General Instructions:

1. Make a team – find a partner. Each project team is of two students.
2. Fill the form provided only once for the team.
3. All projects will involve trying the programs on reasonably large datasets such as square matrices of 50K rows, sparse graphs with millions of vertices and edges, dataset with millions of items, etc. This scale of inputs brings out the features of the implementations nicely.
4. All projects involve comparative studies of one algorithm vs another for the same problem. These comparisons should be plotted on suitable axes using some plotting tools.
5. Datasets for matrices and graphs can be downloaded from the University of Florida Sparse Matrix collection.
6. All the projects below have tremendous scope for continuing further to lead to a publication. Some of these ideas are not tried out so far. Interested teams can continue the project during summer.
7. Most projects can be implemented using C/C++ style of programming including reading/writing from files.
8. We will provide accounts on server class machines to test your programs.
9. Projects will require understanding a couple of algorithms and programming those algorithms.
10. We are not expecting that projects will use any kind of parallel programming. The papers may talk about parallel programming, but the algorithms will run without parallelism too.

Course Project Ideas:

Project 1: Implement the triangle counting algorithm and compare the performance of the algorithm with other leading approaches. As a second step, extend the algorithm to count 4-cycles in the graph.

Projects 2, 3, 4, 5: This project deals with graph biconnectivity testing and listing the biconnected components of the graph. The project compares two different algorithms for this problem and sees how these two algorithms perform on a variety of large graph inputs. Project 2 would consider the Tarjan-Vishkin algorithm and the Jen-Schmidt algorithm, Project 3 would consider Tarjan-Vishkin and Slota-Madduri algorithm, Project 4 would work with the Tarjan-Vishkin and the LCA-based algorithm, Project 5 would work with the Jen-Schmidt algorithm and the Slota-Madduri algorithm.

Project 6. The goal of this project is to study and implement the graph spanner algorithm by Baswana and Sen. The algorithm will be discussed in class but the base algorithm does leave lot of scope for algorithm engineering.

Project 7. The goal of this project is to study the problem of transitive closure of the graph. The transitive closure is closely related to the all-pairs-shortest-paths problem. This project would focus on the algorithm of Ullman and Yannakakis (High-Probability Parallel Transitive-

Closure Algorithms. SIAM J. Comput. 20(1): pg: 100-125, 1991). This paper introduces multiple algorithms that improve on the default algorithms.

Project 8, 9, 10. This set of projects deal with the graph diameter problem. Beyond the algorithm of Crescenzi that we discussed in class, there are multiple algorithms for this problem. Couple of them are the one by Takes and Kosters titled Determining the diameter of small world networks. CIKM 2011: pg. 1191-1196. The paper by Shun is titled An Evaluation of Parallel Eccentricity Estimation Algorithms on Undirected Real-World Graphs. KDD 2015: 1095-1104.

Project 8 would study the algorithm of Crescenzi and the algorithm of Takes and Kosters, implement both algorithms, and see which of these does better based on a variety of graph inputs. Project 9 would study the algorithm of Crescenzi and the k_BFS algorithm from the paper by Shun. Project 10 would study the algorithm of Takes and Kosters and the k-BFS algorithm of Shun. In each of these projects, the intention is to understand the algorithms, implement these algorithms, and compare their performance on various large graph inputs.