Course Information

Welcome to Design and Analysis of Algorithms, one of the most exciting classes in CS! This class covers fundamental algorithmic ideas that have had (and continue to have) a major influence on the evolution of computer science and several other disciplines, ranging from mathematics to biology to the social sciences. We will introduce students to the mathematical modeling and solution of computational problems, and teach them how to design and analyze efficient algorithms. I hope you will enjoy the class! We will:

1. Cover the usual algorithmic paradigms and explore a toolkit of common algorithms.
2. Analyze the correctness and performance of these methods, and learn some principles of good design
3. Explore different ways of thinking about and representing problems.

Detailed coverage: Techniques for the design and analysis of efficient algorithms, emphasizing methods useful in practice. Topics include divide-and-conquer; dynamic programming; greedy algorithms; branch and bound; backtracking; amortized analysis; graph algorithms, especially flow. Advanced topics may include computational geometry, number-theoretic algorithms, polynomial and matrix calculations, caching, and optionally, distributed and parallel computing.

Pre-requisites: Data Structures, Probability and Statistics.

This handout describes basic course information and policies. Most of the sections will be useful throughout the course. The main items to pay attention to **NOW** are:

◇ Make sure you are registered properly through AMS.
◇ Please note, and carefully adhere to, the collaboration policy for homework.

1 Course Management

Check the Google classroom for updates on office hours, class material, problem sets, and announcements. You should visit this site regularly to be aware of any changes in the course.

2 Staff

My policies regarding problem sets and exams are clearly outlined in this handout. If a situation arises that is not addressed in this handout, please ask!

3 Prerequisites

Algorithms prerequisite: We assume knowledge of basic data structures such as heaps, trees, etc., and simple algorithms, such as sorting, graph search, etc. If you did not get an A or A- grade in Data Structures (which is a required prerequisite), you should plan to put in extra time to get up to speed in this course.

4 Lectures & Recitations

Lectures will be held twice a week. You are responsible for material presented in lectures, including oral comments made by me. I will be posting some lecture notes; while this could be helpful in the event of an unavoidable absence, it is unlikely to be as valuable as being present in lecture.

5 Problem Sets

I hope to assign 7-10 problem sets during the semester. The actual due date will always be written on the problem set itself. Homework must be turned in by 4 P.M. on the due date.

- **Grace Days:** You are expected to submit your homework on time. Nonetheless, because unexpected situations might occur (like travel, overload, conflicts, illnesses, and family emergencies), you have a budget of **10 late days** throughout the semester. The budget is spent in increments of 1 day (24 hours), and you may not use more

than 2 grace days per problem set. You do not need to inform us about your use of your budget. The course staff will keep track of the days you have spent. If you submit your problem set later than two days after the deadline, your submission will not count towards your grade. The same will hold true if you are late and have no budget left. Beyond the use of grace days, late homework will not be accepted without a medical note, and even then, only in truly extreme circumstances.

- **Discount Policy:** Your two lowest homework scores will each be counted with half-weight compared to each of the other eight.
- **Office Hours:** There will be no office hours scheduled on the day a problem set is due.
- **Submission Format:** Solutions to all parts of the problem set should be submitted online to gradescope in a single document file. **Your file must be in PDF format prepared in LaTeX using the template provided.** If the PDF file does not clearly indicate which parts the solutions refer to, or has parts missing, it is assumed that the student did not attempt that part of the problem. Therefore, before submitting, make sure all of your work is included in the PDF file.

  Start each question on a new page and mark the top of the page with the following: (1) your name, (2) the question number, and (3) the names of any people you worked with on the problem (see Section 9), or "Collaborators: none" if you solved the problem entirely by yourself.

  The problem sets may include exercises that should be solved but not handed in. These questions will be clearly marked and are intended to help you master the course material. Material covered in exercises will be tested on exams.
- **Regrade Requests:** Any student who feels that a problem set was not graded properly may submit a regrade request through Gradescope within one week of the graded assignment being returned to the student. Please note the following before submitting a regrade request:

  1. You should carefully read the posted solutions for the problem in question.
  2. Indicate which rubric items you deserve (if applicable), where in your solution write-up you address them, and explain why you deserve extra points. Any regrades without justification will not be processed.

6 On the Importance of Clarity

You should be as clear and precise as possible in your write-up of solutions. Understandability of your answer is as desirable as correctness, because communication of technical material is an important skill.

A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error, and because it is easier to read and understand. Sloppy answers will receive fewer points, even if they are correct, so make sure that your solutions are concise and well thought-out.

You will often be called upon to "give an algorithm" to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of your essay should provide the following:

1. A description of the algorithm in English and, if helpful, pseudocode.
2. At least one worked example or diagram to show more precisely how your algorithm works.
3. A proof (or indication) of the correctness of the algorithm.
4. An analysis of the asymptotic running time behavior of the algorithm.

Remember, your goal is to communicate. I will take off points for convoluted and obtuse descriptions.

7 Quizzes & Oral Exam

This course will have two midterm quizzes, and, for some students, a final oral exam (tentative timings):

Quiz 1:	March
Quiz 2:	late-April or early-May
Oral Exam (for some students):	Last week of classes

The oral exam is only for some students at the discretion of the instructor.

There will be no lecture on the two quiz days.

Each of the quizzes and the oral exam will be closed book. Instructions regarding bringing additional materials to quizzes will be explicitly provided, if required.

Attendance at the quizzes is mandatory. Legitimate conflicts can be discussed with the teaching staff but must be due to extenuating circumstances and discussed in advance.

Regrade requests. Any student who feels that something was not graded properly may submit a regrade request. The request must be made online by the announced deadline. The request should include a detailed explanation of why she or he believes that a regrade is warranted.

8 Grading Policy

The final grade will be based on ten problem sets (with the lowest two given half-weight), two quizzes, and a final oral exam.

The grading breakdown is as follows:

Problem sets	40%
Quiz 1	30%
Quiz 2	30%
Oral exam	NA

9 Collaboration Policy

I encourage you to collaborate with your peers to deepen your understanding of the course material. However, you should approach collaboration *on problem sets only* with care, and follow the guidelines below. **Copying from online resources, books, or notes from previous versions of this or other classes is strictly forbidden — copying will be considered a serious offense and dealt with accordingly.**

1. **You should spend at least 30–45 minutes trying to solve each problem entirely by yourself.** If you find yourself unable to solve the problem, you can seek help, either by approaching me or by collaborating with your peers.

2. **Do not be a Spoiler.** If you already solved the problem, do not give away the answer to your friend. The best way you can help your friend is to give hints and allow her or him the pleasure of coming up with the answer her/himself. Our past experience has overwhelmingly shown that students who do not attempt the problem sets on their own generally perform poorly in the exams, and thus in the class overall.

3. **You must write up each problem solution entirely by yourself without assistance,** even if you collaborate with others to solve the problem. Doing otherwise will be considered plagiarism, an academic offence with serious repercussions. You are asked on problem sets to identify your collaborators. If you did not work with anyone, you should write "Collaborators: none."

   It is a serious violation of this policy to submit a problem solution that you cannot orally explain to a member of the course staff. Plagiarism and other dishonest behavior cannot be tolerated in any academic environment that prides itself on individual accomplishment.

If you have any questions about the collaboration policy, or if you feel that you may have violated the policy, please talk to one of the course staff. Although the course staff is obligated to deal with cheating appropriately, we are far more understanding and lenient if we find out from the transgressor himself or herself rather than from a third party or on our own.

Needless to say, **no collaboration whatsoever is permitted on quizzes or exams.**

10 Textbook

We will not be using a textbook as such, but if you really want one, you can use the third edition of the textbook *Introduction to Algorithms* by Cormen, Leiserson, Rivest, and Stein (MIT Press). This is a great textbook and every serious algorithmist should have a copy. Come talk to me before you purchase anything!

In addition, as and when necessary, I will post additional reference material and links on the course homepage.

11 Syllabus

The following topics should ideally be covered as part of the course. Please note that some of the topics/subtopics may change as the course progresses, and the final syllabus will always be restated before an exam.

1. Graph Algorithms. Graph search (DFS, BFS), Bellman-Ford algorithm, connected components algorithm, minimum spanning tree algorithms (Prim's and Kruskal's algorithm) **3-4 hours**

2. Greedy Algorithms. Problem in focus: shortest path; recall Dijkstra, introduce A-star. Discuss TSP. **1.5 hours**

3. Interval Scheduling. Why greedy doesn't always work; hybrid argument. Min and max of an array. **1.5 hours**

4. Median Finding (randomized, median of medians), Integer Multiplication, Fast Fourier Transform (complex, ring, finite), Polynomial Multiplication. (Divide and Conquer.) **3 hours**

5. Amortized Analysis **1 hour**

6. Network Flows: Augmenting Paths, Max-Flow Min-Cut, Ford-Fulkerson Algorithm, Maximum Bottleneck Path Algorithm, Bipartite Matchings, Blocking flows, strongly poly algorithms, min cost flows. **3 hours**
7. Linear Programming and Duality: Simplex. Mention ellipsoid and interior point methods **1.5 hours**
8. Randomized Algorithms: QuickSelect and QuickSort, Matrix Product Verification (also Adversaries, Fiat's Marking Algorithm, Potential Functions, Yao's Minimax Principle) **3 hours**
9. Refresh: Splay trees, buckets, van Emde Boas trees. **0.5 hours**
10. Streaming Algorithms: Reservoir Sampling, Distinct Elements Problem, load balancing, paging, randomization, k-server problem. **3 hours**
11. Dynamic Programming: Alternating Coins Game, Optimal BST, Edit Distance, Knapsack, Longest Common Subsequence, Pseudopolynomiality. **3 hours**
12. Minimax **1.5 hours**
13. N-queens problem (backtracking; mention of alpha-beta pruning) **1.5 hours**
14. MAX-SAT, set cover, set inversion (branch and bound) **3 hours**
15. Random Walks and Markov Chain Monte Carlo (MCMC) Methods **3 hours**
16. (optional) Distributed Algorithms: Leader Election, Maximal Independent Set **1.5 hours**
17. Intractability: P, NP, NP-completeness, Reductions **1.5-3 hours**

These topics will only be covered if time permits:

1. Geometry: recall Range trees and k-server; sweep algorithms, voronoi diagrams; double-coverage
2. Approximation Algorithms: Vertex Cover, Partition, LP Rounding, scheduling relaxation, facility location, randomized rounding
3. Parallel Algorithms: Parallel Integer Multiplication, Maximal Matchings
4. Distributed Algorithms: Leader Election, Maximal Independent Set
5. External memory algorithms

12 Advice and resources for effective learning

Because of the conceptual nature of the material, just attending lectures and doing the homework are unlikely to be sufficient for learning all the concepts. **Setting aside time to do the reading and to study your notes from lecture and recitation is generally necessary to truly learn and internalize the material**, and to be able to apply it in new ways later in the course as well as for the rest of your life.

Homework is essential for learning the material. Rather than thinking of problem sets as just a requirement, recognize them as an excellent means for learning the material, and for building upon it. Spread out the time you have to work the problems. Many people learn best by reading the problems long before they are due, and working on them over the course of a whole week; they find that their minds make progress working the problems in the background or during downtime throughout the day. Few people do their best learning the night before an assignment is due. Work with others if that is helpful, but with the goal of learning first and solving the problems second. **It is worth reading the posted homework solutions, even if you received full credit**. Often the clarity of explanation or details of implementation are different from the way you were thinking about things in ways that can improve your learning.

Don't hesitate to ask for help. This class is largely conceptual, and the concepts tend to build on one another. **If you are having trouble understanding the material, it is important to catch up rather than risk falling further behind**. Office hours are a particularly useful mechanism for learning material and working through difficulties on problem set assignments.

This class has great material, so HAVE FUN!