Szacowanie złożoności kodu oraz pseudokod algorytmu rekurencyjnego

pseudokod

podział na funkcje
- createMatrix (1)
- addMatrices (2)
- subMatrix (3)
- joinMatrices (4)
- recursiveMultiply (5)
- wrapper (6)

(1) Funkcja createMatrix (rows, columns, random)

Utwórz macierz rows x columns wypełnioną 0
Jeśli random = True:
  Do macierzy wpisz liczby z przedziału 0.00000001 do 1.0
zwróć macierz

(6) Funkcja wrapper (A, B, op_count)

A wymiarowo ≠ B -> błąd
op_count = 0
zwróć recursiveMultiply(A, B, op_count)

(2) Funkcja addMatrices

jeśli któraś pusta zwróć niepustą
jeśli niezgodne wielkości -> błąd
Utwórz macierz C o odpowiednich wymiarach
dla ∀ (i, j):
  C[i][j] = A[i][j] + B[i][j]
  zwiększ op_count
zwróć C

(5) Funkcja recursiveMultiply (A, B, op_count)

Jeśli odpowiednio małe mnożymy naiwnie
normalną metodą.

Podziel A i B na 4 części:
a11, a12, a21, a22
b11, b12, b21, b22

Rekurencyjnie policz:
c11_p1 = recursiveMultiply(a11, b11)
c11_p2 = recursiveMultiply(a12, b21)
c12_p1 = recursiveMultiply(a11, b12)
c12_p2 = recursiveMultiply(a12, b22)
c21_p1 = recursiveMultiply(a21, b11)
c21_p2 = recursiveMultiply(a22, b21)
c22_p1 = recursiveMultiply(a21, b12)
c22_p2 = recursiveMultiply(a22, b22)

Sumuj części:
c11 = addMatrices (c11_p1, c11_p2)
c12 = addMatrices (c12_p1, c12_p2)
c21 = addMatrices (c21_p1, c21_p2)
c22 = addMatrices (c22_p1, c22_p2)

połącz bloki w macierz C
zwróć C

(3) Funkcja subMatrix (M, r_s, r_e, c_s, c_e)

Nowa macierz sub o wymiarze (r_e - r_s, c_e - c_s)
Skopiuj odpowiedni fragment z M
zwróć sub

(4) Funkcja joinMatrices (C, c11, c12, c21, c22, m_split, p_split)

ustaw c11 w lewy górny róg C
ustaw c12 w prawy górny róg C
ustaw c21 w lewy dolny róg C
ustaw c22 w prawy dolny róg C

Złożoność

$O(n^3)$ bo to $T(n) = 8\,T\left(\frac{n}{2}\right) + O(n^2) = O(n^3)$

podział na 4 części i łączenie wyników (n^2)