**BÀI TẬP THỰC HÀNH 3**

***Mô tả bài toán***

Phép nhân đa thức là một phép tính cơ bản trong toán học. Giải thuật Strassen là một phương pháp hiệu quả để nhân hai đa thức có bậc nhỏ hơn hoặc bằng 3. Độ phức tạp thời gian của phương pháp Strassen là O(nlog27), nhanh hơn so với phương pháp thông thường O(n2).

***Dữ liệu đầu vào và dữ liệu đầu ra***

Dữ liệu đầu vào:

- A, B: hai đa thức cùng bậc n hoặc bậc nhỏ hơn hoặc bằng 3.

- n: bậc của đa thức

Dữ liệu đầu ra: C: đa thức tích của A và B

***Code Python***

def strassen\_poly(A, B):

n = len(A)

if n <= 3:

return naive\_poly\_mult(A, B)

# chia đa thức A và B thành các đa thức con bằng phép chia đôi

A0, A1 = split\_poly(A)

B0, B1 = split\_poly(B)

# tính các đa thức phụ trợ

P0 = strassen\_poly(A0, B0)

P1 = strassen\_poly(A1, B1)

P2 = strassen\_poly(add\_poly(A0, A1), add\_poly(B0, B1))

P3 = strassen\_poly(subtract\_poly(A0, A1), add\_poly(B0, B1))

P4 = strassen\_poly(add\_poly(A0, A1), B1)

P5 = strassen\_poly(A0, subtract\_poly(B1, B0))

P6 = strassen\_poly(A1, subtract\_poly(B0, B1))

# tính đa thức tích C

C0 = add\_poly(add\_poly(P0, P1), subtract\_poly(P4, P5))

C1 = add\_poly(P2, P3)

C2 = add\_poly(P4, P6)

C = [0] \* (2 \* n - 1)

for i in range(n):

C[i] = C0[i]

C[i + n] = C1[i]

C[i + 2 \* n] = C2[i]

return C

def add\_poly(A, B):

return [a + b for a, b in zip(A, B)]

def subtract\_poly(A, B):

return [a - b for a, b in zip(A, B)]

def split\_poly(P):

n = len(P) // 2

return P[:n], P[n:]

def naive\_poly\_mult(A, B):

n = len(A)

C = [0] \* (2 \* n - 1)

for i in range(n):

for j in range(n):

C[i + j] += A[i] \* B[j]

return C