Sejong University
College of AI Convergence
Department of Artificial Intelligence and Data Science
Fall Semester, 2025

Algorithms and Lab (알고리즘및실습)

Term Project: List of Topics

1. **Topic:** Sorting Playground < Only one team may choose this>

- **Goal:** Implement Insertion, Merge, Heap, and Quick from scratch; compare on different inputs.
- **Core:**
  - Input generators: random, nearly-sorted (≤10% disorder), reversed; variable size n.
  - Report runtime per algorithm and show first/last 10 elements to verify correctness.
  - UI (pick one): Simple desktop GUI (Tkinter/PyQt) or web UI (HTML/JS + Flask/FastAPI) with Start / Stop / Progress and a runtime chart.
- **Tests:**
  - Include at least one edge case: many duplicates; small n≤10; already sorted.
- **Stretch (optional)**
  - Hybrid QuickSort (switch to Insertion Sort for small subarrays), median-of-three pivot, 3-way partition for duplicates; cache-aware mergesort.

2. **Topic:** Stack & Queue Simulator

- **Goal:** Interactive visual tool for stacks and circular queues (capacity n stores at most $n - 1$ items).
- **Core:**
  - Operations: push/pop (stack), enqueue/dequeue (queue); show head/tail indices, overflow/underflow messages, and current contents.
  - UI: Buttons + live state view (desktop or web).
- **Tests:**
  - Show wrap-around behavior in the queue; demonstrate overflow and underflow.
- **Stretch (optional)**
  - Mini "printer queue" simulation with job arrivals and service order.

3. **Topic:** Max-Heap Task Prioritizer

- **Goal:** Build a max-heap priority queue for tasks (title, priority).
- **Core:**
  - Operations: insert, peek-max, extract-max; display internal array after each op.
  - UI: Dashboard (desktop or web) with an operation log and "current max".
- **Tests:**
  - Compare insert/extract time vs. a naïve sorted list on the same workload.
- **Stretch (optional)**
  - Aging (increase priority over time) to avoid starvation; batch insert from file.

4. **Topic:** BST Mini-Dictionary

- **Goal:** Implement a Binary Search Tree dictionary; optional Red-Black Tree comparison.
- **Core:**
  - BST insert/search/delete (no duplicates), load words from file, print in-order.
  - UI: Tree visualizer that redraws after each operation (nodes + edges).

	• **Tests:** - Try worst-case insert order (sorted keys) and compare height. • **Stretch (optional)** - Red-Black Tree insertion + rotations; show how balanced height improves lookups.
5.	**Topic:** Graph Explorer + BFS/DFS
	• **Goal:** Read a graph; implement adjacency list and matrix; run BFS/DFS. • **Core:** - Load from simple edge-list file; run BFS/DFS from a chosen node; report visit order, tree edges, and component count. - Topological sort on DAGs; SCCs for directed graphs. - UI: Animate traversals; step through BFS/DFS. • **Tests:** - Include graphs with multiple components and a directed graph (for DFS edge types).
6.	**Topic:** Shortest Path on a Grid
	• **Goal:** Treat a text grid as a graph; compute paths. • **Core:** - BFS (unweighted) and Dijkstra (weighted cells). Print the path (arrows or coordinates). - UI: Clickable start/goal; live path highlight; simple grid editor. • **Tests** - Walls/blocked cells; unreachable goal; large vs. small grids. • **Stretch (optional)** - A* with Manhattan heuristic; compare explored node counts vs. BFS/Dijkstra.
7.	**Free Topic (Proposal-Based)**
	- 1-page proposal (goal, input, algorithms used), approved by instructor. - UI: Desktop GUI or simple web UI that showcases key algorithm steps.