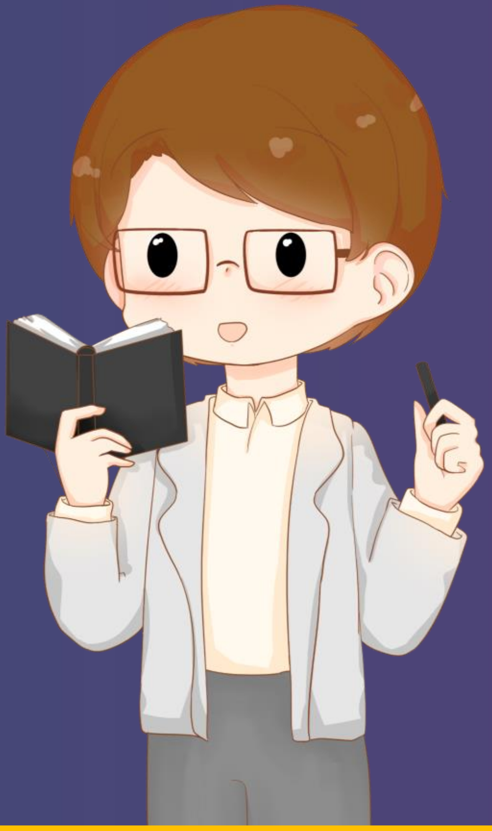
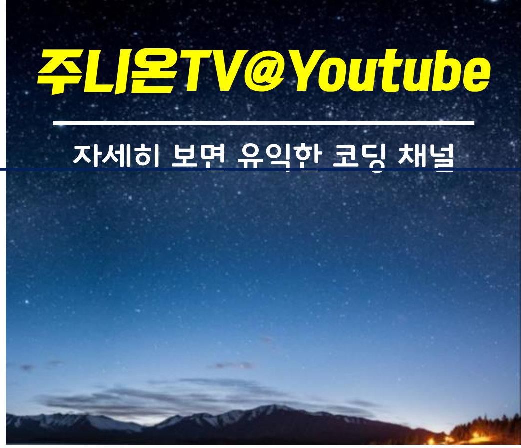
파이썬으로 배우는 알고리즘 기초

Chap 2. 분할정복

2.5

쉬트라센의 행렬 곱셈

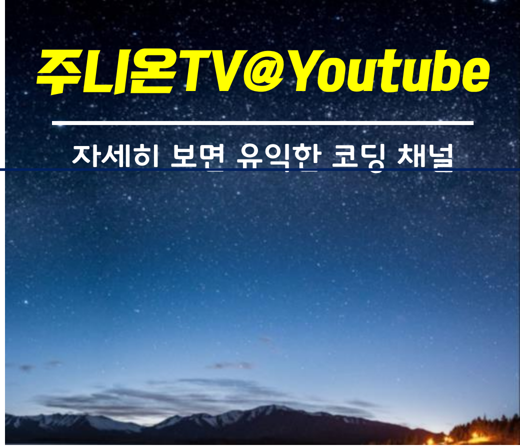




2.5 쉬트라쎈의 행렬 곱셈

행렬 곱셈 문제

- 문제: 두 $n \times n$ 행렬의 곱을 구하시오
- Algorithm 1.4
  - 행렬 곱셈의 정의에 충실하게. 시간 복잡도는 $\in \Theta(n^3)$
- 행렬 곱셈의 시간 복잡도를 더 줄일 수 있을까?
  - Strassen(1969)
- Algorithm 2.8
  - 쉬트라센의 방법을 사용. 시간 복잡도는 $\in \Theta(n^{2.81})$



▪ 쉬트라쎈의 방법

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

8 multiplications

4 additions

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$m_1 = (a_{11} + a_{22})(b_{11} + b_{22})$$
$$m_2 = (a_{21} + a_{22})b_{11}$$
$$m_3 = a_{11}(b_{12} - b_{22})$$
$$m_4 = a_{22}(b_{21} - b_{11})$$
$$m_5 = (a_{11} + a_{12})b_{22}$$
$$m_6 = (a_{21} - a_{11})(b_{11} + b_{12})$$
$$m_7 = (a_{12} - a_{22})(b_{21} + b_{22})$$

7 multiplications

18 additions/subtractions

$$C = \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

2.5 쉬트라센의 행렬 곱셈

- 쉬트라센의 방법: 분할 정복(Divide-and-Conquer)
  - 큰 행렬을 네 개의 부분 행렬로 나누어서 정복하라.

$$\left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \times \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right]$$

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$M_2 = (A_{21} + A_{22})B_{11}$$
$$\dots$$

$$C = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 + M_3 - M_2 + M_6 \end{bmatrix}$$

2.5 쉬트라센의 행렬 곱셈

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \left[\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ \hline 9 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \end{array}\right] \times \left[\begin{array}{cc|cc} 8 & 9 & 1 & 2 \\ 3 & 4 & 5 & 6 \\ \hline 7 & 8 & 9 & 1 \\ 2 & 3 & 4 & 5 \end{array}\right]$$

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22}) = \begin{bmatrix} 3 & 5 \\ 11 & 13 \end{bmatrix} \begin{bmatrix} 17 & 10 \\ 7 & 9 \end{bmatrix} = \begin{bmatrix} 86 & 75 \\ 278 & 227 \end{bmatrix}$$

$$M_2 =$$

$$M_3 =$$

$$M_4 =$$

$$M_5 =$$

$$M_6 =$$

$$M_7 =$$

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} \quad \end{bmatrix}$$

2.5 쉬트라센의 행렬 곱셈

Algorithm 2.8: Strassen's Matrix Multiplication

```
def strassen (A, B):
    n = len(A)
    if (n <= threshold):
        return matrixmult(A, B)
    A11, A12, A21, A22 = divide(A)
    B11, B12, B21, B22 = divide(B)
    M1 = strassen(madd(A11, A22), madd(B11, B22))
    M2 = strassen(madd(A21, A22), B11)
    M3 = strassen(A11, msub(B12, B22))
    M4 = strassen(A22, msub(B21, B11))
    M5 = strassen(madd(A11, A12), B22)
    M6 = strassen(msub(A21, A11), madd(B11, B12))
    M7 = strassen(msub(A12, A22), madd(B21, B22))
    return conquer(M1, M2, M3, M4, M5, M6, M7)
```

2.5 쉬트라쎈의 행렬 곱셈

주니온TV@Youtube

자세히 보면 유익한 코딩 채널

Algorithm 2.8: Strassen's Matrix Multiplication

```
def divide(A):
    n = len(A)
    m = n // 2
    A11 = [[0] * m for _ in range(m)]
    A12 = [[0] * m for _ in range(m)]
    A21 = [[0] * m for _ in range(m)]
    A22 = [[0] * m for _ in range(m)]
    for i in range(m):
        for j in range(m):
            A11[i][j] = A[i][j]
            A12[i][j] = A[i][j + m]
            A21[i][j] = A[i + m][j]
            A22[i][j] = A[i + m][j + m]
    return A11, A12, A21, A22
```

Algorithm 2.8: Strassen's Matrix Multiplication

```
def conquer(M1, M2, M3, M4, M5, M6, M7):
    C11 = madd(msub(madd(M1, M4), M5), M7)
    C12 = madd(M3, M5)
    C21 = madd(M2, M4)
    C22 = madd(msub(madd(M1, M3), M2), M6)
    m = len(C11)
    n = 2 * m
    C = [[0] * n for _ in range(n)]
    for i in range(m):
        for j in range(m):
            C[i][j] = C11[i][j]
            C[i][j + m] = C12[i][j]
            C[i + m][j] = C21[i][j]
            C[i + m][j + m] = C22[i][j]
    return C
```

Algorithm 2.8: Strassen's Matrix Multiplication

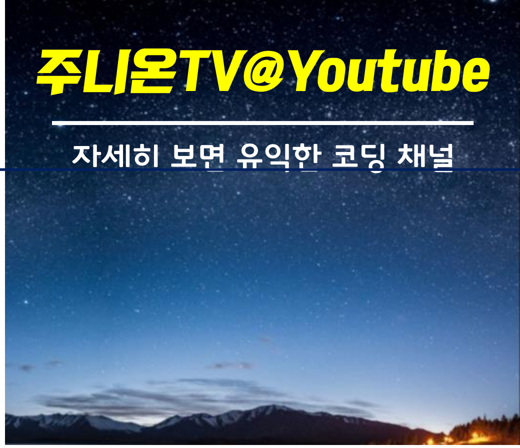
```
def madd (A, B):
    n = len(A)
    C = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            C[i][j] = A[i][j] + B[i][j]
    return C

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

2.5 쉬트라센의 행렬 곱셈

Algorithm 2.8: Strassen's Matrix Multiplication

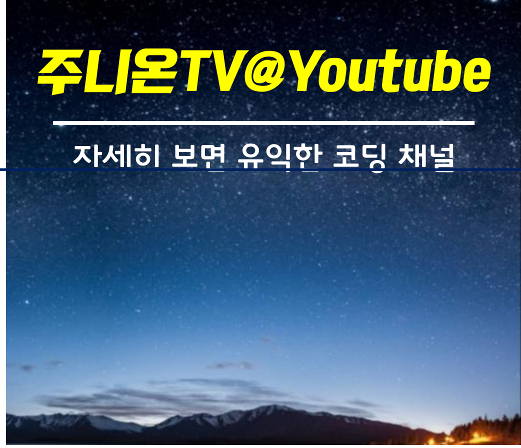
```python
# Algorithm 1.4: Matrix Muiltiplication
def matrixmult (A, B):
    n = len(A)
    C = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            for k in range(n):
                C[i][j] += A[i][k] * B[k][j]
    return C
```



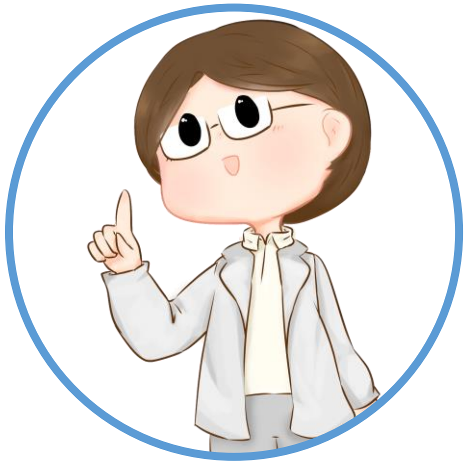

```
A = [                          B = [
    [1, 2, 3, 4],                  [8, 9, 1, 2],
    [5, 6, 7, 8],                  [3, 4, 5, 6],
    [9, 1, 2, 3],                  [7, 8, 9, 1],
    [4, 5, 6, 7],                  [2, 3, 4, 5],
]                              ]

threshold = 2
print('threshold =', threshold)
print('A =', A)
print('B =', B)
C = strassen(A, B)
for i in range(len(C)):
    print('C[%d] = '%(i), C[i])
```

2.5 쉬트라센의 행렬 곱셈



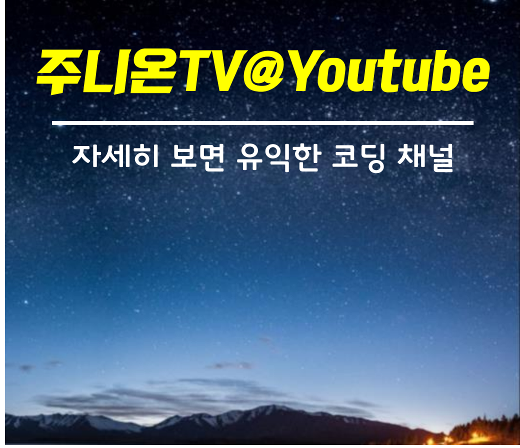

```
A = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 1, 2, 3], [4, 5, 6, 7]]
B = [[8, 9, 1, 2], [3, 4, 5, 6], [7, 8, 9, 1], [2, 3, 4, 5]]
C[0] =  [43, 53, 54, 37]
C[1] =  [123, 149, 130, 93]
C[2] =  [95, 110, 44, 41]
C[3] =  [103, 125, 111, 79]
```



주니온TV@Youtube

자세히 보면 유익한 코딩 채널

https://bit.ly/2JXXGqz



- 여러분의 구독과 좋아요는 강의제작에 큰 힘이 됩니다.
- 강의자료 및 소스코드: 구글 드라이브에서 다운로드 (다운로드 주소는 영상 하단 설명란 참고)

  https://bit.ly/3fN0q8t