1. On slide 20 of lecture 16 we derived a parallel implementation of matrix multiplication $C = AB$ where $A, B, C \in \mathbb{R}^{n \times n}$ with BSP cost

$$C^{MM}(n, p) = \frac{2n^3}{p} + \left(\frac{2n^2}{\sqrt{p}}\right) \cdot g + 2 \cdot L$$

Can you find a way to improve the communication cost?

**SAMPLE SOLUTION**

The cost given above is the result of dividing $C$ into $p$ equal sized squares and for each square fetching all values needed to compute the result, and then computing the result. So each processor’s communication corresponds to fetching $n/\sqrt{p}$ rows of $A$ and $n/\sqrt{p}$ columns of $B$ keeping in mind that rows and columns have $n$ elements. With all processors fetching in parallel, this is a balanced communication with cost $(2n^2/\sqrt{p}) \cdot g$.

To improve the communication, can we divide up the work over $p$ processors so that each processor needs to fetch a smaller amount of data to compute its result? Consider breaking the $n \times n \times n$ matrix multiplication cube into $p$ equal sized subcubes. Each processor fetches $n/p^{1/3} \times n/p^{1/3}$ values from $A$ and from $B$ and computes the result of size $n/p^{1/3} \times n/p^{1/3}$. The communication cost for this step is $(2n^2/p^{2/3}) \cdot g$. However, this doesn’t give our answer yet, since a given element of $C$ is the sum of a value from each of $p^{1/3}$ subcubes. In a second step we partition $C$ into $p$ squares with $n/\sqrt{p}$ elements on a side, and for each element of the result we sum the $p^{1/3}$ corresponding elements from subcubes. The communication cost for this step is $(n/\sqrt{p}) \times (n/\sqrt{p}) \times p^{1/3} = (n^2/p^{2/3}) \cdot g$.

You can verify that these communications are balanced.

So this gives us a total communication cost of $(3n^2/p^{2/3}) \cdot g$. As the denominator of this cost is larger than the denominator of our original cost, for sufficiently large $p$, this will have improved communication cost.
An alternate way to construct a bitonic sort for \( N = np \) elements with \( p = 2^k \) processors is to extend the compare-exchange operation to \textit{sorted sequences}. For \( v, w \) sorted sequences of length \( n \) let \( \text{CE}_{\text{seq}}(v, w) = s, t \) with \( s = \text{merge}(v, w)[1:n] \) and \( t = \text{merge}(v, w)[n+1:2n] \). Observe the \( 2n \) elements of \( s, t \) define a permutation of the \( 2n \) elements of \( v, w \) and \( s, t \) are partitioned, meaning any element in \( s \) is less than or equal to any element in \( t \). More formally the transitive relation \( s \preceq t \equiv \forall_{1 \leq i, j \leq n} s_i \leq t_j \). Using this relation, design a comparison-based bitonic sort of \( N = np \) elements using \( p \) processors with asymptotically optimal work efficiency (in the sense of lecture 16, last slide). Show the complete BSP cost of your solution.

\textit{Sample Solution}

According to the bitonic merge theorem, when we apply the appropriate sequence of CE operations to \( p \) input elements with a total ordering \( \leq \), on termination we will have a permutation of the input elements across processors 1..\( p \) such that \( s_1 \leq s_2 \leq \cdots \leq s_p \).

If we now let the input values be \textit{sorted sequences} of length \( n = N/p \) and define the \( \text{CE}_{\text{seq}} \) operation and the total order \( \preceq \) as given above, then on termination we will have \( s_1 \preceq s_2 \preceq \cdots \preceq s_p \), where each of the \( s_i \) is now a sorted sequence of \( n \) values. Thus, we will have sorted all \( N \) input values.

We sort the input sequences locally at each processor to create sorted sequences before starting the bitonic sort. The work of an initial local sort on all processors is \( \Theta(n \lg n) \) and each \( \text{CE}_{\text{seq}} \) operation in bitonic sort exchanges \( n \) values with a neighboring processor and performs \( \Theta(n) \) work to merge two sorted sequences of length \( n \). The BSP cost of the complete algorithm is

\[
C_{\text{BSP}}(n, p) = \Theta(n \lg n) + L + (\lg^2 p)(\Theta(n) + ng + L)
\]

Combining just the work terms over all steps we have \( \Theta(n \lg n) + (\lg^2 p)\Theta(n) \). Since the local sort can be an optimal sequential sort, the asymptotic efficiency of the work term is given by

\[
\pi = \lim_{N \to \infty} \frac{c_1 n \lg n + (\lg^2 p)\Theta(n)}{(c_1 \lg N)/p} = \lim_{N \to \infty} \frac{c_1 N \lg N - c_1 N \lg p + c_2 N(\lg^2 p)}{c_1 \lg N} = 1
\]

where \( c_1 \) and \( c_2 \) are the leading coefficients in the sort and merge costs, respectively. Since the asymptotic ratio of our work to the optimal work equally distributed over \( p \) processors is 1, our parallel bitonic sort can be work optimal for any number of processors. Note however that \( N \) may have to be very large for \( c_1 N \lg N \) to dominate \( c_2 N(\lg^2 p) \).