Sprawozdanie 1 - Wojciech Smolarczyk, Wiktoria Zalińska

```
def traditional_multiply_matrix(A, B):
    result = np.zeros((len(A), len(B)))
    global counts
    s = len(A)

    for i in range(s):
        for j in range(s):
            sum_j = 0
            for k in range(s):

                sum_j += A[i][k] * B[k][j]
                counts += 1

            result[i][j] = sum_j

    return result
```

Mnożenie macierzy tradycyjną metodą. Zmienna globalna counts zlicza liczbę operacji + , * .

```
def add_matrices(matrix1, matrix2):
    global counts
    if len(matrix1) != len(matrix2) or len(matrix1[0]) != 
len(matrix2[0]):
        raise ValueError("Matrices must have the same dimensions")

    result = [[0 for _ in range(len(matrix1[0]))] for _ in 
range(len(matrix1))]

    for i in range(len(matrix1)):
        for j in range(len(matrix1[0])):
            result[i][j] = matrix1[i][j] + matrix2[i][j]
            counts += 1

    return result
```

```python
    def subtract_matrices(matrix1, matrix2):
    global counts
    if len(matrix1) != len(matrix2) or len(matrix1[0]) != 
len(matrix2[0]):
        raise ValueError("Matrices must have the same dimensions")

    result = [[0 for _ in range(len(matrix1[0]))] for _ in 
range(len(matrix1))]

    for i in range(len(matrix1)):
        for j in range(len(matrix1[0])):
            result[i][j] = matrix1[i][j] - matrix2[i][j]
            counts += 1

    return result
```

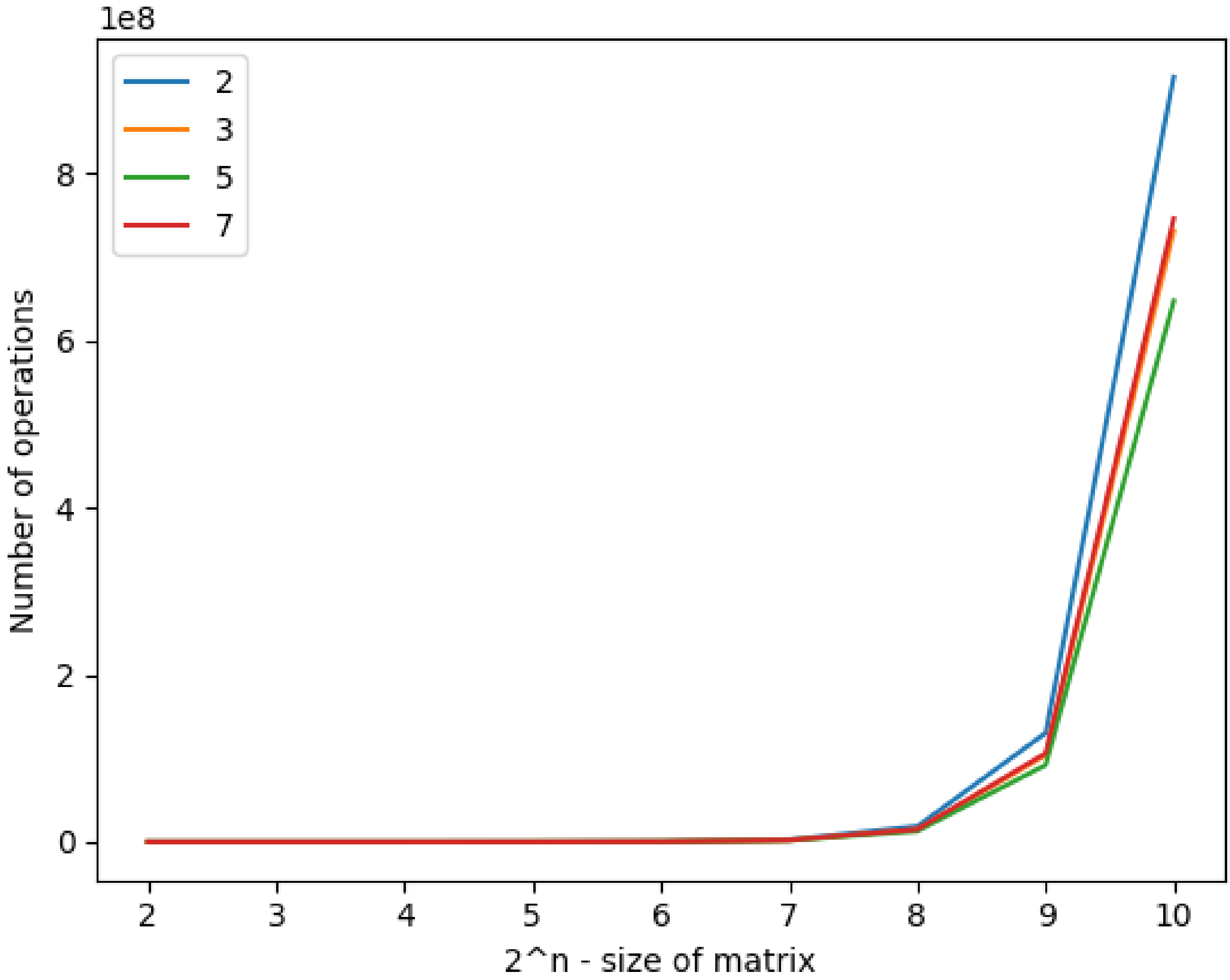
Powyższe funkcje pozwalają dodać i odjąć dwie macierze co będzie przydatne przy metodzie Strassena. Dodatkowo zmienna globalna counts zlicza liczbę operacji +, -, * .

```python
def split_matrix(matrix):
    n = len(matrix) // 2

    return (
        np.array(matrix)[:n, :n],
        np.array(matrix)[:n, n:],
        np.array(matrix)[n:, :n],
        np.array(matrix)[n:, n:],
    )


def strassen_multiply_matrix(A, B, l):

    if len(A) <= 2**l:

        return traditional_multiply_matrix(A, B)

    A11, A12, A21, A22 = split_matrix(A)
    B11, B12, B21, B22 = split_matrix(B)

    P1 = strassen_multiply_matrix(add_matrices(A11, A22),
```

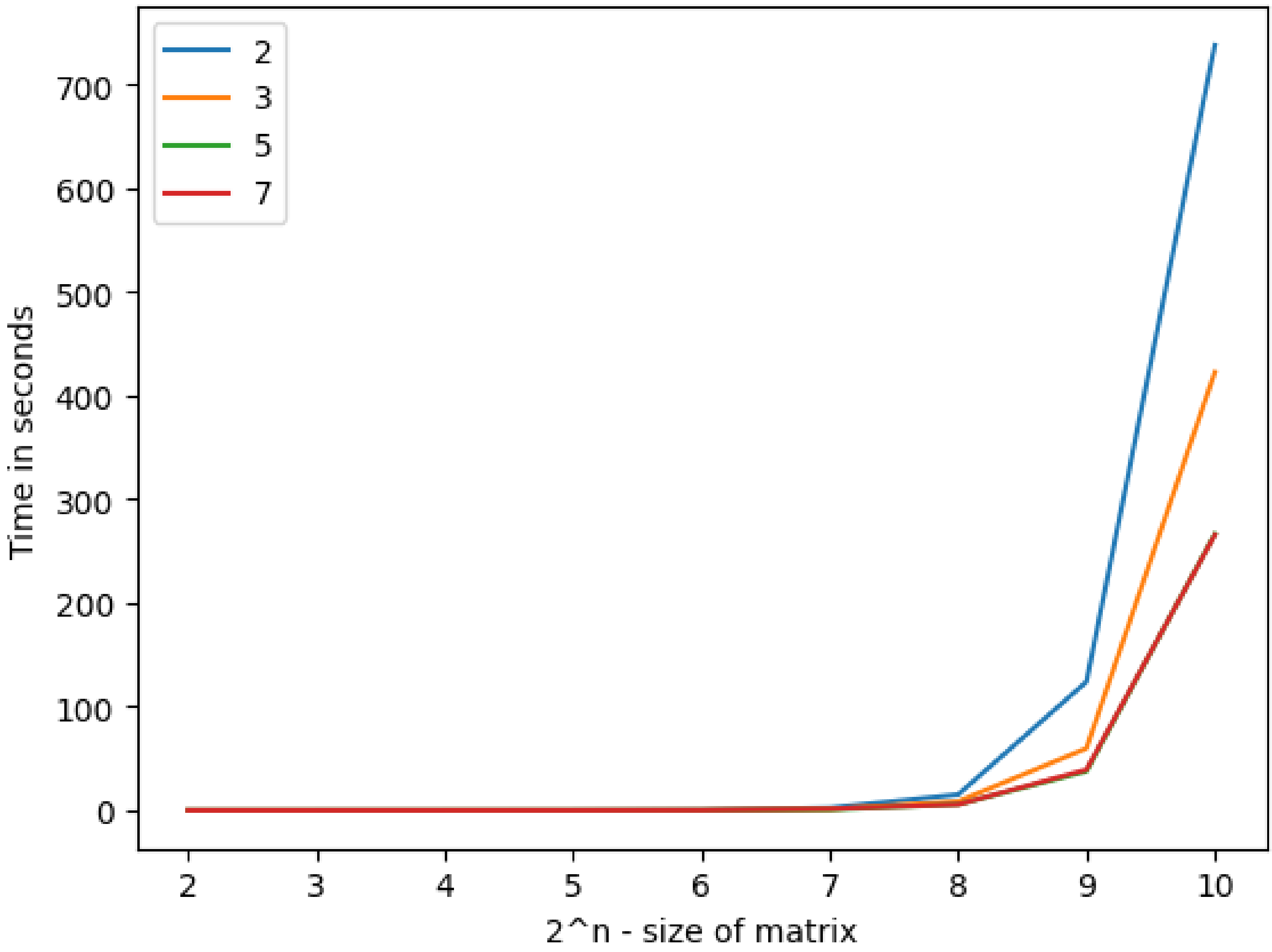
```
add_matrices(B11, B22), l)
    P2 = strassen_multiply_matrix(add_matrices(A21, A22), B11, l)
    P3 = strassen_multiply_matrix(A11, subtract_matrices(B12,
B22), l)
    P4 = strassen_multiply_matrix(A22, subtract_matrices(B21,
B11), l)
    P5 = strassen_multiply_matrix(add_matrices(A11, A12), B22, l)
    P6 = strassen_multiply_matrix(
        subtract_matrices(A21, A11), add_matrices(B11, B12), l
    )
    P7 = strassen_multiply_matrix(
        subtract_matrices(A12, A22), add_matrices(B21, B22), l
    )

    C11 = add_matrices(subtract_matrices(add_matrices(P1, P4),
P5), P7)
    C12 = add_matrices(P3, P5)
    C21 = add_matrices(P2, P4)
    C22 = add_matrices(subtract_matrices(add_matrices(P1, P3),
P2), P6)

    C = np.vstack((np.hstack((C11, C12)), np.hstack((C21, C22))))

    return C
```

Zaimplementowany algortym Strassena. Zmienna "l" określa przy jakim rozmiarze tablicy "przełączamy" się z algortymu Strassena na tradycyjne mnożenie macierzy. Alogrytm na początku sprawdza czy macierze otrzymane jako argumenty są większego rozmiaru niż 2^l. Jeśli tak to dzieli obie macierze. Każdą na cztery podmacierze równej wielkości (przykład macierz 4x4 -> 4 macierze 1x1 , 8x8 -> 4 macierze 2x2). Nastepnie rekurencyjnie używając algorytmu Strassena wyznaczamy siedem macierzy a następnie za ich pomocą konstruujemy 4 macierze, które po złożeniu dają macierz wynikową.

Wykresy



Co naturalne im większe tablice wejściowe tym więcej operacji należy wykonać. Im mniejsze jest "l" tym później (niżej) przełączamy się na tradcyjne mnożenie macierzy. Zauważalne różnice widać przy rozmiarze tablic 2^8, a różnica w liczbie operacji jest ogromna gry rozmiar tablicy przekracza 2^10. Warto również zauważyć na relatywnie małe różnice pomiędzy wynikami dla l=3 i l=7.



Bardzo podobne wnioski można wyciągnać dla czasu wykonania algortymu. Znów widać niewielke różnice (a nawet ich relatywny brak) dla l=5 i l=3 oraz im mniejsze l tym dłużej wykouje się algortym. Jedyną większą różnicą jest fakt, że relatywny czas wykonania algortymu dla l=2 i macierzy startowej rozmiaru 2^10 jest znacznie większy niż liczba wykonanych operacji w stosunku do innych wartości l.