#Write Python program for implementing Strassen's Matrix multiplication using Divide and Conquer method. Discuss the complexity of algorithm.

import numpy as np

import time

start=time.time()

N=int(input('\nEnter the value of N for NxN matrix : '))

#checking condition

if N % 2 != 0:

print("Not Possible.")

else:

print('\nEnter ',N\*N,' elements of matrix A ')

A= [[0 for x in range(N)] for y in range(N)]

for i in range(0,N):

for j in range(0,N):

A[i][j]=int(input())

print('\nEnter ',N\*N,' elements of matrix B ')

B= [[0 for x in range(N)] for y in range(N)]

for i in range(0,N):

for j in range(0,N):

B[i][j]=int(input())

a= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

b= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

c= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

d= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

for i in range(0,int(N/2)):

for j in range(0,int(N/2)):

a[i][j]=A[i][j]

for i in range(0,int(N/2)):

for j in range(0,int(N/2)):

b[i][j]=A[i][j+int(N/2)]

for i in range(0,int(N/2)):

for j in range(0,int(N/2)):

c[i][j]=A[i+int(N/2)][j]

for i in range(0,int(N/2)):

for j in range(0,int(N/2)):

d[i][j]=A[i+int(N/2)][j+int(N/2)]

a=np.matrix(a);

b=np.matrix(b);

c=np.matrix(c);

d=np.matrix(d);

e= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

f= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

g= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

h= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

for i in range(0,int(N/2)):

for j in range(0,int(N/2)):

e[i][j]=B[i][j]

for i in range(0,int(N/2)):

for j in range(0,int(N/2)):

f[i][j]=B[i][j+int(N/2)]

for i in range(0,int(N/2)):

for j in range(0,int(N/2)):

g[i][j]=B[i+int(N/2)][j]

for i in range(0,int(N/2)):

for j in range(0,int(N/2)):

h[i][j]=B[i+int(N/2)][j+int(N/2)]

e=np.matrix(e);

f=np.matrix(f);

g=np.matrix(g);

h=np.matrix(h);

p1= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

p2= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

p3= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

p4= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

p5= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

p6= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

p7= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

p1=np.matmul(a,(f-h))

p2=np.matmul((a+b),h)

p3=np.matmul((c+d),e)

p4=np.matmul(d,(g-e))

p5=np.matmul((a+d),(e+h))

p6=np.matmul((b-d),(g+h))

p7=np.matmul((a-c),(e+f))

d1= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

d2= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

d3= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

d4= [[0 for x in range(int(N/2))] for y in range(int(N/2))]

d1=p5+p4-p2+p6

d2=p1+p2

d3=p3+p4

d4=p1+p5-p3-p7

d1=np.matrix(d1);

d2=np.matrix(d2);

d3=np.matrix(d3);

d4=np.matrix(d4);

C= [[0 for x in range(N)] for y in range(N)]

for i in range(0,int(N/2)):

for j in range(0,int(N/2)):

C[i][j]=d1.item(i,j)

for i in range(0,int(N/2)):

for j in range(int(N/2),N):

C[i][j]=d2.item(i,j-int(N/2))

for i in range(int(N/2),N):

for j in range(0,int(N/2)):

C[i][j]=d3.item(i-int(N/2),j)

for i in range(int(N/2),N):

for j in range(int(N/2),N):

C[i][j]=d4.item(i-int(N/2),j-int(N/2))

print("\nMatrix A")

print(np.matrix(A))

print("\nMatrix B")

print(np.matrix(B))

print("\nMatrix C")

print(np.matrix(C))

end=time.time()

diff=end-start

print("Time Complexity Of Algorithm: ", diff)