Szacowanie złożoności kodu oraz pseudokod algorytmu rekurencyjnego

pseudokod
└ podział na funkcje:
- create Matrix (1)
- add Matrices (2)
- sub Matrix (3)
- join Matrices (4)
- recursiveMultiply (5)
- wrapper (6)

(1) Funkcja create Matrix (rows, columns, random)
Utwórz macierz rows x columns wypełnioną 0
Jeśli random = True
  Do macierzy wpisz liczby z przedziału 0.0000001 do 1.0
zwróć macierz

(3) Funkcja sub Matrix (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 join Matrices (C, c11, c12, c21, c22, m-split, j-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

(6) Funkcja wrapper (A, B, op-count)
A wymnożone ≠ B → błąd
op-count = 0
zwróć recursive Multiply (A, B, op-count)

(2) Funkcja add Matrices
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 recursive Multiply (A, B, op-count)

Jeśli odpowiednio małe wtedy mnożymy naiwną
naiwną metodą

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

Rekurencyjnie policz:

c11_p1 = recursive Multiply (a11, b11)

c11_p2 = recursive Multiply (a12, b21)

c12_p1 = recursive Multiply (a11, b12)

c12_p2 = recursive Multiply (a12, b22)

c21_p1 = recursive Multiply (a21, b11)

c21_p2 = recursive Multiply (a22, b21)

c22_p1 = recursive Multiply (a21, b12)

c22_p2 = recursive Multiply (a22, b22)

Sumy części:
  c11 = add Matrices (c11_p1, c11_p2)
  c12 = add Matrices (c12_p1, c12_p2)
  c21 = add Matrices (c21_p1, c21_p2)
  c22 = add Matrices (c22_p1, c22_p2)

połącz bloki w macierz C
zwróć C

Złożoność

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

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