CIS 667 and CIS 467H: Homework 3 (Due: Mar 24, 2005)

Problem 1. (10 points)
Show that there are \( n^2 - n + 2 \) bitonic sequences of 0’s and 1’s.

Problem 2. (10 points)
The Fibonacci sequence is given by the following recurrence:
\[
F_{n+1} = F_n + F_{n-1} \quad \text{for } n > 1 \quad \text{and} \quad F_0 = F_1 = 1.
\]
(i) Show how to compute \( F_n \) in \( O(n) \) time.
(ii) Given an \( n \times n \) matrix \( A \) show how you can find \( A^n \) in \( O(n^3 \log n) \) time.
(iii) Can you improve the time bound in (i)? In particular, prove that \( F_n \) can be computed in \( O(\log n) \) time.

**Hint:** You may need to use the result of part (b), i.e. formulate the \( F_n \) as a matrix problem. The discussion on page 902 and 903 (Problem section at the end of the Chapter on Number-Theoretic Algorithms may offer you some insight).

Problem 3. (20 points)
You are given a real number \( x \), a positive integer \( n \), and 2 processors. Design a parallel algorithm that computes \( x^n \) in \( \log n + O(1) \) parallel multiplication steps. You are only allowed to use the floating-point/integer operations +, -, *, and the integer operations SHR, LSB.

SHA(x) shifts x right one bit position and inserts a zero into the leftmost bit position. LSB(x) returns the least significant bit of x. For example if \( x = 1111 \), then after SHR(x), x becomes 0111. If \( x = 1110 \), LSB(x) is 0.

Problem 4. (20 points)
(a) Show that any sorting network must have \( \Omega(\log n) \) depth, and \( \Omega(n \log n) \) comparators.
(b) Then show that any sorting network must have depth at least \( \log n \).

Note: In (a) we don’t care about constants; in (b) we do care.

Problem 5. (20 points)
(a) Show that an \( n \)-input sorting network must contain at least one comparator between the \( i \)-th and the \( i + 1 \)-st wire for all \( i \) such that \( 1 \leq i \leq n - 1 \).

(b) Give a sorting network that sorts 4 keys with 5 comparators.

Problem 6. (20 points)
This is Problem 11-3 of CLRS (page 250-251). Suppose that we are given a key \( k \) to search for in a hash table with positions 0, \ldots, \( m - 1 \), and suppose that we have a hash function \( h \) mapping the key space into the set \( \{0, \ldots, m - 1\} \). The search scheme is as follows.
1. Compute the value \( i = h(k) \) and set \( j = 0 \).
2. Probe in position \( i \) for the desired \( k \). If you find it, or if this position is empty, terminate the search.
3. Set \( j = (j + 1) \mod m \) and \( i = (i + j) \mod m \) and return to step 2.

Assume that \( m \) is a power of 2.

(a) Show that this scheme is an instance of the general quadratic probing scheme by exhibiting the appropriate constant \( c_1, c_2 \) for \( h(k, i) = (h(k) + c_1 i + c_2 i^2) \mod m \).

(b) Prove that this algorithm examines every table position in the worst case.
CIS 667 and CIS 467H: Homework 3 (Due: Mar 24, 2005)

Option 1. (100 points)
(a) Implement a sorter based on bitonic sorting. It must be able to handle key sizes other than power of two. The interface must be that of the Standard C library function `qsort`. For details on `qsort` do a `man qsort` on an AFS or Linux System. In Visual C++ do a Help | Search for `qsort`. More information on how to use `qsort` is available in the posted source code.
(b) Write a simple (recursive or non-recursive) FFT program that works for \( n \) a power of two with the following definition `comp` of a complex number

```c
typedef struct {
    double i, r;
} comp;
```

For example the high level function call may look like

```c
my_fft(comp *input, comp *output, int n);
```

if `output` has been preallocated or

```c
my_fft(comp *input, comp **output, int n);
```

if `output` is to be allocated inside `my_fft`.

All programs to be sent to alg667@cs.njit.edu or alg467@cs.njit.edu.

Revised Homework Schedule.
HW4: Out on Mar 24 (changed) and in Apr 7 (as planned).
HW5: Out on Apr 7 (as planned) and in Apr 21 (as planned).