ECS223a Parallel Algorithms
Homework 2

You are encouraged to talk to other people about these problems, but please **write up the solutions by yourself**. Cite any conversations you had with others, as well as books, papers or Web sites you consulted.

Always explain the answer in **your own words**; do not copy text from anywhere. Explain your solution as you would to someone who does not understand it, for instance to a beginning graduate student or an advanced undergraduate. Do not give several solutions to one problem; pick the best one and explain it.

Please type your homework and submit it using SmartSite. If you know LaTeX, use that. If you don't know it, this might be a good time to learn. Include pictures if appropriate; you can scan them in or produce them with a computer sketching program.

1. Do problem 5.13.

2. Do problem 5.22.

3. The sorting algorithm in the 1992 paper, "Parallel sort by regular sampling" by Shi and Schaeffer has come to be called "sample sort" and it is a leading technique for sorting on clusters. In this problem we'll do a BSP analysis of a slightly simplified version of the algorithm in their paper.

   They assume the processors have access to a shared global memory, which initially contains the input array. The input array is of size n and the number of processors is p.

   > 1. Each processor reads a contiguous group of n/p elements from the input array.
   > 2. Each processor sorts its array sequentially, and then selects $p-1$ evenly spaced local pivots, which it writes to global memory.
   > 3. Processor 1 reads all of the local pivots and sorts them. It then selects $p-1$ evenly spaced global pivots from the sorted list of local pivots, and writes the global pivots to global memory. These global pivots segment the output into almost equally-sized sub-problems, as we will argue below.
   > 4. Each processor reads the global pivots, uses them to divide its sorted array into segments, and writes the sorted array, with a flag array describing the segments, into global memory.
   > 5. Processor i computes the ith segment of the output array, by merging the p segments belonging to that segment from each of the p outputs of Step 4.

   a) The algorithm hinges on the idea that each output segment has size nearly n/p. Give the best lower and upper bounds you can on the size of each output segment. Feel free to consult the paper.

b) Looking at the number of instructions required in each step of the algorithm, for what values of n, as a function of p, is the number of instruction executed by the parallel algorithm optimal?

c) In the BSP analyses we did in class, we described the running time in terms of the machine parameter g, which is roughly the inverse of memory bandwidth, as well as the algorithm parameters h for the maximum number of messages per superstep, L for the maximum number of instructions per superstep, and and k for the number of supersteps. Give such a description for this algorithm. Describe, where necessary, how best to implement each of the five steps of the algorithm.

d) Based on part c), for what range of g, as a function of n and p, is this algorithm optimal? Does this suggest that communication or computation will be the bottleneck?