CSE221 Data Structure

Fall, 2025, Homework #2
Due: December 7, 2025 (Sun), 23:59 KST

Instructor: Taesik Gong (taesik.gong@unist.ac.kr), TA: Ahin Lee (ahin@unist.ac.kr)

==========

Submission Instructions. Compress all the relevant files into a single zip file named STUDENT_ID_HW2.zip and submit it.
Your zip must include:

- *src/* folder containing all source codes (datastructure.cpp, baseline_*.cpp, headers, etc.)
- *Makefile* that correctly compiles all files using make
- *README.txt* explaining compilation, execution, and result reproduction
- *report.pdf* containing your write-up (see below)

Build Commands:

- *make* → compiles all executables
- *make run* → runs your main test or benchmark
  Alternatively, you may run your executable directly, e.g., *./main <optional arguments>*

If your program requires multiple runs or arguments to reproduce results, please include a simple script file named run_bench.sh that executes all necessary commands.

Collaboration Policy. You are welcome to discuss concepts with classmates, but all code, figures, and written content must be entirely your own. Copying others' work or sharing your own directly is plagiarism and will result in zero points for all parties.

==========

Introduction

In programming assignments, you mainly practiced implementing existing data structures to understand how they work and how to use them correctly. However, in the real world, you are usually asked to solve new and unfamiliar problems that do not have ready-made solutions. This homework helps you think creatively, design your own data structure, and evaluate it systematically, just as you would in research or practical software development.

In this homework is a **mini project**—you will design, implement, and evaluate your own data structure. Your goal is to demonstrate creativity in design and rigor in evaluation by comparing your data structure with existing ones.

Note that this homework accounts for the **largest portion** among all assignments (homework, programming assignments, quizzes, etc.). Since designing a new data structure and thoroughly evaluating it take a long time, **please start as soon as possible!**

Report Requirements (80%)

Your report.pdf should include the following sections. Each part contributes to the overall grading of your report.

1. Design of Your Data Structure

Clearly describe your data structure, including its purpose, internal representation, and key operations such as insert, delete, and search. Explain what problem it aims to solve or what advantage it provides compared to existing data structures.

Novelty is an important grading criterion. I understand that it can be difficult to come up with something completely new. It is perfectly fine if your design is not entirely original — you can build upon existing ideas to

form a new one (e.g., modify data structures we learned in the class, combining two data structures, etc.). Since there is no single structure that performs best for all cases, thinking about a specific use case or application scenario can make your idea more meaningful and valuable.

2. Related Work and Comparison

Identify existing data structures that are similar to yours. Explain how yours differs in functionality, efficiency, or design goals. A clear understanding of related structures will help position your idea and demonstrate awareness of existing approaches. This part is important to emphasize the novelty of your own data structure. If you overlook important related data structures, points may be deducted.

3. Evaluation and Analysis

Compare the performance of your data structure with existing ones through experiments or analysis. You must implement proper baseline data structures, including those covered in class and those you found during your research, to ensure fair comparisons.

It is acceptable if your structure is slower or requires more memory than existing ones in general. However, try to find and discuss specific cases, scenarios, or applications where your design performs better or more efficiently. Clearly discuss when and why your structure performs well and when it does not.

Design appropriate test cases to evaluate efficiency, scalability, and usability.

4. References

Include **all** references (papers, articles, lecture notes, etc.) for any sources or ideas that inspired your design. For instance, if your design is inspired by DoublyLinkedList we learned in the class, add the lecture note number.

5. Readability and Presentation

Present your report in a clear, organized, and visually understandable way. Use figures to illustrate your data structure and tables or graphs to summarize comparisons or results. The ability to communicate your idea effectively to others is an essential part of this homework.

Code Requirements and Grading (20%)

1. Compilation

Your code must compile and run without errors.

2. Reproducibility

Include test cases so that your results in the report can be reproduced. Make sure your code produces sufficient output to verify your analysis.

3. README File

Include a README file that clearly explains how to compile and run your code, how to reproduce the results presented in your report, and what test cases are provided.

The README should include:

- A short description of your data structure
- Compilation and execution instructions
- Example commands or scripts to reproduce key experiments
- Expected outputs or results for verification

A clear README is an important part of the grading, as it ensures the reproducibility and clarity of your work.

Language and Library Restrictions

- You must use C++ only (minimum C++11, C++17 or higher recommended).
- You may freely use the C++ Standard Library (namespace std), including containers such as std::vector, std::map, std::unordered_map, and std::set, as well as standard algorithms and utilities.
- However, your proposed/novel data structure must be implemented manually. You may build ideas upon existing structures, but you may not simply wrap, alias, or rename an existing std container as your new structure.
- No third-party/external libraries (e.g., Boost, OpenCV, Eigen, TBB, fmt, nlohmann/json) are allowed.
- Using standard containers as baselines for comparison (e.g., comparing against std::vector or std::unordered_map) is allowed, in order to reduce the load of reimplementing existing data structures.

Report Format & Length

- Use the report.docx template, which follows the ACM format
- Write your main report in less than 8 pages, excluding appendix and references.
- Figures, tables, and captions are included in the page count.
- Don't spend time perfecting formatting; what matters is clarity of your motivation, design, and results.

***What is the *ACM Format*?**
The ACM (Association for Computing Machinery) format is the standard writing style used for research papers in computer science. It provides specific rules for how to organize a paper (title, abstract, introduction, related work, results, etc.) and how to cite references. Many top computer science conferences and journals use this format, so learning it helps you prepare professional and well-structured research papers.

Grading & Evaluation Process

- Total: 100 points
- Report (80 pts), Code (20 pts)
- Evaluated by: Both TA and Instructor will jointly assess HW2 using a shared rubric. Final grades reflect combined evaluation.
- Manual and automatic and checks will both be used. If compilation fails, score is zero. If runtime errors occur, partial credit may apply depending on execution progress.