

# Meeting Note 2

---

**Date:** 2025-11-05 **Time:** 14:00 - 14:30 **Location:** Supervisor's Office (PQ706) **Attendees:** LIN Zhanzhi, CAO Yixin (Supervisor)

## 1. Objectives

- Review progress on the initial C++ implementation of Dual-Pivot Quicksort.
- Discuss findings from the literature review (Wild and Aumüller papers).
- Plan the structure for the upcoming Interim Report.

## 2. Progress Report

- **Proposal:** Final project proposal submitted on Oct 24.
- **Literature Review:** Completed reading of "Dual-Pivot Quicksort: Optimality and Analysis" (Aumüller) and Wild's work on scanned elements.
- **Implementation:** Created a basic working prototype of 1-pivot and 2-pivot quicksort in C++.
  - Current status: Works for `std::vector<int>`.

## 3. Discussion & Feedback

### Key Discussion Points

- **Code Quality:** Showed the initial code. Supervisor noted that while it works for `int`, it needs to be generic.
- **C++23 Concepts:** Discussed using `std::sortable` and `std::random_access_iterator` concepts to ensure type safety.
- **Interim Report:** Supervisor outlined the expectations for the Interim Report.

### Supervisor Feedback

"Your current implementation is too specific to integers. You must use C++ templates and Concepts to make it a true library candidate. Ensure it handles custom comparators correctly."

"For the Interim Report, don't just paste code. Focus on the *design decisions*. Why did you choose this specific partitioning scheme? Use diagrams to explain the 3-way partition."

## 4. Issues & Challenges

- **Generic Implementation:** I am struggling slightly with the syntax for C++20/23 Concepts for custom iterators.
  - *Resolution:* Supervisor suggested looking at the `std::ranges::sort` implementation in the GCC or LLVM open-source repositories for reference.
- **Benchmarking Setup:** Setting up Google Benchmark is proving tricky with the custom build system.
  - *Resolution:* Advised to stick to a simple `CMake` setup to manage dependencies.

## 5. Action Items

- Refactor code to use C++ Templates and Concepts (Generic programming). (Due: 2025-11-20)
- Implement a "Benchmark Runner" using Google Benchmark. (Due: 2025-11-25)
- Draft the "Methodology" and "Literature Review" sections for the Interim Report. (Due: 2025-12-01)

## 6. Next Meeting Plan

- **Target Date:** 2025-12-03
- **Focus:** Review of Interim Report Draft and preliminary benchmark results.