MatrixMultiplication

November 26, 2021

1 Matrix Multiplication using Devide and Conquer

1.0.1 T(n) = 8T(n/2) + O(n^2) = O(n^3)

1.0.2 T(n) = 7T(n/2) + O(n^2) = O(N^2.8…)

[1]:

```
import numpy as np
```

[34]:

```
def Mul (A, B):

    if len(A) == 1:
        return A * B

    #split A to a, b, c, d
    arow, acol = A.shape
    a = A[:arow//2, :acol//2]
    b = A[:arow//2, acol//2:]
    c = A[arow//2:, :acol//2]
    d = A[arow//2:, acol//2:]

    #split B to e, f, g, h
    brow, bcol = B.shape
    e = B[:brow//2, :bcol//2]
    f = B[:brow//2, bcol//2:]
    g = B[brow//2:, :bcol//2]
    h = B[brow//2:, bcol//2:]

    #calculate p0, p1, ... p6
    p = []
    p.append(Mul(a, f - h))
    p.append(Mul(a+b, h))
    p.append(Mul(c + d, e))
    p.append(Mul(d, g - e))
    p.append(Mul(a + d, e + h))
    p.append(Mul(b - d, g + h))
    p.append(Mul(a - c, e + f))

    # c = x y
    #      z  k
    x = p[4] + p[3] - p[1] + p[5]
    y = p[0] + p[1]
    z = p[2] + p[3]
    k = p[0] + p[4] - p[2] - p[6]

    return np.vstack((np.hstack((x, y)), np.hstack((z, k))))
```

[47]:

```
A = np.array ([[1, 2],
               [3, 4]])

B = np.array ([[4, 4],
               [1, 2]])
print (Mul(A, B))
print ("###")
print(np.matmul(A, B))
```

```
[[ 6  8]
 [16 20]]
###
[[ 6  8]
 [16 20]]
```

[36]:

```
A = np.array ([[1, 0, 0, 0],
              [0, 1, 0, 0],
              [0, 0, 1, 0],
              [0, 0, 0, 1]])

B = np.array ([[100, 100, 100, 1],
              [5, 5, 5, 1],
               [1, 1, 1, 1],
              [2, 2, 2, 2]])

print (Mul(A, B))
print ("###")
print(np.matmul(A, B))
```

```
[[100 100 100   1]
 [  5   5   5   1]
 [  1   1   1   1]
 [  2   2   2   2]]
###
[[100 100 100   1]
 [  5   5   5   1]
 [  1   1   1   1]
 [  2   2   2   2]]
```