Intro to Algorithms: Homework #5

Due on March 11, 2021

Prof. Zaki

Jared Gridley

Jared Gridley HW5

2.27)

a) Compute square of 2x2 matrix with 5 multiplications

$$\begin{bmatrix} x & y \\ z & m \end{bmatrix} \times \begin{bmatrix} x & y \\ z & m \end{bmatrix} \qquad \begin{bmatrix} A & C \\ B & D \end{bmatrix}\begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$\begin{bmatrix} x^2 + yz & xy + ym \\ zx + z^2 & zy + zm \end{bmatrix} = \begin{bmatrix} \underset{1}{x^2} + \underset{2}{yz} & \underset{3}{y(x+m)} \\ \underset{4}{z(x+z)} & \underset{5}{z(y+m)} \end{bmatrix}$$

b) Straussen method but with 5 instead of 7 of the same size

① the math for that will only work if we have 3 of the same calculations.

$p_1 = A(F-H)$
$p_2 = A(B+H)$
$p_3 = (C+D)\cdot E$
$p_4 = D(G+E)$
$p_5 = (A+D)(E+H)$
$p_6 = (B-D)(G+H)$
$p_7 = (A-C)(E+F)$

7 multiplications of size $n/2$

$$\overset{X}{\begin{bmatrix} A & B \\ C & D \end{bmatrix}} \times \overset{Y}{\begin{bmatrix} E & F \\ G & H \end{bmatrix}} \Rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix}\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$P_1 = A(B-D)$
$P_2 = A(B+D)$
$P_3 = (C+D)A$
$P_4 = D(A+C)$
$P_5 = (A+D)(A+D)$
$P_6 = (B-D)(C-D)$
$P_7 = (A-C)(A+B)$

we can see that this does not reduce it at all, it is still 7 sub-problems. The only way to reduce the Straussen method equations would be if certain elements were equal, ex: B = C = D.

Since it still consists of the 7 subproblems, the runtime is still $O(n^{\log_2 7})$

Figure 1: Page 1

c) Squaring an nxn matrix is the same as multiplying by itself $(n \times n)(n \times n)$.

Squaring
$$\begin{bmatrix} x & y \\ z & w \end{bmatrix}^2 = \begin{bmatrix} x & y \\ z & w \end{bmatrix}\begin{bmatrix} x & y \\ z & w \end{bmatrix} = \begin{bmatrix} x^2 + yz & yx + yw \\ zx + zw & zy + zw \end{bmatrix}$$

Multiplying
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

Only difference is that squaring can be simplified because of common values, but both will follow the same Algorithm, Squaring will be slightly faster if they simplify multiplications, but otherwise same Big-Oh:

<u>Algorithm 1 (A,B):</u>

```
for i in 1...n:
    for j in 1...n:
        //C[i,j] = A[i:] • B[:j]
        for k in 1...m:
            C[i,j] = C[i,j] + A[i,k] • B[k,j]
```

$$C[i,j] = C[i,j] \underset{O(1)}{+} \underbrace{A[i,k] \cdot B[k,j]}_{O(1)}$$

Machine Instr. 64-bits

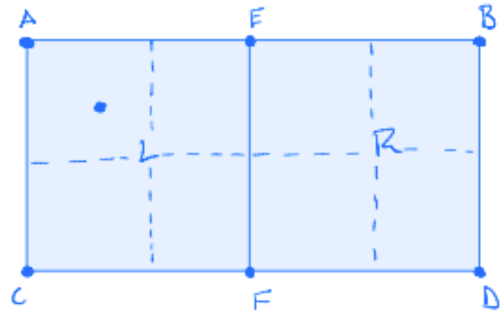
$$O(n \cdot n \cdot n) = O(n^3)$$

Figure 2: Page 2

2.32)

a) Prove only max of 4 points in L.

If we assume that there are more than 4 points in L. Let's say we have 5 points in L.



We know that in order for L to be a square, All 4 of its points must be the same distance apart. However, if there is a 5^{th} point inside of L, then it will create a pair of points with a shorter distance than the squares edge. This contradicts the assumption of a $d \times d$ square.

b) If we have $p_1 \in L$ and $p_2 \in R$, then for the algorithm to be correct, it must test these 2 points' distance. So since we have already calculated a distance d, that is the min between 2 points. So we can cap off the points we test as those within $x-d$ and $x+d$ as well as in $y-d$ and $y+d$. We can split this middle section into L and R. Both of size $d \times d$. We know that from part a that each of the squares contain at most 4 points (including the pair we are looking for, (p_{sL}, p_{sR})). Since the algorithm will check the distance between P_{sL} and the 7 other points in L and R, it will be tested with P_{sR} to get the smallest distance, so the algorithm will be correct.

If the smallest pair points are Both in L or R then when it initially splits and tests it, it will test with both of the points to get the smallest distance. So it will be correct.

Figure 3: Page 3

c)

Closest Pair (Pts):

Input: Set of points in a space

Output: Pair of closest points

1. Find the median x value, x_m — $O(n)$
2. Split points at median value in L and R sets — $O(n)$

   $L = x_i < x_m$

   $R = x_i \geq x_m$
3. Recursively split until length 1 ← Recursive $2T(\frac{n}{2})$

   Base case: len (Pts) ≤ 1 : return Array

   p_L = Closest Pair (L)

   p_R = closest Pair (R)

   d = min (distance (p_L) , distance (p_R))
4. Split middle section by points in $[x_m - d, x_m + d]$, sort by y-coordinate ← $O(n\log n)$ - sorting

   - Compute distances between point, check against d for smaller value $O(1) \times O(n)$

return d;

$$T(n) = 2T(\tfrac{n}{2}) + O(3n\log n)$$
$$= 2T(\tfrac{n}{2}) + O(n\log n)$$

↓ Master's Theorem

$$O(n\log^2 n)$$

3) $A = [2, 5, 4, 2, 0, 0, 0, 0]$, $\omega = \frac{1}{\sqrt{2}}(1+i)$

FFT Algo: (a, ω)

if ω = 1: return a

$s[\,] = FFT((a_0, a_2, \ldots, a_{n-2}), \omega^2)$

$s'[\,] = FFT((a_1, a_3, \ldots, a_{n-1}), \omega^2)$

for j in range(0, n/2 - 1):

$r_j = s_j + \omega^j s'_j$

$r_{j+\frac{n}{2}} = s_j - \omega^j s'_j$

$S = [2, 4, 0, 0]$ $S' = [5, 2, 0, 0]$ $\omega = \frac{1}{\sqrt{2}}(1+i)$, $\omega^2 = i$

$\boxed{r = [13,\ 4+4i,\ -2+3i,\ i,\ -1,\ -i,\ -2-3i,\ 4-4i]}$

FFT([2,4,0,0], i)

$S_1 = [2, 0]$ $S'_1 = [4, 0]$ $\omega = i$, $\omega^2 = -1$

n = 4, range(0, 1)

$r = [6, 2+4i, -2, 2-4i]$

s[0] + 1 s'[0] = 6

s[0] - 1 s'[0] = -2

s[1] + i s'[1] = 2 + 4i

FFT([2,0], -1)

$S_2 = [2]$, $S'_2 = [0]$, $\underbrace{\omega^2 = 1}_{\text{will trigger return on next loop}}$

$r = [2, 2]$

FFT([4,0], -1)

$S_3 = [4]$ $S'_3 = [0]$ $\underbrace{\omega^2 = 1}_{\text{Trigger Return}}$

$r = [4, 4]$

FFT([5,2,0,0], i)

$S_4 = [5, 0]$ $S'_4 = [2, 0]$ $\omega = i$, $\omega^2 = -1$

$r = [7, 5+2i, 3, 5-2i]$

FFT([5,0], -1)

$S_5 = [5]$ $S'_5 = [0]$ $\underbrace{\omega^2 = 1}_{\text{Triggers Return}}$

$r = [5, 5]$

FFT([2,0], -1)

$S_6 = [2]$ $S'_6 = [0]$ $\underbrace{\omega^2 = 1}_{\text{Triggers Return}}$

$r = [2, 2]$

Figure 4: Page 4

Lab Results:
Coefficients of A (Alg1):
[5, 5, 7, 6, 9, 9, 3, 7, 7, 0, 7, 6, 3, 3, 6, 5, 9, 8, 0, 6, 1, 4, 3, 5, 6, 5, 7, 7, 6, 5, 2, 2, 1, 9, 2, 7, 8, 0, 5, 6, 6, 2, 0, 3, 7, 4, 5, 9, 2, 4, 8, 1, 3, 3, 3, 0, 5, 9, 9, 8, 6, 7, 3, 4, 0, 3, 1, 4, 9, 4, 4, 2, 9, 3, 3, 6, 1, 3, 6, 3, 5, 7, 9, 5, 2, 2, 6, 4, 1, 1, 2, 6, 1, 9, 0, 6, 3, 4, 1, 9]
Coefficients of B (Alg1):
[4, 5, 0, 7, 4, 5, 8, 9, 2, 9, 4, 7, 2, 8, 7, 6, 9, 3, 0, 7, 7, 5, 7, 6, 5, 1, 5, 9, 8, 8, 7, 0, 3, 1, 4, 1, 8, 7, 6, 8, 0, 8, 3, 4, 8, 5, 1, 2, 8, 5, 5, 4, 1, 4, 3, 8, 5, 3, 5, 7, 1, 6, 0, 8, 8, 7, 5, 1, 6, 8, 0, 2, 4, 1, 5, 8, 4, 2, 9, 2, 2, 9, 4, 0, 3, 3, 2, 7, 7, 1, 8, 7, 8, 8, 6, 4, 3, 7, 5, 7]
Coefficients of A (FFT):
[0, 0, 1, 3, 1, 7, 1, 0, 0, 5, 1, 2, 7, 5, 4, 3, 7, 0, 7, 0, 9, 9, 4, 6, 3, 5, 9, 0, 9, 6, 0, 4, 4, 0, 3, 0, 7, 4, 5, 0, 2, 4, 8, 4, 7, 2, 4, 3, 6, 0, 3, 0, 2, 3, 5, 9, 0, 0, 9, 9, 1, 4, 2, 0, 4, 6, 7, 4, 5, 4, 7, 0, 7, 6, 1, 3, 2, 4, 5, 5, 1, 3, 1, 1, 6, 0, 0, 9, 2, 6, 2, 3, 5, 9, 3, 3, 6, 3, 6, 5]
Coefficients of B (FFT):
[6, 2, 4, 3, 9, 9, 8, 6, 8, 4, 9, 4, 5, 2, 8, 6, 0, 3, 1, 3, 8, 4, 9, 5, 1, 0, 6, 2, 3, 4, 7, 1, 6, 3, 9, 9, 3, 6, 4, 5, 2, 0, 8, 3, 0, 3, 0, 1, 3, 6, 8, 0, 2, 9, 6, 6, 8, 0, 2, 7, 2, 3, 0, 5, 4, 0, 8, 4, 3, 1, 3, 4, 6, 6, 9, 4, 7, 8, 2, 0, 6, 4, 5, 0, 9, 9, 7, 0, 1, 0, 8, 6, 9, 4, 2, 6, 0, 5, 5, 7]

Results for d = 100

Alg1: [20, 45, 53, 94, 121, 175, 192, 250, 303, 303, 339, 434, 355, 416, 482, 466, 522, 603, 538, 577, 628, 638, 634, 695, 738, 677, 760, 797, 808, 830, 972, 905, 943, 923, 921, 878, 966, 950, 998, 1075, 1083, 1067, 1118, 1137, 1189, 1141, 1194, 1160, 1143, 1284, 1219, 1247, 1294, 1377, 1277, 1415, 1332, 1284, 1352, 1378, 1500, 1447, 1460, 1649, 1529, 1624, 1660, 1599, 1596, 1657, 1618, 1633, 1774, 1695, 1776, 1687, 1748, 1777, 1710, 1822, 1816, 1841, 1901, 1931, 1893, 1938, 1963, 1863, 2109, 1993, 1837, 2081, 2052, 1944, 2179, 2229, 2120, 2251, 2258, 2158, 2160, 2139, 2337, 1984, 2196, 2014, 1985, 1968, 2001, 1989, 1860, 1857, 1757, 1850, 1816, 1755, 1784, 1603, 1655, 1652, 1639, 1659, 1648, 1634, 1459, 1620, 1543, 1470, 1394, 1280, 1253, 1401, 1348, 1248, 1330, 1306, 1302, 1365, 1273, 1116, 1162, 1171, 1046, 1129, 1104, 991, 1120, 1074, 1006, 1051, 1078, 944, 916, 876, 925, 886, 967, 804, 832, 795, 832, 677, 783, 662, 663, 674, 583, 646, 638, 569, 530, 616, 526, 560, 598, 472, 425, 491, 408, 393, 430, 367, 245, 332, 294, 300, 288, 267, 215, 249, 255, 186, 208, 121, 124, 75, 96, 52, 63]

FFT: [6, 20, 16, 59, 42, 69, 69, 135, 122, 133, 144, 197, 184, 222, 254, 279, 301, 312, 387, 334, 392, 418, 462, 503, 588, 507, 583, 521, 564, 611, 622, 621, 614, 682, 645, 577, 678, 589, 713, 651, 771, 748, 744, 657, 861, 795, 948, 744, 884, 866, 877, 777, 883, 949, 858, 833, 1096, 942, 967, 906, 1007, 1086, 1042, 1064, 1062, 934, 1163, 1053, 1178, 1140, 1114, 1176, 1216, 1230, 1353, 1259, 1337, 1389, 1298, 1375, 1292, 1124, 1351, 1280, 1252, 1476, 1299, 1437, 1376, 1348, 1550, 1499, 1517, 1604, 1587, 1732, 1701, 1535, 1696, 1649, 1718, 1656, 1688, 1461, 1505, 1565, 1623, 1424, 1473, 1475, 1404, 1503, 1461, 1348, 1261, 1248, 1446, 1354, 1288, 1357, 1303, 1136, 1132, 1121, 1239, 1174, 1233, 1123, 1010, 1103, 1142, 1041, 1019, 1013, 982, 878, 905, 968, 991, 898, 884, 991, 825, 926, 899, 845, 798, 831, 901, 837, 849, 791, 756, 659, 686, 731, 702, 645, 619, 616, 606, 677, 680, 647, 498, 557, 529, 604, 518, 517, 476, 485, 421, 380, 359, 389, 366, 393, 294, 294, 321, 326, 313, 254, 266, 206, 259, 205, 217, 206, 143, 112, 96, 87, 76, 67, 35]

Runtimes:
Alg1 Runtimes: 100: 0.00187, 1000: 0.15106, 10000: 15.25722
FFT Runtimes: 100: 0.00579, 1000: 0.0481, 10000: 0.95342