Data Structures

Fall 2022 (M1522.000900 / section 001)

Instructor: *Prof.* Bongki Moon
Contact: bkmoon@snu.ac.kr (✉), 02-880-1842 (☎), 02-886-7589 (FAX), Engr 301-402 (☕)
Office hours: Wed 13:00-14:00, or by appointment.

Teaching Assistants: TBA

Class schedule: Mon/Wed 11:00-12:15, Engr. Bldg. 302, Room 105

Course keywords: Data structure, Algorithm, Hashing, Trees, Graphs, Sort, Search.

Overview and Topics: This is an introductory data structure course at the undergraduate level that covers fundamental algorithms and data structures used in computer programming. Data structures are ways of organizing data within computer's storage so that some desired operations may be performed on the data easily or efficiently. Algorithms are a sequence of operations that usually take some input data and produce some desired output. Together, they form the foundation of computer programming. The topics to be covered include the following.

- Abstract data types, and algorithm analysis techniques.
- Trees: traversal, search tree, Huffman coding, heap, binary search trees, balanced trees, UNION/FIND.
- Hashing: open hashing, closed hashing and probing techniques.
- Graphs: traversals, shortest path algorithms, minimum spanning tree algorithms.
- Sorting: bubblesort, insertion sort, quicksort, mergesort, heapsort, shellsort, radixsort.
- String match: trie and patricia.
- Algorithm design techniques: greedy, divide and conquer, dynamic programming.

See *Topics and Tentative Schedule* for the detailed description and tentative lecture schedule.

Textbook and Lecture slides: The lectures will be based on the following textbook.

- Cliff A. Shaffer, *Data Structures and Algorithm Analysis*, Edition 3.2 (Java version), Mar/28/2013. (This e-book is free for downloading at `http://people.cs.vt.edu/~shaffer/Book/`.)
- All lecture notes will be available to students in advance at eTL.

Grading: This course is offered for three credits. The grade is broken down as follows.

• Attendance	5%	
• Homework assignments	0%	(three assignments planned)
• Programming assignments	35%	(three assignments planned)
• Exams	60%	(three exams planned, 20% each)

The final grades may be curved, but a weighted total of 90% and above is guaranteed to be an A, 80% and above at least a B, 70% and above at least a C, and 60% and above at least a D. No determinations will be made until the final exam is graded.

Prerequisites:

- 4190.102A (Computer Programming) or consent of instructor.
- Knowledge of the Java programming language enough to be able to design, code, and debug programs consisting of hundreds of lines of code.

Web-based Learning Management System:

The SNU eTL system is an open source Learning Management System (LMS). You can access eTL at: `http://etl.snu.ac.kr` and login with your *mySNU* id and password.

The eTL server will have the course syllabus, lecture notes, homework and programming assignments. It will also be used for submission of assignments, and posting grades.

Homework assignments: Homework assignments will not be graded. The purpose of homework assignments is to provide students with additional opportunities to review the materials covered in the class. A homework assignment will be handed out before each exam.

Programming assignments: In writing your program code, *correctness* is the most important attribute, followed by *efficiency*. Code documentation will not be evaluated. It is very important for you to test your program thoroughly before turning it in, as even a small bug arising early in a test can result in the subsequent test phases failing (and hence, a poor grade).

Programming assignments must be submitted electronically at eTL, and *neither paper copies nor email submissions* will be accepted. When you submit your programming assignment, if your solution has more than one source file, submit them as a single .zip, .tar.gz, or .tar.bz2 archive, so that TAs can download it quickly.

Your program will be tested and graded only on a **Linux** platform to verify/evaluate its completeness, correctness and efficiency. Be certain that your program runs correctly on a Linux platform before you submit it.

You can request a re-grade on a program. Points lost due to failing test cases can be partially regained, depending on the amount of change in your code from the original submission. The formula for this is:

90% of the full credit for less than 2% changed,
70% of the full credit for less than 10% changed,
50% of the full credit for less than 25% changed,
0% of the full credit for 25% or more changed.

The amount of change will be measured based on the lines of code; comments and blank lines will not be counted. We will count Java statements that are added to, removed or changed from the original submission.

To request a re-grade on a program, you will first need to fix your code. Next, upload your revised program in the re-grade folder of eTL. Third, send TAs an email message requesting the re-grade. All three steps will need to be completed within seven calendar days from the return of assignments.

Use of Java language: Java has been taught in the introductory courses offered by the Department; this object-oriented language has many advantages for implementing data structures and algorithms (*e.g.*, encapsulation for abstract data types). So, Java will be the language you will use for the programming assignments.

However, that does not necessarily mean that we will be teaching the language during the course. As a matter of fact, this course is meant for teaching data structures, but not for teaching a specific language. You are rather expected to have an adequate level of knowledge about the language, which is actually one of the prerequisites of the course. You can expect the instructor and TAs to help you understand data structures, but you should understand we will not help you much with your Java programs, not to mention debugging them.

Note that, for the lectures, the traditional pseudo-code will be used to describe data structures and algorithms, because it is often cumbersome and confusing to use Java to define data structures and describe algorithms in lectures.

Late assignment policy: A late assignment may be turned in within 24 hours after the deadline for a 10% penalty (*i.e.*, 10% deduction of your credits). *No assignment late for more than 24 hours* will be accepted unless a valid excuse (*e.g.*, documented illness or family emergency) is given to the instructor by the day prior to the due date. For fairness, this policy will be strictly enforced.

Exam Policy: Exams will consist primarily of short-answer and problem-solving questions, but code-writing questions are also possible.

All students are expected to take the exams at the announced exam times. Make-up exams will be given only in *extreme* circumstances. The instructor will decide whether a circumstance is extreme or not. For example, being in a documented car accident on the way to the exam is likely to count as an extreme circumstance, but having a cold or flu is not. A zero will be given for an exam missed under a sub-extreme circumstance.

Students are required to present identification upon request. Failure to do so will forfeit your right to take an exam. A student ID or a government issued ID bearing a recent recognizable photograph will be accepted as valid identification. Additional ground rules for taking an exam include: (1) Step out of the room and

no return is allowed. (2) No electronic devices with communication capability (*e.g.*, mobile phones) are allowed. (3) Both the exam sheet and bluebook must bear your name when they are turned in. (4) No one is allowed to leave during the first 30 minute period or to enter after that period.

Attendance policy: The University requires attendance to be recorded. If a student is absent from more than one third of the class sessions, then, by the University regulations, a failing grade shall be given. However, we realize that absence from class is sometimes unavoidable due to medical or personal reasons. It is the responsibility of the student to report the reason for his/her absence to the instructor and provide documentation if requested. The instructor will decide whether the student's absence should be excused. Each student is accountable for all work missed because of absence.

Language policy: This course may be taught in English. In case it is, all lectures as well as exams and assignments will be given in English. Students will be allowed to use Korean, English or both for answering exam questions and doing assignments.

Withdrawal policy: Students who wish to withdraw from the course should do it *before* the week of the first mid-term exam. No withdrawal request submitted during or after the week of the first mid-term exam will be approved by the instructor unless it is inevitable due to an extreme circumstance.

Academic integrity and conduct: Academic integrity is a commitment to five fundamental values: honesty, trust, fairness, respect, and responsibility. Students are encouraged to discuss the course work with classmates; both giving and receiving advice will help you learn. However, students need understand that academic integrity must not be violated under any circumstance.

Briefly, you will not accept solutions from others, you will not give solutions to others, and you will not tamper with graded papers, code, or exams. You must develop and write your own code and homework solutions. You must turn in your own work. The one exception is test cases for programming assignments. You are allowed to share test cases and their expected results. It is OK to talk with other students about **what** the problems are but it is not OK to talk about **how** the problems are to be solved. Excessive collaboration is considered a violation.

In a typical programming assignment, the problems could be about writing code, designing algorithms, or a combination of both. For the part of writing code, it is generally assumed that the algorithms are already given or presented in the lecture. In such a case, discussing the algorithms is fine, but discussing implementation details is forbidden. For the part of designing algorithms, sharing detailed algorithm logics is considered a violation because that is essentially equivalent to sharing solutions. (For example, if you tell another student how exactly Knuth's monotonicity should be applied to the OBST algorithm for $\mathcal{O}(n^2)$ running time, that is a violation.)

Students who violate academic integrity will be subject to penalties. There is no violation that is not serious. Every single violation will result in *failing grade in course* and will be reported to the Department.

Disruptive behavior: The instructor seeks to promote a teaching and learning environment free from classroom disruptions, and has the authority and responsibility to effectively manage the classroom environment. Disruptive student behavior is conduct that substantially interferes with or obstructs teaching or learning process in the classroom or any other educational setting. Examples of disruptive behavior are: a student who is persistently tardy or leaves early, a student who stands or walks around in the classroom, a student who talks incessantly while a lecture is being delivered, a student who makes persistent and unreasonable demands for time and attention, a student who reads or sends a text message or chats online, a student with a cell phone ringing in the classroom, a student who becomes belligerent when his or her inappropriate behavior is confronted.

Students who engage in disruptive behavior may be directed by the instructor to leave the classroom for the class period. Students who are involved in involuntary removal for more than one class period will be reported to the Student Discipline Committee for more serious sanctions.

Topics and Tentative Schedule

Week[1]	Topic Description	Book Chapters[2]	Milestones[3][4]
Sep/05	Course introduction, Basic concepts, Abstract data types, Recursion, Mathematical induction	1.1, 1.2, 1.4	Program 1 out
Sep/12	More analysis techniques: summations and recurrence relations	2.4-2.6	
Sep/19	Algorithm analysis: counting, upper and lower bounds, O, Ω, Θ notations; Stack and Queue: list and array implementations, Postfix	3.1-3.8, 4.1-4.3	Program 1 due
Sep/26	Binary Trees: Full binary tree theorem, tree traversal, implementation; Threaded tree; Huffman coding	5.1-5.3, 5.6	Program 2 out
Oct/03	Binary search trees (BST); Balanced search trees: AVL tree, Splay tree	5.4, 13.2	Exam 1 *(Oct/05)*
Oct/10	Hashing: open hashing, closed hashing, probing techniques, linear hashing	9.4	
Oct/17	Heap and priority queues	5.5, 6.2	
Oct/24	Graphs: definitions and representations, traversal algorithms: BFS and DFS, topological sort	11.1-11.3	Program 2 due
Oct/31	Shortest path algorithms: Dijkstra's single source shortest path algorithm, the correctness of Dijkstra's	11.4	Program 3 out
Nov/07	Shortest path algorithms: negative weights and Bellman-Ford's algorithm, Floyd's all-pair shortest path algorithm	16.1.2	Exam 2 *(Nov/09)*
Nov/14	Minimum spanning tree algorithms: Prim's and Kruskal's; UNION/FIND	11.5	
Nov/21	Sorting: insertion, bubble, selection, quicksort, mergesort, heapsort	7.1-7.2, 7.4-7.6	
Nov/28	Sorting: radixsort, shellsort (Donald and Hibbard); Lower bound, External sort	7.3, 7.7, 7.9, 8.5	Program 3 due
Dec/05	String match: Trie and Patricia, Longest common subsequence (LCS)	13.1	
Dec/12	Algorithm design techniques: greedy, divide and conquer, dynamic programming, randomized algorithms	16.1-16.2	Exam 3 *(Dec/14)*

[1] A week is a Monday-Wednesday sequence. For example, the week of Sep/05 refers to Sep/05 Monday and Sep/07 Wednesday.
[2] Cliff A. Shaffer, A Practical Introduction to Data Structures and Algorithm Analysis, Edition 3.2.
[3] The exact dates are subject to change, but programming assignments are expected to be handed out on Monday and due on Sunday.
[4] Exams will be administered on Monday.