Programowanie Współbieżne

Algorytmy

Sortowanie przez scalanie (**mergesort**)

Algorytm :

1. JEŚLI jesteś rootem TO: pobierz/wczytaj tablice do posortowania

   JEŚLI_NIE to pobierz tablicę do posortowania od rodzica

Sortowanie przez scalanie (**mergesort**)

2. JEŚLI ilość elementów tablicy > 2 i (ilość procesów < max procesów) [tu można dodać czy tablica nie jest już posortowana lub inny warunek zatrzymania dzielenia] TO

   - twórz 2 procesy i wyślij im po jednej części tablicy (najlepiej w miarę równe).
   - czekaj na posortowane 2 tablice
   - scal w jedną posortowaną

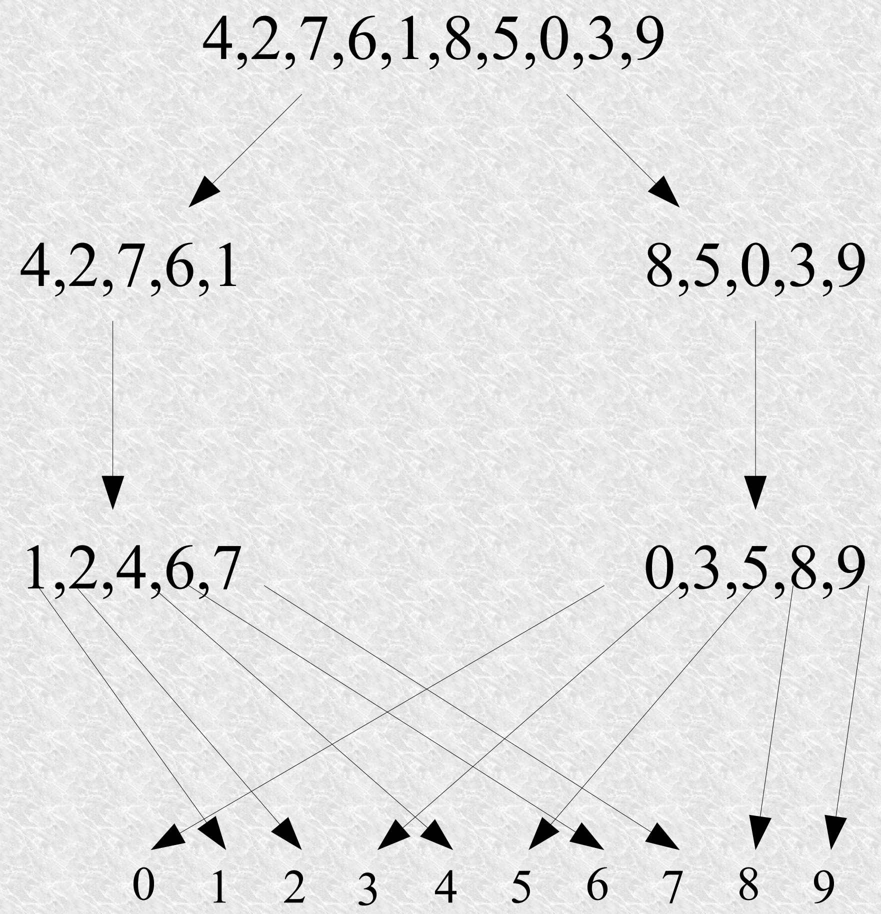
   JEŚLI_NIE to posortuj to co masz. (w szczególnym przypadku będzie to 1 liczba do oddania lub dwie do porównania).

Sortowanie przez scalanie (**mergesort**)

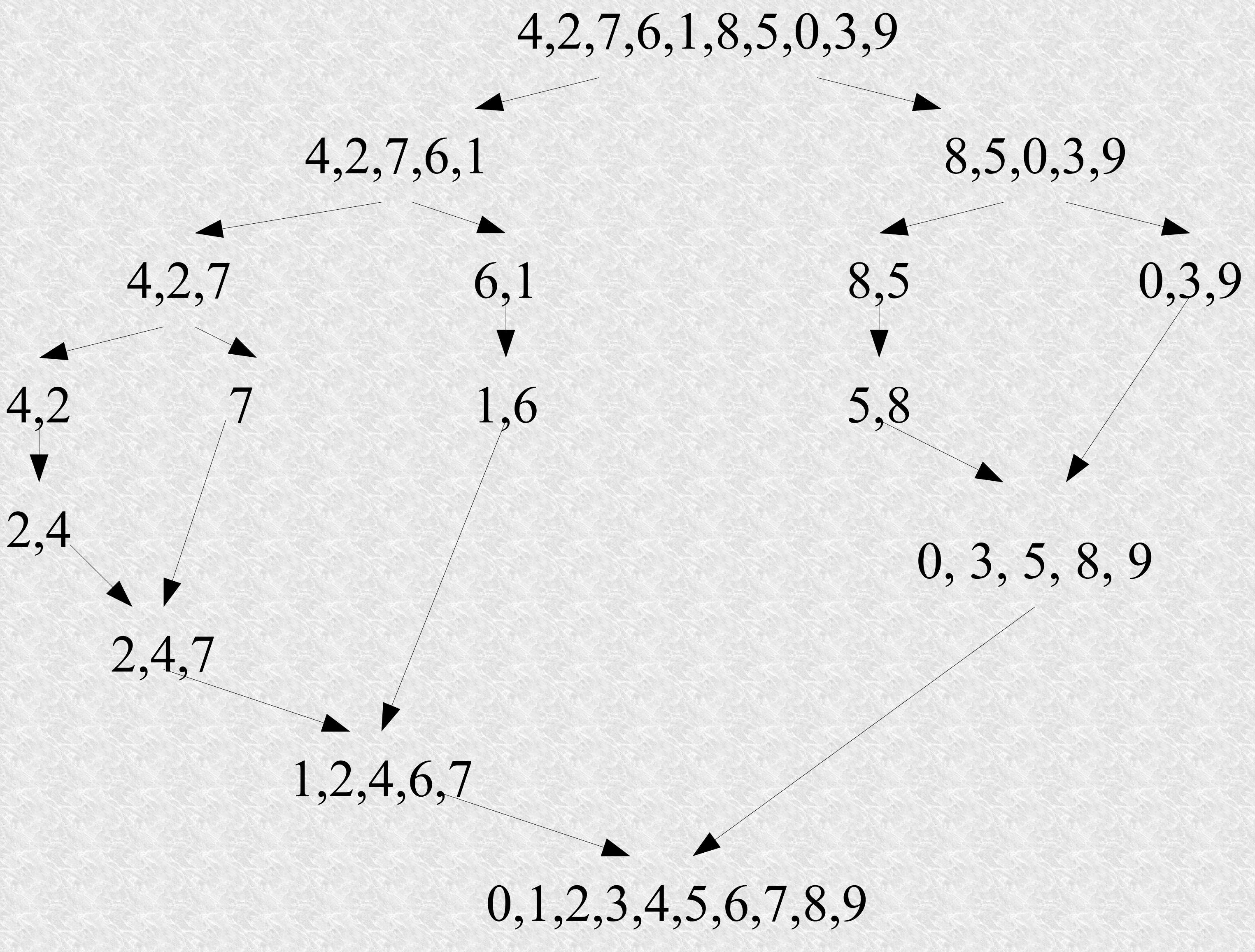
3. JEŚLI jesteś rootem TO wyświetl/zapisz wynik

   JEŚLI_NIE to odeślij rodzicowi posortowaną tablicę.

Sortowanie przez scalanie



Sortowanie przez scalanie

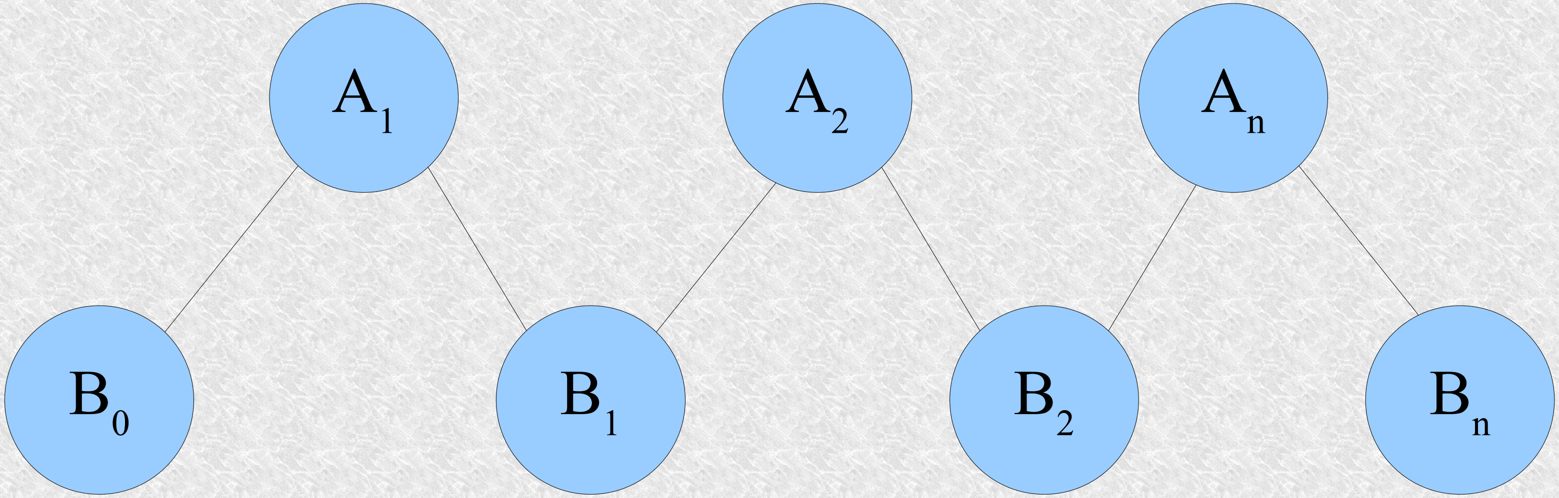


Sortowanie oscylacyjne

Dla ciągu długości **n** tworzymy dwa zestawy procesów.

$\mathbf{A_1, A_2, \ldots A_n}$

$\mathbf{B_0, B_1, \ldots B_n}$



Sortowanie oscylacyjne

Zadania typu A_i działają następująco:

- otrzymują 2 liczby
- mniejszą przesyłają do B_{i-1}
- większą do B_i.

Sortowanie oscylacyjne

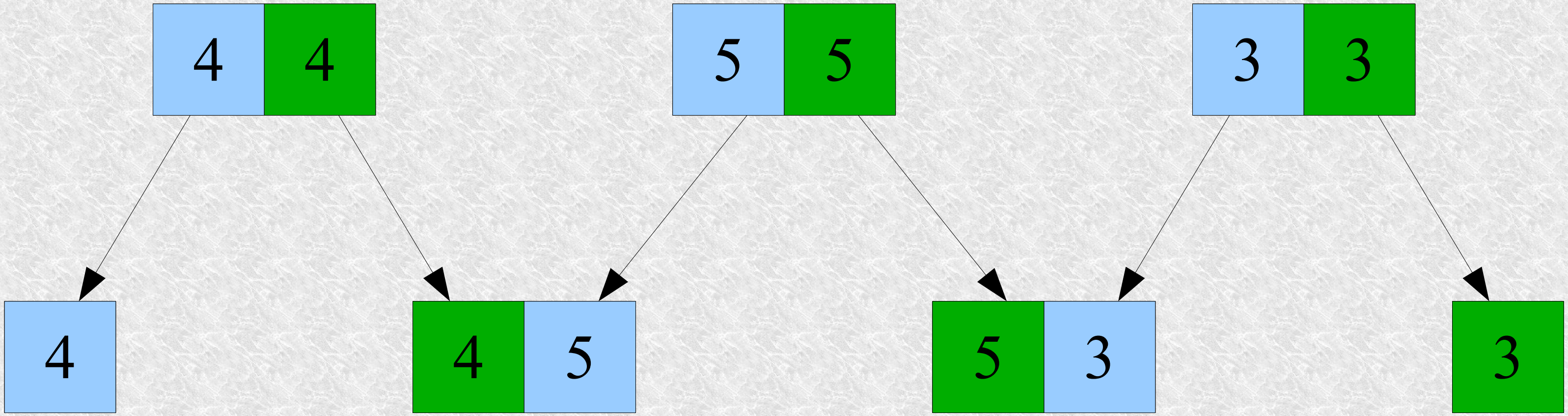
Zadania typu B_i działają następująco:

- otrzymują 2 liczby
- mniejszą przesyłają do A_i
- większa do A_{i+1}

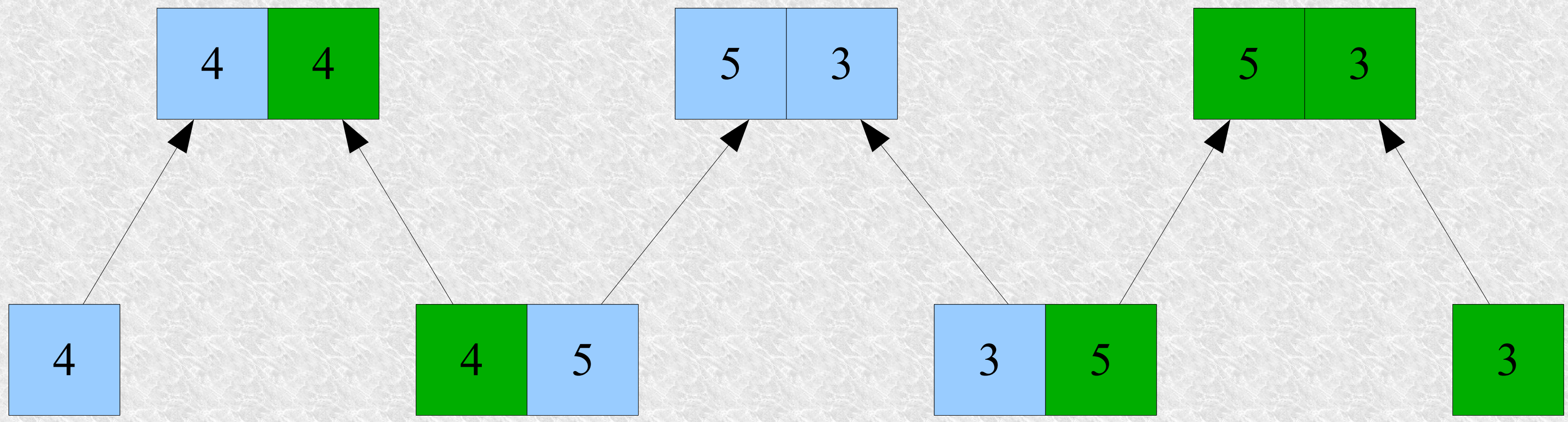
Elementy skrajne nic nie robią tylko oddają liczbę

Po **2n** cyklach mamy gotowy wynik

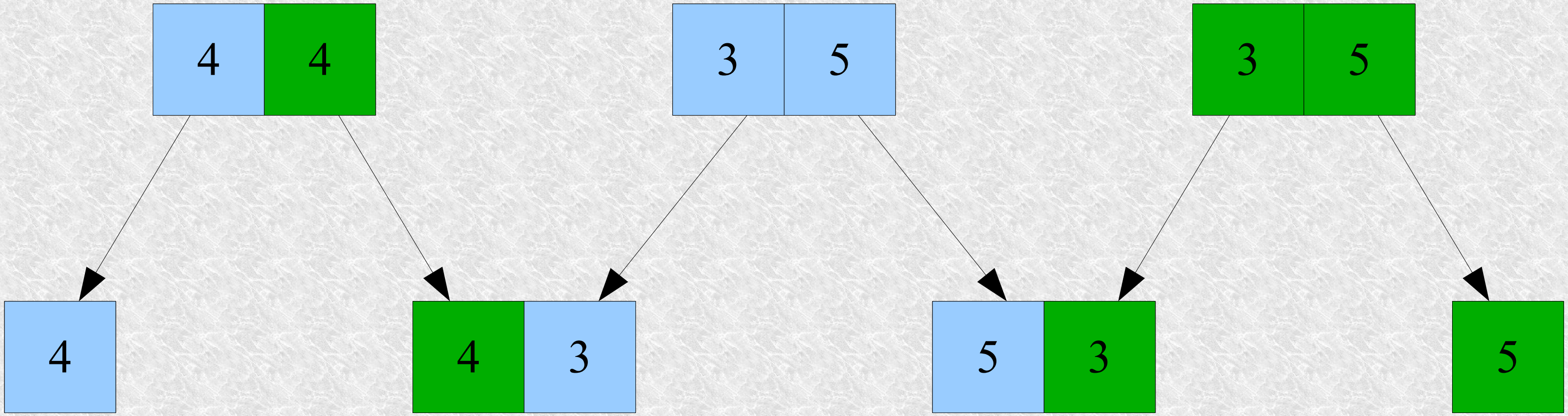
Sortowanie oscylacyjne



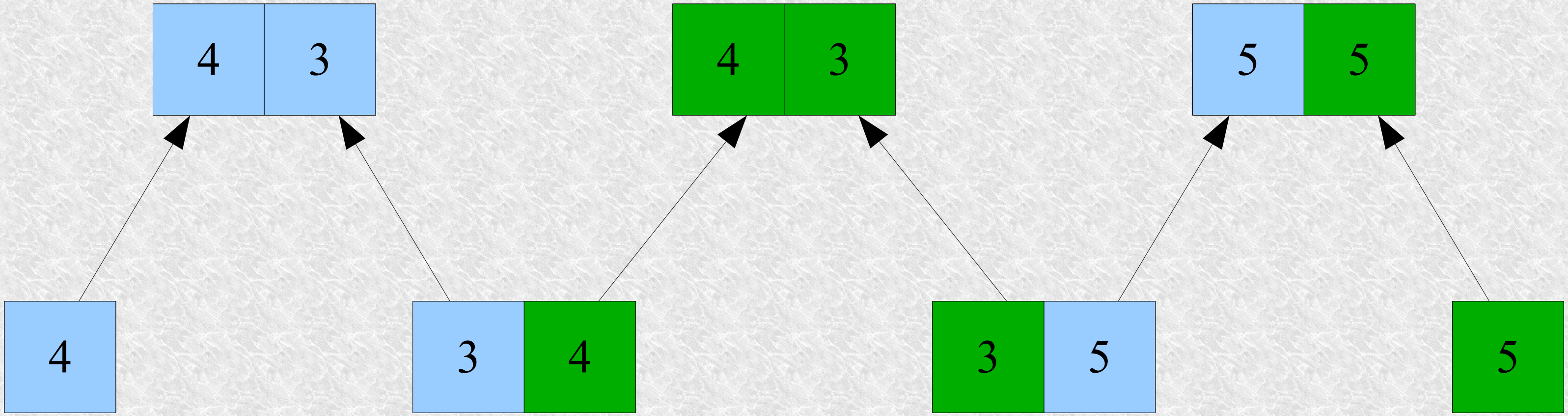
Sortowanie oscylacyjne



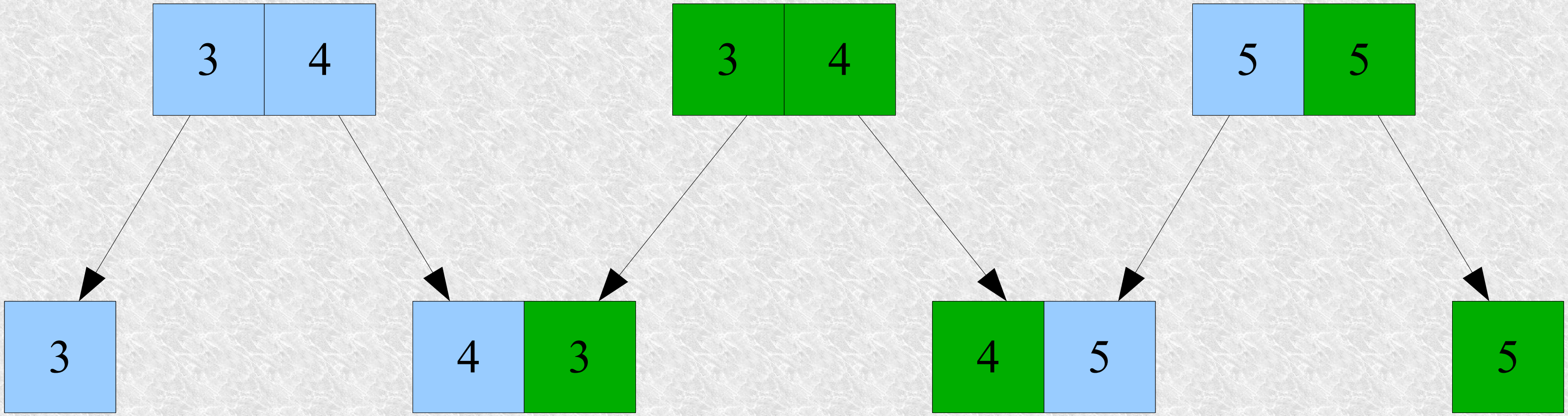
Sortowanie oscylacyjne



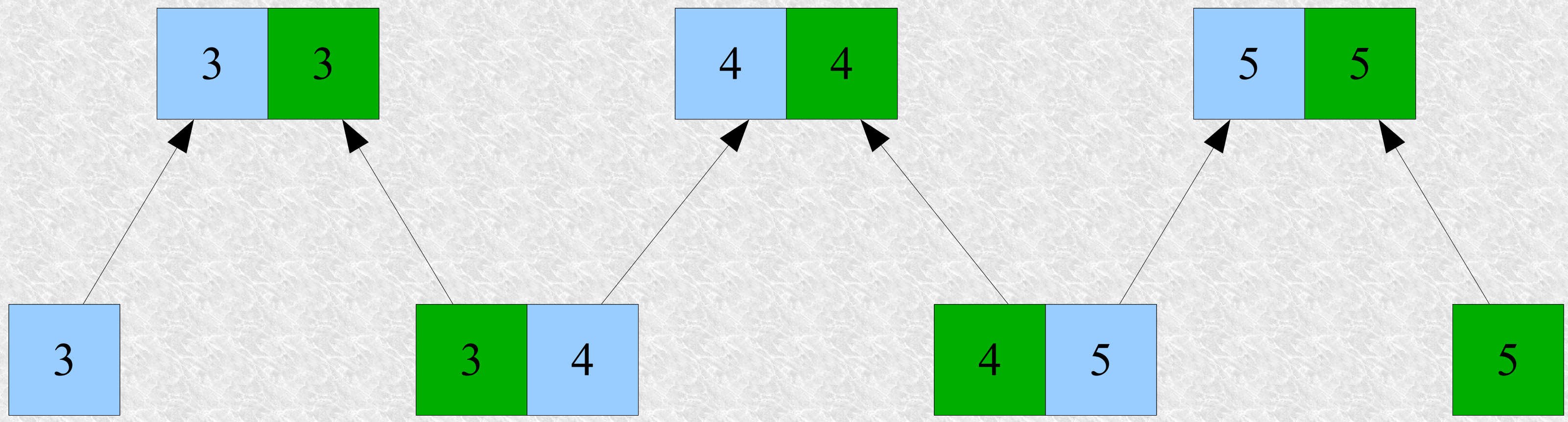
Sortowanie oscylacyjne



Sortowanie oscylacyjne



Sortowanie oscylacyjne



Mnożenie macierzy

Zastosowanie:

- w matematyce, rozwiązywanie układów równań liniowych Metodą Gaussa
- Zapisanie obiektów geometrycznych w przestrzeni liniowej w fizyce tzw. Tensory
- Grafika trójwymiarowa, transformacje

Mnożenie macierzy

Definicja

Iloczynem macierzy **A**=$[a_{ij}]_{nxp}$ przez macierz **B**=$[b_{ij}]_{pxm}$ nazywamy taką macierz **C**=$[c_{ij}]_{nxm}$ piszemy **C=A•B**, że

$$c_{ij} = \sum_{k=1}^{p} a_{ik} \bullet b_{kj}$$ dla i=1,2,...,n;j=1,2,...,m

Mnożenie macierzy

$A \in M_{m\times k}$ $\quad$ $B \in M_{k\times n}$ $\quad$ $C \in M_{m\times n}$

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ & (\dots) & & \\ a_{m1} & a_{m2} & \dots & a_{mk} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ & (\dots) & & \\ b_{k1} & b_{k2} & \dots & b_{kn} \end{pmatrix} = \begin{pmatrix} \boxed{c_{11}} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ & (\dots) & & \\ c_{m1} & c_{m2} & \dots & c_{mn} \end{pmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ (\dots) \\ b_{k1} \end{bmatrix} = \boxed{c_{11}}$$

Mnożenie macierzy

Kilka przydatnych właściwości:

Jeżeli $\mathbf{A}$, $\mathbf{B}$ *oraz* $\mathbf{C}$ *są macierzami o odpowiednich wymiarach to:*

1. $\mathbf{A(BC)=(AB)C}$
2. $\mathbf{(AB)=(A)B}$
3. $\mathbf{(A+B)C=AC+BC}$
4. $\mathbf{C(A+B)=CA+CB}$
5. $\mathbf{IA=A}$, gdy $\mathbf{A}_{nxn}$ i $\mathbf{I}_{nxn}$

Mnożenie macierzy

Algorytm „dziel i rządź”

Uzyskanie macierzy **C** jest wynikiem niezależnych operacji arytmetycznych na wierszach macierzy **A** i kolumnach **B**. Stąd intuicyjny sposób podziału zadania na wiele wątków, tak by każdy obliczył niezależnie element macierzy C. Takich podziałów w tym wypadku musi być $m*n$ (liczba wątków). Koszt operacji wynosi $\boldsymbol{O(n^3)}$.

Mnożenie macierzy

Każdy element A_{ij} B_{jk} C_{ik} to mała podmacierz na której wykonujemy takie same operacje jak na pojedynczych elementach.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \quad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

Mnożenie macierzy

Przykład:

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 4 & 1 & 2 \\ 0 & 2 & 2 & 1 \\ 1 & 2 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 2 & 0 \\ 2 & 3 & 1 & 2 \\ 1 & 1 & 2 & 3 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 8 & 5 & 9 \\ 9 & 16 & 8 & 13 \\ 6 & 9 & 6 & 11 \\ 5 & 10 & 6 & 9 \end{bmatrix}$$

Takie mnożenie można rozbić na 4 działania analogiczne do poniższego

$$\begin{bmatrix} 0 & 1 \\ 1 & 4 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 8 & 13 \end{bmatrix} + \begin{bmatrix} 2 & 5 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 4 & 8 \\ 9 & 16 \end{bmatrix}$$

Mnożenie macierzy

Metoda **„Strassena”**

- Z tak podzielonej macierzy wylicz 7 pomocniczych macierzy **m_i** o rozmiarze **n/2**

$m_1=(A_{12}-A_{22})*(B_{21}+B_{22})$
$m_2=(A_{11}+A_{22})*(B_{11}+B_{22})$
$m_3=(A_{11}-A_{21})*(B_{11}+B_{12})$
$m_4=(A_{11}+A_{12})*B_{22}$
$m_5=A_{11}*(B_{12}-B_{22})$
$m_6=A_{22}*(B_{21}-B_{11})$
$m_7=(A_{21}+A_{22})*B_{11}$

Mnożenie macierzy

Oblicz składowe Cij macierzy wynikowej C

$C_{11}=m_1+m_2-m_4+m_6$
$C_{12}=m_4+m_5$
$C_{21}=m_6+m_7$
$C_{22}=m_2-m_3+m_5-m_7$

Koszt powyższego algorytmu szacuje się na $\boldsymbol{O(n^{\log_2 7})}$

Mnożenie macierzy

Algorytm **„canona"**

- Zakładamy że mamy sieć zadań m x m.
- Każdy proces (t_{ij} gdzie $0<i,j<m$) wewnątrz zawiera bloki C_{ij}, A_{ij} i B_{ij}.
- Na wstępie algorytmu proces na przekątnej diagonalnej (t_{ij} gdzie i=j) przesyła swój blok A_{ij} do wszystkich innych procesów w rzędzie i.
- Po transmisji A_{ii}, wszystkie zadania obliczają $A_{ii}xB_{ij}$ i dodają wynik do C_{ij}.
- W kolejnym kroku kolumna bloków macierzy B jest obracana. Tzn. t_{ij} przesyła swój blok $t_{(i-1)j}$. Proces t_{0j} przesyła swój blok B do $t_{(m-1)j}$.

Mnożenie macierzy

- Teraz procesy powracają do kroku pierwszego.
- $A_{i(i+1)}$ jest podstawową informacją dla wszystkich innych procesów w rzędzie i.

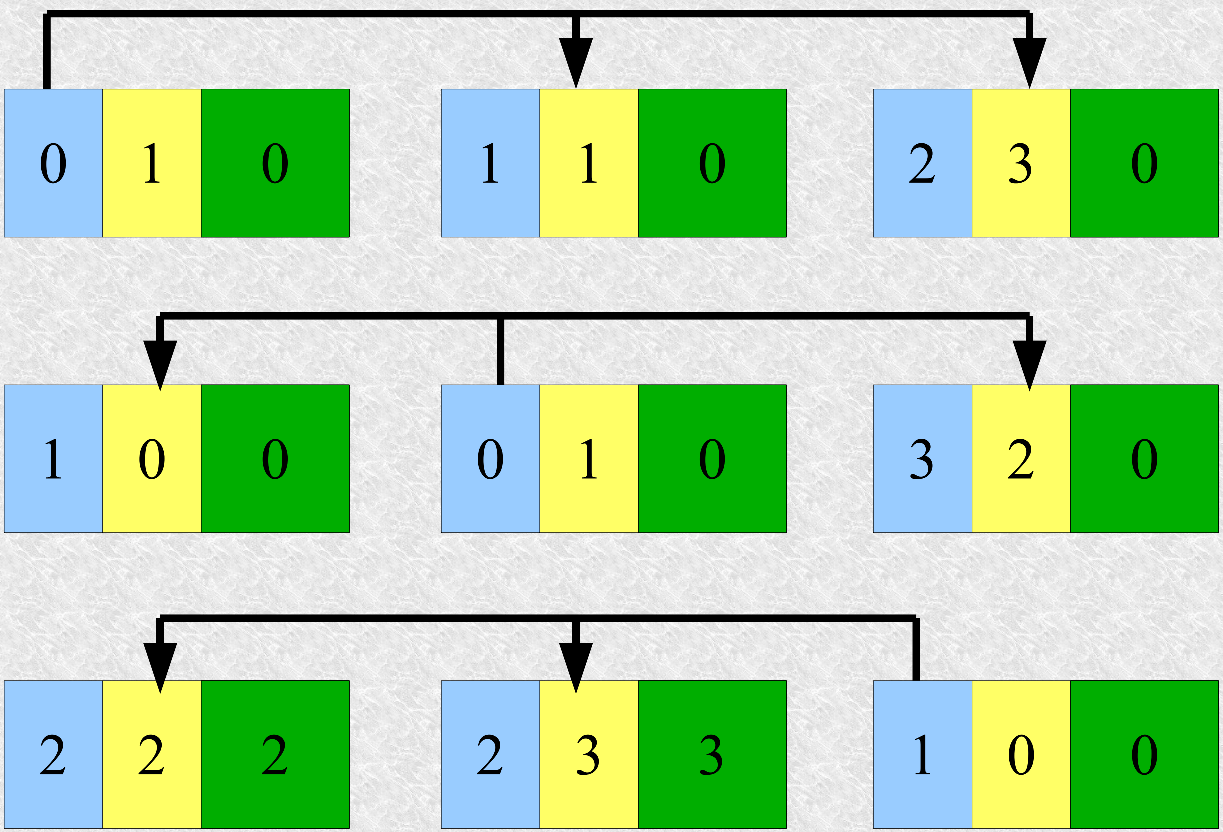
- Algorytm jest dalej kontynuowany. Po m iteracjach macierz C zawiera wynik mnożenia AxB, a obracana macierz B przyjmuje swoją początkową postać.

Mnożenie macierzy

$$\overset{A}{\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 2 & 1 \end{bmatrix}} \cdot \overset{B}{\begin{bmatrix} 1 & 1 & 3 \\ 0 & 1 & 2 \\ 2 & 3 & 0 \end{bmatrix}} = \overset{C}{\begin{bmatrix} 4 & 7 & 2 \\ 7 & 10 & 3 \\ 4 & 7 & 10 \end{bmatrix}}$$

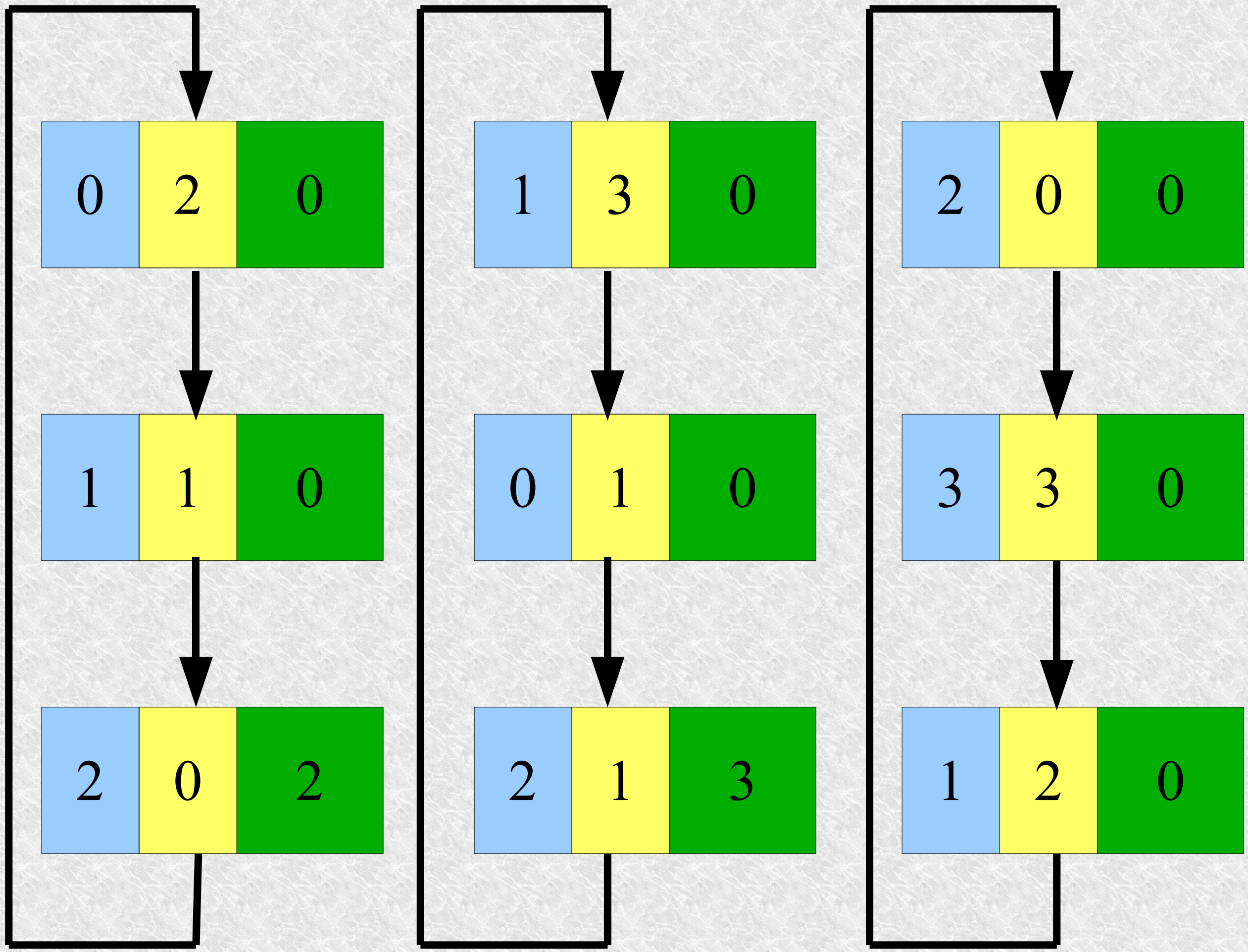
0	1	0	1	1	0	2	3	0
1	0	0	0	1	0	3	2	0
2	2	0	2	3	0	1	0	0

Mnożenie macierzy



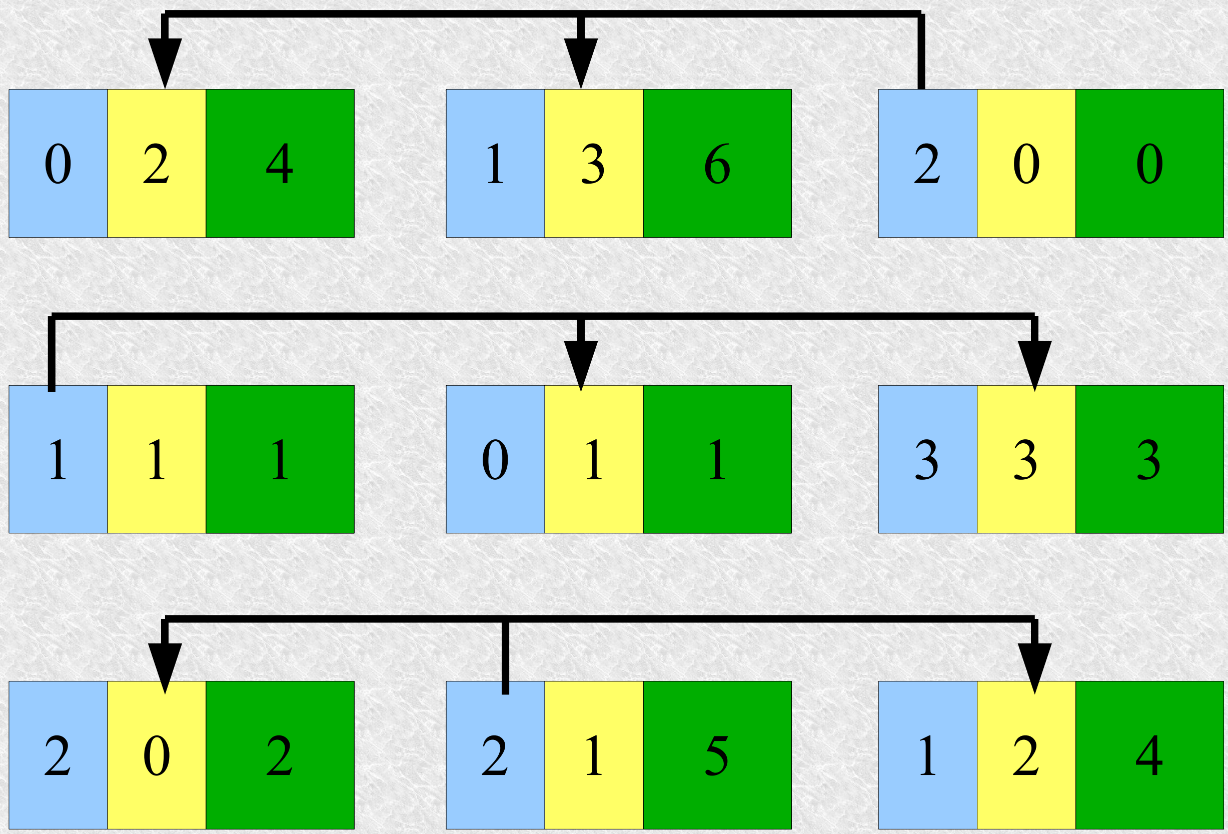
Z macierzy A bierzemy wartości leżące na diagonalnej
Przysyłamy do sąsiadów w tym samym wierszu.
Mnożymy wartości przesyłane z wartościami macierzy B i dodajemy do wyniku w C

Mnożenie macierzy



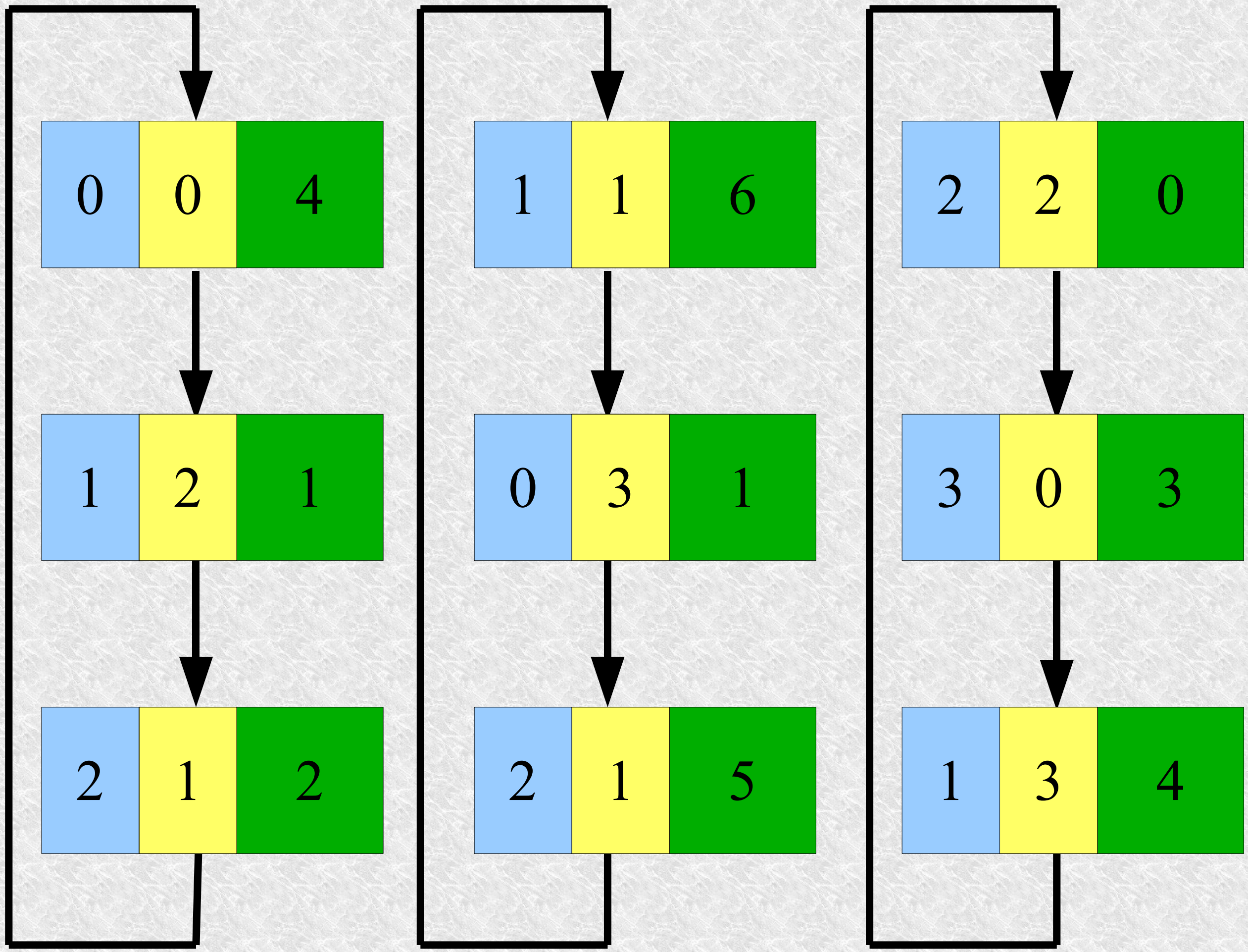
Kolumny macierzy B „rolujemy” w dół.

Mnożenie macierzy



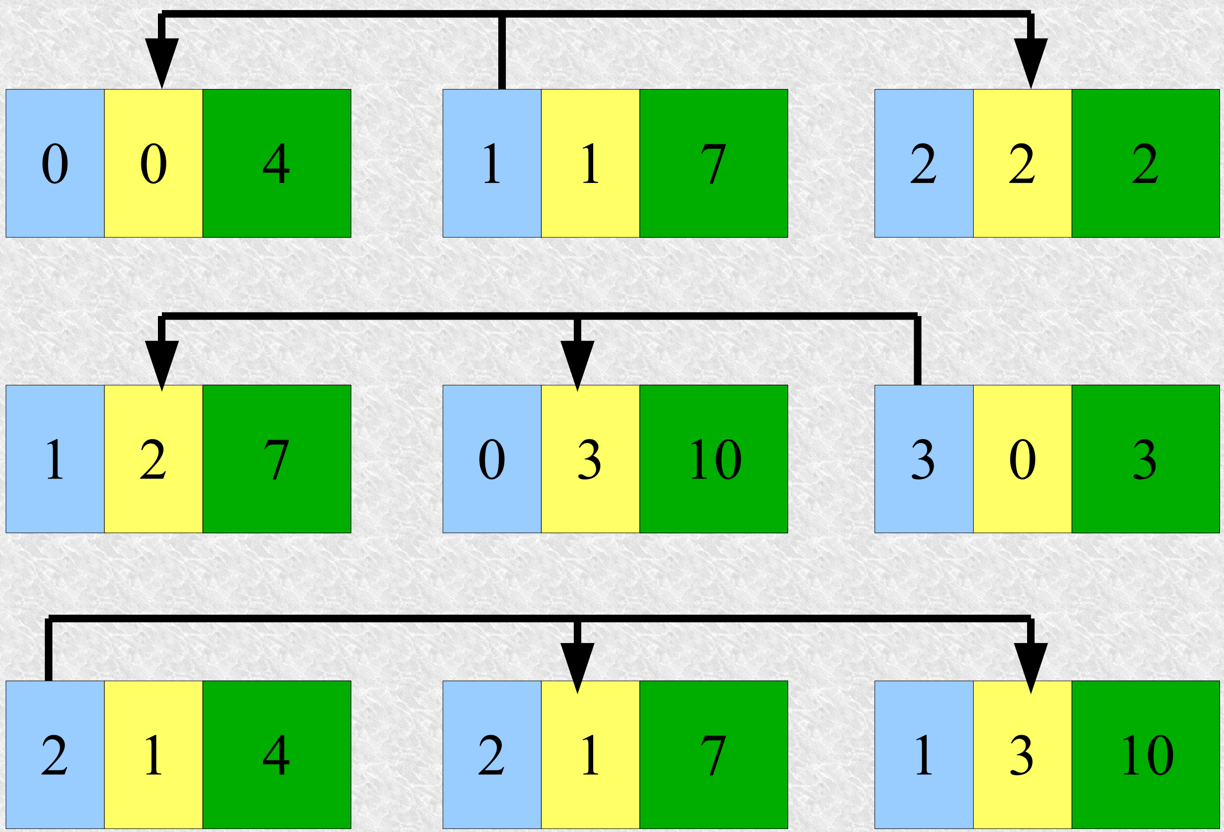
Obniżamy diagonalną o jeden w dół.
Przysyłamy do sąsiadów w tym samym wierszu.
Mnożymy wartości przesyłane z wartościami macierzy B i dodajemy do wyniku w C

Mnożenie macierzy



Kolumny macierzy B „rolujemy” w dół.

Mnożenie macierzy



Obniżamy diagonalną o jeden w dół.
Przysyłamy do sąsiadów w tym samym wierszu.
Mnożymy wartości przesyłane z wartościami macierzy B i dodajemy do wyniku w C

Mnożenie macierzy w 3D

$$\underset{A}{\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 2 & 1 \end{bmatrix}} \cdot \underset{B}{\begin{bmatrix} 1 & 1 & 3 \\ 0 & 1 & 2 \\ 2 & 3 & 0 \end{bmatrix}} = \underset{C}{\begin{bmatrix} 4 & 7 & 2 \\ 7 & 10 & 3 \\ 4 & 7 & 10 \end{bmatrix}}$$