CIS 668-102 Parallel Algorithms	**Homework 4** **Due: Week 15, Monday May 10**	Dr. David Nassimi Spring 2004

(Topic: Parallel Sorting)

1. Consider sorting a random sequence of N elements on an N-PE *linearly-connected* computer without end-around connections, $N = 2^P$. Analyze the complexity for each of the following sorting algorithms:

   (a) Odd-Even-Transposition sort;

   (b) Batcher's Bitonic sort.

   In each case, find the total number of comparison steps, $C(N)$, and unit-distance-moves, $U(N)$. (You don't have to write the algorithms. For (a), the algorithm is obvious. For (b), you can take the algorithm you wrote for *cube* and analyze it for the linearly-connected architecture.)

   For bitonic sort, carry out the analysis formally using recurrence relations. (First analyze the bitonic merge algorithm which merges a bitonic sequence of length 2^k into a sorted sequence.) Which of the two sorting algorithms is more efficient?

2. (a) Show the *bitonic merge* network for $N = 16$. (The input is a bitonic sequence of length N. The output is sorted in increasing order.) Apply the input

   $$A = (3, 5, 6, 7, 9, 11, 14, 15, 13, 12, 10, 8, 4, 2, 1, 0).$$

   (b) Show the bitonic sorting network for $N = 16$. (The input is a random sequence). Apply the input

   $$A = (11, 5, 15, 9, 3, 14, 6, 7, 4, 8, 1, 10, 0, 2, 12, 13).$$

   (c) Write the *bitonic sort* algorithm for an N-PE *cube* to sort a *random* sequence $A[0 : N-1]$, where $N = 2^P$. (The mask may be written in terms of index bits as well as local data.) Find the number of comparisons and routing steps.

   (d) Write the sorting algorithm for a PSC with N PEs. Find the total number of routing steps. (Note that the number of comparison steps will be the same as on cube.) Illustrate the PSC sorter for $N = 8$ and $A = (2, 5, 0, 3, 7, 4, 1, 6)$. (The illustration should be in the form of a sorting network, with shuffles/ unshuffles between stages of comparators. Be sure to indicate the polarity of each comparator.)

3. Write an algorithm to implement bitonic-merge on an $M \times M$ *mesh* without end-around connections, where $N = M^2$ and $M = 2^P$. (The input is a bitonic sequence A of length N stored in row-major order. The output is A sorted in row-major order.) Find the number of routing instructions, unit-distance moves, and comparison steps. Illustrate the algorithm for $M = 4$ and

   $$A = \begin{bmatrix} 3 & 5 & 6 & 7 \\ 9 & 11 & 14 & 15 \\ 13 & 12 & 10 & 8 \\ 4 & 2 & 1 & 0 \end{bmatrix}$$

4. This problem deals with Batcher's *odd-even merge* algorithm where input size is $N = 2^P$. The input sequence is concatenation of two sorted sequences of length $N/2$, both sorted in increasing order.

(a) Show the merging network for $N = 16$. Number the inputs 0..N-1. Show the action for input sequence

$$A = (3, 5, 6, 7, 9, 11, 14, 15,\ 0, 1, 2, 4, 8, 10, 12, 13).$$

(b) Is this algorithm suitable for implementation on an N-PE cube? Explain.

(c) Consider an N-PE parallel computer with an interconnection scheme called *Plus-Minus-*2^b (abbreviated as PM2b). Let PEs be numbered $0 : N-1$. Each PE[i] is connected to the following PEs (if they exist): PE[$i+2^b$] and PE[$i-2^b$], $0 \le b \le \log N - 1$. Note that each PE is connected to (at most) $2 \log N$ other PEs. Also note that this interconnection is a superset of the cube interconnection. (Some authors consider a variant of this with end-around connections. That is, PE[i] is connected to PE[$(i \pm 2^b) \bmod N$]. Thus each PE is connected to exactly $2 \log N$ others. For this problem, we don't have such end-around connections.) Write the odd-even merging algorithm for PM2b architecture. What is the total number of routing steps?

(d) An *Ultracomputer*, defined by J.T. Schwartz [1], is similar to a PSC as we have defined except that instead of only even-odd connections, there is a linear connection (with end-around). That is, PE[i], $0 \le i \le N-1$, is connected to 4 other PEs:

shuffle(i), *unshuffle*(i), $(i+1) \bmod N$, and $(i-1) \bmod N$.

An Ultracomputer can "simulate" a PM2b much the same way as a PSC simulates a cube. Implement the odd-even merge algorithm on this architecture. Illustrate for $N = 16$ and $A = (3, 5, 6, 7, 9, 11, 14, 15,\ 0, 1, 2, 4, 8, 10, 12, 13)$.

Additional Exercises (Not to be handed-in)

5. (a) Given a sorting network A which produces N outputs sorted in ascending order. Prove that by reversing the direction of every comparator, we obtain a sorting network B which sorts in descending order.

Hint: Use 0/1 principle. Corresponding to any input vector X to network A, consider the complement vector X' as input to network B.)

(b) Illustrate the theorem on Batcher's bitonic sort network A (which sorts any random sequence in $O(\log^2 N)$ steps) for $N = 8$ and input $X = (1, 0, 1, 0, 1, 1, 1, 0)$, and the corresponding network B.

6. Given a bitonic sequence $A = (A_0, \ldots, A_{N-1})$ where N is even. Prove there exists an integer k, $0 \le k \le N-1$, such that if A is left-rotated by k places, the resulting sequence

$$(A_k, \ldots, A_{N-1}, A_0, \ldots, A_{k-1})$$

will satisfy the following properties:

(a) Every element in the left half will be $\le$ every element in the right half; and

(b) Each half will be a bitonic sequence.

7. (a) Show the *odd-even-transposition* sorting network for $N = 8$. Apply the input sequence $A = (8, 7, 6, 5, 4, 3, 2, 1)$.

[1] J.T. Schwartz, "Ultracomputers," *ACM Transactions on Programming Languages and Systems*, Vol. 2, No. 4, Oct. 1980, pp. 484-521.

(b) Show the sorting network for $N = 4$. Suppose each input is a vector of 2 elements, and each comparator is replaced by a merge-split module. Show how the following input is sorted:

$$A = [(4, 5), (1, 4), (3, 4), (0, 2)].$$

(c) Consider sorting kN items on a linearly-connected SIMD computer with N PEs and without end-around connections. (We want to do this by adapting an N-input transposition sorter, with each input a vector of k elements.)

i. First write a subroutine

LOCAL-MERGE(A, B, C, k)

to locally merge in each PE$[i]$ two sorted sequences of length k, $A[i, 0 : k - 1]$ and $B[i, 0 : k - 1]$, into a sorted sequence $C[i, 0 : 2k - 1]$. Analyze the time complexity.

ii. Write the algorithm to sort kN arbitrary items $A[0 : N - 1, 0 : k - 1]$, where $A[i, 0 : k - 1]$ is in PE$[i]$. At the end, the smallest k items must be in PE[0], the next k items in PE[1], etc. Assume another subroutine

LOCAL-SORT(A, k)

is available which locally sorts a k-element sequence $A[i, 0 : k-1]$ in each PE$[i]$ in $O(k \log k)$ steps. (You don't need to write the LOCAL-SORT procedure.) Analyze the overall time complexity.

8. This problem explores an alternative proof of correctness for the odd-even-transposition sorting network, using a modified version of the zero-one principle applicable to *primitive* networks. (Recall a comparator-network is called *primitive* if every comparator is between an adjacent pair $[i : i + 1]$.)

(a) Prove that if a primitive network with n inputs sorts every input sequence of type 1^k0^{n-k} (that is, the first k inputs are 1 and the remaining $n - k$ are 0) , where $0 \leq k \leq n$, then the network correctly sorts every input sequence.

Hint: One way to prove this is to use the following two proven properties:

i. If a primitive network properly sorts the reverse sequence $(n, n - 1, \cdots, 1)$, then it sorts every input.

ii. If a network with input X produces output Y, and if $f()$ is a monotonic function, then the network with input $f(X)$ will produce output $f(Y)$.

(b) Prove that the odd-even-transposition network correctly sorts every input sequence of type 1^k0^{n-k}, where $0 < k < n$. First find the trajectory of the first 0 input (i.e., input $k + 1$) Then deduce the trajectory of each of the subsequent 0 inputs.