CS 7200 Algorithm Design and Analysis

- **Instructor:** T. K. Prasad
- **Phone No.:** (937)-775-5109
- **Email:** t.k.prasad@wright.edu
- **Home Page**:https://cecs.wright.edu/people/faculty/tkprasad/
- **Semester:** Spring, 2025
- **Class Hrs / Location:** 2:30pm-3:25pm MWF / 146 RC (Synchronous as well as recorded Asynchronous Remote Teaching)

- (WebEx recorded lectures will be available via CS7200 course page on Pilot.
- The two tests will be conducted in-person for uniformity and fairness.)

- **Office Hrs / Location:** 12:30pm - 1:50pm MWF / 395 Joshi Res.Ctr. (and if necessary by appointment)
- **Instructor/Class Communication:** By Email (for personal matters)/Discord class account (for course related matters)
- **GTAs** : TBA

Course Objectives

To provide a solid foundation in algorithm design and analysis. Specifically, the student learning outcomes include:

- Able to apply graphs and matching algorithms.
- Ability to design algorithms using greedy strategy, divide and conquer approach, dynamic programming, and max flow - min cut theory.
- Ability to analyze asymptotic runtime complexity of algorithms including formulating recurrence relations.
- Defining and applying basic concepts of computational complexity.
- Preliminary knowledge of approximation and randomized algorithms.

Prerequisites **CS 3100/5100 Data Structures and Algorithms**

Course Description

This course introduces concepts related to the design and analysis of algorithms. Specifically, it discusses recurrence relations, and illustrates their role in asymptotic and probabilistic analysis of algorithms. It covers in detail greedy strategies, divide and conquer techniques, dynamic programming and max flow - min cut theory for designing algorithms, and illustrates them using a number of well-known problems and applications. It also covers popular graph and matching algorithms, and basics of randomized algorithms and computational complexity. However, the depth of coverage of complexity classes and intractability, approximation algorithms, and randomized algorithms, will depend on the available time. The programming assignments can be coded in Python.

Course Load

The course load includes homeworks and programming assignments, a midterm exam, and a final exam.

(1) Assignments will be graded. You may work in teams of at most three students (four members must split into two teams of two each) and share your work with each other to improve your understanding of the material. Avoid *plagiarizing code* from other teams or online sources.
(2) The exams will be conducted in-person in the class. The exam will provide helpful information and code snippets from the text book or course notes that are deemed necessary for answering questions.
(3) The entire class is expected to take the exam simulaneously so that everyone can be tested on the same exam or variants of the same exam. (Exceptions may be made only under extraordinary circumstances (such as documented medical emergency) and a different exam may be given.)
(4) Any academic impropriety during an exam (which includes copying from other students, or accessing online resources that are prohibited) will have a minimum penalty of an 'F' grade, plus additional disciplinary action for unethical behavior. See http://www.wright.edu/students/judicial/integrity.html for details.

Tentatively, the weightage for the assignment is 20%, the (earlier and shorter) midterm exam will be 30% and the final exam will be 50% for determining the final grade. If this policy is modified for any reason, you will be informed about it in class.

Required Text

1. J. Klienberg and Eva Tardos: Algorithm Design. 1st Edition. Addison Wesley, 2005. ISBN 0-321-29535-8 (Lecture Slides)

Reference Text

1. T. H. Corman, C. E. Leiserson, R. L. Rivest, and C. Stein: Introduction to Algorithms. 3rd Edition. MIT Press, 2009. ISBN 978-0-262-53305-8

Grading

The letter grades will be assigned using the following scale: A[90-100], B[80-90), C[70-80), D[60-70), and F[0-60). However, I reserve the right to adjust the scale somewhat to utilize the gaps in the distribution.

Applicable University Policy on Generative AI

You are not permitted to use generative AI tools for any work for this course. This includes the use of popular tools like ChatGPT, Midjourney, GitHub Co-Pilot, as well as all other tools built on generative AI technologies. Due to the nature of this course, the professor can only fairly and accurately evaluate work that is not assisted by generative AI. Use of generative AI for assigned work in this course will be considered a violation of the university's academic integrity policies. In general, if you have any questions about whether or not use of a particular tool or technology is allowed, check with your instructor first. (Also note that the output generated by these tools can be invalid (due to hallucination) and you will not be able to detect it without first knowing what the "correct" answer is, which is what this course is trying to provide.)

Tentative Class Schedule and Syllabus

	Topics
Class 1	Introduction; Stable Matching Problem

Class 2	Gale Shapley Algorithm
Class 3	Algorithm Analysis; Graph Search; DemoTO;
Class 4	Greedy Algorithms : Interval Scheduling/Partitioning
Class 5	Shortest Paths Algorithm; MST
Class 6	Minimum Spanning Trees; Huffman Code
Class 7	Divide and Conquer : Inversions, Closest Pair of Points
Class 8	Recurrence Relations and Master Theorem
Class 9	Divide and Conquer : Multiplication
Class 10	Fast Fourier Transform
Class 11	Quicksort : Average Case Analysis; Randomization
Class 12	Select / Median Algorithm : Probabilisitic Analysis
	Test 1: Feb 17, Monday, 2:30pm-3:25pm, Loc: TBA
Class 13	Dynamic Programming (incl. *Iteration vs Recursion*) : Weighted Intervals
Class 14	Segmented Least Squares; Knapsack
Class 15	RNA Secordary Structure; Sequence Alignment / *String Matching*
Class 16	Sequence Alignment in Linear-space
Class 17	Bellman-Ford Algorithm
Class 18	(continue)
Class 19	Maximum Flow Theory: Ford-Fulkerson Algorithm (Demo1) (Demo2)
Class 20	Maximum Flow Applications
Class 21	Intractability: Polynomial-Time Reduction
Class22	NP-Completeness: Definitions and Proofs; More reductions
Class 23	Complexity classes
Class 24	PSPACE
Class 25	Extending Limits of Tractability
Class 26	Approximation Algorithms
Class 27	Local Search (*Reading*: *Hill Climbing*)
Class 28	Randomized Algorithms
	Test 2/Meeting: May 2, Friday, 2:45pm-4:45pm, Loc: TBA

Old Exams (Fall 2013, Spring 2020)

- Midterm (pdf)(pdf).
- Final (pdf)(pdf).

The exact nature of the exam for this semester, based on what has been covered, will be shared before the exam..

Assignments (Fall 2021)

Assignments will be emailed.

Lecture slide decks have been linked to this page. Lecture videos recorded in WebEx will be posted on Pilot. Furthermore, old recordings in Panopto and/or WebEx will also be made available as a fall back in case there is some recording snafu.