Projekt 1 - Mnożenie macierzy

Temat 2 - Dla macierzy o rozmiarze mniejszym lub równym $2^l \times 2^l$ algorytm tradycyjny. Dla macierzy o rozmiarze większym od $2^l \times 2^l$ algorytm rekurencyjny Strassena.

Bartłomiej Jamiołkowski, Cyprian Neugebauer

March 14, 2024

1 Pseudokod

1.1 Kod wyboru algorytmu na podstawie parametru l

Podstawową ideą naszego programu jest wybór algorytmu w zależności od rozmiaru macierzy. Dla macierzy o rozmiarze mniejszym lub równym $2^l \times 2^l$ wykonywany jest algorytm tradycyjny, w przeciwnym razie wykonywany jest algorytm Strassena.

```python
def matmul(A: np.ndarray, B: np.ndarray) -> np.ndarray:
    if np.log2(A.shape[0]) <= l:
        return traditional_matmul(A, B)
    else:
        return strassen_matmul(A, B)
```

Rysunek 1: Kod wyboru algorytmu na podstawie parametru l

1.2 Tradycyjny algortym mnożenia macierzy

W implementacji tradycyjnego mnożenia macierzy wykonujemy następujące kroki:

1. Tworzymy pustą macierz wynikową C o wymiarach odpowiadających liczbie wierszy macierzy A i liczbie kolumn macierzy B, aby przechowywać wyniki operacji.
2. Rozpoczynamy proces mnożenia od iteracji przez wszystkie wiersze macierzy A.
3. Dla każdego wiersza z macierzy A przeprowadzamy iterację po wszystkich kolumnach macierzy B.
4. Dla każdej pary (wiersz z A, kolumna z B):
   (a) Mnożymy odpowiadające sobie elementy z wiersza macierzy A i kolumny macierzy B.
   (b) Sumujemy wyniki tych mnożeń.
5. Zapisujemy wynik sumy w macierzy C na pozycji odpowiadającej wierszowi z macierzy A i kolumnie z macierzy B. Dla i-tego wiersza macierzy A i j-tej kolumny macierzy B, wynik zapisujemy na pozycji $C[i, j]$.

```python
def traditional_matmul(A: np.ndarray, B: np.ndarray) -> np.ndarray:
    rows_first_matrix, cols_first_matrix = A.shape
    cols_second_matrix = B.shape[1]

    C = np.empty((rows_first_matrix, cols_second_matrix))

    for i in range(rows_first_matrix):
        for j in range(cols_second_matrix):
            sum = 0
            for k in range(cols_first_matrix):
                sum += A[i, k] * B[k, j]
                counter["+"] += 1
                counter["*"] += 1

            C[i, j] = sum

    return C
```

Rysunek 2: Kod tradycyjnego algorytmu mnożenia macierzy

1.3 Algorytm rekurencyjny Strassena

Algorytm Strassena do mnożenia macierzy jest metodą, która pozwala na redukcję liczby operacji mnożenia przez rekurencyjny podział macierzy i zastosowanie specjalnych kombinacji sum i różnic ich podmacierzy.

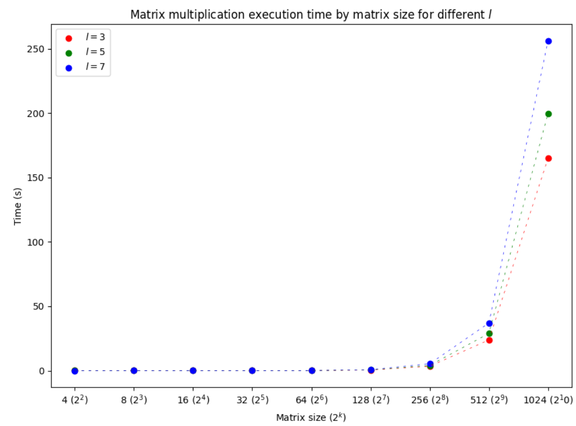
1. **Warunek bazowy:** Gdy macierz ma wymiar 1×1, elementy A i B są mnożone bezpośrednio, a wynik jest zwracany.
2. **Podział macierzy:** Macierz A i B jest dzielona na cztery podmacierze: $A_{11}, A_{12}, A_{21}, A_{22}$ dla A oraz $B_{11}, B_{12}, B_{21}, B_{22}$ dla B.
3. **Obliczanie produktów pomocniczych:** Za pomocą określonych kombinacji podmacierzy A i B, obliczane jest siedem produktów pomocniczych P_1 do P_7.
4. **Konstrukcja macierzy wynikowej C:** Korzystając z operacji dodawania i odejmowania na produktach P_1 do P_7, formowane są cztery podmacierze wynikowe $C_{11}, C_{12}, C_{21}, C_{22}$.
5. **Łączenie podmacierzy w macierz wynikową:** Z podmacierzy $C_{11}, C_{12}, C_{21}, C_{22}$ składana jest pełna macierz wynikowa C, przy użyciu funkcji `np.block`.

```python
def strassen_matmul(A: np.ndarray, B: np.ndarray) -> np.ndarray:
    if (n := A.shape[0]) == 1:
        counter["*"] += 1
        return A * B

    n //= 2

    A_11, A_12, A_21, A_22 = A[:n, :n], A[:n, n:], A[n:, :n], A[n:, n:]
    B_11, B_12, B_21, B_22 = B[:n, :n], B[:n, n:], B[n:, :n], B[n:, n:]

    P_1 = matmul(A_11 + A_22, B_11 + B_22)
    P_2 = matmul(A_21 + A_22, B_11)
    P_3 = matmul(A_11, B_12 - B_22)
    P_4 = matmul(A_22, B_21 - B_11)
    P_5 = matmul(A_11 + A_12, B_22)
    P_6 = matmul(A_21 - A_11, B_11 + B_12)
    P_7 = matmul(A_12 - A_22, B_21 + B_22)

    C_11 = P_1 + P_4 - P_5 + P_7
    C_12 = P_3 + P_5
    C_21 = P_2 + P_4
    C_22 = P_1 - P_2 + P_3 + P_6

    counter["+"] += 18 * n**2

    return np.block([[C_11, C_12], [C_21, C_22]])
```

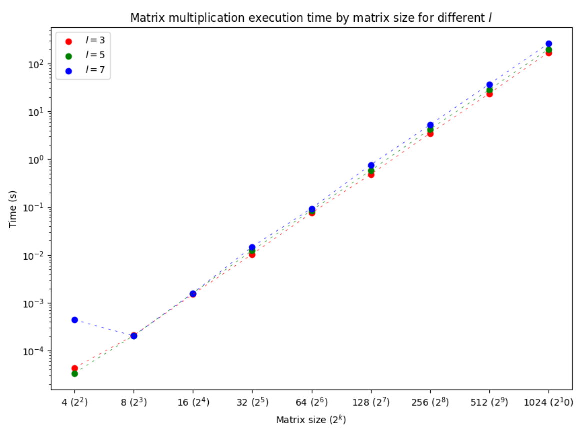
Rysunek 3: Kod mnożenia macierzy algortymem Strassena

2 Wykresy

2.1 Wykresy czasu mnożenia macierzy dla różnych l

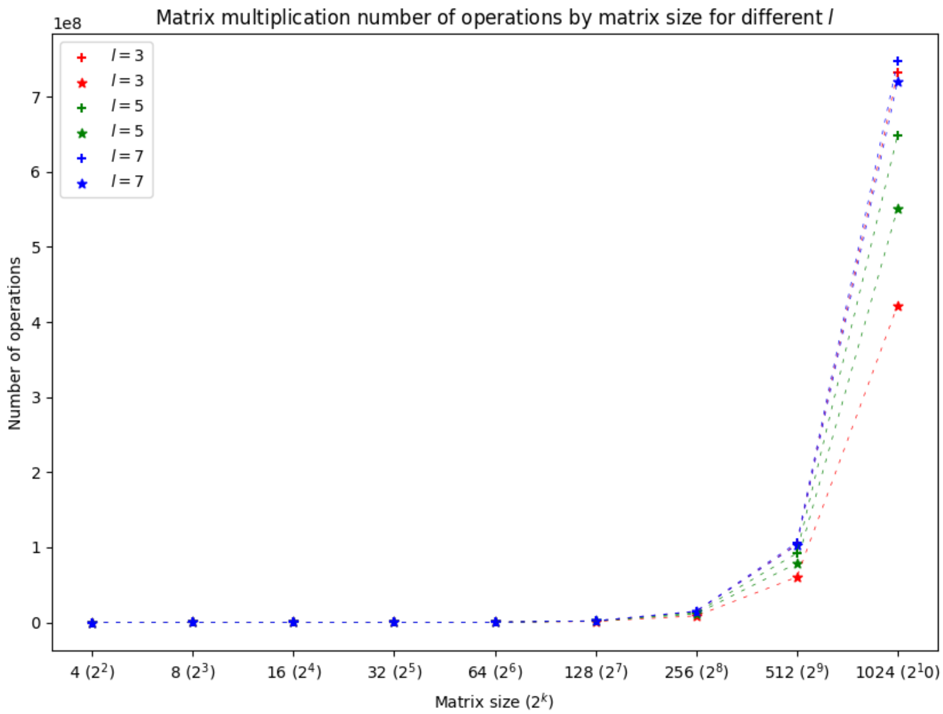


Rysunek 4: Wykres czasu mnożenia macierzy w zależności od rozmiaru macierzy.

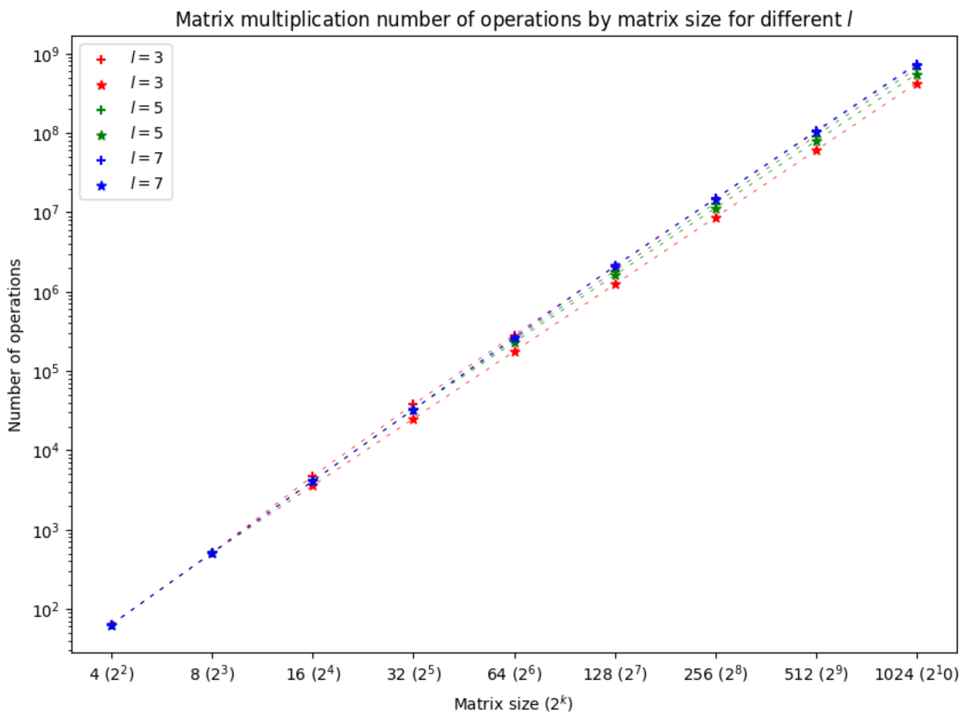


Rysunek 5: Wykres czasu w skali logarytmicznej w zależności od rozmiaru macierzy

2.2 Wykresy liczebności operacji zmiennoprzecinkowych dla różnych l



Rysunek 6: Wykres liczebności operacji zmiennoprzecinkowych w zależności od rozmiaru macierzy



Rysunek 7: Wykres liczebności operacji zmiennoprzecinkowych w zależności od rozmiaru macierzy