Final Project

CSCI 5454: Algorithms
CU Boulder

Fall 2020

1 Description

The final project is a rigorous investigation of a problem and (an) algorithm(s) for solving it. Applying principles from the class, a typical project will ask:

- What is the formal **description** of the problem? What does it mean to solve this problem correctly or optimally?
- Is the algorithm **correct**, optimal, or approximately so?
- What are the important **resources** available to the algorithm (such as time and space), and how **efficiently** does it use those resources?
- How does the algorithm compare to alternatives, e.g. naive brute-force approaches or other proposed algorithms?

Resources used to accomplish this may include textbooks, research papers, open source code, or more.

2 Logistics and Assessment

Project groups may consist of 1, 2, or 3 people.

- By **Thu, October 22**, the group will submit a project proposal on Gradescope, consisting of one to two paragraphs.
- By **Thu, November 5**, the group will submit a project update on Gradescope of about one page.
- By **Tue, November 23**, the group will submit a project update on Gradescope of about one page.

- By **Tue, December 2**, the group will create a one-page slide or "poster" summarizing their work, and present it to the class in a poster session (more details to be announced, including for asynchronous students).
- By **Fri, December 11, 7:00pm Mountain time**, the group will submit a project summary and report on Gradescope.

Remember that the final report is *not* your project. Your project is a deep investigation over time with your group. The final report is only a condensed summary of your investigation and a writeup of your rigorous findings.

3 Kinds of Projects

Projects generally focus on at least one of these kinds of approaches:

- **Theoretical** investigation, including some proofs of correctness (or approximation, etc) and/or efficiency.
- **Empirical** investigation, including implementations and experiments to understand an algorithm and/or setting.
- **Expository** to explain key concepts and intuition behind a particularly difficult algorithm.

Note that **it is not enough to implement an algorithm**, no matter how complicated. The project must include some rigorous **investigation** of the algorithm. For example, it may implement the algorithm on a variety of real-world datasets, and compare its performances to some baseline or the theoretical performance guarantee. In this case it should include discussion on e.g. how well the algorithm scales, or factors of the input instances that affect the algorithm's performance.

It is also not required to implement an algorithm, if the project focuses on a theoretical investigation. This could include coming up with new proofs, or simplifying or improving existing proofs in your own words. As a rule of thumb, if the problem is written up in a textbook, then rewriting proofs is probably not enough; if the problem is only written up in research papers, it may be enough.

4 Example Topics

Groups are encouraged to propose their own topic, especially one relevant to their outside interests.

Some of the below topics are easier than others (e.g. the algorithm is simpler). Generally, the easier the topic, the higher the bar. For example, for a quite difficult algorithm such as the AKS primality test, a mostly-expository project may be sufficient. For a simpler algorithm

or data structure, we would expect significant theoretical and empirical investigation, ideally in a real-world context.

- **Hashing, Streaming**
  - Implementations of hash functions: efficiency, collisions, resistance to manipulation, etc.
  - Advanced implementations of hash tables: linear and quadratic probing, cuckoo hashing, power of two choices, etc.
- **Data Structures**
  - Fibonacci heap
  - Union-find and Kruskal's minimum spanning tree algorithm
  - Skip lists
  - Tries, hash tries
- **Approximation, Randomized, Online Algorithms**
  - Prophet inequalities or secretary problems
  - Kelly betting
  - Submodular optimization algorithms
  - k-server problem
  - Variants of online bipartite matching
  - For students who've studied linear programming: Primal-dual based algorithms; randomized (or deterministic) rounding, ...
  - Online convex optimization: mirror descent, follow the regularized leader, follow the perturbed leader, ...
  - More on Las Vegas vs Monte Carlo
  - Concentration inequalites / tail bounds and their uses in randomized algorithms
- **Core Algorithms, Theory**
  - Greedy algorithms: matroids, subset systems, optimality of greedy.
  - Matrix multiplication
  - Graph isomorphism (!)
  - Max flows via electrical networks
  - Approximations for traveling salesman: planar, Euclidean, other metrics, etc.
  - Min-cost max flow; min-cost bipartite matching; Hungarian algorithm

  - Maximum matching in general graphs; blossom algorithm
  - Semidefinite programming and max-cut approximation algorithm
  - Steiner tree problems, parameterized complexity
  - Geometric algorithms: convex hull, use of quaternions, ...

- **Cryptography, Number Theory**
  - Primality testing: AKS primality test; Miller-Rabin probabilistic primality test.
  - Cryptographic protocols: RSA, Diffie-Helman, elliptic curve crypto, blockchain tech (e.g. Merkle trees and digital signatures), or more advanced/theoretical topics like zero-knowledge proofs, obfuscation, or homomorphic encryption.
  - Attacks on cryptographic protocols, e.g. finding collisions in cryptographic hash functions, or finding preimages. Pollard's Rho algorithm, tortoise-and-hare cycle-finding algorithm.

- **Quantum Computing**. Model, Grover's search or Shor's factoring algorithm.

- **Game Theory, Markets**
  - Game-playing: Monte Carlo Tree Search (e.g. AlphaZero chess and Go bots), poker playing bots, etc.
  - Auctions and resource allocation (ask prof. or TA for more ideas here).
  - Market equilibrium, tatonnement.
  - Resource allocation problems; reducing to network flow or linear programming.
  - Prediction markets, market scoring rules

- **Big Data, Machine Learning**
  - Bio-related projects: DNA sequencing problems, protein folding, ...
  - Data mining in graphs: community detection, finding structures or subgraphs. Strongly connected components, condensation graphs, etc.
  - Advanced optimization topics and techniques; non-convex optimization
  - Clustering, k-means, k-medians
  - Bandits algorithms for online learning settings

- **Artificial Intelligence**
  - A* search in graphs, general graph search
  - Anytime algorithms
  - Multi-armed bandits; Gittins index theorem

  - Expectation-maximization algorithm

- **Distributed or Parallel Computing**
  - Distributed data stores: consistency, consensus, CAP theorem.
  - Parallel algorithms (pick one or a small group)

- **Functional programming** algorithms and data structures.