Algorithms
Computer Science 321

Instructor:	Erik Krohn
E-mail:	krohne@uwosh.edu
Text Message:	920-644-3745
Class Time:	Monday, Wednesday & Friday: 12:40pm - 1:40pm or 1:50pm - 2:50pm
Classroom:	Halsey 309
Office Location:	Halsey 216
Office Hours:	Monday: 2:50pm - 3:50pm Thursday: 10:15am - 1:15pm Friday: 9:20am - 10:20am
Prerequisites:	CS271 and Math212 with a grade of C or better
Course Website:	`http://www.uwosh.edu/d2l`
Textbooks:	*Algorithm Design* by Kleinburg and Tardos. ISBN 9780321295354 *Introduction to Algorithms* by Cormen, Leiserson, Rivest and Stein. ISBN 9780262033848

Course Information
Algorithm design techniques including brute-force, backtracking, divide-and-conquer, dynamic programming and greedy algorithms. Other topics include big-O and amortized analysis, recurrence relations in the analysis of recursive algorithms, numerical algorithms, pattern matching, data integrity, authentication, and encryption.

Course Website
You should check d2l on a regular basis - it will contain lecture notes, handouts, assignments, announcements, and grades. Ill do my best to let you know when something new and important comes up, but it is your responsibility to check the web site frequently for information that you might not get otherwise.

Mini Assignments
You will have daily mini assignments. Mini assignments are generally short and should take less than an hour to complete. You will be assigned a mini assignment every lecture to ensure you are staying current with the material. Mini assignments must be completed in LaTeX and the resulting pdf uploaded to the dropbox before the due date. I will drop your 2 lowest mini assignments. Not all mini assignments will be graded. **No late mini assignments will be accepted.**

Assignments
All assignments requiring "written" work must be written in LaTeX and the resulting pdf submitted electronically via d2l. It is your responsibility to ensure your submission was submitted correctly. **There are no late submissions**.

Exams

Exam material will come from the lecture notes, mini assignments, book and assignments. There will be more information about each exam as it approaches. The *tentative* exam dates are listed below. All exams will be taken during the regular class period. These may change, so as the date approaches make sure you've got the most recent information.

- **Exam One** - Wednesday, October 8^{th}, 2015
- **Exam Two** - Wednesday, November 11^{th}, 2015
- **Exam Three** - Wednesday, December 16^{th}, 2015

If you are unable to take a scheduled exam, it may be possible to take a make-up exam provided that you do both of the following, which are then subject to my approval:

1. Make arrangements prior to the scheduled exam. For last minute emergencies, telephone me at 424-7080 or leave a message at the computer science office, 424-2068 or send me a text message. No after-the-fact notifications will be accepted.

2. Have a written medical excuse signed by the attending physician OR have a note of justification from the Dean of Students Office.

If allowed, only one make-up exam will be given. It will be a comprehensive exam given at an arranged time during the last week of the semester.

Grading

Course grades will be based on assignments, mini assignments and exams. Your final grade will be computed with the following percentages:

- 45% - assignments
- 15% - mini assignments
- 40% - exams

If you believe anything was graded incorrectly or unfairly and would like to have it regraded, you must let me know about it within *one week* of having the item graded. I will regrade the entire assignment or exam and you may gain or lose points.
Grading will be on a plus/minus system. Grading may be done on a curve depending on the overall performance of the class. If no curve is used, your grade will be computed based on the following:

Percentage	Grade	Percentage	Grade
≥ 92	A	72 - 78	C
90 - 92	A-	70 - 72	C-
88 - 90	B+	68 - 70	D+
82 - 88	B	62 - 68	D
80 - 82	B-	60 - 62	D-
78 - 80	C+	< 60	F

Topic Coverage

- Analysis of algorithms
  - Efficiency
  - Correctness
- Algorithm design strategies
  - Divide-and-conquer
  - Dynamic programming
  - Greedy algorithms
  - Brute-force
  - Backtracking
  - Randomization
  - Approximation
- Algorithms from specific problem domains including:
  - Graph algorithms
  - Numerical algorithms
  - Number theory
  - Geometry
- Data integrity, authentication, and encryption
- Pattern matching
- Parallel algorithms

Learning Outcomes

- The student will be able to prove the correctness of an algorithm using loop invariants and mathematical induction.
- Given a recursive algorithm, the student will be able to examine its recursive structure, determine and mathematically solve the corresponding recurrence relation, and infer the asymptotic runtime of the algorithm using big-O notation.
- The student will be able to use appropriate asymptotic notations for bounding algorithm running times from above and below.
- Given a data structure with an occasional high-cost operation, the student will be able to select an appropriate potential function and use amortized analysis techniques to determine the amortized cost of the operations on the data structure.
- The student will be able to identify problems amenable to divide-and-conquer solutions, derive the details of a solution to such a problem, and analyze the run-time behavior of the corresponding solution.
- Given a problem amenable to a dynamic programming solution, the student will be able to determine the underlying recursion that solves the problem, determine why this recursion is suited to dynamic programming, implement the algorithm using dynamic programming to cache subproblem solutions, and determine the efficiency of the resulting algorithm.
- The student will be able to identify problems amenable to greedy solutions, derive the details of a solution to such a problem, and analyze the run-time behavior of the corresponding solution.
- Given a string, the student will be able to trace the execution of an advanced string search algorithm (such as Knuth-Morris-Pratt, Boyer-Moore, or Rabin-Karp) on this data and compare the efficiency of this algorithm to a more straight-forward version of a string search algorithm.
- The student will be able to simulate the extended Euclidean algorithm on paper analyze its computational costs.
- The student will be able to simulate the RSA encryption/decryption algorithm on paper (for data that is reasonable to hand-calculate) and analyze the computational costs of the algorithm when the data scales to what is used in actual practice.

Academic Dishonesty

Academic dishonesty of any kind will not be tolerated. All assignments, mini-assignments and exams are to be completed individually unless otherwise specified. While high-level discussion of ideas and problems with fellow students is encouraged, all work must be done individually. In certain circumstances, code fragments from the instructor may be provided to eliminate tedious coding or to provide a common framework for all students. **All other code must be original**. Online resources may be used to help you understand the material, but you may not copy online code nor can you "borrow" code from other students, past or present. If you use a book, website or any reference to help you solve a problem, you must cite the reference in your assignment.

Any suspected academic dishonesty will be dealt with on a case-by-case basis. Any clarification of what does or does not constitute academic dishonesty must take place **before** you turn in questionable work. For clarification on what constitutes academic dishonesty, contact me or consult the printed policy in the UWO Student Discipline Code, Chapter UWS 14.