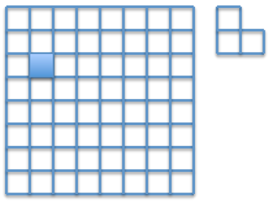
Algorithms Tutorial Problems 2
Divide-and-Conquer, FFT and Convolution

1. You are given a $2^n \times 2^n$ board with one of its cells missing (i.e., the board has a hole); the position of the missing cell can be arbitrary. You are also given a supply of “dominoes” each containing 3 such squares; see the figure:
 Your task is to design an algorithm which covers the entire board with such



 “dominoes” except for the hole.

2. You and a friend find yourselves on a TV game show! The game works as follows. There is a **hidden** $N \times N$ table A. Each cell $A[i, j]$ of the table contains a single integer and no two cells contain the same value. At any point in time, you may ask the value of a single cell to be revealed. To win the big prize, you need to find the N cells each containing the **maximum** value in its row. Formally, you need to produce an array $M[1..N]$ so that $A[r, M[r]]$ contains the maximum value in row r of A, i.e., such that $A[r, M[r]]$ is the largest integer among $A[r, 1], A[r, 2], \ldots, A[r, N]$. In addition, to win, you should ask at most $\mathbf{O}(\mathbf{N} \log \mathbf{N})$ many questions. For example, if the hidden grid looks like this:

	Column 1	Column 2	Column 3	Column 4
Row 1	**10**	5	8	3
Row 2	1	**9**	7	6
Row 3	-3	**4**	-1	0
Row 4	-10	-9	-8	**2**

then the correct output would be $M = [1, 2, 2, 4]$.
Your friend has just had a go, and sadly failed to win the prize because they asked N^2 many questions which is too many. However, they whisper to you a hint: they found out that M is **non-decreasing**. Formally, they tell you that $M[1] \leq M[2] \leq \cdots \leq M[N]$ (this is the case in the example above).
Design an algorithm which asks at most $\mathbf{O(N \log N)}$ many questions that produces the array M correctly, even in the very worst case.

Hint: Note that you do not have enough time to find out the value of every cell in the grid! Try determining $M[N/2]$ first, and see if divide-and-conquer is of any assistance.

3. Let us define the Fibonacci numbers as $F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$. Thus, the Fibonacci sequence looks as follows: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

   (a) Show, by induction or otherwise, that

   $$\begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$$

   for all integers $n \geq 1$.

   (b) Hence or otherwise, give an algorithm that finds F_n in $O(\log n)$ time.

   Hint: You may wish to refer to Example 1.5 on page 28 of the "Review Material" found as lecture notes for Topic 0 on the Course Resources page of the course website.

4. Design an algorithm which multiplies a polynomial of degree 16 with a polynomial of degree 8 using only 25 multiplications in which both operands (which both depend on the coefficients of the polynomial) can be arbitrarily large.

5. Multiply the following pairs of polynomials using at most the prescribed number of multiplications of large numbers (large numbers are those which depend on the coefficients and thus can be arbitrarily large).

   (a) $P(x) = a_0 + a_2x^2 + a_4x^4 + a_6x^6$; $Q(x) = b_0 + b_2x^2 + b_4x^4 + b_6x^6$ using at most 7 multiplications of large numbers;

   (b) $P(x) = a_0 + a_{100}x^{100}$ and $Q(x) = b_0 + b_{100}x^{100}$ with at most 3 multiplications of large numbers.

6. (a) Multiply two complex numbers $(a + i\,b)$ and $(c + i\,d)$ (where a, b, c, d are all real numbers) using only 3 real number multiplications.

   (a) Find $(a + i\,b)^2$ using only two multiplications of real numbers.

   (b) Find the product $(a + i\,b)^2(c + i\,d)^2$ using only five real number multiplications.

7. Describe all k which satisfy $i\,\omega_{64}^{13}\omega_{32}^{11} = \omega_{64}^{k}$ (i is the imaginary unit).

8. What are the real and the imaginary parts of $e^{i\,\frac{\pi}{4}}$? Compute the DFT of the sequence $A = (0, 1, 2, 3, 4, 5, 6, 7)$ by applying the FFT algorithm by hand.

9. Describe how you would compute all elements of the sequence $F(0), F(1), F(2), \ldots, F(2n)$ where

   (a)
$$F(m) = \sum_{\substack{i+j=m\\0\le i,j\le n}} i^3 j^2$$

   (b)
$$F(m) = \sum_{\substack{i+j=m\\0\le i,j\le n}} \log(j+1)^i$$

   in time $O(n \log n)$.

10. (a) Compute by any method you wish the (linear) convolution $s * s$ of the sequence $s = \langle 1, 2, 0, 4\rangle$ with itself. (Note that there is no requirement on the efficiency of your method, and that the sequence is really short!)

    (b) If a sequence x has n terms and sequence y has k terms, how many terms does the convolution $x * y$ of these two sequences have?

    (c) Is it true that $s * t = t * s$ for any two sequences s and t? Explain why or why not.

    (d) Describe how we compute **efficiently** the convolution of two (long) sequences?

11. In this part we will extend the convolution algorithm described in class to multiply multiple polynomials together (not just two).
    Suppose you have K polynomials $P_1, \ldots, P_K$ each of degree at least one, and that
$$\text{degree}(P_1) + \cdots + \text{degree}(P_K) = S$$

(i) Show that you can find the product of these K polynomials in $O(KS \log S)$ time.
Hint: consider using divide-and-conquer; a tree structure might be helpful here as well. Also, remember that if x, y, z are all positive, then $\log(x+y) < \log(x+y+z)$

(ii) Show that you can find the product of these K polynomials in $O(S \log S \log K)$ time. Explain why your algorithm has the required time complexity.
Hint: consider using divide-and-conquer!

12. You have a set of N coins in a bag, each having a value between 1 and M, where $M \geq N$. Some coins may have the same value. You pick two coins (**without replacement**) and record the sum of their values. Determine what possible sums can be achieved, in $O(M \log M)$ time.
For example, if there are $N = 3$ coins in the bag with values 1, 4 and 5 (so we could have $M = 5$), then the possible sums are 5, 6 and 9.
Hint: if the coins have values $v_1, \ldots, v_N$, how might you use the polynomial $x^{v_1} + \cdots + x^{v_N}$?

13. Consider the polynomial

$$P(x) = (x - \omega_{64}^0)(x - \omega_{64}^1)(x - \omega_{64}^2) \ldots (x - \omega_{64}^{63})$$

(a) Compute $P(0)$;

(b) What is the degree of $P(x)$? What is its coefficient of the highest degree of x present in $P(x)$?

(c) What are the values of $P(x)$ at the roots of unity of order 64?

(d) Can you represent $P(x)$ in the coefficient form without any computation?

Hint: two polynomials of the same degree which have the same coefficient in front of the largest power and have the same zeros are equal.

14. To apply the FFT to the sequence $(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$ we apply recursively FFT and obtain $FFT(a_0, a_2, a_4, a_6)$ and $FFT(a_1, a_3, a_5, a_7)$. Proceeding further with recursion, we obtain $FFT(a_0, a_4)$ and $FFT(a_2, a_6)$ as well as $FFT(a_1, a_5)$ and $FFT(a_3, a_7)$. Thus, from bottom up, $(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$ is obtained using permutation $(a_0, a_4, a_2, a_6, a_1, a_5, a_3, a_7)$ as the leaves of the recursion tree of the original input sequence. Given any input $(a_0, a_1, a_2, \ldots, a_{2^n-1})$ describe the permutation of the leaves of the recursion tree.

Hint: write indices in binary and see what the relationship is of the bits of the i^{th} element of the original sequence and the i^{th} element of the resulting permutation of elements as they appear on the leaves on the recursion tree. From there use induction to prove the general statement

15. You are given a sequence

$$A = \langle a_0, a_1, \ldots, a_{n-1} \rangle$$

of length n and sequence

$$B = \langle 1, \underbrace{0, 0, \ldots, 0}_{k}, -1 \rangle$$

of length $k+2$, where $1 \leq k \leq n/4$.

   (a) Compute the DFT of sequence B.

   (b) Compute the convolution sequence $C = A * B$ in terms of the elements of sequence A.

   (c) Show that in this particular case such a convolution can be computed in time $O(n)$.

16. (a) Compute the DFT of the sequence $(1, -1, -1, 1)$ by **any method** you wish.

   (b) Compute the linear convolution of sequences $(1, -1, -1, 1)$ and $(-1, 1, 1, -1)$ by **any method** you wish.

17. Assume that you are given a polynomial $P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ and a polynomial $Q(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3$ whose coefficients can be arbitrarily large numbers. Let $R(x) = P(x)^2 - Q(x)^2$. Compute the coefficients of $R(x)$ using only 7 large number multiplications.

18. Recall that the DFT of a sequence

$$\langle a_0, a_1, \ldots, a_{n-1} \rangle$$

is the sequence of values

$$\langle P(\omega_n^0), P(\omega_n^1), P(\omega_n^2), \ldots, P(\omega_n^{n-1}) \rangle$$

of the associated polynomial $P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$. Compute directly (i.e., without using the FFT) and maximally simplify the DFT of the following sequences:

(a) $\langle 1, \underbrace{0, \ldots, 0}_{n-1} \rangle$

(b) $\langle \underbrace{0, \ldots, 0,}_{n-1} 1 \rangle$

(c) $\langle \underbrace{1, \ldots, 1}_{n} \rangle$