Spring 2013, CSE 392/CS 395 T, Homework 3
Due 11pm Tuesday April 9th , electronically (ZIP)

In this homework, you can work in groups. (No more than three people per group.) Each group member must submit the **same** write-up and source files for the code. (For logistical reasons, every member of the group must submit the homework.) Every group should select a "`group`" name.

Submission: Please submit a single ZIP file named `group_lastname_firstname_hwk3.zip`. Please do not use uppercase and do use the underscores.
The zip file should uncompress to a directory named "`group_lastname`" that contains: (a) your write-up "`group.pdf`"; and (b) the makefile and sources to builds three executables `scan`, `search`, and `sort`.

For all questions that involve implementations you are welcome to use TACC Longhorn's GPUs. You can either use it instead of OpenMP or you use GPUs in addition to OpenMP to provide extra parallelism.

1. (Graphs.) An undirected, unweighted graph can be defined by the number of vertices and a compressed row storage sparse matrix (see notes) for its adjacency matrix. (Although the matrix is symmetric, for simplicity you can use a generic CRS format). Give a shared memory implementation for Luby's maximal independent sets algorithm. It is up to you to demonstrate the scalability of your algorithm by selecting datasets and what numbers to report.

2. (Graphs.) Give MPI pseudo-code for parallel graph matching. Assume that the input graph is colored, undirected, and unweighted. Given an estimate for the complexity of the algorithm assuming the graph is sparse and that the initial partitioning of the vertices is random.

3. (Multigrid.) Derive the work-depth complexity of a V-cycle multigrid for the 3-point, 5-point, and 7-point Laplacian in 1D,2D, and 3D.

4. (Morton orderings on nonuniform octrees). Let T be an octree decomposition of the cube for which we have the level and Morton index of every node (leaf and internal). Assume that we have N nodes in T.

   (a) Sketch an $O(1)$ algorithm, which given two nodes $n_1, n_2 \in T$ at levels $l_1 < l_2$, determines whether n_1 is an ancestor of n_2.

   (b) Sketch an $O(\log N)$ algorithm that returns all the colleagues of a node n_1. The colleagues of a node n_1 are all nodes in T that are in the same level as n_1 and share an edge or a face or corner with n_1. Is there a way to reduce the complexity to $O(1)$?

   (c) Given two nodes n_1 and n_2 with $(l_1 \geq l_2)$, sketch an algorithm to compute the least common ancestor of these nodes. Give the complexity of your algorithm. (The least common ancestor of two nodes is a common ancestor whose level is closest to l_1).

   (d) Assume a quadtree stored as an unsigned integer array `a` of length N in which `a[i]` is the morton id of the node `i`. Assume the array is sorted. Given rank `i`, give an $O(\log N)$-depth algorithm to compute the ranks of the children of the node `i`.

5. (2D N-BODY SIMULATION). You will develop a shared (OpenMP) memory implementation of a two-dimensional N-body approximation scheme using the Barnes-Hut scheme. That is, given source points $\{y_j\}_{j=1}^N$, densities d_j and target points $\{x_i\}_{i=1}^M$ you will approximate $u_i = \sum_j G(x_i, y_j)d_j, \quad \forall i$. The function pointer `G` is the exact evaluation kernel. For your experiments use $G(x_i, y_j) = -1/2\pi \log|x_i - y_j|$ if $x_i \neq y_j$ and 0 otherwise.

   Recall that in the Barnes-Hut scheme, "averaging" in a node is done by computing the total density $D = \sum d_j$ and the center of "mass", $\bar{y} = 1/D \sum y_j d_j$. For a leaf node, the sum is over the individual points. For an internal node, the sum is over the centers of mass of its children.

   You should use data structures and algorithms that support non-uniform trees. Please document and explain how to compile and use the code (a text README file is suffcient).

   To assist you with the design, I've provided a complete (sequential) implementation in MATLAB. See `nbody.tgz`. You should implement any of the parallelization approaches we discussed in the class. But all phases (tree construction, tree averaging, evaluation) should be parallelized.

   (a) Give pseudocode for all major functions in your implementation. Give expected work and depth complexities (for the unform tree case); In particular, make sure you explain how you parallelized the code.

   (b) Report weak and strong scalability results on 1,2,4,8,16 threads for $N = 2^{18}$, 2^{20}, 2^{22} points (for uniform and nonuniform distributions of points). For simplicity, you can assume that the target points set is the same as the input point set.

   (c) Suggest an inexpensive way ($O(N)$) to estimate the approximation error introduced by averaging (compared to a direct evaluation). Estimate the speed-up over the direct sum implementation.