0, 1)$, gdzie $+$ i $\cdot$ – dodawanie i mnożenie wielomianów.

Pierścienie

Wn. Jeśli $(S, \oplus, \odot, \bar{0}, \bar{1})$ jest pierścieniem i $n \geq 1$, to $(S^{n\times n}, \oplus, \odot, \bar{0}, \bar{1})$ też jest pierścieniem.

D-d. Ćwiczenie $\Box$

Tw. Algorytm Strassena mnożenia macierzy działa prawidłowo, dla macierzy utworzonych nad pierścieniem.

D-d. Algorytm Strassena zależy od poprawności algorytmu dla macierzy 2×2, który wymaga tylko aby elementy należały do pierścienia. Elementy macierzy należą do pierścienia. Z Wniosku – podmacierze na każdym poziomie rekursji też są elementami pierścienia. $\Box$

Mnożenie macierzy boolowskich

Tw. Niech $M(n)$ – liczba operacji artymetycznych potrzebna do pomnożenia dwóch macierzy $n \times n$ liczb całkowitych. Wtedy dwie macierze boolowskie $n \times n$ można pomnożyć przy pomocy $O(M(n))$ operacji arytmetycznych.

D-d. Niech A, B, $C = AB$ – macierze boolowskie $n \times n$, czyli: $c_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$. Zamiast przeprowadzać operacje w boolowskim quasi-pierścieniu obliczamy C' w pierścieniu liczb całkowitych, t.j.: $c'_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$.

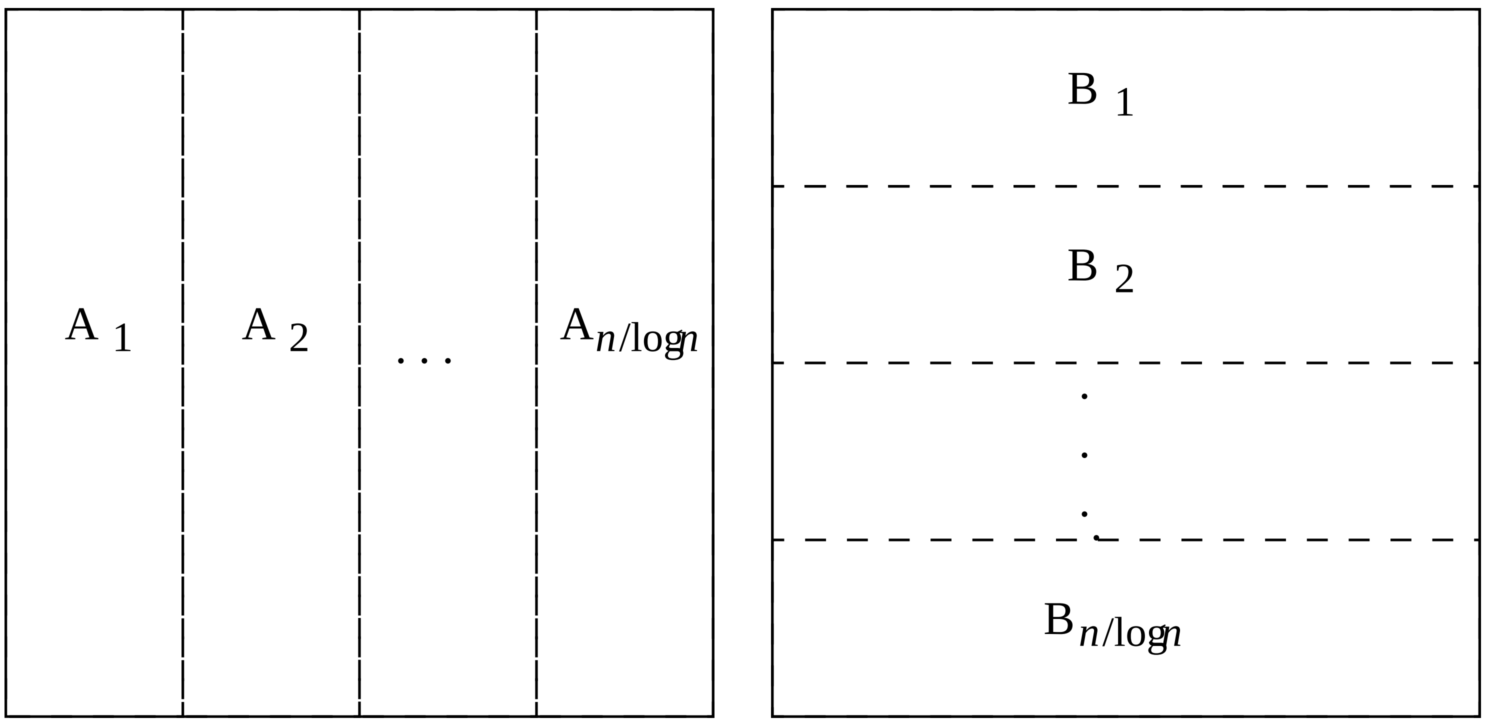
$a_{ik} b_{kj} = 0 \equiv a_{ik} \wedge b_{kj} = 0$ oraz $a_{ik} b_{kj} = 1 \equiv a_{ik} \wedge b_{kj} = 1$. Stąd $c_{ij} = 0 \equiv c'_{ij} = 0$. Macierz C może więc być odtworzona przy pomocy $\Theta(n^2)$ operacji arytmetycznych z C'. Ponieważ $M(n) = \Omega(n^2)$ całkowita liczba operacji jest $M(n) + \Theta(n^2) = O(M(n))$. $\Box$

Mnożenie macierzy boolowskich

Wniosek: Można stosować algorytm Strassena do mnożenia macierzy boolowskich w czasie $O(n^{\lg 7})$.
Uwaga: Obliczenia można przeprowadzać w pierścieniu liczb całkowitych modulo $n+1$ (ćw.)

Algorytm czterech Rosjan

Chcemy pomnożyć dwie macierze logiczne A i B rozmiaru $n \times n$. Dzielimy A i B na odpowiednio pionowe i poziome paski szerokości $\lg n$. (zakładamy, że $n = 2^m$ i $m|n$).



Wtedy $AB = \sum_{i=1}^{n/\lg n} A_i B_i$. (zauważmy, że każde $A_i B_i$ jest macierzą $n \times n$.)

Algorytm czterech Rosjan

Aby policzyć $C_i = A_i B_i$ liczymy $a_j B_i$ dla każdego wiersza a_j macierzy A_i, następująco: znajdujemy wszyskie wiersze b_k macierzy B_i takie, że k-ty element a_j jest $= 1$ i dodajemy je jako wektory n-bitowe. (*Wymaga to czasu $O(n \lg n)$, bo długość a_j jest $\lg n$. Wynik jest j-tym wierszem C_i. Zatem (naiwne) policzenie C_i wymaga $O(n^2 \lg n)$.*) Zauważmy, że jest $2^{\lg n} = n$ możliwych różnych wierszy a_j. Liczymy i zapisujemy w tablicy wszystkie możliwe sumy wierszy B_i i zamiast obliczać $a_j B_i$ pobieramy wynik z miejsca wskazanego przez a_j.

Wszystkie sumy wierszy B_i liczymy w czasie $O(n^2)$: Każdy z n podzbiorów wierszy B_i jest albo $= \emptyset$, albo 1-elementowy, albo jest sumą zbioru o jeden mniejszego i 1-elementowego.Stosujemy kolejność niemalejącą ze wzgl. na liczność zbioru: wynik dla danego zbioru wymaga dodania 1 wiersza do wcześniejszego wyniku.

Algorytm czterech Rosjan

$a_j^{(i)}$ $b_j^{(i)}$ $c_j^{(i)}$ oznaczają odpowiednio j-ty wiersz A_i, B_i, C_i.

R – n-elementowa tablica wektorów n-bitowych.

$([x_1 \ldots x_m])_2$ – liczba o binarnej reprezentacji $x_m \ldots x_1$.

$m \leftarrow \lg n$
for $i \leftarrow 1$ **to** n/m **do** $\triangleright$ $n/\lg n$ razy
    $R[0] \leftarrow [0, \ldots, 0]$ $\triangleright$ wektor n-bitowy
    for $j \leftarrow 1$ **to** $2^m - 1$ **do** $\triangleright$ $n-1$ razy
        $k \leftarrow \lfloor \lg j \rfloor$
        $R[j] \leftarrow R[j - 2^k] \vee b_{k+1}^{(i)}$ $\triangleright$ suma wektorów: $O(n)$
    for $j \leftarrow 1$ **to** n **do** $\triangleright$ n razy
        $c_j^{(i)} \leftarrow R\left[\left(a_j^{(i)}\right)_2\right]$ $\triangleright$ $O(\lg n + n) = O(n)$
$C \leftarrow \sum_{i=1}^{n/m} C_i$ $\triangleright$ $O(n^3/\lg n)$

Czas: $O(n^3/\lg n)$.