Algorithms - Ex7 Solutions

1 Question 1:

1. We want to calculate $(a + bi)(c + di)$. We need $(ac - bd)$ and $(ad + bc)$. Calculate $m_1 = (a + b)(c + d)$, $m_2 = ac$, $m_3 = bd$. So $ac - bd = m_2 - m_3$ and $ad + bc = m_1 - m_2 - m_3$, and three multiplications suffice.

2. We have to n-digit numbers a, b. Suppose n is even, and write $a = a_1 + a_2k$ and $b = b_1 + b_2k$ where a_1, a_2, b_1, b_2 are $n/2$-digit numbers, and $k = 10^{n/2}$. Now $ab = (a_1 + a_2k)(b_1 + b_2k) = (a_1b_1) + (a_2b_1 + a_1b_2)k + (a_2b_2)k^2$. Seeing the similarity to the previous problem, we shall calculate $m_1 = (a_1 + a_2)(b_1 + b_2)$, $m_2 = a_2b_2$, $m_3 = a_1b_1$. Now $a_2b_1 + a_1b_2 = m_1 - m_2 - m_3$, so we have reduced the multiplication of two n-digit numbers to three multiplications of $n/2$-digit numbers, followed by some additions of n-digit numbers ($O(n)$ operations), and also multiplication of an n-digit number by k of k^2 (also $O(n)$ operations, since multiplying by k or k^2 is just a shift left by $n/2$ and n digits respectively).

   Write $f(n)$ as the number of operations you need in order to multiply two n-digit numbers. We have shown that $f(n) = 3f(n/2) + O(n)$ Therefore, $f(n) = O(n^{log_2 3})$ instead of $O(n^2)$ of the "school method" algorithm. The best known algorithms uses the FFT, and takes around $n \log^3 n$ in the algorithm we saw. This can be improved to almost $n \log n$ in a more careful design.

   We are cheating a bit in two places: We tacitly assume that n is a power of two so our idea will work all the way down the recursion, and we assume that $a + b$ is an $n/2$-digit number when it is actually an $n/2 + 1$ digit number, but these two problems are only technical.

2 Question 2:

Here is an algorithm for computing the determinant, when we have a black-box that calculates the LUP decomposition.

We are given a square matrix A. Run the LUP decomposition algorithm to get a representation $A = L \cdot U \cdot P$. Where U is a lower triangular matrix, U is an upper triangular matrix, and P is a permutation matrix. Notice that the determinant satisfies the following properties:

- $det(A \cdot B) = det(A) \cdot det(B)$
- For a lower diagonal matrix the determinant is just the multiple of the elements on the diagonal.
- Same for upper diagonal.
- The determinant of a permutation matrix is easy to calculate (just to the development of the determinant by the first row. Then you get only one determinant of an $(n - 1) \times (n - 1)$ permutation matrix to calculate. So this takes $O(n^2)$ operations.

So calculate the three determinants in time $O(n^2)$, and multiply the three results to get $det(A)$. So

$$D(n) \leq L(n) + O(n^2)$$

Since $L(n) = \Omega(n^2)$ because we have to at least look at all the matrix elements and there are n^2 of those, this gives the required result.

3 Question 3:

We want to find a vector $c = (c_1, c_2, c_3)$ such that

$$c_1 + c_2 x \log x + c_3 e^x$$

Is the best least-squares approximation the the points

$$(1,1), (2,1), (3,3), (4,8)$$

Define a matrix

$$A = \begin{pmatrix} 1 & 1\log 1 & e^1 \\ 1 & 2\log 2 & e^2 \\ 1 & 3\log 3 & e^3 \\ 1 & 4\log 4 & e^4 \end{pmatrix}$$

For a vector $c = (c_1, c_2, c_3)$ define

$$(y_1, y_2, y_3, y_4) = (A \cdot c)^T.$$

Then our goal is to find the vector c such that

$$(y_1 - 1)^2 + (y_2 - 1)^2 + (y_3 - 3)^2 + (y_4 - 8)^2$$

Is minimal. As you have learned in class, the solution for c is:

$$c = (A^T A)^{-1} \cdot A^T \cdot y$$

And using matlab to do this calculation gives

$$c = (0.4117, -0.2049, 0.1695)$$

The sum of squares is 0.1297. To check that we worked correctly, we note that for our solution c it should be true that the vector $y - Ac$ is perpendicular to any vector Ax, since the best c is such that Ac is the perpendicular projection of y to the subspace $\{Ax : x \in \mathbb{R}^3\}$.
For example, calculate this for $x = (1, 2, 3)$. Then

$$Ax = (9.1548, 27.1672, 70.7664, 180.7945)$$

and

$$y - Ac = (0.1274, -0.2548, 0.1570, -0.0296)$$

You can check that $< y - Ac, Ax >= 0$ as required.

4 Question 4:

We use the fact that the l_1 and l_∞ norms are dual, thus for every vector x, $\|Ax\|_1 = \max_{\|y\|_\infty \le 1} < y, Ax >$. So we have:

$$\max_{\|x\|_\infty \le 1} \|Ax\|_1 = \max_{\|x\|_\infty \le 1} \max_{\|y\|_\infty \le 1} < y, Ax >= \max_{\|x\|_\infty, \|y\|_\infty \le 1} y^t Ax$$