Emily Crilley

Analysis of Algorithms - Homework 3

1. See hw3.py
2. Calling this incorrect binary search with values such as a = [25,2,23,4,10,40] and v=10 would cause the code to run in an infinite loop. With each iteration, the h and l values are changing to m. Step 1: h=5, l=0 10>4 → search(a,v,m,h) Step 2: h=5, l=3, …. This continues and the code is unable to exit the while loop, as l is never greater than h, causing an infinite loop.
3.
    a.
    $S_1 =$ 8-2 = 6 $\quad P_1 = 1 * 6 = 6$
    $S_2 =$ 1+3 = 4 $\quad P_2 = 4 * 2 = 8$
    $S_3 =$ 7+5 = 12 $\quad P_3 = 12 * 6 = 72$
    $S_4 =$ 4-6 = -2 $\quad P_4 = 5 * -2 = -10$
    $S_5 =$ 1+5 = 6 $\quad P_5 = 6 * 8 = 48$
    $S_6 =$ 6+2 = 8 $\quad P_6 = -2 * 6 = -12$
    $S_7 =$ 3-5 = -2 $\quad P_7 = -6 * 14 = -84$
    $S_8 =$ 4+2 = 6
    $S_9 =$ 1-7 = -6
    $S_{10} =$ 6+8 = 14

    $$\begin{pmatrix} 18 & 14 \\ 62 & 66 \end{pmatrix}$$

    $C_{1,1} = 48 + (-10) - 8 + (-12) = 18$
    $C_{1,2} = 6 + 8 = 14$
    $C_{2,1} = 72 + (-10) = 62$
    $C_{2,2} = 48 + 6 - 72 - (-84) = 66$

    b. def strassen(A,B)
    n = A.rows
    if n == 1
        $C_{1,1} = A_{1,1}B_{1,1}$
    else
        $S_1 = B_{1,2} - B_{2,2}$
        $S_2 = A_{1,1} + A_{1,2}$
        $S_3 = A_{2,1} - A_{2,2}$
        $S_4 = B_{2,1} - B_{1,1}$
        $S_5 = A_{1,1} + A_{2,2}$
        $S_6 = B_{1,1} + B_{2,2}$

$S_7 = A_{1,2} - A_{2,2}$
$S_8 = B_{2,1} + B_{2,2}$
$S_9 = A_{1,1} - A_{2,1}$
$S_{10} = B_{1,1} + B_{1,2}$

$P_1 = strassen(A_{1,1}, S_1)$
$P_2 = strassen(S_2, B_{2,2})$
$P_3 = strassen(S_3, B_{1,1})$
$P_4 = strassen(A_{2,2}, S_4)$
$P_5 = strassen(S_5, S_6)$
$P_6 = strassen(S_7, S_8)$
$P_7 = strassen(S_9, S_{10})$

$C_{1,1} = P_4 + P_5 + P_6 - P_2$
$C_{1,2} = P_1 + P_2$
$C_{2,1} = P_3 + P_4$
$C_{2,2} = P_1 + P_5 - P_3 - P_7$
return ($C_{1,1}$, $C_{1,2}$ $C_{2,1}$ $C_{2,2}$)

c. $C_{2,1} = P_3 + P_4 = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$
$P_3 = S_3 B_{1,1}$
$P_4 = S_4 A_{2,2}$
$S_3 B_{1,1} + S_4 A_{2,2} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$
$S_3 = A_{2,1} + A_{2,2}$
$S_4 = B_{2,1} - B_{1,1}$
$(A_{2,1} + A_{2,2})B_{1,1} + (B_{2,1} - B_{1,1})A_{2,2} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$
$B_{1,1}A_{2,1} + B_{1,1}A_{2,2} + A_{2,2}B_{2,1} - A_{2,2}B_{1,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$
$A_{2,1}B_{1,1} + A_{2,2}B_{2,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$

d. $C_{2,2} = P_5 + P_1 - P_3 + P_7 = A_{2,1}B_{1,2} + A_{2,2}B_{2,2}$
$P_5 = S_5 S_6$
$P_1 = S_1 A_{1,1}$
$P_3 = S_3 B_{1,1}$
$P_7 = S_9 S_{10}$
$S_5 S_6 + S_1 A_{1,1} - S_3 B_{1,1} - S_9 S_{10}$

$S_5 = A_{1,1} + A_{2,2}$
$S_6 = B_{1,1} + B_{2,2}$
$S_1 = B_{1,2} - B_{2,2}$
$S_3 = A_{2,1} - A_{2,2}$
$S_9 = A_{1,1} - A_{2,1}$
$S_{10} = B_{1,1} + B_{1,2}$

$(A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2}) + A_{1,1}(B_{1,2} - B_{2,2}) - B_{1,1}(A_{2,1} - A_{2,2}) - (A_{1,1} - A_{2,1})(B_{1,1} + B_{1,2})$

Factor, Expand, and Simplify to get:

$A_{2,1}B_{1,2} + A_{2,2}B_{2,2} = A_{2,1}B_{1,2} + A_{2,2}B_{2,2}$

e. T(n) = $7T(\frac{n}{2}) + \frac{9}{2}n^2$

$7(\frac{7T(\frac{n}{2}) + \frac{9}{2}n^2}{2}) + \frac{9}{2}n^2$

$7(7\mathrm{T}(\frac{n}{2^2}) + \frac{9}{2^2}n^2) + \frac{9}{2}n^2$

$7^2\,\mathrm{T}(\frac{n}{2^2}) + \frac{9}{2^2}n^2) + \frac{9}{2}n^2$

$\frac{9}{2}n^2 + (\frac{7}{2^2})n^2 + 7^2T(\frac{9}{2^2}n)$

$\frac{9}{2}n^2 + (\frac{7}{2^2})n^2 + 7^2((\frac{9}{2^2})7T(\frac{n}{2}) + \frac{9}{2}n^2)$

$\frac{9}{2}n^2 + (\frac{7}{2^2})n^2 + (\frac{7}{2^2})n^2 + (\frac{7}{2^2})n^3 \ldots + 7^{\,log(2n) - log(2)}n^2 - 6n$

$\sum_{i=0}^{log(2n) - log(2)} (7)^i n^2 - 6n^2 = \Theta(n)^{\,log(2n)/log(2)} - 6n^2 = \mathrm{T(n)}$

$\mathrm{T}(2^m) = 7^{log(2(2^m))\,/\,log(2)} - 6(2^m)^2$

f. You would be able to apply strassen’s algorithm by using padding with 0s. You would have to assume $2^{k-1} < n < 2^k = m$, then you would be able to use strassen’s algorithm. The resulting algorithm would then be $\theta(2n)^{lg7} = \theta n^{lg7}$

g. Using three multiplications:

1= a*c
2= b*d
3= (a+b)*(c+d)

To produce real component: 1 - 2 = ac-bd
To produce imaginary component: 3 - 2 - 1 = ad+bc

4.

a. D(1) = 2 , D(n) = D (floor(n/2)) +1
D(floor(2/2)) +1 = floor(lg(2) + 2
D(1)+1 = 1+2
3=3
D(k) = D(floor(k/2) + 1

$D(n) = D(floor(n/2)) + 1$
$D(n) = floor(lg(n)) + 2$

b. $T(n) - T(1) = D(n-1)$
   $T(n) - T(1) = T(n-1+1) - T(n-1)$
   $T(n) - T(1) = T(n) - T(1)$
c. $T(n) = D(n)$, implying the time complexity is O(nlogn)

5. See hw3.py
6. See hw3.py