Algorithm Analysis & Design: Final Project Guidelines

1. Overview

The final project is the capstone of this course. It is a group-based assignment designed to provide you with an opportunity to deeply explore a set of related algorithms, implement them from scratch, and rigorously analyze their performance.

Your group's submission will be evaluated on three components:

1. **A comprehensive written report.**
2. **A code implementation.**
3. **A final 50-minute presentation and evaluation.**

The goal is not just to make something that *works*, but to analyze *how* and *why* it works, and to empirically validate the theoretical concepts of complexity and efficiency discussed in class.

2. Submission Deliverables

2.1. Code Implementation

This is the core technical component. The central requirement is that all algorithms must be implemented from scratch.

- **"From Scratch" Mandate:** You **must** implement the core logic of all algorithms yourself.
    1. **Allowed:** Using standard library data structures (i.e., lists, hashmaps, priority queues, dictionaries, vectors) and language built-ins.
    2. **Not Allowed:** Using an external library package that implements the algorithm for you (i.e., networkx.dijkstra, scipy.spatial.ConvexHull).
- **Programming Language:** You may use any programming language.
    1. **Warning:** Be prepared to answer detailed questions about your code, including language-specific syntax, data structures, and library choices. Choosing an obscure language invites questions about its mechanics.
- **Submission Format:** A link to a **public** GitHub repository attached in the report.
- **Repository Requirements:**
    1. **README.md:** A clear guide on how to compile, install dependencies, and run your project.

2. **Well-Commented Code:** Your code must be readable, well-structured, and commented to explain complex sections.
3. **Test/Benchmarking Harness:** Your code must include the scripts you used to run your experiments and generate the data for your report.
4. **Docstrings:** All functions must contain docstrings to explain their purpose and specify the data type of the input and output of the function.
5. **Modularized code:** Each algorithm should be in a separate file. You are allowed to create other files to store helper functions.

2.2. Project Report

This is a formal, academic-style report detailing your project. It should be structured as follows:

1. **Title Page & Abstract:** Project title, team members, and a brief 150-250 word summary of the project, its methods, and its findings.
2. **Introduction:** Define the problem(s) you are solving, discuss their real-world relevance, and state the objectives of your project.
3. **Algorithm Descriptions:** For *each* algorithm implemented, provide:
    - A clear, theoretical explanation of how it works.
    - *Correct* asymptotic analysis (Time and Space complexity).
4. **Implementation Details:** Discuss your key design choices. What data structures did you use and why? What were the most significant implementation challenges?
5. **Experimental Setup (if applicable):**
    - **Environment:** Your hardware and software (language version, libraries).
    - **Datasets:** Describe the datasets you used (i.e., synthetic data generation, real-world data sources).
6. **Results & Analysis:**
    - **This is the most important section.**
    - Present your results clearly using **graphs, charts, and tables.**
    - **Metrics:** Mention what metrics you used i.e., wall-clock time, memory usage, solution quality, number of comparisons).
    - Compare the empirical performance of your algorithms against each other and against their theoretical complexities.
    - Discuss *why* you see these results.
7. **Conclusion:** Summarize your findings, discuss the limitations of your work, and suggest potential future improvements.
8. **References:** Cite any papers, books, or online resources you used.
9. **NOTE: Clearly add a section mentioning what parts of the report are bonus. Label this under the heading "Bonus Disclosure" before the References section. Mention specifically which algorithms, metrics, etc. should be used for bonus evaluation. You cannot change this post-submission.**

3.3. Final Presentation

This is a slide-based presentation that summarizes your project for the class and instructor.

- **Content:**
    - Brief introduction to the problem.
    - High-level overview of the algorithms you implemented (no deep-dives into code).
    - **Focus on the results:** Show your most important graphs and charts.
    - On slides covering bonus content (mention “BONUS” in large font at the top right of the slide)
    - Discuss your key takeaways, interesting findings, and challenges.
    - A final conclusion.

4. Evaluation Structure (45 Minutes Total)

The final evaluation is a 45-minute timed slot for your group. The time will be strictly managed.

- **Phase 1: Group Presentation (15 minutes)**
    - Your group will deliver your prepared presentation.
    - This portion is led by the group.
- **Phase 2: Code Demo & Walkthrough (10 minutes)**
    - The code will be run live according to instructions provided in the README.
    - This may involve:
        - Running your benchmarking script.
        - Demonstrating a visual component (if applicable).
        - Walking through 1-2 key sections of your "from-scratch" code to prove authorship and explain the logic.
- **Phase 3: Group Q&A (10 minutes)**
    - The panel will ask questions to the group as a whole.
    - These questions will focus on your design choices, analysis, and high-level concepts.
- **Phase 4: Individual Q&A (15 minutes total; 3 minutes per member)**
    - The panel will ask 2-3 specific questions to each group member individually.
    - **Individual Responsibility:** You are expected to be an expert on the part of the project you were primarily responsible for. You are also expected to have a high-level understanding of all other parts. General, top-level questions about any part of the project are fair game.

5. Marking Scheme (100 Marks)

Your final project grade will be calculated out of 100 marks, broken down as follows.

Category	**Component**	**Marks**	**Description**
A. Project Report	**(40)**		
	Theoretical Analysis	10	Correctness of proof and accurate, detailed complexity analysis.
	Experimental Design	5	Quality of datasets, metrics, and test environment.
	Results & Empirical Analysis	20	Quality of graphs/tables. Depth of discussion linking theory to practice.
	Writing & Structure	5	Professionalism, clarity, formatting, and proper citations.
B. Code Implementation	**(25)**		
	Correctness & Functionality	10	The code runs, produces correct results, and solves the problem.
	"From-Scratch" Adherence	5	All core algorithm logic is original and not from an external library.
	Code Quality	5	Code is readable, well-commented, and well-structured.
	Test Harness & Reproducibility	5	README is clear and experiments are easy to reproduce.
C. Final Presentation & Defense	**(35)**		
	Presentation Quality	10	Clarity of slides, professionalism, pacing, and time management.

	Group Q&A	10	Depth and correctness of answers to group-level questions.
	Individual Q&A	15	Demonstrated expertise in your own component and project-wide knowledge.
Total		**100**	

NOTE: the 100 marks listed here is for the 20% requirement. Bonus marks are up to the discretion of the secondary TA based on the evaluation.