# CSCI 411 - Advanced Algorithms and Complexity
# Project

January 19, 2024

This project is an opportunity for you to investigate an algorithm that interests you. The main goal is for you to analyze and implement a complex, real-world algorithm on your own and to communicate your findings effectively to a technical audience.

The report, implementation, and feedback response components of the project will be due on **April 7th by 11:59 pm**. Submissions will be through Canvas. The report, presentation, and feedback responses should be submitted as PDFs while your implementation should be submitted as a .cpp file. Presentations will take place one to two weeks before the due date. More information regarding the format of each piece of the project is below.

This is a big project but you have a lot of freedom in the algorithm that you choose to study. Pick an algorithm in a field that you are excited about! Be sure to get started early and have fun!

## (60 pts) Report

There is no length requirement for the report. As a general guideline, the report, including pseudocode and figures, will likely be between 3 and 10 pages. It should be as clear and concise as possible while providing a complete, self-contained narrative. Below are some suggested sections for the report along with what they could contain.

- Introduction - the introduction may include a brief history of the algorithm along with a description of its purpose and potential applications. This section should address why the algorithm is important.
- Intuition - a high level but complete description of how the algorithm works.
- Pseudocode and Detailed Description - formal pseudocode of the algorithm along with a more specific description. While not necessarily a formal proof of correctness, this description should include an argument that the algorithm works as expected. Expectations regarding inputs and outputs should be unambiguous and clear.
- Run Time Analysis - a tight asymptotic analysis of the algorithm's run time. Make sure notation and variable meanings are clear. This analysis does not need to consider each line of the algorithm individually but it must address all major components of the algorithm including data structure creation and upkeep as well as any pre- and post-processing steps. This section may also include descriptions and results of different experiments. For example, figures showing run time as a function of input size or a brief comparison of this algorithm against other approaches to the same problem. Such experiments are not required.

Do not forget to include proper citations of any resources that you use.

## (25 pts) Implementation

Implement your algorithm, following the intuition and pseudocode provided in the report, in C++. Your code should conform to standard style guidelines, should be well commented, and must include test cases verifying that the implementation works as expected. This includes both inputs and expected outputs. Depending on the complexity and size of the algorithm you choose, 5 to 10 test cases should be sufficient. Make sure to include a brief comment in `main` or in a README.md file describing how to run your tests and how to run your code on different inputs.

## (15 pts) Presentation

One to two weeks before the project is due, you will give at least one 5-10 minute presentation to a small group of classmates via Zoom. The goal of the presentation is for you to communicate the important aspects of your algorithm to your peers. Your presentation should touch on intuition, pseudocode, and run time but does not need to include the full detail expected in your report. However, it should be well practiced. You must submit materials used for your presentation (e.g. slides) as a PDF on Canvas.

After each presentation, the audience will give the presenter feedback, verbal and written, to help improve their communication. This feedback should be used to enhance the report before submission. In written feedback, the audience can use the following questions as a guide:

- What parts of the presentation were most effective?
- Did you find any aspects of the algorithm particularly confusing or unclear? If so, what might help to clarify those parts?
- After the presentation, do you know why the algorithm is important, what the general idea behind the algorithm is, and how efficient it is asymptotically?

The presenter should submit a copy of this feedback along with names of the audience members and brief comments on how the feedback affected the final report.

## Example Algorithms

Here is a short list of algorithms that you might consider investigating. You are also more than welcome to pick an algorithm not on this list. Feel free to use tools like ChatGPT to explore ideas but be sure to cite them appropriately. No matter which algorithm you decide on, please discuss your choice with me before getting started. Pick an algorithm early so that you have plenty of time to work on the project. You must choose and have approval of an algorithm by **March 8th**.

- Hopcroft-Karp algorithm
- Gale–Shapley algorithm
- Christofides algorithm
- Fortune's algorithm
- Bron–Kerbosch algorithm
- Sudoku solver

- Decision tree learning
- $k$-nearest neighbors (non-naive implementation)
- Agglomerative hierarchical clustering
- Hedge algorithm

Some algorithms can be implemented using different data structures or techniques. These differences may or may not affect the run time. You do not need to focus on the fastest or most efficient approach for this project. However, all parts of the project should consider the same approach. In particular, your intuition, pseudocode, run time analysis, and implementation should match. If you want, you may mention other approaches, and how they differ from the one you chose, briefly in the report.

It is also worth mentioning that some algorithms are easier to analyze and implement than others. Be aware of this when picking an algorithm. Some consideration will be given to exceptionally difficult projects and the bar will be higher for exceptionally easy projects. This project is worth 15% of your final grade. As a result, I expect you to spend a significant amount of time between March 8th and April 7th polishing your pseudocode and your analysis and fine tuning your implementation. If you are worried about picking an algorithm that is too hard or too easy, or if you have any other questions, don't hesitate to ask.