Sorting Algorithm Functions Proficiency Rubric

C++ implementations and Big-O efficiency analysis

EMERGING

Missing algorithms or major errors in code and analysis

- Fewer than 5 algorithms completed; code has major errors or does not compile
- Big-O values incorrect for most cases; explanations missing or wrong
- Bubble sort line-by-line analysis missing or shows no understanding of counting operations
- Submits more than 3 days late, wrong format, or major sections left empty

DEVELOPING

All algorithms attempted but code or analysis has errors

- All 5 functions written but some have bugs or do not sort correctly
- Big-O values mostly correct but explanations are vague or incomplete
- Bubble sort analysis attempted but line counts or Big-O per line has errors
- Submits 1–2 days late or last-minute extension; most sections filled but rushed

PROFICIENT

All algorithms work correctly with accurate Big-O analysis

- All 5 sorting functions compile and sort arrays correctly
- Big-O tables complete with correct values and clear explanations for best/average/worst
- Bubble sort line-by-line analysis correctly counts operations and derives total Big-O
- Submits on time as notebook, communicates delays, helper functions included where needed

EXTENDING

Deep understanding shown through code quality and insightful analysis

- Code is clean, well-commented, and uses efficient implementations (e.g., early exit in bubble sort)
- Explanations connect Big-O to algorithm behavior: why best case differs from worst case
- Line-by-line analysis shows understanding of nested loops and how input size affects runtime
- Submits quality work early; could explain the analysis to another student

Assessment Note: Complete the notebook with working C++ functions for all 5 sorting algorithms. Fill in the Big-O summary tables with best, average, and worst case analysis. Bubble sort requires a detailed line-by-line analysis showing how you calculated the total Big-O.