The class project should involve reading, summarizing, analyzing, and critiquing on one or more research paper giving an application of algorithms. Generally speaking, a two-person project involves one research paper; a five -person project should compare two or more related research papers. An exception can be made if a single paper is long or difficult. I would suggest 5 pages as a bare minimum, with say an additional 2 pages per person in the group over 2, but there are no strict length limits.

The project must have the following elements:

1. An application where algorithms are important.

2. A discussion of the application and associated constraints on algorithms.

3. A formal statement of (at least one) algorithmic problem to be solved. Be sure to distinguish between the problem specification and the algorithmic model. Don’t confuse constraints/costs for the ALGORITHM (e.g, “Must be on-line”) with constraints on /costs of the SOLUTION (e.g., “Must minimize the number of cache misses”).

4. A clear presentation of at least one algorithm.

5. A time analysis or other cost (parallel time, memory, distributed time, stability) analysis as appropriate. If the time analysis is not clear, give the best upper bound you can. Think about using data structures to improve the time over naive implementations. Give as detailed a time analysis as possible, in as many parameters as are relevant. (O(nm) is more revealing than O(n^3).) and avoid treating parameters as “constants” just because they are fixed in standards. If nothing is proven about the algorithm’s time, this might be a place to try experiments with implementations.

6. A correctness discussion/ proof for at least one algorithm. Prove correctness if correctness is true and known. Otherwise, give examples where the algorithm is incorrect or not optimal. Discuss the approximation ratio for the algorithm. How far off from optimal can the algorithm be? What is guaranteed about the algorithm’s performance? If nothing is proven about the algorithm’s performance, this might be a place to try experiments with implementations.

7. A discussion of how well the algorithms presented fit the application. A critique of assumptions made by the algorithm designers. A comparison of alternative algorithm approaches for the application. This can also involve implementation and experiment, or just comparing the analyses but with realistic parameters taken from “real-world” instances.

Suggested topics:
Parallel algorithms Look at how algorithms for basic problems like matrix multiplication and sorting need to be modified to take advantage of parallelism. Starting place: see Scott Baden’s CSE 260 syllabus.

Cryptanalysis: lattice reductions How is the lattice reduction algorithm of Lenstra, Lenstra and Lovasz used in cryptanalysis?

Cryptanalysis: factoring The quadratic sieve method for factoring large integers.


Vision Shape recognition and object recognition using shape contexts. Belongie, Malik, Puzichar. Concentrate on the problem of finding the closest match spatial transformation.


Graphics Inverse Kinetics. See Sam Buss’s website.


Circuit design Arithmetic algorithms for hardware implementation. See CSE 246 syllabus.

Circuit design Wang, Zhang, Chen, Yang, Cheng, Graham. Bus Matrix synthesis ased on Steiner Graphs for Power Efficient System on a chip

Circuit design Albrecht, Kahng, Mandiou, Zelikowsky, Multicommodity Flow Algorithm for Buffering Global Routing, 2005. See also prior ISPD paper by Albrecht.


Data Mining and other applications The Johnson-Lindenstrauss transform. Starting place: Dasgupta paper.
**Data Mining** Differential privacy. Starting place: papers by Cynthia Dwork.


**On-line algorithms** Starting place: Online computation and Competitive Analysis, Borodin and El-Yaniv (book, project should be based on e.g., first chapter).

**Streaming algorithms** Starting place: S. Muthukrishnan, Data Streams: Algorithms and Applications, Foundations and Trends in TCS. (Pick one problem.)

**Computational economics** Algorithmic mechanism design. Starting place: Nisan and Ronen, 1999.


**Coding** Achyara, Das, Orlitsky, Pan. Algebraic computation of pattern maximum likelihood.

**Network routing** Levchenko, Voelker, Paturi, Savage: XI: an efficient network routing algorithm.

**Improved SAT algorithms** Compare approaches of Paturi, Pudlak, Zane and of Schoning.