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

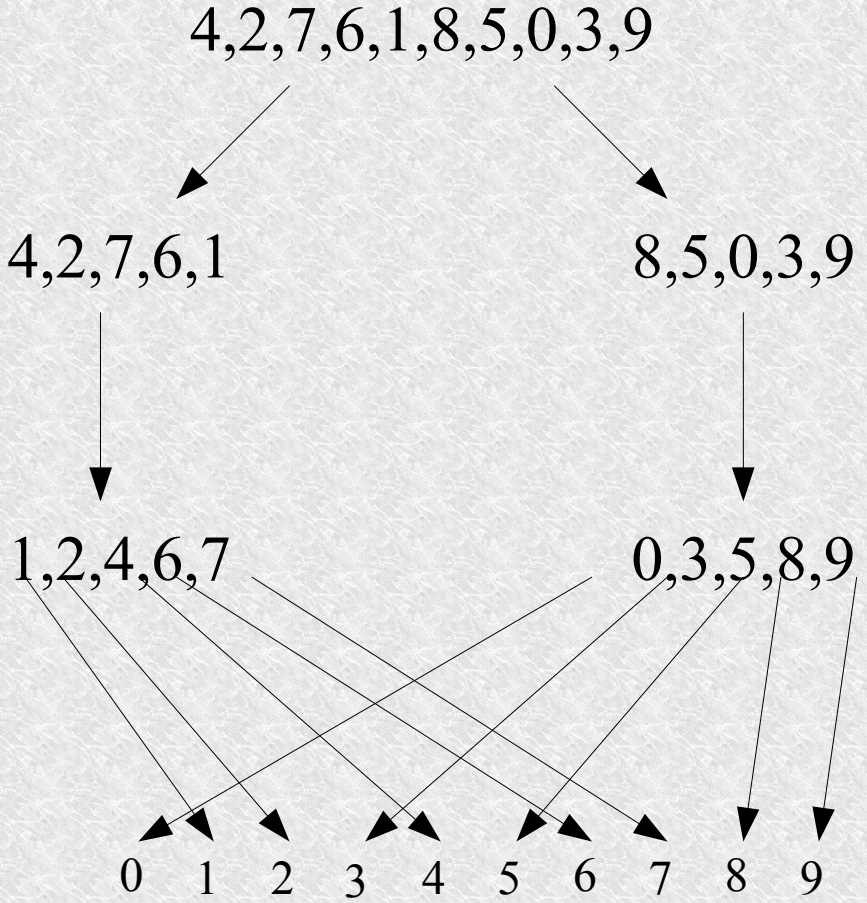
   JEŚLI_NIE to posortuj to co masz. (w szczególnym przypadku będzie to 1 liczba do oddania lub dwie do porównania).

Sortowanie przez scalanie (**mergesort**)

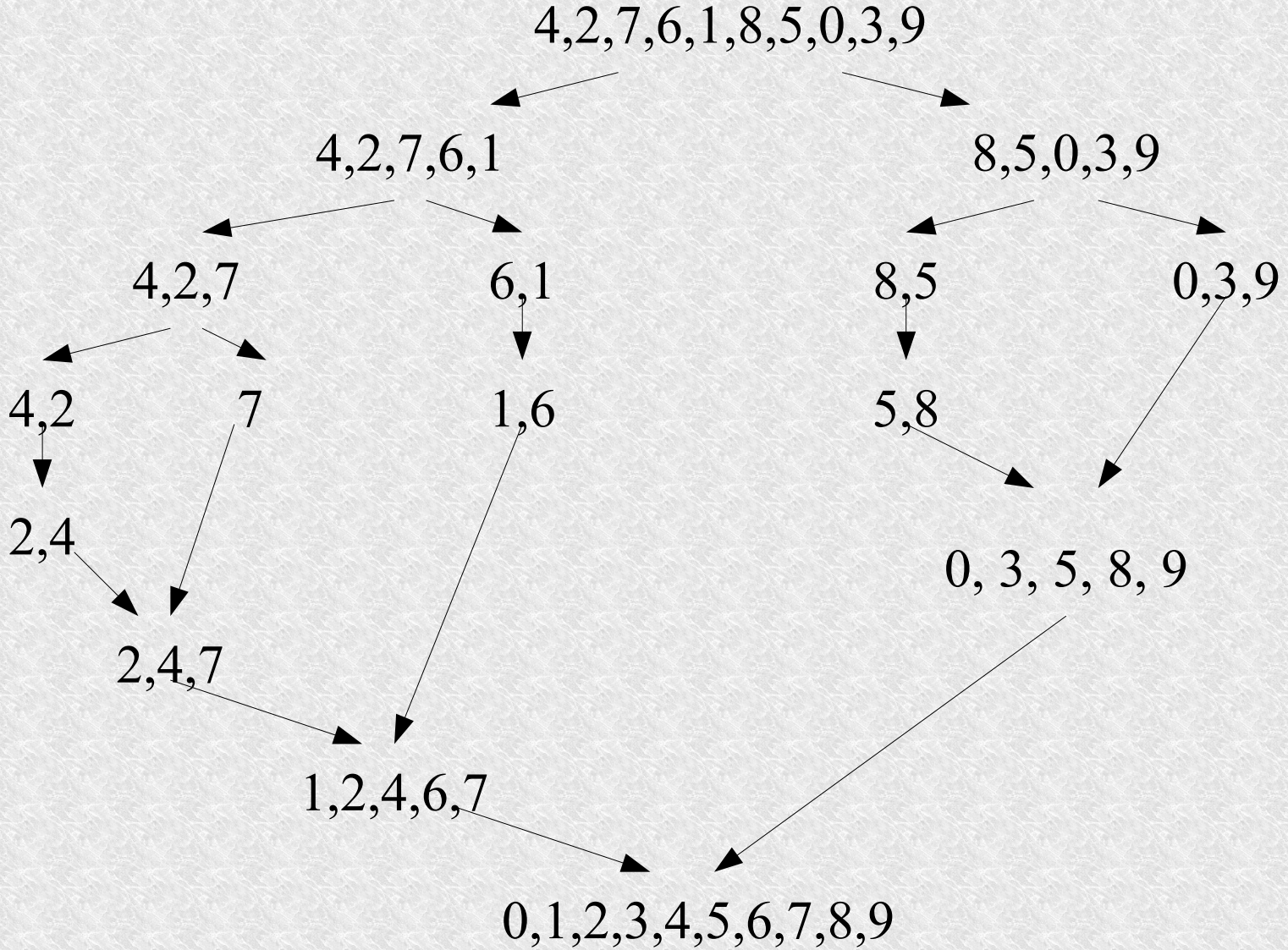
3. JEŚLI jesteś rootem TO wyświetl/zapisz wynik

   JEŚLI_NIE to odeślij rodzicowi posortowaną tablicę.

Sortowanie przez scalanie



Sortowanie przez scalanie

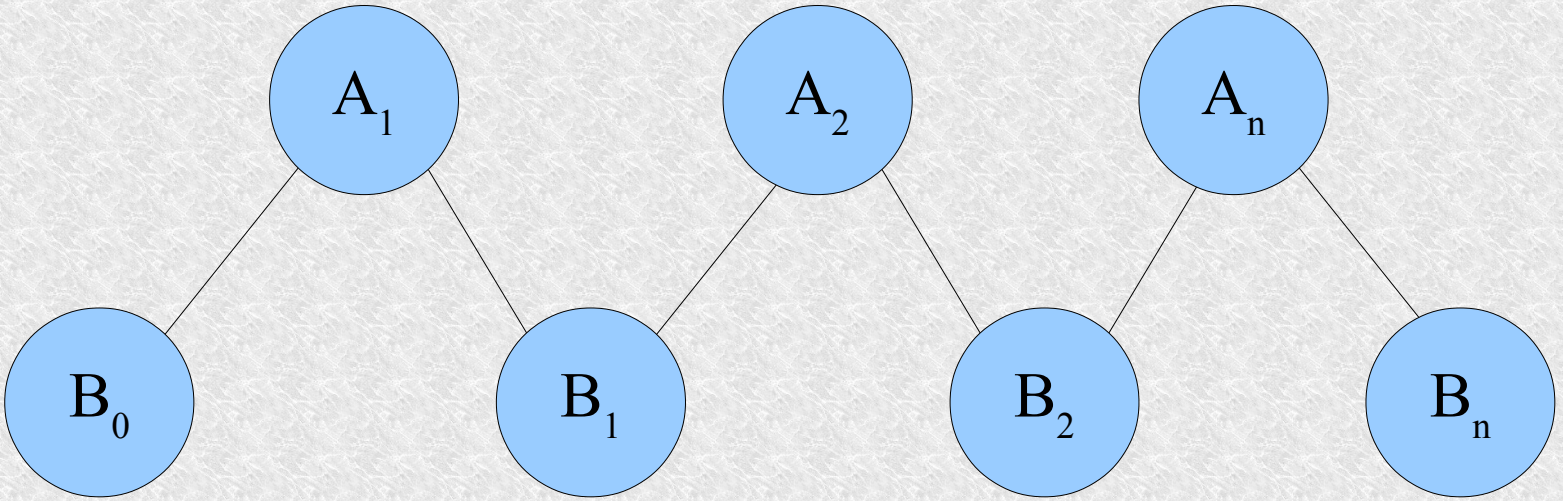


Sortowanie oscylacyjne

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

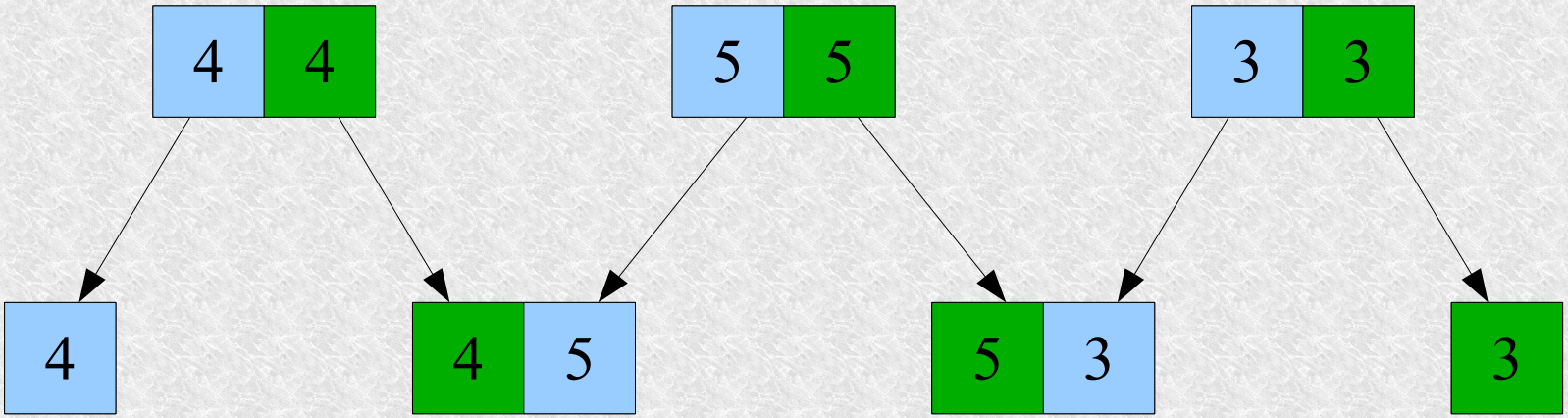
Zadania typu B_i działają następująco:

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

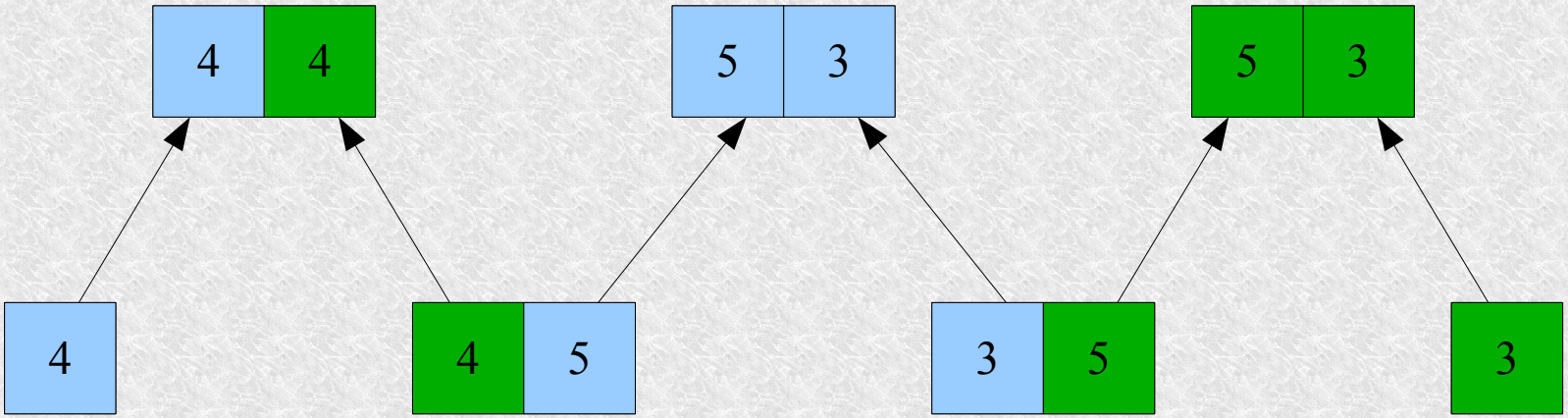
Elementy skrajne nic nie robią tylko oddają liczbę

Po **2n** cyklach mamy gotowy wynik

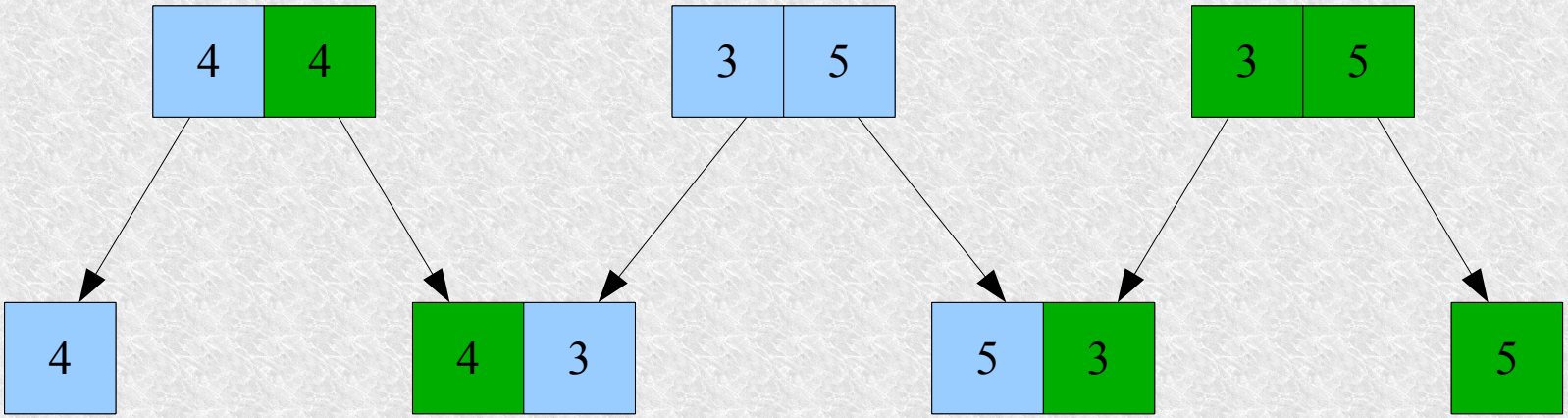
Sortowanie oscylacyjne



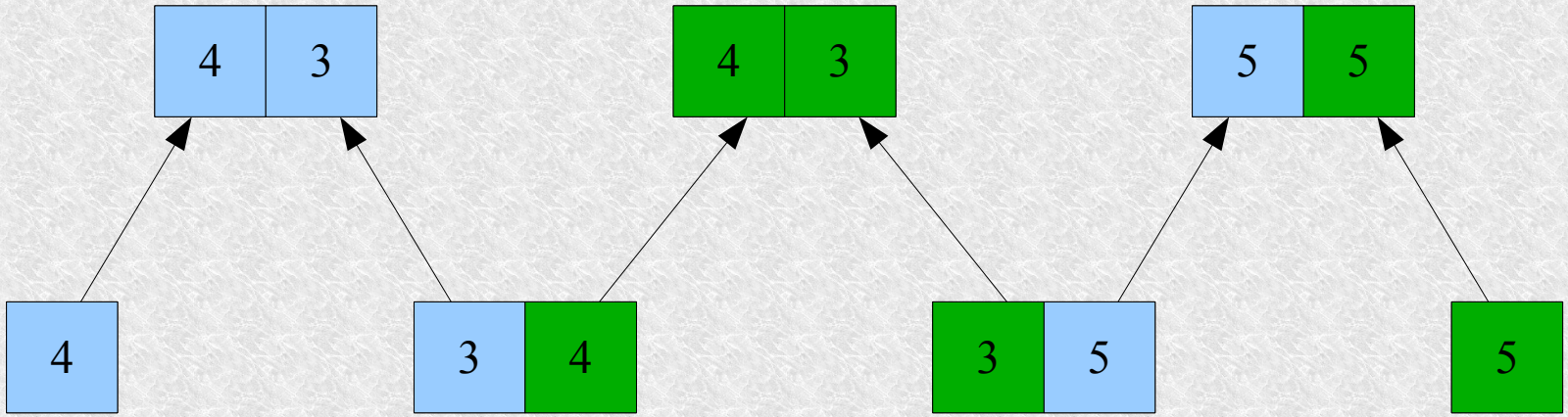
Sortowanie oscylacyjne



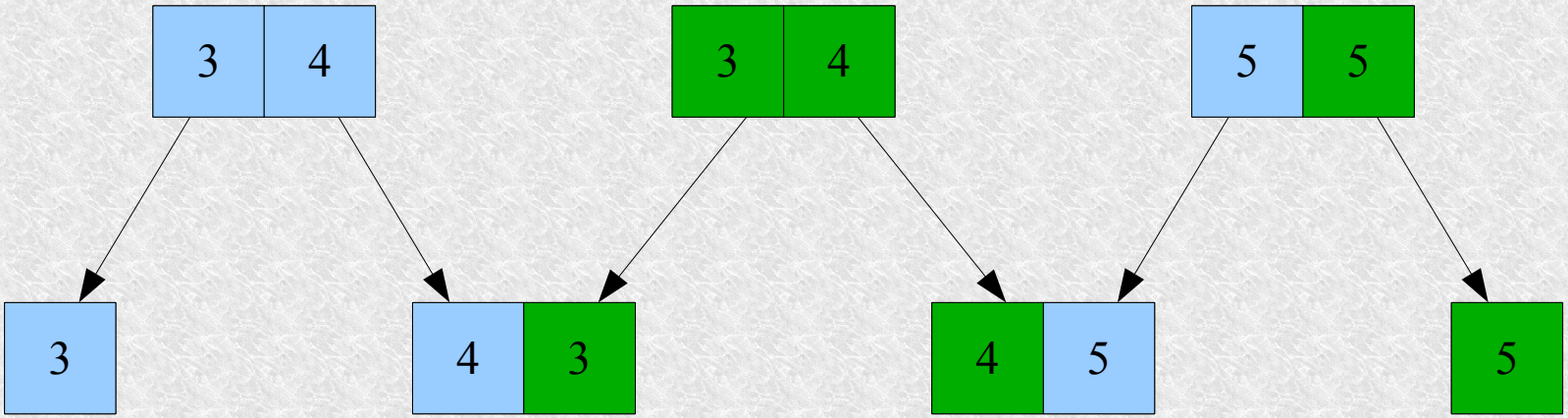
Sortowanie oscylacyjne



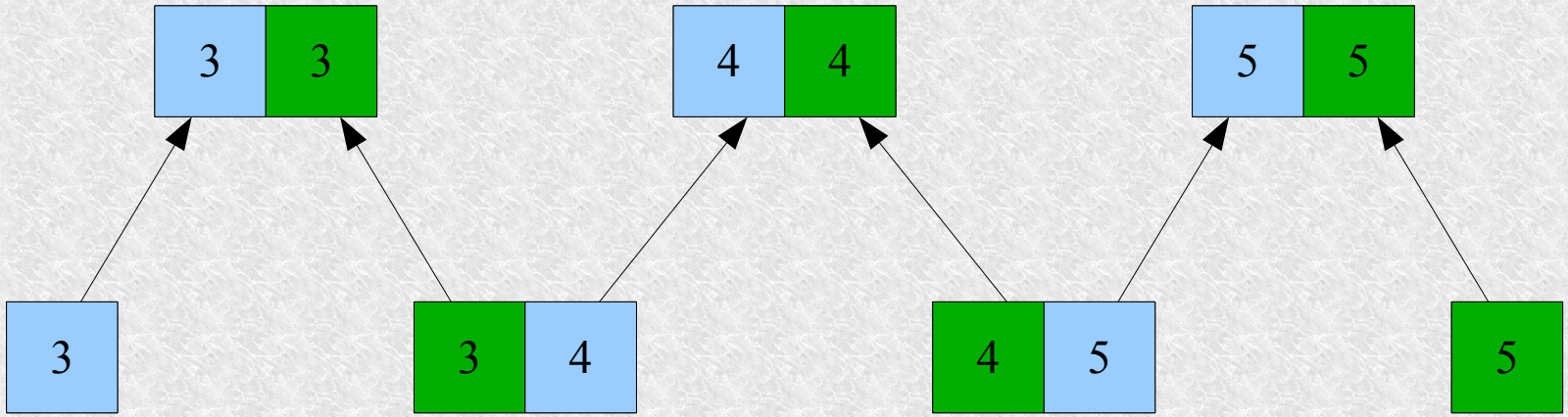
Sortowanie oscylacyjne



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 $\mathbf{A}=[a_{ij}]_{nxp}$ przez macierz $\mathbf{B}=[b_{ij}]_{pxm}$ nazywamy taką macierz $\mathbf{C}=[c_{ij}]_{nxm}$ 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

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

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

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

Mnożenie macierzy

Kilka przydatnych właściwości:

*Jeżeli **A**,**B** oraz **C** są macierzami o odpowiednich wymiarach to:*

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

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 $O(n^3)$.

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:

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

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 $\mathbf{m_i}$ 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”**

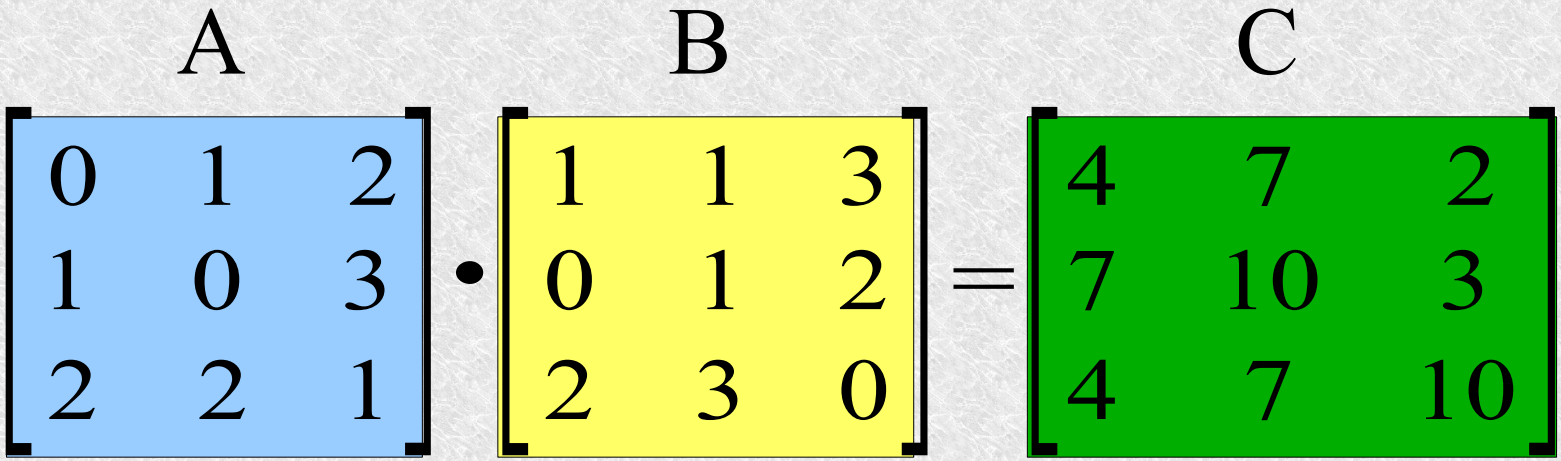
- Zakładamy że mamy sieć zadań m x m.
- Każdy proces (t_{ij} gdzie $0<i,j<m$) wewnątrz zawiera bloki C_{ij}, A_{ij} i B_{ij}.
- Na wstępie algorytmu proces na przekątnej diagonalnej (t_{ij} gdzie i=j) przesyła swój blok A_{ij} do wszystkich innych procesów w rzędzie i.
- Po transmisji A_{ii}, wszystkie 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 $t_{(m-1)j}$.

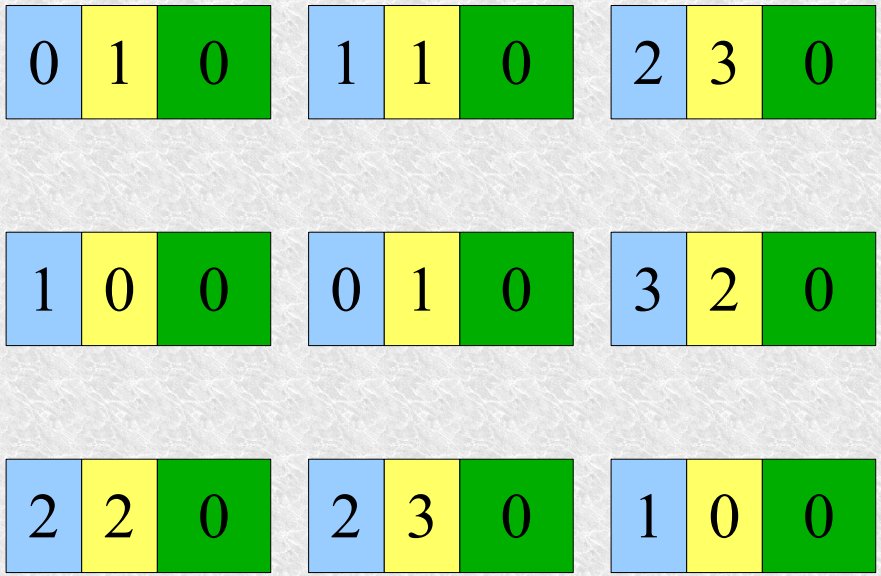
Mnożenie macierzy

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

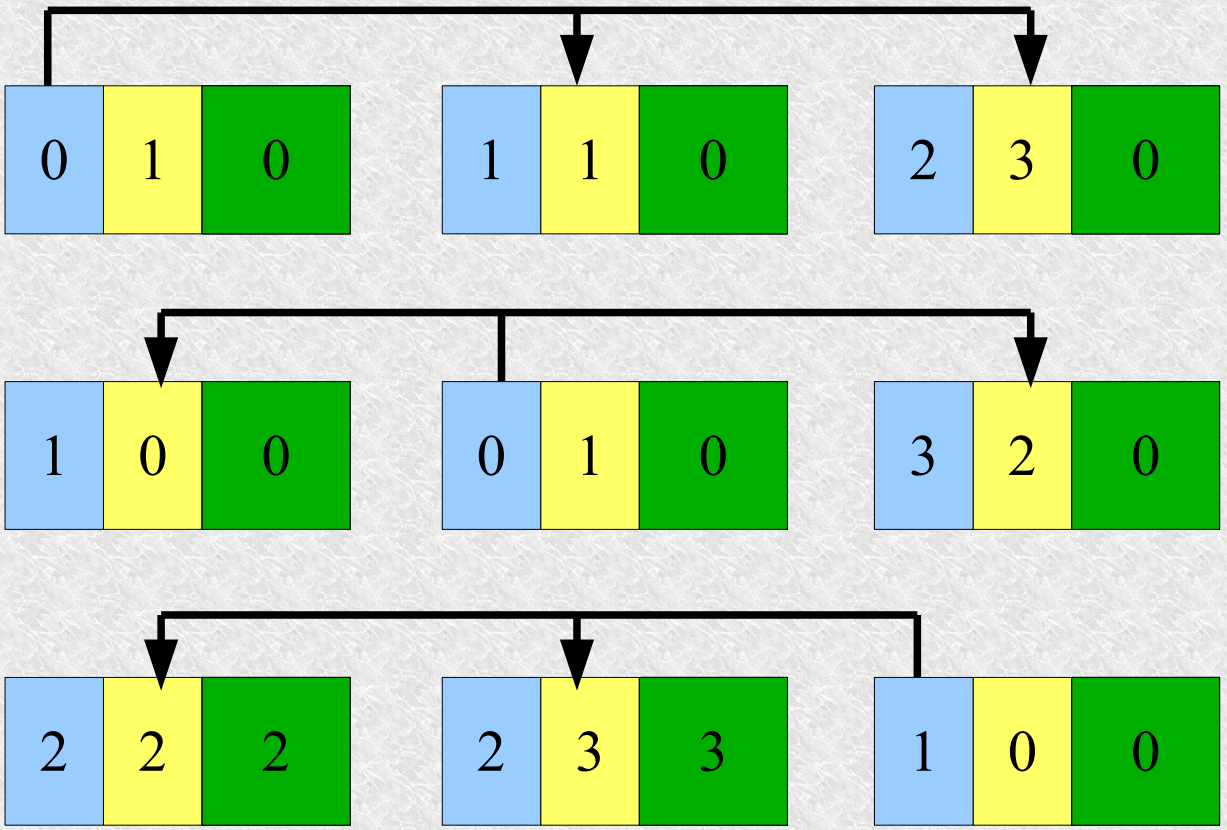
- Algorytm jest dalej kontynuowany. Po m iteracjach macierz C zawiera wynik mnożenia AxB, a obracana macierz B przyjmuje swoją początkową postać.

Mnożenie macierzy



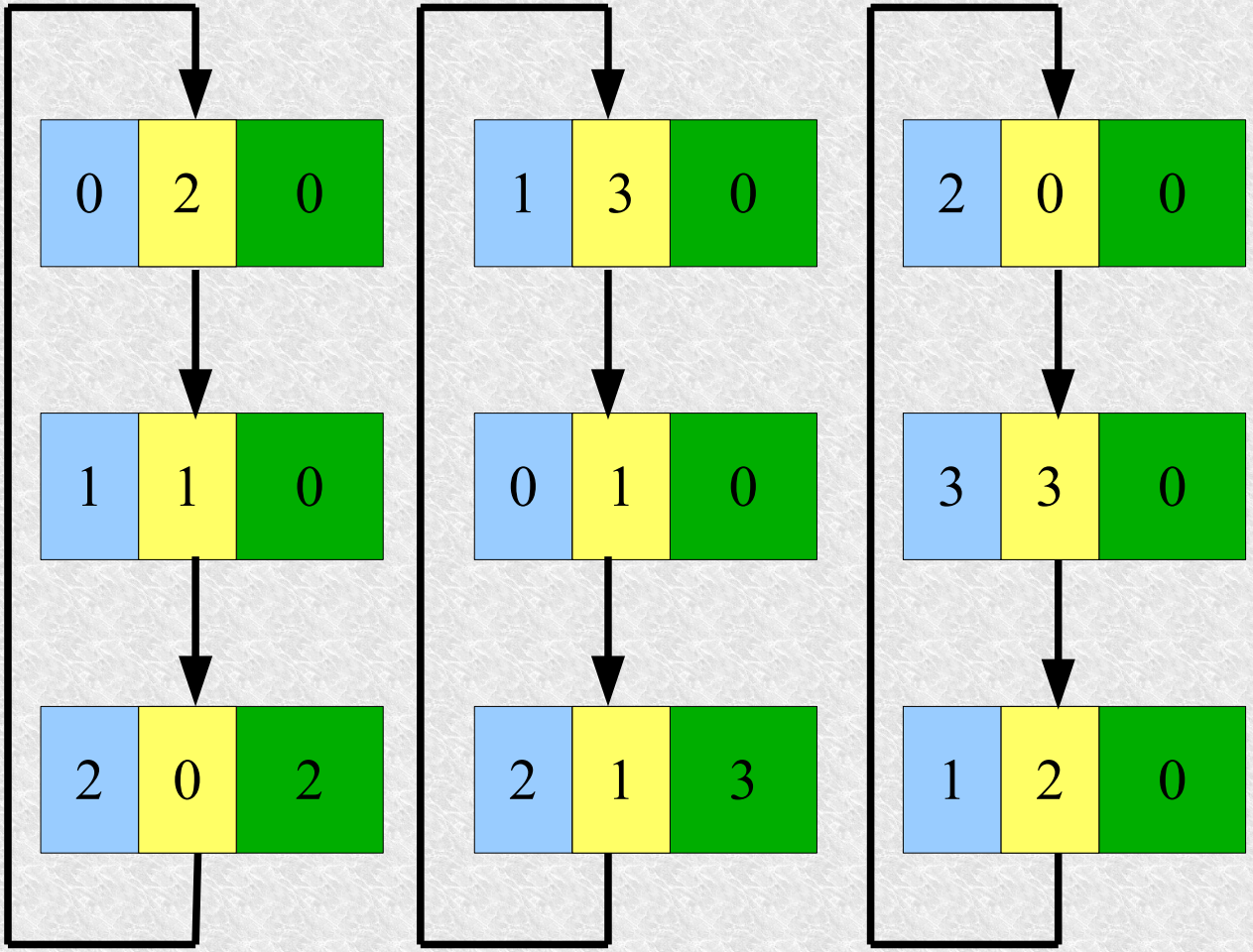


Mnożenie macierzy



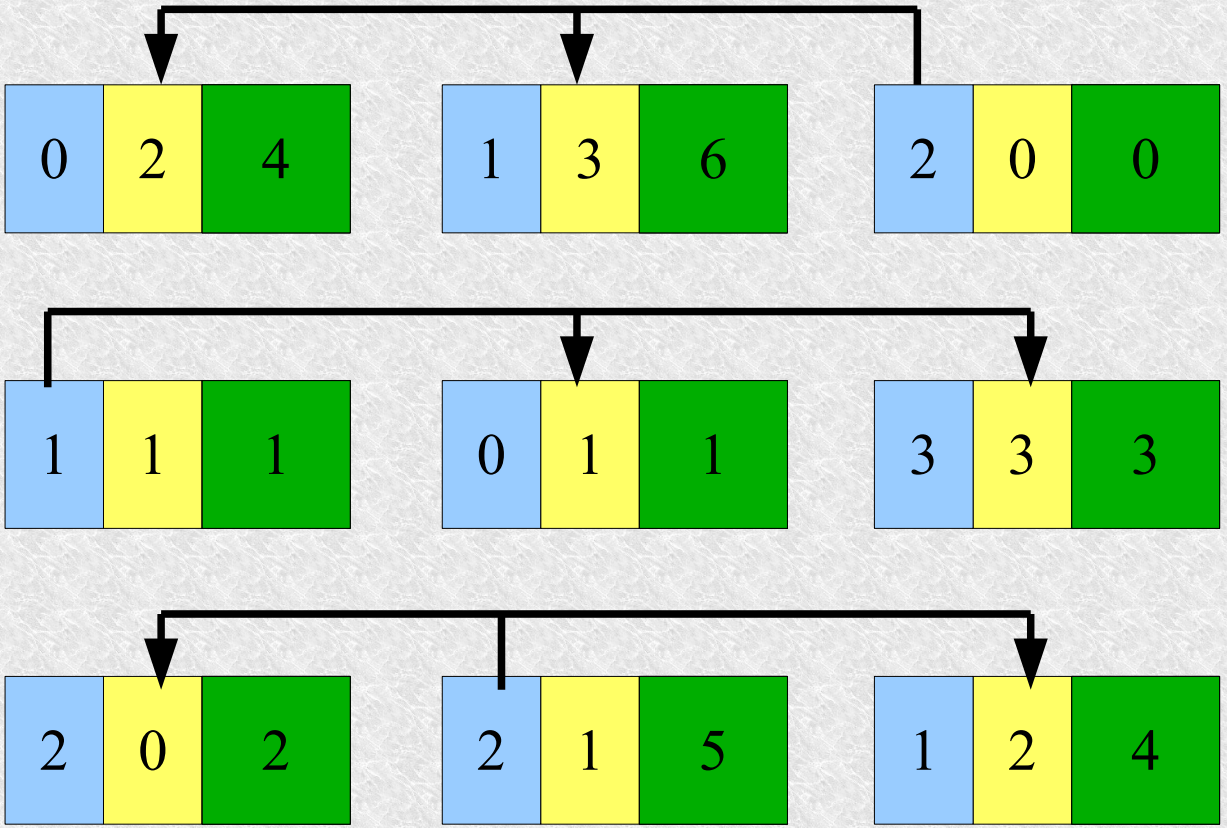
Z macierzy A bierzemy wartości leżące na diagonalnej
Przysyłamy do sąsiadów w tym samym wierszu.
Mnożymy wartości przesyłane z wartościami macierzy B i dodajemy do wyniku w C

Mnożenie macierzy



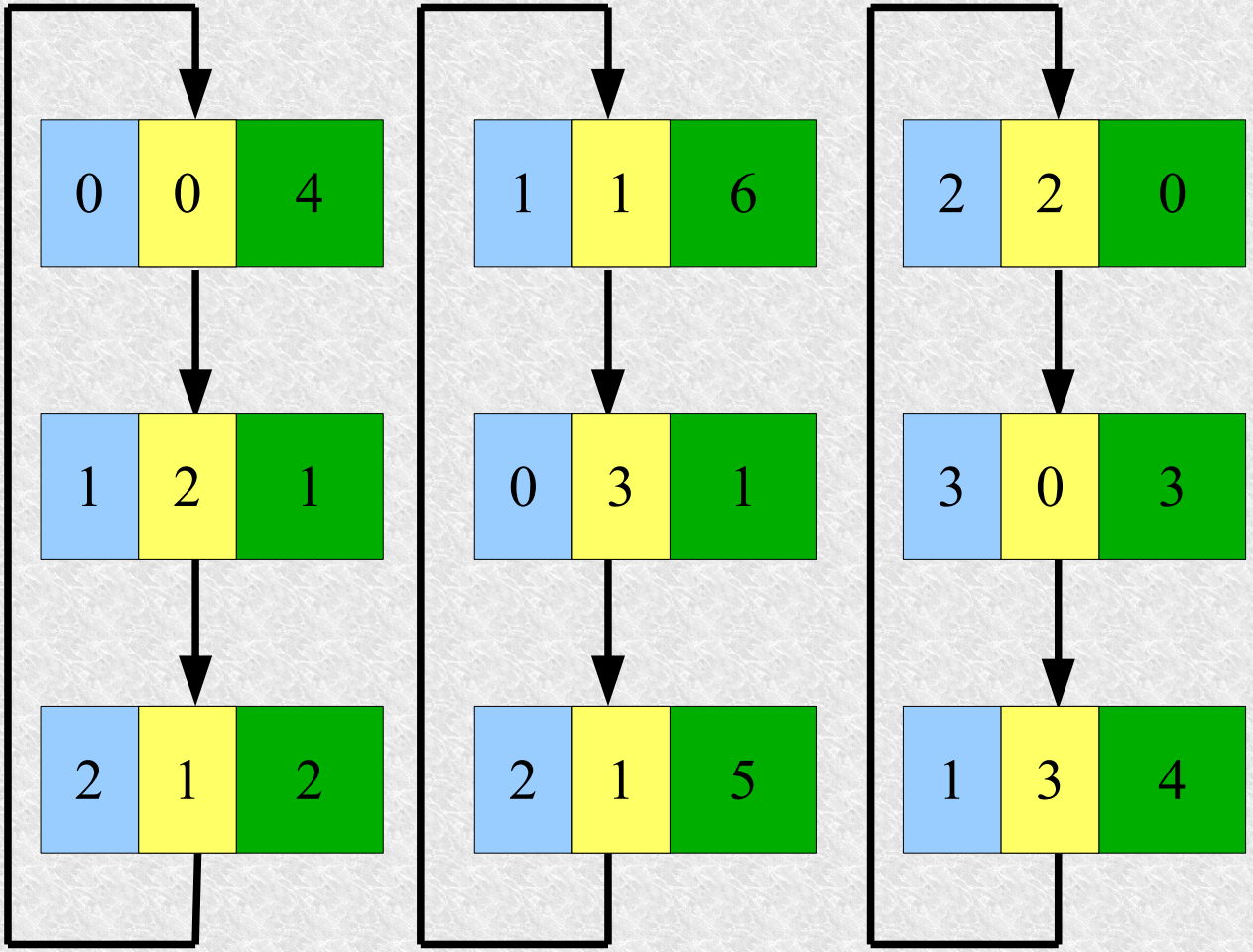
Kolumny macierzy B „rolujemy” w dół.

Mnożenie macierzy



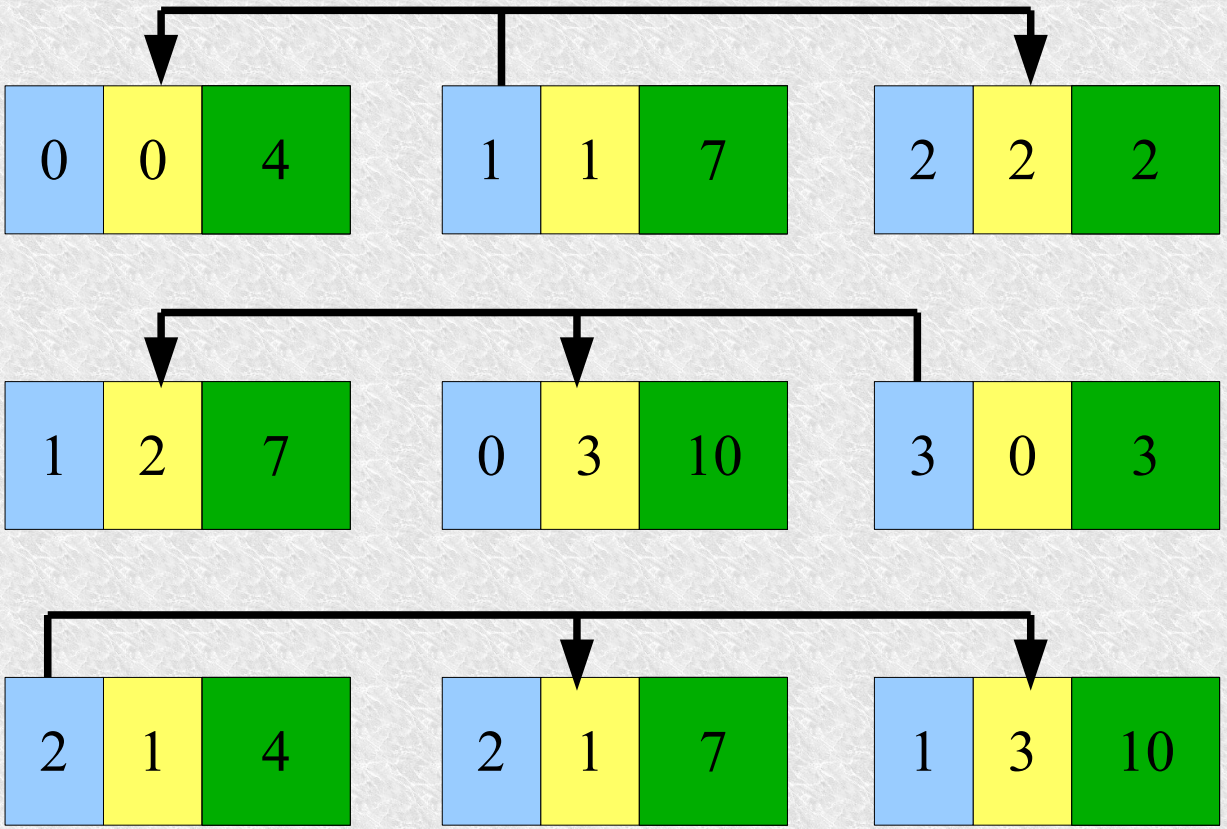
Obniżamy diagonalną o jeden w dół.
Przysyłamy do sąsiadów w tym samym wierszu.
Mnożymy wartości przesyłane z wartościami macierzy B i dodajemy do wyniku w C

Mnożenie macierzy



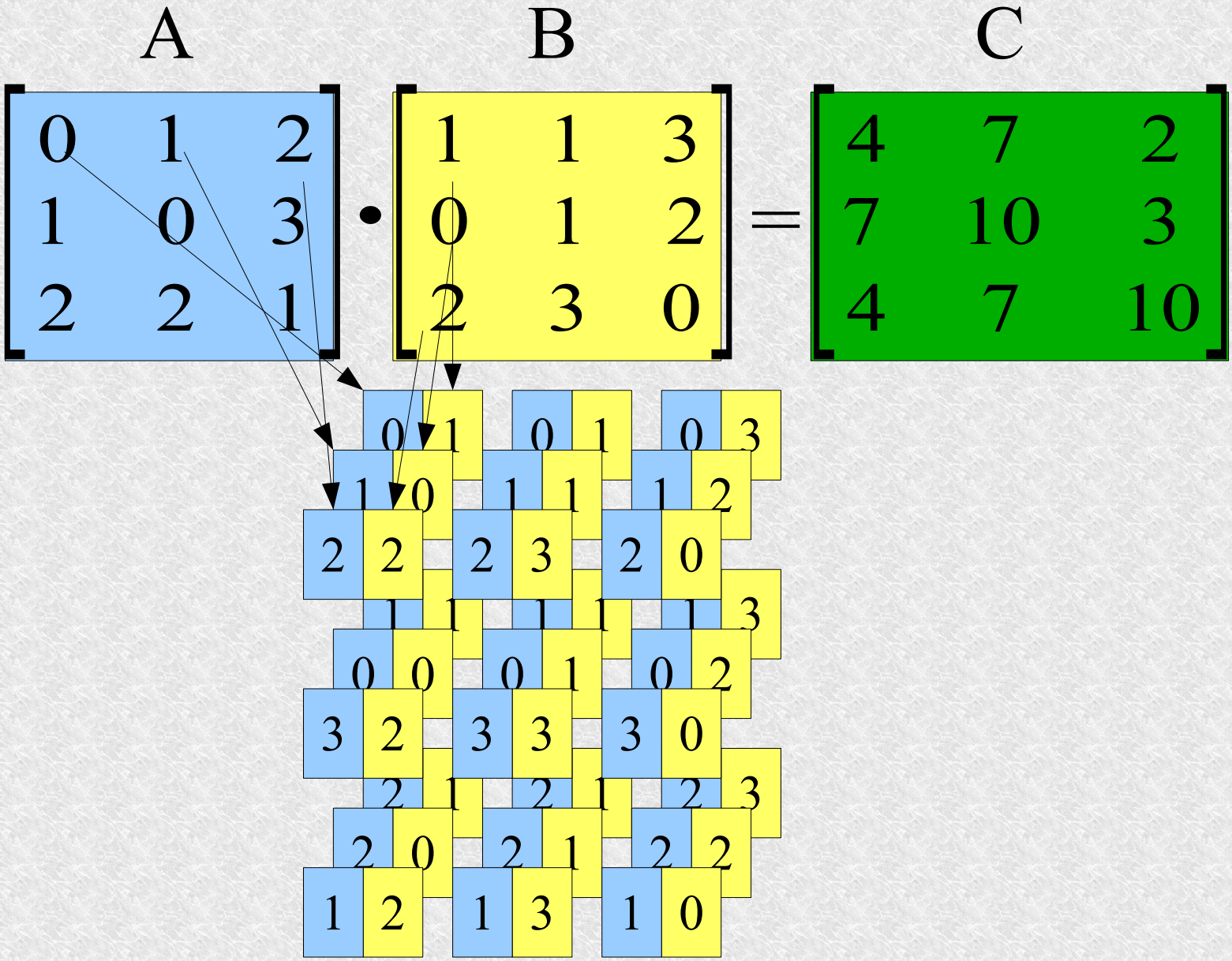
Kolumny macierzy B „rolujemy” w dół.

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



Algorytm Genetyczny

- Algorytm genetyczny - rodzaj algorytmu przeszukującego przestrzeń alternatywnych rozwiązań problemu w celu wyszukania rozwiązań najlepszych.
- Zaliczany do grona algorytmów ewolucyjnych
- Twórcą jest John Henry Holland

Źródło: wikipedia

Algorytm Genetyczny

- Problem definiuje środowisko
- Populacja – zbiór osobników
- Genotyp – zbiór informacji przypisany do pojedynczego osobnika
- Fenotyp – zbiór cech osobnika podlegającemu ocenie funkcji przystosowania
- Funkcja przystosowania modeluje środowisko
- Chromosomy – części składowe genotypu
- Geny – elementy chromosomu

Źródło: wikipedia

Algorytm Genetyczny

Wspólne cechy algorytmów genetycznych

- Stosowanie operandów genetycznych takich jak krzyżowanie i mutacja, które dostosowane są do postaci rozwiązań
- Równoległe przeszukiwanie przestrzeni rozwiązań, startujące z różnych punktów
- Ukierunkowanie przestrzeni poszukiwań następuje na podstawie informacji o jakości rozwiązania (funkcja przystosowania)
- Celowe wprowadzenie elementów losowych

Źródło: wikipedia

Algorytm Genetyczny

Podstawowy algorytm

- Losowana populacja początkowa (inicjacja)
- Rozwój osobników, ocena ich przystosowania
- Sprawdzenie warunku wyjścia
- Losowanie (najlepsi mają największe szanse)
- Zastosowanie operatorów genetycznych
  - Krzyżowanie
  - Mutacja
- Powstaje nowe pokolenie. Tu algorytm wraca do punktu 2.

Źródło: wikipedia

Algorytm Genetyczny

Kodowanie

- Chromosom przedstawiamy zwykle jako wektor genów
- Geny reprezentowane są przez liczby binarnych, całkowitych lub rzeczywistych (genów)
- Kodowanie logarytmiczne
- Struktury drzewiaste

Źródło: wikipedia

Algorytm Genetyczny

Ocena

- Chromosomy są parametrem funkcji oceny f(x)
- Badając wartość f(x) wiemy w jaki sposób jest przystosowany osobnik o chromosomie x
- Przy optymalizacji, szukamy maximum lub minimum f(x)

Źródło: wikipedia

Algorytm Genetyczny

Dobór naturalny

- Według koła ruletki
  - Całe koło ruletki odpowiada sumie wartości funkcji przystosowania wszystkich chromosomówdanej populacji.
  - Dla każdego chromosomu ch_i, dla $i=1,2,...,N$, gdzie N liczebność populacji, odpowiada wycinek koła zgodnie ze wzorem $Ch_i = p_s * 100$

    Gdzie prawdopodobieństwo wyrażone jest

$$p_s(ch_i) = \frac{F(ch_i)}{\sum_{i=1}^{N} F(ch_i)} \; przy \; (i = 1,2,...,N)$$

  - $F(ch_i)$ oznacza funkcje przystosowania chromosomu ch_i
  - Losowanie odbywa się N razy

Algorytm Genetyczny

Dobór naturalny

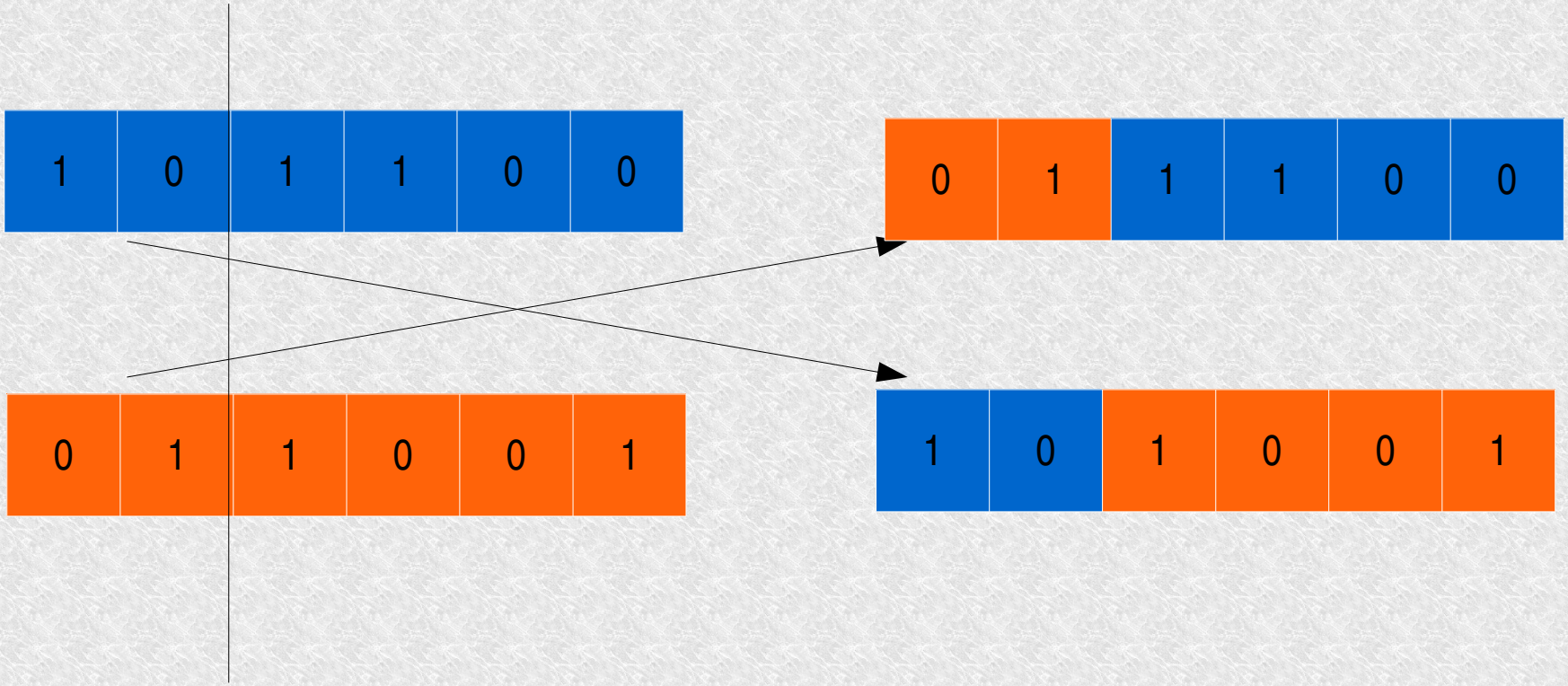
- Metoda elitarna
  - najlepszy osobnik przechodzi bez zmian
- Metoda turniejowa
  - Osobniki dzielone są na grupy i z poszczególnych grup wybierane najlepsze rozwiązania.
- Metoda rankingowa.
  - Rozwiązania sortowane są według jakości rozwiązania i nadawane rangi. Potem zwykle funkcją liniową wybiera się odpowiednią liczbę osobników poczynając od najlepszego

Źródło: wikipedia

Algorytm Genetyczny

Krzyżowanie

- Podstawowe poprzez cięcie w wybranym punkcie (locus)



Źródło: wikipedia

Algorytm Genetyczny

Inne metody krzyżowania

- Krzyżowanie z częściowym odwzorowaniem (PMX)
- Krzyżowanie z porządkowaniem (OX)

Algorytm Genetyczny

Przykład

- Znalezienie maximum funkcji $f(x)=x^2$ w przedziale 0..31
- Wybieramy system kodowania zmiennej decyzyjnej x (5b)
- Populacja złożona z 4 ciągów
  01101
  11000
  01000
  10011

Algorytm Genetyczny

Przykład

Nr ciągu	Populacja początkowa	Wartość x	f(x)	prawdopodobieństwo	Oczekiwana liczba kopii	Liczba kopii wylosowanych	Wylosowana pula rodzicielska	Losowo wybrany partner	Losowy punkt krzyżowania	Nowa populacja	x	f(x)
1	01101	13	169	0,14	0,58	1	011|01	2	4	01100	12	144
2	11000	24	576	0.49	1,92	2	110|00	1	4	11001	25	625
3	01000	8	64	0,06	0,22	0	11|000	4	2	11011	27	729
4	10011	19	361	0,31	1,23	1	10|011	3	2	10000	16	256
suma			1170	1,00	4	4						1754
średnia			293	0,25	1	1						439
maksimum			576	0,49	1,97	2						729

Algorytm Genetyczny

Zastosowanie w programowaniu współbieżnym

- Głównie układ master-slaves
  - Master zleca procesom roboczym rozwój populacji, całych grup lub tylko pojedynczych osobników na proces
  - Procesy Slave zwracają wartość funkcji przystosowania
  - Master decyduje o doborze naturalnym
- W układzie P2P
  - Dobór naturalny bez procesu zarządzającego.
  - Wymiana chromosomów bezpośrednia.

Algorytm Mrówkowy

Kilka słów na temat mrówek

- Są praktycznie ślepe
- Mają bardzo dobre zmysły węchu
- Mają znikome mózgi.
- Gdy działają w stadzie potrafią znaleźć najkrótszą drogę pomiędzy pożywieniem a mrowiskiem
- Korzystają z feromonów i zjawiska stygmergii (zjawisko pośredniej komunikacji poprzez wywoływanie zmian w środowisku i odczytywanie ich)
- Mrówki maszerując zostawiają na swej drodze feromon
- Feromony nieustannie parują
- Mrówki podczas decyzji którą z dróg wybrać kierują się intensywnością zapachu feromonu.

Algorytm Mrówkowy

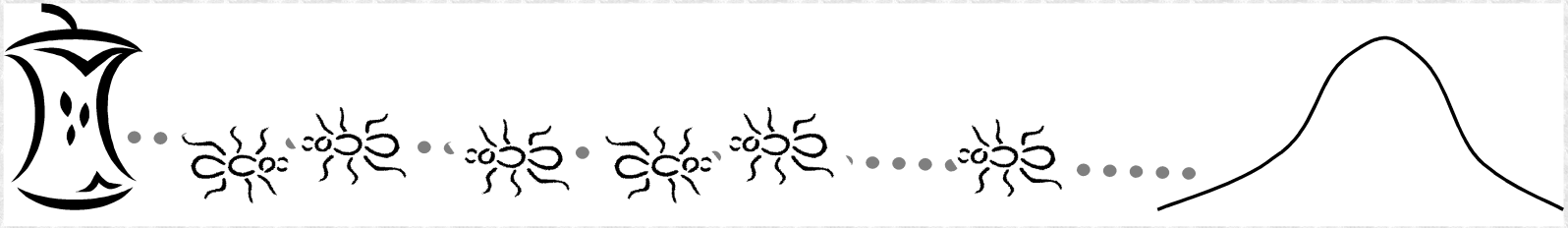
Działanie algorytmu

- Mrówki losowo rozchodzą się z mrowiska szukając pożywienia.
- Te mrówki które trafią krótszą drogą, szybciej wrócą po swoich śladach
- Na krótszych trasach liczba mrówek w jednostce czasu jest większa a więc i odświeżanie śladu feromonowego jest intensywniejsze
- Trasy dłuższe, jako rzadziej odwiedzane wyparowują.

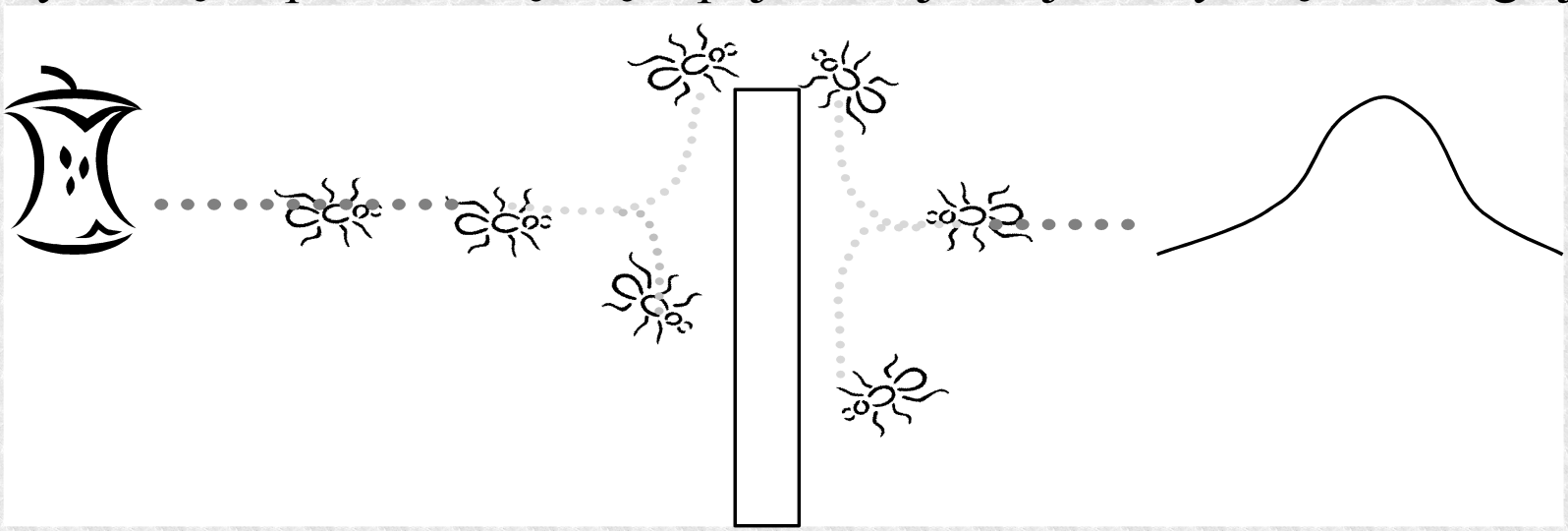
Algorytm Mrówkowy

Działanie algorytmu w przypadku podjęcia decyzji

- Mrówki mają ustaloną trasę



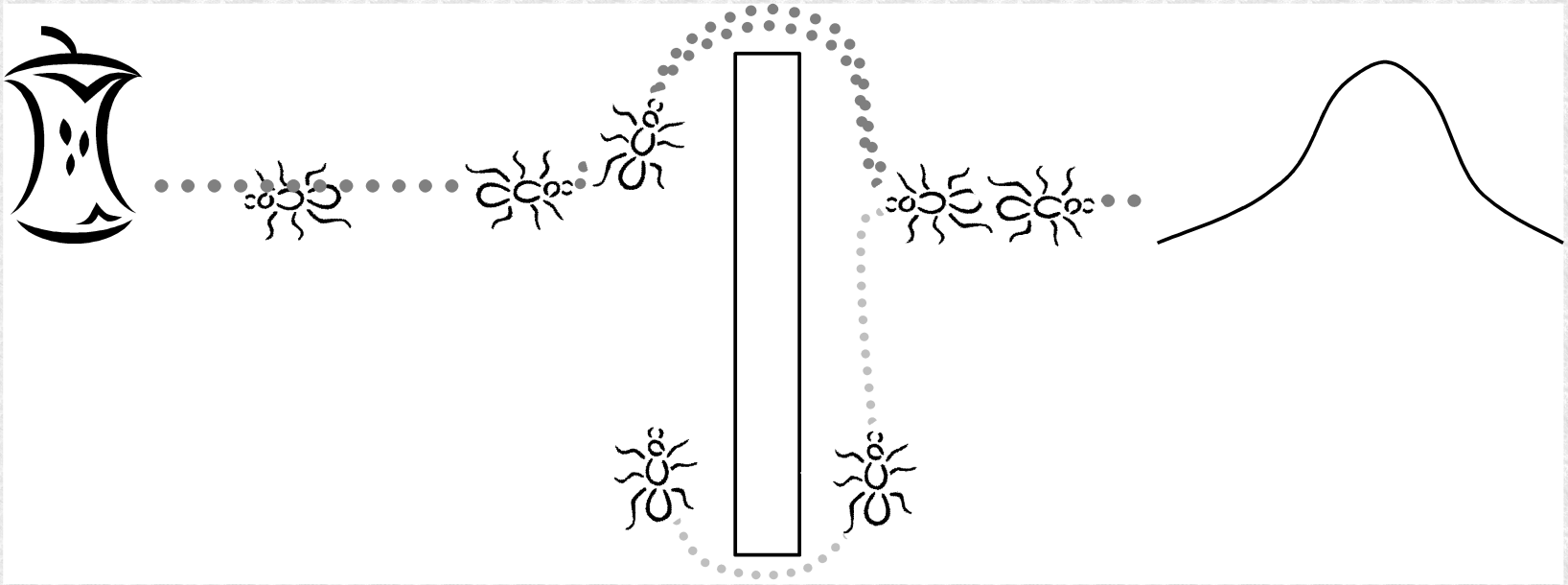
- Gdy trafią na przeszkodę część pójdzie z jednej strony część z drugiej



Algorytm Mrówkowy

Działanie algorytmu w przypadku podjęcia decyzji

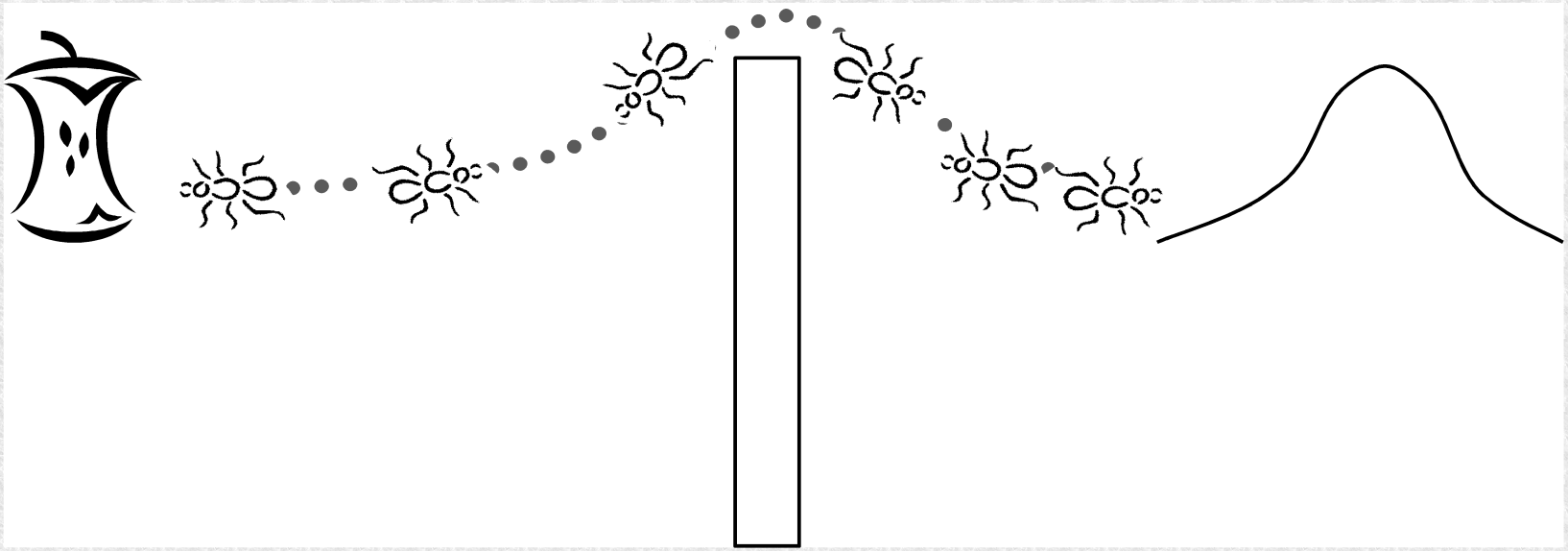
- Mrówki idące krótszą drogą szybciej połączą ślady feromonowe dając jednocześnie sygnał dla pozostałych mrówek która trasa jest lepsza.



Algorytm Mrówkowy

Działanie algorytmu w przypadku podjęcia decyzji

- Z czasem mrówki zoptymalizują swoją trasę.



Algorytm Mrówkowy

Wyszukiwanie najlepszej trasy w problemie komiwojażera

- Punkt startu - dowolny
- Cel pojedynczej mrówki – odwiedzić wszystkie miasta tylko 1 raz
- Poruszanie się po krawędziach grafu
- Mrówki działają iteracyjnie
- Feromon nakładany dopiero po znalezieniu tras przez wszystkie mrówki w danej iteracji (algorytm CAS – Cycle Ant System)
- Ilość feromonu odwrotnie proporcjonalna do znalezionej drogi lub nakładana tylko przez najlepszą mrówkę

Algorytm Mrówkowy

Wyszukiwanie najlepszej trasy w problemie komiwojażera

- Na początku każda krawędź ma równą ilość feromonu
- Każda mrówka wybiera sobie dowolny punkt startu
- Wybierając kolejny i-ty węzeł na swojej trasie kieruje się tablicą decyzyjną

$$A_i = [a_{ij}(t)]_{[N_i]}$$

Algorytm Mrówkowy

Elementy tablicy decyzyjnej określone są wzorem:

$$a_{ij} = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i}[\tau_{ij}(t)]^{\alpha}[\eta_{il}]^{\beta}} \forall j \in N_i$$

- τ_{ij} - ilość feromonu pomiędzy węzłem i oraz j
- $\eta_{ij} = \frac{c}{d_{ij}}$ Wartość heurystyczna informująca o atrakcyjności danego połączenia punktu *i* oraz *j*. *c* to stała *d* odległość pomiędzy *i* a *j*
- N_i - zbiór możliwych połączeń wychodzących z węzła *i*
- αβ – parametry dostrajające algorytm, można nimi regulować jaki udział w podjęciu decyzji ma heurystyka a jaki ilość feromonu.

Algorytm Mrówkowy

Prawdopodobieństwo wybrania węzła $j \in N_i^k$ dla k mrówek w pozycji i podczas t-iteracji jest:

$$p_{ij}^k(t) = \frac{a_{ij}(t)}{\sum_{l \in N_i^k} a_{il}(t)}$$

Gdzie $N_i^k \subseteq N_i$ jest zestawem sąsiednich węzłów nie odwiedzonych.

Algorytm Mrówkowy

Gdy wszystkie mrówki znajdą drogę wybieramy najlepszą z nich lub proporcjonalnie dla wszystkich uaktualniamy ilość feromonu na każdej krawędzi wchodzącej w skład trasy poprzez dodanie m/L feromonu gdzie m stała, L – długość znalezionej trasy.

Do tego dochodzi jeszcze parowanie. Z każdej krawędzi grafu odparowujemy pewną stałą część feromonu:

$$\tau_{ij} = \tau_{ij} * \rho$$