CS 330: Algorithm Analysis

Contact Information

Instructors:	Dmitri Volper
Office Phone Number:	7017
Email address:	dvolper@digipen.edu
Home page (DigiPen):	`faculty.digipen.edu/~dvolper`
Office hours:	T/Th 9-10 am, or by appointment

Course Description This course provides students with an introduction to the analysis of algorithms, specifically proving their correctness and making a statement about their efficiency. Topics for discussion may include loop invariants, strong mathematical induction and recursion, asymptotic notation, recurrence relations, and generating functions. Students will examine examples of algorithm analysis from searching and sorting algorithms. Second part of the course will concentrate on classification of algorithms and building a strong knowledge base of existing algorithms.

Course Objectives This course is designed to promote individual learning and analytical thinking skills. It combines lecture, reading and projects implementation.

Upon successful completion students should

- be able to prove correctness of algorithms
- have deep understanding of differences and advantages of iterative and recursive algorithms
- have a knowledge base of existing algorithms
- understand speed vs. space trade-off, importance of data-structures, and data preprocessing
- be able to design new algorithms using the ideas from class

Course	Day and Time	Room
CS330	M/W 9-11:40	Zoom

Textbooks and References
Required

- *Introduction to the Design & Analysis of Algorithms*, by Anany V. Levitin, ISBN 9780201743951
- `http://jeffe.cs.illinois.edu/teaching/algorithms/` Erickson
- `http://people.cs.vt.edu/shaffer/Book/C++3elatest.pdf` Shaffer
- Slides on class web-site

Grading policy and Grades

Grades will be derived from homework assignments and exams.

Programming assignments (5)	35%	+3 extra projects for graduate students. Submit online, graded offline
Labs (6-7)	10 %	small programming exercises - graded online
Homework (5-6)	5 %	written homework - submit online
Midterm (1)	25%	obvious
Final (1)	25%	obvious. Not-cumulative (covers second half of semester only)

You can see all your current grades through my submission page. Here is a link to some guidelines: `https://faculty.digipen.edu/~dvolper/grade-estimation.html`

Letter grade distribution

x%	Grade
$x \geq 93$	A
$90 \leq x < 93$	A-
$87 \leq x < 90$	B+
$83 \leq x < 87$	B
$80 \leq x < 83$	B-
$77 \leq x < 80$	C+
$73 \leq x < 77$	C
$70 \leq x < 73$	C-
$60 \leq x < 70$	D
$x < 60$	F

Attendance is mandatory. There are no makeup exams or quizzes. Also, for every lecture that is missed, you will lose one point from your final grade (e.g. a 90 becomes an 89). The only exceptions are if you notify me prior to your absence with a valid reason. (Sleeping, studying for another class, working on your game, etc., are not valid reasons for an absence.) Class participation will boost your grade if you are on the border. (e. g. It is possible to get an A- with an overall average of 88.5%)

Classroom policies.

- No food.
- Drinks are allowed, unless prohibited by School policies.
- No loud noises.
- Laptops are allowed if used to display lecture material.
- No strong smells.

Tentative Schedule

Background:

- logarithms, big-O notation
- iterative vs. recursive algorithms
- algorithms for generating permutations, subsets, and combinations

List of problems to be used in class

- sorting a collection
- maximum, minimum elements, other statistics
- searching in a collection
- multiplication of large numbers
- string matching
- closest pair of points in a plain
- convex hull of a set of points
- longest increasing subsequence
- longest common substring of 2 strings (LCS)
- knapsack
- shortest path between 2 vertices of a graph
- all-pairs shortest path

- minimum spanning tree of a graph (MST)
- connected components (bidirectional graph)
- transitive closure (similar to above, but directional graph)
- 3-SAT

Analysis of algorithms:

- proving correctness. Examples: fast exponentiation, remainder.
- time complexity (worst, best, average). Example - fast exponentiation.
- time complexity - recursive algorithms. Solving recurrences.
- case study - Fibonacci numbers.

Brute-force algorithms

- sorting - snail, selection, and bubble sorts
- searching - sequential search
- string matching
- convex hull
- closest-pair
- knapsack

Clever brute-force - backtracking: generating tuples

Divide-and-conquer:

- general structure (pre-processing, recursive call(s), post-processing)
- sorting - insertion sort (divide into n-1 and 1 piles)
- mergesort
- quicksort
- binary search (application to debugging)
- convex hull
- closest-pair
- knapsack - cannot be (reasonably) solved using divide-and-conquer. Discussion.

Dynamic programming:

- introductory example - computing Fibonacci numbers
- LCS
- knapsack
- transitive closure - Warshall's algorithm
- all-pairs - Floyd's algorithms
- comparison with recursive algorithms and memory functions

Greedy algorithms:

- definition and discussion of sub-optimality

- shortest path (Dijkstra)
- MST (Prim, Kruskal)

Midterm

Iterative improvement algorithms:

- definition and discussion of convergence
- solving non-linear equations - Newton's method

Speeding-up algorithms by using special data-structures:

- AVL, 2-3-4, RB trees (search).
- binary heaps (sort)
- binomial heaps - only idea. Connected components problem as example of JOIN operation.
- Fibonacci heaps - only idea. Dijkstra as example.

Speeding-up algorithms by transforming a problem:

- Horner's rule
- FFT - just an idea

Speeding-up algorithms by space-time tradeoff:

- hashes
- Example: counting bits in a binary representation of an integer
- string-matching
  - prebuilding a finite-state machine - Knuth-Morris-Pratt
  - precalculating offsets Horspool and Boyer-Moore

Sweep-style algorithms:

- Graham's scan (convex-hull)
- visibility graph

Theoretical limitations of algorithm power:

- decision trees
- fake-coin problems
- proof of best average $O(N \log N)$ for sorting algorithms.

Computational complexity:

- Turing machine
- P vs. NP

Heuristics:

- using with backtracking

Hybrid algorithms:

- A* - BFS and greedy

Submitting Homework Programming assignments will use C++ language. More specifically, all programs must adhere to Standard C++. Assignments will be graded using GNU's gcc/g++ compiler.

The source files must be submitted electronically through the course submission page - use your DigiPen login and student number to login.

`https://pontus.digipen.edu/cgi-bin/submission.cgi`

Your source code should be archived in zip format.

There will be several small programming assignments during the semester, with the first one being assigned during the second week. You will be given no less than 14 days to complete each assignment. This gives you adequate time to manage your workload. The amount of time actually required to complete an assignment is much less than the time allotted and is generally between 5 and 20 hours. Depending on your grasp of the subject matter during the lectures, some of you will require more or less time to complete the assignments. In any event, you should plan on devoting 5 hours per week to this course (outside of the lectures).

Academic integrity Academic dishonesty, or cheating, occurs when a student represents someone else's work as their own, or assists another student in doing so. This can happen on exams, quizzes, homework, or projects. Academic dishonesty also may occur when a student uses any prohibited reference or equipment in the completion of a task. For example, the use of a calculator, notes, books or the internet when it is prohibited. Plagiarism is a common form of academic dishonesty. This can take the form of copying and pasting excerpts from the web, and representing them as original work. The type and severity of any occurrence, as well as the legitimacy of any claim of academic dishonesty, will be judged by the instructor and the disciplinary committee. All students are asked to help in promoting a culture of academic integrity by discouraging cheating in all forms.

Submissions The following must be submitted electronically:

- source code for programming assignments
- multiple-choice answers to quizzes, midterm(s), and final exam. Note that hard-copies of quizzes, midterm(s), and final exam should be returned in class (use a separate sheet of paper to copy multiple-choice answers and test ID).

Programming Assignment Late Submissions No late submissions are allowed. The only reason I will accept late homework is if you had an emergency. Too much work because of other classes is not an emergency.

Homework Late Submissions Late homework is not accepted.

Lab Late Submissions Late lab submissions are not accepted.

Disability Support Services If students have disabilities and will need formal accommodations in order to fully participate or effectively demonstrate learning in this class, they should contact the Disability Support Services Office at 425-629-5015 or dss@digipen.edu. The DSS Office welcomes the opportunity to meet with students to discuss how the accommodations will be implemented. Also, if you may need assistance in the event of an evacuation, please let the instructor know.

Religious Accommodation DigiPen Institute of Technology provides reasonable accommodations to students who may be absent from activities or incur significant hardship due to religious holidays or observances. These holidays or observances must be part of a religious denomination, church, or religious organization, and the course instructor must be notified in writing during the first two weeks of the course. The institutes policy for grievances is published in the course catalog.