Solutions to Assignment 2

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

   Solution: Note that when n is even, $M^n = (M^{\frac{n}{2}})^2$, and when n is odd, $M^n = (M^{\frac{n-1}{2}})^2 \times M$. Hence, we can proceed by divide and conquer. If n is even, we recursively compute $M^{\frac{n}{2}}$ and then square it. If n is odd, we recursively compute $M^{\frac{n-1}{2}}$, square it and then multiply by another M. Since n is (approximately) halved in each recursive call, there are at most $O(\log n)$ recursive calls, and since we perform only one or two multiplications in each recursive call, the algorithm performs $O(\log n)$ many multiplications, as required.

   Alternative Solution: Any positive integer n is the sum of a subset of the powers of 2 ($\{1, 2, 4, 8, 16, ...\}$). Thus, M^n is the product of a subset of powers of M where the power is a power of 2 ($\{M, M^2, M^4, M^8, ...\}$). We can obtain these powers of M in $O(\log n)$ time by repeated squaring and then multiply together the appropriate powers to get M^n. The appropriate powers to multiply are the powers M^{2^i} such that the i^{th} least significant bit of the binary representation of n is 1. For example, to obtain M^{11}, the binary representation of 11 is 1011, and hence we should multiply together M, M^2, and M^8.

2. 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)

   Solution: Note that using the substitution $y = x^{100}$ reduces $P(x)$ to $P^*(y) = A_0 + A_1y^1 + A_2y^2$, and it is clear that $P^*(y)^2$ will have the same coefficients as $P(x)^2$. The polynomial $Q^*(y) = P^*(y)^2$ is of degree 4 so to uniquely determine $Q^*(y)$ we need five of its values. Thus, we evaluate $Q^*(y)$ at five values: -2, -1, 0, 1, and 2 (this is where the 5 large integer multiplications occur). Then, using these five values, we obtain the coefficients of $Q^*(y)$ by solving the corresponding system of linear equations. After we have found the coefficients of $Q^*(y)$, we can substitute y back with x^{100} to obtain $P(x)^2$.
   Alternative, cleverer solution by a student: Note that

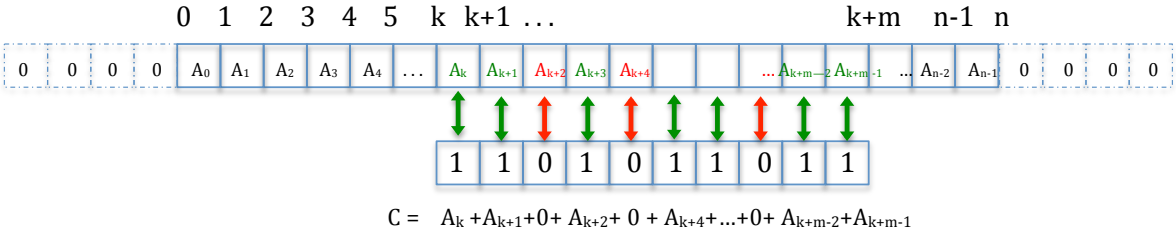
   $$(A_0+A_1x^{100}+A_2x^{200})^2 = A_0^2+2A_0A_1x^{100}+(A_1^2+2A_0A_2)x^{200}+2A_1A_2x^{300}+A_2^2x^{400}$$

   and that

   $$(A_0 + A_1 + A_2)^2 - A_0^2 - A_2^2 - 2A_0A_1 - A_1A_2 = A_1^2 + 2A_0A_2$$

   Thus, we only need 5 multiplications of large numbers to compute A_0^2, A_2^2, A_0A_1, A_1A_2 and $(A_0 + A_1 + A_2)^2$ to obtain all the needed coefficients.

3. Assume you are given a map of a straight sea shore of length $100n$ meters as a sequence of $100n$ numbers such that A_i is the number of fish between the i^{th} meter of the shore and the $(i+1)^{th}$ meter, $0 \leq i \leq 100n-1$. You also have a net of length n meters but unfortunately it has holes in it. Such a net is described as a sequence N of n ones and zeros, where 0's denote where the holes are. If you throw such a net starting at meter k and ending at meter $k+n$, then you will catch only the fish in one meter stretches of the shore where the corresponding bit of the net is 1; see the figure.



Find the spot where you should place the left end of your net in order to catch the largest possible number of fish using an algorithm which runs in time $O(n \log n)$. (30 pts)

Hint: Let N' be the net sequence N in the reverse order; look at the sequence $A * N'$.

Solution: We are given that the sea shore is described by the sequence $A = \langle A_0, A_1, \ldots, A_{100n-1} \rangle$, where A_i is the number of fish in the i^{th} meter of the shore, and the net is described by the sequence $N = \langle N_0, N_1, ..., N_{n-1} \rangle$, where N_i is 0 if there is a hole in the i^{th} meter of the net, and 1 otherwise. Hence, if we place the left end of our net at meter i, then the number of fish we will catch is given by $A_i N_0 + A_{i+1} N_1 + \ldots + A_{i+n-1} N_{n-1} = \sum_{k=0}^{n-1} A_{i+k} N_k$. Note that the left end of our net can be placed anywhere between meter $-n+1$ and meter $100n-1$ ($-n+1 \leq i \leq 100n-1$), but if any part of the net is placed outside the shore, that part does not yield any fish, so $A_i = 0$ for all values of i outside the range $[0, 100n-1]$. Since we want to catch as many fish as possible, we want to find the value of i that maximises $\sum_{k=0}^{n-1} A_{i+k} N_k$. Let us define a sequence N^* as the reversed sequence N, i.e., $N_i^* = N_{n-1-i}$ for all $0 \leq N-1$. Then the number of fish we catch if we throw the net starting at A_i is equal to

$$\begin{aligned} A_i N_0 + A_{i+1} N_1 + \ldots + A_{i+n-1} N_{n-1} &= A_i N_{n-1}^* + A_{i+1} N_{n-2}^* + \ldots + A_{i+n-1} N_0^* \\ &= \sum_{j+k=i+n-1} A_j N_k^* \end{aligned}$$

But the last sum is just the value of the convolution $A * N'$ at point $i+n-1$. Thus, we convolve sequences A and N^* and look for m such that $(A * N')[m]$ has the largest value which represents the largest number of fish you can catch, and solving $m = i+n-1$ for i, i.e., $i = m-n+1$ tells you where your net should start. Note that negative values of i indicate that the beginning of the net is on the shore and only a part of it is in the sea. Similarly, if $m > 100n$ this indicates that the right end of the net will be on the shore and only its left part in the sea.

4. (a) Compute the convolution $\langle 1, \underbrace{0, 0, \ldots, 0}_{k}, 1\rangle * \langle 1, \underbrace{0, 0, \ldots, 0}_{k}, 1\rangle$. (10 pts)

Solution: The sequence $\langle 1, \underbrace{0, 0, \ldots, 0}_{k}, 1\rangle$ corresponds to the polynomial $P(x) = 1 + x^{k+1}$, and the convolution of this sequence with itself is the sequence of coefficients of the polynomial $P(x)^2 = (1 + x^{k+1})^2 = 1 + 2x^{k+1} + x^{2k+2}$, i.e., the sequence $\langle 1, \underbrace{0, 0, \ldots, 0}_{k}, 2, \underbrace{0, 0, \ldots, 0}_{k}, 1\rangle$.

(b) Compute the DFT of the sequence $\langle 1, \underbrace{0, 0, \ldots, 0}_{k}, 1\rangle$. (10 pts)

Solution: The corresponding polynomial is $P(x) = 1+x^{k+1}$, and hence

$$\begin{aligned} DFT(\langle 1, \underbrace{0, 0, \ldots, 0}_{k}, 1\rangle) &= \langle P(\omega_{k+2}^0), P(\omega_{k+2}^1), \ldots, P(\omega_{k+2}^{k+1})\rangle \\ &= \langle 1 + \omega_{k+2}^{0\cdot(k+1)}, 1 + \omega_{k+2}^{1\cdot(k+1)}, \ldots, 1 + \omega_{k+2}^{(k+1)\cdot(k+1)}\rangle \\ &= \langle 2, 1 + \omega_{k+2}^{1(k+1)}, 1 + \omega_{k+2}^{2(k+1)}, \ldots, 1 + \omega_{k+2}^{(k+1)^2}\rangle \end{aligned}$$

5. Find the sequence x satisfying $x * \langle 1, 1, -1\rangle = \langle 1, 0, -1, 2, -1\rangle$. (20 pts)

Hint: What polynomials correspond to the given sequences?

Solution: The sequence $\langle 1, 1, -1\rangle$ corresponds to the polynomial $Q(y) = 1 + y - y^2$, and the sequence $\langle 1, 0, -1, 2, -1\rangle$ corresponds to the polynomial $R(y) = 1 - y^2 + 2y^3 - y^4$.

x is simply the sequence of coefficients of the polynomial $P(y)$ that satisfies $P(y) \cdot Q(y) = R(y)$. Polynomial division gives $P(y) = 1 - y + y^2$, so $y = \langle 1, -1, 1\rangle$.