Final Project Guidelines

Fall 2024

Final Project Summary

The final project will be done either individually or in groups of 2.

The idea of the group project is to have similar scope to a Homework or Assignment: it should involve some amount of understanding an algorithm and implementing it to achieve improved performance. It's up to each student what topic they work on; it could be a topic from earlier in the class explored in more depth, or a new topic.

The final deliverable will be the code from the project, and a writeup explaining what the group did.

The writeup should explain any background research that the students did for the project, as well as the results, and what the actual code being submitted is. The length of the writeup may very: a project with straightforward but substantial code will likely have a shorter writeup; a project that consists mostly of background research combined with a small amount of code will have a longer writeup; a project focusing on comparing performance of different approaches will have a longer writeup but one consisting significantly of experiments. With this variation in mind, a length to shoot for would be roughly 3-5 pages of text (and potentially additional space for figures).

On the last day of class, we'll also have each group do a very short ($\approx$ 5–10 minute) presentation on the topic they've been working on.

1 Schedule

- **Thu, Nov 21:** project proposals due (consists of an email to me (Sam) saying who your group is and the topic.)
- **Tue, Dec 3:** Each group will meet with the instructor (again, Sam) to check in on current progress, and make sure their plan for what to do has technical content matching the learning goals of the project.
- **Fri, Dec 6:** Each group will give a $\approx$ 5 minute presentation on their topic and their progress so far.
- **Tue, Dec 10:** Final project due.

2 Grading

The project will be graded using letter grades, much like the homeworks and assignments.

In terms of technical content, the idea is for the total amount done to be roughly similar to an average homework or assignment in the course. Some groups may emphasize theory or code more heavily; similarly, it may either emphasize understanding previous work or coming up with new results. Any of these are fine so long as the overall contribution demonstrates sufficient knowledge of the topic.

Of course, terms like "sufficient" and "similar" are not objective. The idea of the meeting on December 3rd is to check in with each group and specifically discuss to make sure that the project has the correct scope. The instructor is available over email or Slack if you have questions as well.

Grading Breakdown:

- Final submission: 80%
- Presentation: 10%
- Initial plan and Dec 3 meeting: 10%

3 Potential Topic Ideas

3.1 Space/Time Tradeoff Algorithms

- The two towers problem can be solved in $2^{n/4}$ rather than $2^{n/2}$ space (the running time remains the same asymptotically). Implement this algorithm and compare it to the algorithm from class. The priority queue should have degree more than 2 for cache-efficiency.
  - Original paper: https://dspace.mit.edu/bitstream/handle/1721.1/148974/MIT-LCS-TM-147.pdf
  - More recent exposition (in Appendix A): https://arxiv.org/pdf/2010.08576
- Use SIMD instructions to speed up your edit distance implementation. The entire computation (or very close to it) should take place using SIMD.

3.2 Randomized Algorithms

- Implement an adaptive cuckoo filter. Compare the performance of this filter with the filter you had in class in insert performance, query performance, and false positive rate. Investigate how changes to the underlying table (e.g. using linear probing) affect performance.
  - Adaptive Cuckoo Filter Paper: https://arxiv.org/pdf/2105.10622
- Implement the SimHash cross-polytope LSH for cosine similarity. Generate data to test your implementation (the data should generate a large number of random vectors, and plant one close pair of vectors). How many repetitions (what value of k) did you find works best?
  - Lecture notes: http://madscience.ucsd.edu/notes/lec9.pdf
  - Blog post: https://aerodatablog.wordpress.com/tag/simhash/
  - Original paper (Section 3): https://www.cs.princeton.edu/courses/archive/spr04/cos598B/bib/CharikarEstim.pdf
- Generate set data from a real-world application where Jaccard similarity roughly corresponds to the similarity of real-world elements. Implement a heuristic to search for close pairs (for example, by looking for elements with similar rare elements first). How does it compare to MinHash?

3.3 LP/ILP/MIP

- Apply LP or MIP to a real-world problem
- Compare LP or MIP algorithms/libraries on difficult-to-solve problem instances. Is there a time when one does better than the other?
- Investigate the two towers problem using ILP. Can you get an accurate solution without rounding issues using `C`? Are there inputs for which the ILP does give an accurate solution, and is faster than the meet in the middle approach?

3.4 String Algorithms

- Implement the Burrows-Wheeler transform with an efficient ($O(n \log n)$ or better) suffix array algorithm, and apply it to compress the `proust.txt` dataset using the move-to-front algorithm.

3.5 Misc

- Compare several Minimum Spanning Tree Algorithms (Prim's, Kruskal's, Boruvka's; applying a Fibonacci heap to Prim's). Which is fastest? Does the answer change on different types of graphs?
- Implement a linear-space Van Emde Boas tree and compare its performance to a binary search tree