

T(2) =

T(2) =

T(4) =

T(4) = 4c + 4 =

T(8) =

T(8) =

T(16) =

T(16) =

T(n)

=

=

=

=

=

=

…

=

= 1 => k =

…

=]

…

]

]

# note

# note and

# note and

…

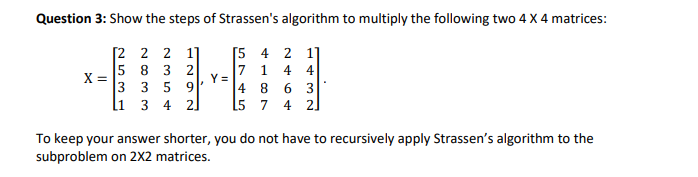


A = ac, B = bd, C = (a+b)(c+d) = (ac + ad + bc +bd)

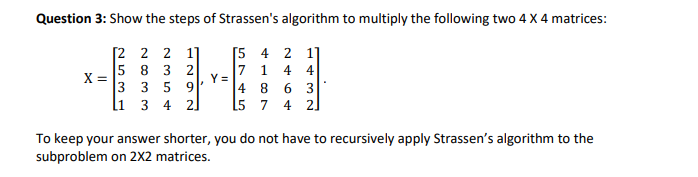
D = A – B = ac –bd

E = C – B – A = (ac + ad + bc +bd)– bd –ac = ad +bc

X x Y = D + iE = (ac-bd) + i(ad+bc)



* Divide each matrix into 4 parts. Let = top left matrix, = top right matrix, = bottom left matrix, and = bottom right matrix. Y is split into similarly



­

* According to Strassen’s algorithm,
* For a matrix X = [ a b ]  
   [ c d ]
* For a matrix Y = [ e f ]  
   [ g h ]
* P1= a \* ( f – h )
* P2 = ( a + b )\*h
* P3 = ( c + d )\*e
* P4 = d \* ( g – e )
* P5 = ( a + d ) \* ( e + h )
* P6 = ( b – d ) \* ( g + h )
* P7 = ( a – c ) \* ( e + f )
* X \* Y = Z
* Z = [P6 + P5 + P4 – P2] [P1 + P2]  
   [P3 + P4] [P1 + P5 – P7 – P3]
* Our result matrix will be Z
* Using Strassen’s algorithm to get Z,
* P1 = \* (
  + P1 = [2 2 ] \* ( [2 1] - [6 3] ) = [2 2] \* [-4 -2] = [-8 0]  
     [5 8 ] [4 4] [4 2] [5 8] [0 2] [-20 6]
* P2 = (\*
  + P2 = ( [2 2] + [2 1] ) \* [6 3 ] = [4 3] \* [6 3] = [36 18]  
     [5 8] [3 2] [4 2 ] [8 10] [4 2] [88 44]
* P3 = ( \*
  + P3 = ( [3 3] + [5 9] ) \* [5 4] = [8 12] \* [5 4] = [124 44]  
     [1 3] [4 2] [7 1 ] [5 5] [7 1] [60 25]
* P4 = \* (
  + P4 = [5 9 ] \* ( [4 8] - [5 4] ) = [5 9] \* [-1 4] = [-23 74]  
     [4 2 ] [5 7] [7 1] [4 2] [-2 6] [-8 28]
* P5 = \* (
  + P5 = ( [2 2] + [5 9] ) \* ( [5 4] + [6 3] ) = [7 11] \* [11 7] = [198 82]  
     [5 8] [4 2] [7 1] [4 2] [9 10] [11 3] [209 93]
* P6 = \* ()
  + P6 = ( [2 1] - [5 9] ) \* ( [4 8] + [6 3] ) = [-3 -8] \* [10 11] = [-102 -105]  
     [3 2] [4 2] [5 7] [4 2] [-1 0] [ 9 9] [ -10 -11]
* P7 = \* (
  + P7 = ( [2 2] - [3 3] ) \* ( [5 4] + [2 1] ) = [-1 -1] \* [7 5] = [-18 -10]  
     [5 8] [1 3] [7 1] [4 4] [4 5] [11 5] [ 83 45]
* = P6 + P5 + P4 – P2
  + = [-102 -105] + [198 82] + [-23 74] - [36 18] = [37 33]  
     [ -10 -11] [209 93] [-8 28] [88 44] = [103 66]
* = P1 + P2
  + = [-8 0] + [36 18] = [28 18]

[-20 6] [88 44] [68 50]

* = P3 + P4
  + = [124 44]+ [-23 74] = [101 118]

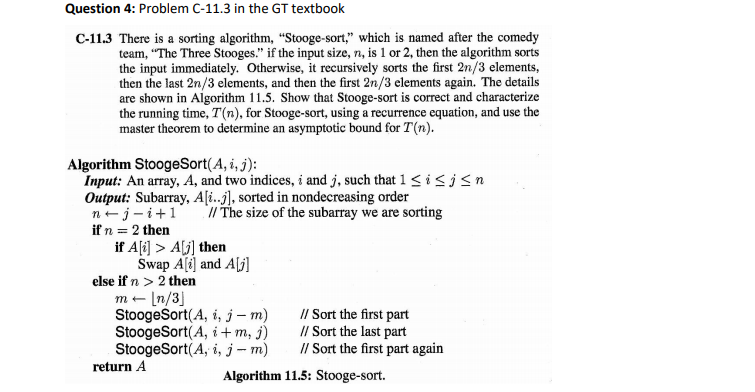
[60 25] [-8 28] [52 53]

* = P1 + P5 – P7 – P3
  + = [-8 0] + [198 82] - [-18 -10] - [124 44] = [84 48]  
     [-20 6] [209 93] [ 83 45] [60 25] = [46 29]
* Z = [ 37 33 28 18 ]

[ 103 66 68 50 ]

[ 101 118 84 48 ]

[ 52 53 46 29 ]



If n <= 2, the algorithm clearly works. If we call the first third of the array X, the second third Y, and the final third Z, then after sorting the first 2/3 of the array recursively, then we know that all elements in X are less than all elements in Y. After recursively sorting the final 2/3 elements, all the elements in Y are less than all the elements in Z. Since all the elements in Y were greater than the elements in X, then all the elements in Z are greater than all the elements in X. Therefore, the elements in Z are in the correct location. Then after sorting the first 2/3 of the array again, we know get the all the elements in X and Y are now in the correct location. Now we have a sorted array and we see that StoogeSort works.

, since StoogeSort is recursively called 3 times on 2/3 of the size of the original input.

Master’s Theorm:

f(n) = 1 =

c < 2.71, so T(n)