# I zestaw zadań - Mnożenie macierzy

Kacper Kozubowski, Mateusz Podmokły
III rok Informatyka WI

2 październik 2024

## 1 Treść zadania

Należy wygenerować macierze losowe o wartościach z przedziału otwartego $(10^{-8}, 1.0)$ i zaimplementować

1. Rekurencyjne mnożenie macierzy metodą Binét'a
2. Rekurencyjne mnożenie macierzy metodą Strassena
3. Mnożenie macierzy metodą AI na podstawie artykułu w Nature

Proszę zliczać liczbę operacji zmienno-przecinkowych wykonywanych podczas mnożenia macierzy.

## 2 Specyfikacja użytego środowiska

Specyfikacja:

- Środowisko: Jupyter Notebook,
- Język programowania: Python,
- System operacyjny: Microsoft Windows 11,
- Architektura systemu: x64.

## 3 Działanie algorytmów

### 3.1 Wykorzystane biblioteki

W realizacji rozwiązania wykorzystane zostały następujące biblioteki:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import time
```

### 3.2 Pseudokod

**Algorithm 1** Algorytm Binét'a

```
Input: A, B
Output: C
function BINET(A, B)
    n = size(A)
    if n = 1 then
        return A * B
    end if

    mid = n div 2
    A11 = A[1:n, 1:n] // Górny lewy blok
    A12 = A[1:n, n+1:2n] // Górny prawy blok
    A21 = A[n+1:2n, 1:n] // Dolny lewy blok
    A22 = A[n+1:2n, n+1:2n] // Dolny prawy blok
    B11 = B[1:n, 1:n]
    B12 = B[1:n, n+1:2n]
    B21 = B[n+1:2n, 1:n]
    B22 = B[n+1:2n, n+1:2n]

    C11 = BINET(A11, B11) + BINET(A12, B21)
    C12 = BINET(A11, B12) + BINET(A12, B22)
    C21 = BINET(A21, B11) + BINET(A22, B21)
    C22 = BINET(A21, B12) + BINET(A22, B22)

    C[1:n/2, 1:n/2] = C1 // Górny lewy blok
    C[1:n/2, n/2+1:n] = C2 // Górny prawy blok
    C[n/2+1:n, 1:n/2] = C3 // Dolny lewy blok
    C[n/2+1:n, n/2+1:n] = C4 // Dolny prawy blok
    return C
end function
```

**Algorithm 2** Algorytm Strassen'a

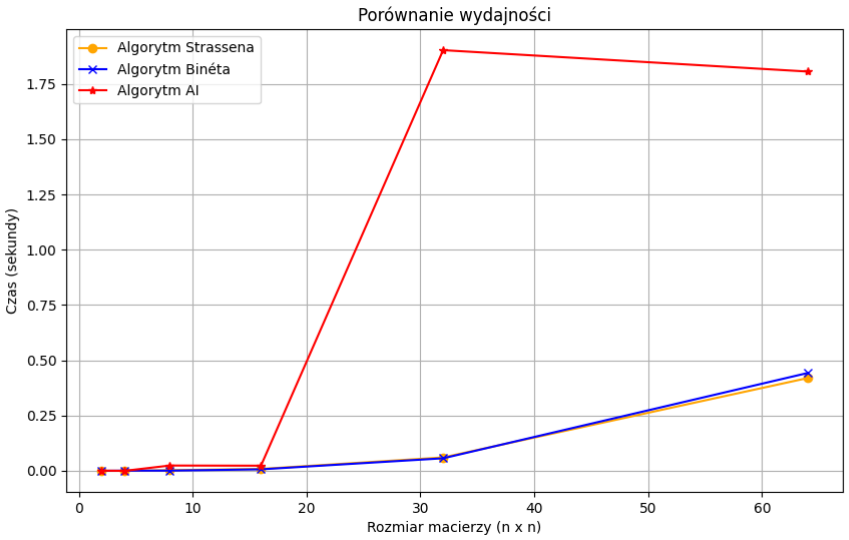
```
Input: A, B
Output: C
function STRASSEN(A, B)
    n = size(A)
    if n = 1 then
        return A * B
    end if

    mid = n div 2
    A11 = A[1:n, 1:n] // Górny lewy blok
    A12 = A[1:n, n+1:2n] // Górny prawy blok
    A21 = A[n+1:2n, 1:n] // Dolny lewy blok
    A22 = A[n+1:2n, n+1:2n] // Dolny prawy blok
    B11 = B[1:n, 1:n]
    B12 = B[1:n, n+1:2n]
    B21 = B[n+1:2n, 1:n]
    B22 = B[n+1:2n, n+1:2n]

    P1 = STRASSEN(A11 + A22, B11 + B22)
    P2 = STRASSEN(A21 + A22, B11)
    P3 = STRASSEN(A11, B12 - B22)
    P4 = STRASSEN(A22, B21 - B11)
    P5 = STRASSEN(A11 + A12, B22)
    P6 = STRASSEN(A21 - A11, B11 + B12)
    P7 = STRASSEN(A12 - A22, B21 + B22)

    C11 = P1 + P4 - P5 + P7
    C12 = P3 + P5
    C21 = P2 + P4
    C22 = P1 + P3 - P2 + P6

    C[1:n/2, 1:n/2] = C1 // Górny lewy blok
    C[1:n/2, n/2+1:n] = C2 // Górny prawy blok
    C[n/2+1:n, 1:n/2] = C3 // Dolny lewy blok
    C[n/2+1:n, n/2+1:n] = C4 // Dolny prawy blok
    return C
end function
```

# 4 Porównanie działania algorytmów

## 4.1 Czas działania

Czas działania algorytmów dla $n = 2^m$, gdzie

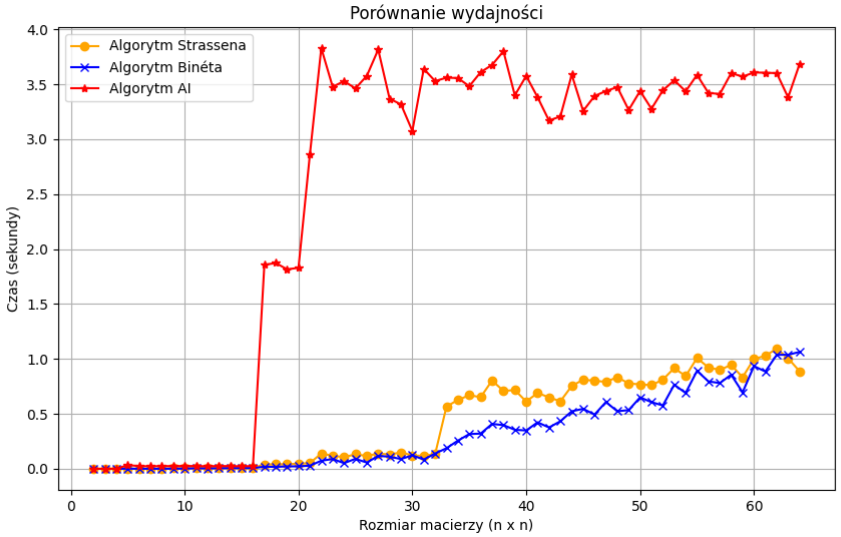
$$m \in \{2, 4, 8, 16, 32, 64\}$$




Rysunek 1: Porównanie czasu działania.

Czas działania algorytmów dla $n = 2^m$, gdzie

$$m \in [2, 64]$$

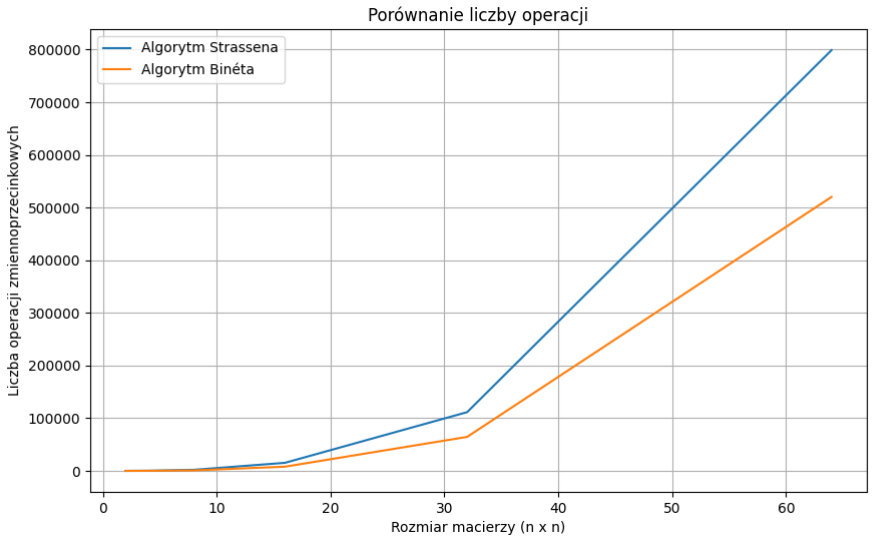



Rysunek 2: Porównanie czasu działania.

## 4.2 Liczba operacji zmienno-przecinkowych

Liczba operacji zmienno-przecinkowych wykonanych podczas działania algorytmów dla $n = 2^m$, gdzie

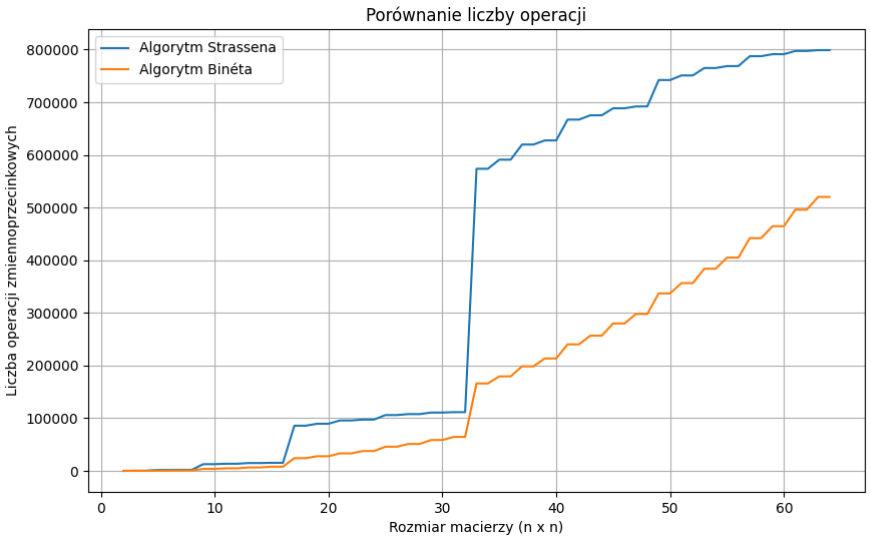
$$m \in \{2, 4, 8, 16, 32, 64\}$$




Rysunek 3: Porównanie liczby operacji zmienno-przecinkowych.

Liczba operacji zmienno-przecinkowych wykonanych podczas działania algorytmów dla $n = 2^m$, gdzie

$$m \in [2, 64]$$




Rysunek 4: Porównanie liczby operacji zmienno-przecinkowych.

# 5 Oszacowanie złożoności obliczeniowej

Dla każdego algorytmu zmierzony został czas obliczeń dla następujących wielkości macierzy:

$$n \in \{2, 4, 8, 16, 32, 64, 128\}$$

oraz dopasowana krzywa złożoności obliczeniowej.

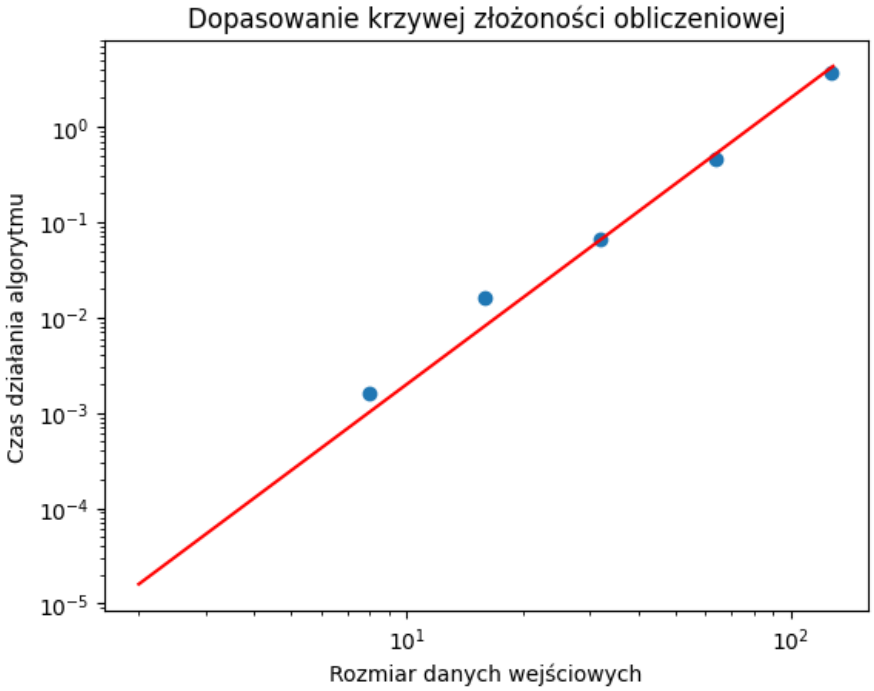
## 5.1 Algorytm Binét'a

Dopasowana krzywa:

$$y = 2 * 10^{-6} * x^3$$

Zatem złożoność wynosi

$$O(n) \approx n^3$$




Rysunek 5: Krzywa dopasowana w skali logarytmicznej dla algorytmu Binét'a.

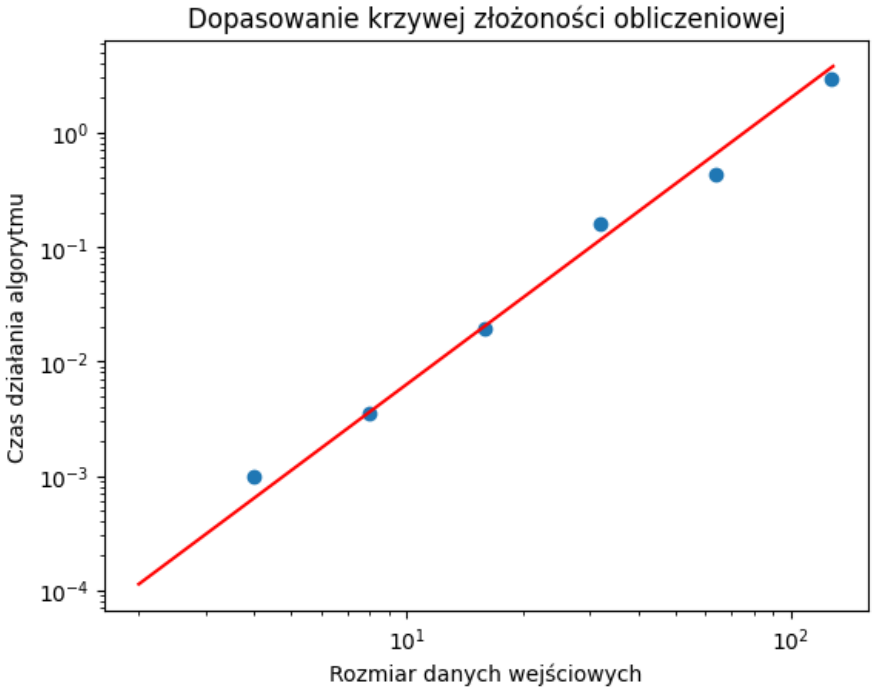
## 5.2 Algorytm Strassena

Dopasowana krzywa:

$$y = 2 * 10^{-5} * x^{2.5}$$

Zatem złożoność wynosi

$$O(n) \approx n^{2.5}$$




Rysunek 6: Krzywa dopasowana w skali logarytmicznej dla algorytmu Strassena.

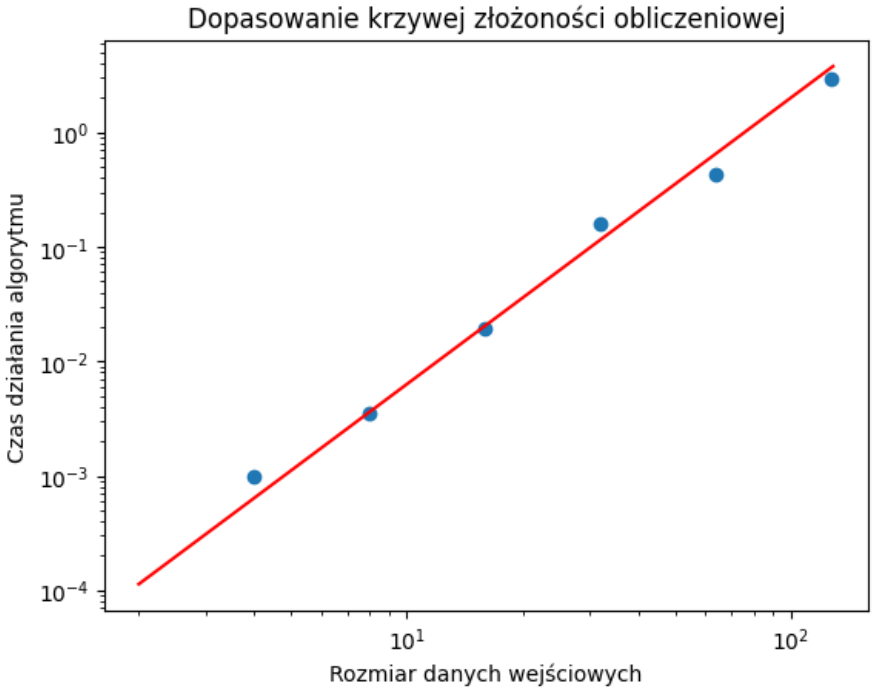
### 5.3 Algorytm AI

Dopasowana krzywa:

$$y = 1.8 * 10^{-5} * x^3$$

Zatem złożoność wynosi

$$O(n) \approx n^3$$




Rysunek 7: Krzywa dopasowana w skali logarytmicznej dla algorytmu AI.

## 6 Porównanie obliczeń ze srodowiskiem Matlab

Na wejściu mamy macierz $A$ i $B$. Wartości wewnątrz macierzy:

```
A =
[[0.88 0.65 0.41 0.29]
 [0.8  0.66 0.23 0.75]
 [0.76 0.45 0.03 0.47]
 [0.99 0.1  0.88 0.35]]
B =
[[0.84 0.78 0.24 0.42]
 [0.94 0.08 0.71 0.03]
 [0.74 0.1  0.36 0.06]
 [0.68 0.83 0.97 0.03]]
```

Rysunek 8: Wartości macierzy $A$ i $B$.

Wyniki mnożenia macierzy uzyskane za pomocą każdego z zaimplementowanych algorytmów:

```
Binet algorithm:
[[1.8508 1.0201 1.1016 0.4224]
 [1.9726 1.3223 1.4709 0.3921]
 [1.4032 1.0219 0.9686 0.3486]
 [1.8148 1.1587 0.9649 0.4821]]
Strassen algorithm:
[[1.8508 1.0201 1.1016 0.4224]
 [1.9726 1.3223 1.4709 0.3921]
 [1.4032 1.0219 0.9686 0.3486]
 [1.8148 1.1587 0.9649 0.4821]]
AI algorithm:
[[1.8508 1.0201 1.1016 0.4224]
 [1.9726 1.3223 1.4709 0.3921]
 [1.4032 1.0219 0.9686 0.3486]
 [1.8148 1.1587 0.9649 0.4821]]
```

Rysunek 9: Wyniki algorytmów.

Wyniki mnożenia macierzy obliczone za pomocą środowiska Matlab:

```
A =

    0.8800    0.6500    0.4100    0.2900
    0.8000    0.6600    0.2300    0.7500
    0.7600    0.4500    0.0300    0.4700
    0.9900    0.1000    0.8800    0.3500

>> B = [0.84 0.78 0.24 0.42;
0.94 0.08 0.71 0.03;
0.74 0.1 0.36 0.06;
0.68 0.83 0.97 0.03]

B =

    0.8400    0.7800    0.2400    0.4200
    0.9400    0.0800    0.7100    0.0300
    0.7400    0.1000    0.3600    0.0600
    0.6800    0.8300    0.9700    0.0300

>> A * B

ans =

    1.8508    1.0201    1.1016    0.4224
    1.9726    1.3223    1.4709    0.3921
    1.4032    1.0219    0.9686    0.3486
    1.8148    1.1587    0.9649    0.4821
```

Rysunek 10: Wyniki środowiska Matlab.

Wyniki uzyskane za pomocją zaimplementowanych algorytmów są identyczne z wynikami otrzymanymi ze środowiska Matlab. Można wnioskować, że przedstawione algorytmy działają poprawnie.