General Instructions:

1. 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.
2. 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.
3. Datasets for matrices and graphs can be downloaded from the University of Florida Sparse Matrix collection.
4. Any of the projects below have tremendous scope for continuing further to lead to a publication. Some of these ideas are not tried out so far.
5. Most projects can be implemented using C/C++ style of programming including reading/writing from files.
6. We will provide accounts on server class machines to test your programs.
7. Projects will require understanding a couple of algorithms and programming those algorithms.

Course Project Ideas:

Project 1: Study the biconnectivity algorithms of Tarjan and Schmidt for their practical differences in run time, sparse vs dense graphs, and the like.

References: Tarjan algorithm from the CLRS book, Schmidt (IPL, 2015)

Skills: C/C++, graphs as adjacency lists (not matrices),

Project 2: t-Spanner construction

In this project, we study algorithms for constructing a t-spanner of a given graph. These algorithms, for instance the one by Sen and Baswana, use randomization too to thin out the edges of G. There are several possible engineering choices in these algorithms.

Skills: C/C++, can try parallelism also, graphs as adjacency lists (not matrices),

Project 3: Diameter or Eccentricity computation based on Crescenzi et al. and other parallel algorithm papers by Shun et al.

Project 4: Approximate Distance Oracles (Zwick 2001),

Project 5: Library sort – read the paper by Bender, and compare it other standard sorting algorithms such as insertion sort, quick sort, etc.

Project 6: Modify the algorithm of Tarjan-Vishkin to incorporate Union-Find data structure as we add edges to G'. Check how this works in practice wrt other algorithms such as Tarjan's algorithm.

Project 8: The transitive closure algorithm of Yannakakis et al. This is a good parallel algorithm that can be compared with other approaches.

This requires studying the algorithm and implementing the same in parallel.

Project 10: Implement Bader-Cong along with Tarjan-Vishkin and Union-Find. Check also Project 6 – Is Bader-Cong essential in light of Union-Find?