TUẦN 5

Họ & Tên: Phạm Nguyễn Phương Duy
MSSV: 19110290

Bai 1

1/

[1]
```
1 import random
2 import numpy as np
```

```
1 a = int(input('Nhập cấp của ma trận vuông A: '))
2 b = int(input('Nhập cấp của ma trận vuông B: '))
```

```
Nhập cấp của ma trận vuông A: 2
Nhập cấp của ma trận vuông B: 2
```

[3]
```
1 rowsA, colsA = (a, a)
2 A = [[random.randint(1,1000) for i in range(colsA)] for j in range(rowsA)]
3 A = np.array(A)
4
5 rowsB, colsB = (b, b)
6 B = [[random.randint(1,1000) for i in range(colsB)] for j in range(rowsB)]
7 B = np.array(B)
```

[4]
```
1 print("Ma trận A: ",A, "\n\n", "Ma trận B: ",B)
```

```
Ma trận A:  [[977 431]
 [462  14]]

 Ma trận B:  [[758 157]
 [114 889]]
```

[5]
```
1 # Nhân 2 ma trận A, B theo cách thông thường:
2 C = A.dot(B)
3 print(C)
```

```
[[789700 536548]
 [351792  84980]]
```

2/ Dùng kỹ thuật **Chia để trị** để nhân 2 ma trận A, B với độ phức tạp $O(n^{log7})$

```python
def add(A, B):
    n = len(A)
    C = [[0 for j in range(0, n)] for i in range(0, n)]
    for i in range(0, n):
        for j in range(0, n):
            C[i][j] = A[i][j] + B[i][j]
    return C


def subtract(A, B):
    n = len(A)
    C = [[0 for j in range(0, n)] for i in range(0, n)]
    for i in range(0, n):
        for j in range(0, n):
            C[i][j] = A[i][j] - B[i][j]
    return C
```

```python
def multiply_matrix_level_2(A, B):
    n = len(A)
    C = [[0 for i in range(n)] for j in range(n)]
    for i in range(n):
        for k in range(n):
            for j in range(n):
                C[i][j] += A[i][k] * B[k][j]
    return C
```

```python
def strassenR(A, B):
    n = len(A)

    if n <= 2:
        return multiply_matrix_level_2(A, B)
    else:
        # initializing the new sub-matrices
        new_size = n // 2
        a11 = [[0 for j in range(0, new_size)] for i in range(0, new_size)]
        a12 = [[0 for j in range(0, new_size)] for i in range(0, new_size)]
        a21 = [[0 for j in range(0, new_size)] for i in range(0, new_size)]
        a22 = [[0 for j in range(0, new_size)] for i in range(0, new_size)]

        b11 = [[0 for j in range(0, new_size)] for i in range(0, new_size)]
        b12 = [[0 for j in range(0, new_size)] for i in range(0, new_size)]
        b21 = [[0 for j in range(0, new_size)] for i in range(0, new_size)]
        b22 = [[0 for j in range(0, new_size)] for i in range(0, new_size)]

        aResult = [[0 for j in range(0, new_size)] for i in range(0, new_size)]
        bResult = [[0 for j in range(0, new_size)] for i in range(0, new_size)]

        # dividing the matrices in 4 sub-matrices:
        for i in range(0, new_size):
            for j in range(0, new_size):
                a11[i][j] = A[i][j]  # top left
                a12[i][j] = A[i][j + new_size]  # top right
                a21[i][j] = A[i + new_size][j]  # bottom left
                a22[i][j] = A[i + new_size][j + new_size]  # bottom right
```

```
                a22[i][j] = A[i + new_size][j + new_size]  # bottom right

                b11[i][j] = B[i][j]  # top left
                b12[i][j] = B[i][j + new_size]  # top right
                b21[i][j] = B[i + new_size][j]  # bottom left
                b22[i][j] = B[i + new_size][j + new_size]  # bottom right

        # Calculating p1 to p7:
        aResult = add(a11, a22)
        bResult = add(b11, b22)
        p1 = strassenR(aResult, bResult)  # p1 = (a11+a22) * (b11+b22)

        aResult = add(a21, a22)  # a21 + a22
        p2 = strassenR(aResult, b11)  # p2 = (a21+a22) * (b11)

        bResult = subtract(b12, b22)  # b12 - b22
        p3 = strassenR(a11, bResult)  # p3 = (a11) * (b12 - b22)

        bResult = subtract(b21, b11)  # b21 - b11
        p4 = strassenR(a22, bResult)  # p4 = (a22) * (b21 - b11)

        aResult = add(a11, a12)  # a11 + a12
        p5 = strassenR(aResult, b22)  # p5 = (a11+a12) * (b22)

        aResult = subtract(a21, a11)  # a21 - a11
        bResult = add(b11, b12)  # b11 + b12
        p6 = strassenR(aResult, bResult)  # p6 = (a21-a11) * (b11+b12)

        aResult = subtract(a12, a22)  # a12 - a22
        bResult = add(b21, b22)  # b21 + b22
        p7 = strassenR(aResult, bResult)  # p7 = (a12-a22) * (b21+b22)
```

```
        p7 = strassenR(aResult, bResult)  # p7 = (a12-a22) * (b21+b22)

        # calculating c21, c21, c11 e c22:
        c12 = add(p3, p5)  # c12 = p3 + p5
        c21 = add(p2, p4)  # c21 = p2 + p4

        aResult = add(p1, p4)  # p1 + p4
        bResult = add(aResult, p7)  # p1 + p4 + p7
        c11 = subtract(bResult, p5)  # c11 = p1 + p4 - p5 + p7

        aResult = add(p1, p3)  # p1 + p3
        bResult = add(aResult, p6)  # p1 + p3 + p6
        c22 = subtract(bResult, p2)  # c22 = p1 + p3 - p2 + p6

        # Grouping the results obtained in a single matrix:
        C = [[0 for j in range(0, n)] for i in range(0, n)]
        for i in range(0, new_size):
            for j in range(0, new_size):
                C[i][j] = c11[i][j]
                C[i][j + new_size] = c12[i][j]
                C[i + new_size][j] = c21[i][j]
                C[i + new_size][j + new_size] = c22[i][j]
        return C
```