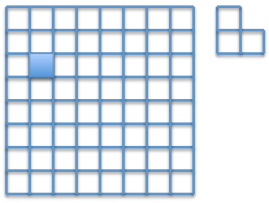
Algorithms Tutorial 2
Exercises

Divide and Conquer and polynomial multiplication

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. Each dominoe can be rotated.

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

$2N\lceil \log N \rceil$ 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. Given positive integers M and n compute M^n using only $O(\log n)$ many multiplications. (15 pts)

4. 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" booklet.

5. 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.

6. 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.

7. (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.

   Solution:

   (a) This is again the Karatsuba trick:

   $$(a+i\,b)(c+i\,d) = ac-bd+(bc+ad)\,i = ac-bd+((a+b)(c+d)-ac-bd)\,i$$

   so we need only 3 multiplications: ac and bd and $(a + b)(c + d)$.

   (a) $(a + i\,b)^2 = a^2 - b^2 + 2ab\,i = (a + b)(a - b) + (a + a)b\,i$.

   (b) Just note that $(a + i\,b)^2(c + i\,d)^2 = ((a + i\,b)(c + i\,d))^2$; you can now use (a) to multiply $(a + i\,b)(c + i\,d)$ with only three multiplications and then you can use (b) to square the result with two additional multiplications.

8. Assume that you are given a polynomial $P(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ and a polynomial $Q(x) = b_0 + b_1x + b_2x^2 + b_3x^3$ 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.

9. You are given a polynomial $P(x) = A_0 + A_1x^{100} + A_2x^{200}$ where A_0, A_1, A_2 can be arbitrarily large integers. Design an algorithm which squares $P(x)$ using only 5 large integer multiplications. (15 pts)