Algorytmy Macierzowe

Laboratorium nr. 1 - Rekurencyjne Mnożenie Macierzy

Artur Gęsiarz, Błażej Kapkowski

1. Cel Zadania:
   Celem zadania było zaimplementowanie rekurencyjnych algorytmów mnożenia macierzy oraz porównanie ich z tradycyjnymi metodami. W zadaniu zaimplementowane zostały trzy metody:
   a. Algorytm Binét'a - rekurencyjna metoda mnożenia macierzy.
   b. Algorytm Strassena - bardziej zoptymalizowana metoda rekurencyjna redukująca liczbę mnożeń.
   c. Algorytm AI (AlphaTensor) - oparty na wynikach badań opublikowanych w czasopiśmie Nature, gdzie sztuczna inteligencja opracowała nową, bardziej optymalną metodę mnożenia macierzy.

2. **Algorytm Binét'a:**
   Algorytm stosuje podejście rekurencyjne, które dzieli macierz na cztery równe podmacierze i rekurencyjnie mnoży je między sobą, zgodnie ze wzorem dla mnożenia blokowego. Złożoność obliczeniowa tego algorytmu wynosi O(n^3).

   Pseudokod:

```
binet(A, B):
        if A and B are 1x1 matrices:
                return A * B

        Divide A and B into submatrices:
        A11, A12, A21, A22 = submatrices of A
        B11, B12, B21, B22 = submatrices of B

        Calculate submatrices of result:
        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)

        Combine C11, C12, C21, C22 into one matrix C
        return C
```

3. **Algorytm Strassena**
   Algorytm redukuje liczbę mnożeń, wykorzystując zależności między podmacierzami. Zamiast tradycyjnych 8 mnożeń dla podmacierzy, algorytm Strassena wykorzystuje tylko 7. Złożoność tego algorytmu wynosi = , co oznacza, że jest on szybszy od klasycznego algorytmu dla dużych macierzy

Pseudokod:

```
strassen(A, B):
        Divide A and B into submatrices

        Calculate the 7 Strassen products:
        M1 = strassen(A11 + A22, B11 + B22)
        M2 = strassen(A21 + A22, B11)
        M3 = strassen(A11, B12 - B22)
        M4 = strassen(A22, B21 - B11)
        M5 = strassen(A11 + A12, B22)
        M6 = strassen(A21 - A11, B11 + B12)
        M7 = strassen(A12 - A22, B21 + B22)

        Calculate submatrices of result:
        C11 = M1 + M4 – M5 + M7
        C12 = M3 + M5
        C21 = M2 + M4
        C22 = M1 – M2 + M3 + M6


        Combine C11, C12, C21, C22 to get C
        return C
```

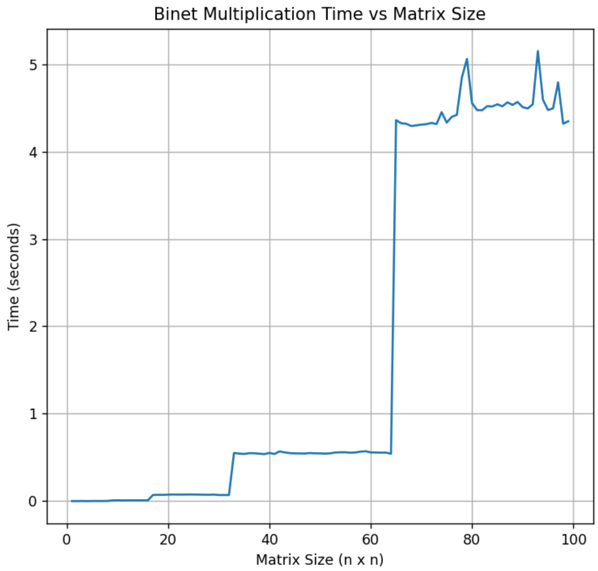
4. **Algorytm AI (AlphaTensor)**

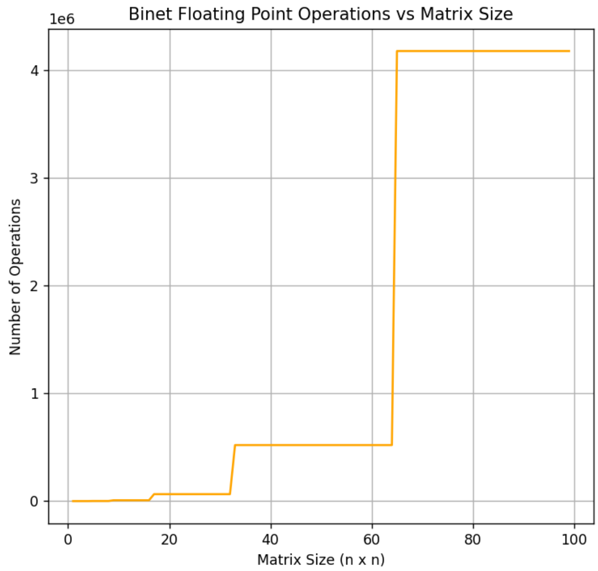
   Algorytm AI to wynik zastosowania sztucznej inteligencji w celu opracowania nowego, zoptymalizowanego sposobu mnożenia macierzy. Ten algorytm jest dostosowany do mnożenia macierzy 4x5 i 5x5. Oparty jest na zoptymalizowanych równaniach, które minimalizują liczbę operacji.

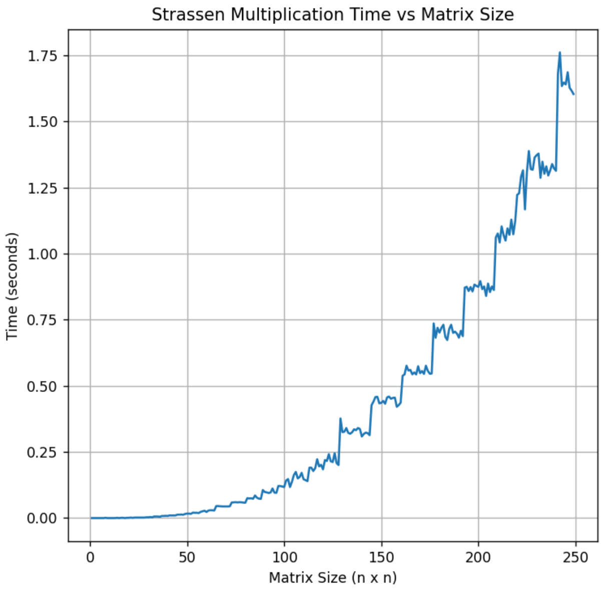
   Pseudokod:

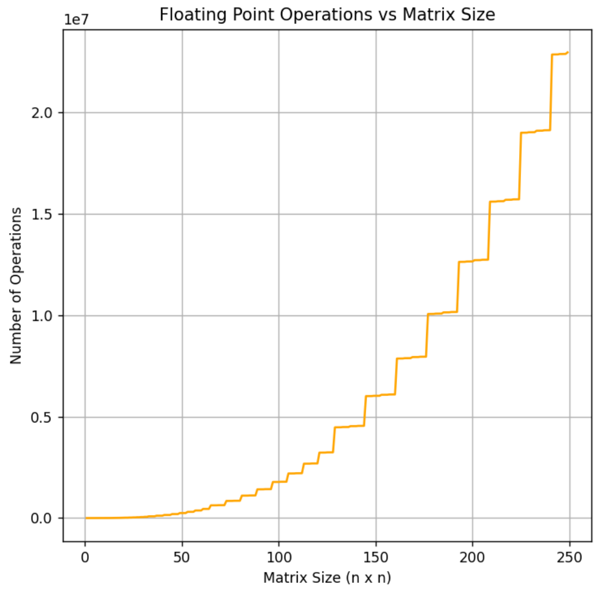
```
Compute constants h1 to h76 using elements of A and B
h1 = a32 * (-b21 - b25 - b31)
h2 = (a22 + a25 - a35) * (-b25 - b51)
h3 = (-a31 - a41 + a42) * (-b11 + b25)
...

Compute result matrix C using the constants
c11 = -h10 + h12 + h14 - h15 - h16 + h53 + h5 - h66 - h7
c21 = h10 + h11 - h12 + h13 + h15 + h16 - h17 - h44 + h51
...

return C
```

Wykresy:

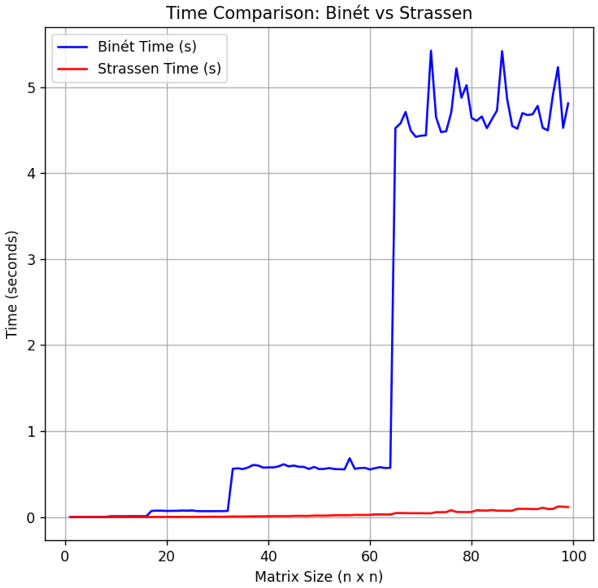

Binet Multiplication Time vs Matrix Size
Time (seconds)
Matrix Size (n x n)
0
1
2
3
4
5
0
20
40
60
80
100


Binet Floating Point Operations vs Matrix Size
1e6
Number of Operations
Matrix Size (n x n)
0
1
2
3
4
0
20
40
60
80
100

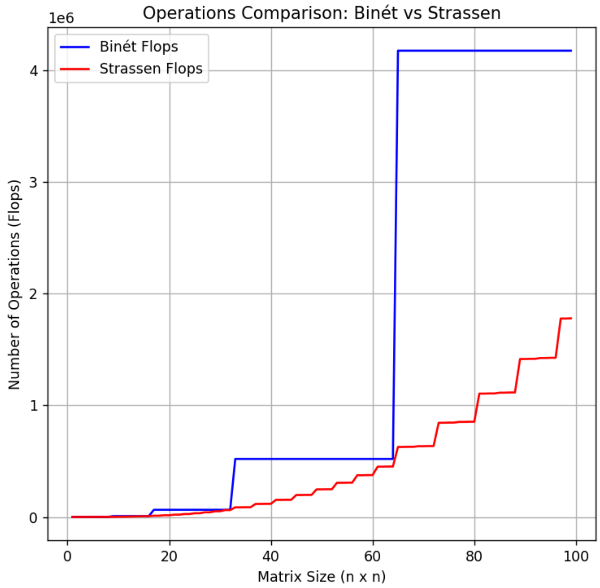

Strassen Multiplication Time vs Matrix Size
Time (seconds)
Matrix Size (n x n)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0
50
100
150
200
250


Floating Point Operations vs Matrix Size
1e7
Number of Operations
Matrix Size (n x n)
0.0
0.5
1.0
1.5
2.0
0
50
100
150
200
250

Porównanie:



Time Comparison: Binét vs Strassen
Binét Time (s)
Strassen Time (s)
Time (seconds)
Matrix Size (n x n)
0
1
2
3
4
5
0
20
40
60
80
100



Operations Comparison: Binét vs Strassen
1e6
Binét Flops
Strassen Flops
Number of Operations (Flops)
Matrix Size (n x n)
0
1
2
3
4
0
20
40
60
80
100

Referencje:

1. Wykład - Algortymy Macierzowe - Maciej Paszyński, prof. dr hab
2. Artykuł o Algortymie AI- https://deepmind.google/discover/blog/discovering-novel-algorithms-with-alphatensor/#:~:text=In%20our%20paper,%20published%20today%20in%20Nature,%20we