Meeting Note 1

Date: 2025-10-01 **Time:** 14:00 - 14:45 **Location:** Supervisor's Office (PQ706) **Attendees:** LIN Zhanzhi, CAO Yixin (Supervisor)

1. Objectives

- Review the initial draft of the Capstone Project Proposal.
- Discuss the feasibility and scope of implementing Dual-Pivot Quicksort in C++.
- Identify key academic and technical requirements for the project.

2. Progress Report

- Selected project topic: "Implementing Dual Pivot Quicksort in C++".
- Drafted a preliminary proposal outlining the basic implementation plan.
- Conducted initial reading on Yaroslavskiy's 2009 paper.

3. Discussion & Feedback

Key Discussion Points

- **Scope Definition:** The initial proposal focused primarily on the coding aspect. We discussed the need to elevate this to an academic research project by adding rigorous performance analysis and theoretical backing.
- **Language Standards:** Discussed the choice of C++23. Supervisor agreed it is a good choice to demonstrate modern software engineering practices (e.g., using `std::ranges` and concepts).
- **Benchmarking:** Simply implementing the algorithm is not enough. The core value lies in comparing it against existing giants like `std::sort` (Introsort) and PDQSort.

Supervisor Feedback

> "The current draft is too engineering-focused. You need to justify *why* this is a research gap. Java adopted Dual-Pivot years ago; why hasn't C++? You should investigate the 'scanned elements model' by Wild to explain cache behavior, rather than just counting comparisons."

> "Expand the methodology to include a systematic comparative analysis. Don't just measure time; measure cache misses and branch mispredictions."

4. Issues & Challenges

- **Theoretical Depth:** I admitted I was unfamiliar with the mathematical analysis of Dual-Pivot Quicksort (e.g., Aumüller’s framework).
  - *Resolution:* Supervisor provided references to key papers by Aumüller and Dietzfelbinger to strengthen the literature review.
- **Benchmarking Fairness:** How to ensure a fair comparison between my implementation and the highly optimized STL?

    - *Resolution:* We agreed to use a standardized benchmarking framework (like Google Benchmark) and test across various distributions (random, sorted, reverse).

5. Action Items

- [ ] Read Aumüller and Dietzfelbinger’s paper on Dual-Pivot analysis. (Due: 2025-10-10)
- [ ] Read Wild’s paper on the "scanned elements model". (Due: 2025-10-10)
- [ ] Rewrite the "Background" section to highlight the gap in C++ standard libraries. (Due: 2025-10-15)
- [ ] Expand "Methodology" to include specific metrics (L1/L2 cache misses, branch prediction). (Due: 2025-10-15)
- [ ] Submit the revised, final proposal. (Due: 2025-10-24)

6. Next Meeting Plan

- **Target Date:** 2025-11-05
- **Focus:** Review of the revised proposal and initial code structure.