The theme of these projects is the implementation spatial data structures on the GPU using CUDA. The algorithm for building these data structures are given in the Foundations of Multidimensional and Metric Data Structures Book. Please refer to the discussions in the book. You may also have to look up papers referenced in the book. The project for this class is not meant to be hard. My intention in assigning this project is for you to learn a new data structure as well as to become comfortable working with CUDA. Your solution should include a problem definition, pseudo code as well as the codebase, which should be emailed to me. A complexity analysis of your solution as well as an empirical comparison with a serial implementation will earn extra credits. Note that your solution should not be so trivial that I can come up with a better algorithm within a few minutes. Please note that some of the problems will ask you to build multiple data structures because they involve similar solutions. Your grade will reflect the thought you have put into designing the solution as well as the quality of the report. Reports are due on or before December 8.

As each of these projects is to be done by one student rather than as a group, the assignment of the project is on a first come, first serve basis, although I reserve the right to choose among competing proposals based on my assessment of the likelihood of success by matching the proposed project to the student’s background and skill set. Notice that few of the problems have already been chosen due to their appropriateness to the skill sets of certain individuals in class.

You are to execute your programs in the OpenGPU Lab at opengpu00.umiacs.umd.edu through opengpu03.umiacs.umd.edu. If you need support, please send a message to the UMIACS staff at staff@umiacs.umd.edu.

1. Given a large collection of line segments of the form \(< x_1, y_1, x_2, y_2 >\) devise a parallel algorithm that outputs the decomposition of the space induced by the construction of a PM_1, PM_2, and PM_3 data structures on the input. Your output should be a set of Morton blocks along with the associated line segments. Note that a Morton block encapsulates both the position and size of a leaf block.

2. Given a large collection \(C\) of overlapping rectangles, for every rectangle in \(C\), determine the minimum enclosing quadtree block in the case of (a) a loose quadtree with an expansion factor \(p\), and (b) a partition fieldtree. Ask me for hints, if you are struggling. Your output should be a set of corresponding Morton blocks.

3. Given a region quadtree represented as a collection of Morton blocks, an expansion distance \(\varepsilon\), devise a region expansion algorithm. Your output should be a set of Morton blocks corresponding to the expanded regions.
4. Given a raster image which is represented as an array of pixels where 0 is black and 1 is white, perform a connected component labeling operation. Your output should indicate for every pixel its corresponding region identifier. Make your algorithm as efficient as possible. Credit will not be given for brute-force solutions.

5. Given a quadtree decomposition of space in the form of a set of Morton blocks, transform it into a decomposition that satisfied the restricted quadtree criterion.

6. Given a set of points in two-dimensional space, construct k-d and BSP trees for them. Define your output appropriately.

7. Given a region quadtree of a binary image containing several components, find its Euler Number and perimeter. The input is a set of Morton blocks.

8. Construct well-separated pair decomposition for a given set of points at arbitrary positions in a two-dimensional space of a given separation factor $s > 2$. The output is a pair of Morton blocks such that they are well-separated. Note that this is different from some of your work in fast multipole methods where the input is assumed to be regular. (Adam O’Donovan).

9. Given $n$ objects and their $n^2$ inter-object distances, construct a Lipschitz embedding for it. Your output should be the vector positions of the $n$ objects in a high-dimensional space.

10. Given a set of two-dimensional points as input, construct an AVD(3,0) on it. Next, modify your code to build an AVD $(3, \varepsilon)$ for a suitably defined $\varepsilon$. The output should be a set of Morton blocks along with their associated Voronoi sites.

11. Perform online document clustering on the GPU. The input is a set of document feature vectors. The output is a cluster identifier for every input document. Also try a distributed clustering involving all of the GPU computers in the GPU Openlab (Benjamin Teitler).