

## COURSE DESCRIPTION

**Department and Course Number** CMPS 261      **Course Coordinator** Dr. Mark Radle

<b>B. Course Title</b>	Advanced Data Structures and Software Engineering	<b>Total Credits</b>	3
<b>C. URL</b>	<a href="http://web.cacs.louisiana.edu/~mgr/261">http://web.cacs.louisiana.edu/~mgr/261</a>	<b>Semester hours</b>	

### Current Bulletin Description

Programming methodology, software testing techniques, and algorithm analysis. Data structures topics include construction, traversal, and modification of trees, heaps, and hash tables. Sorting and searching techniques on linear structures including arrays and sequential files. Fa, Sp. Prereq: CMPS 260 and MATH 270 with a grade of C.

### Textbook

Weiss, M., "Data Structures and Algorithm Analysis in C++", Third Edition, Addison-Wesley, 2006.

### D. References

"Programming Documentation Standard", Object-Oriented Version, available from <http://web.cacs.louisiana.edu/~mgr/261>

### Course Goals

- A. To provide a framework of software engineering models and issues concerned with the development of high quality software.
- B. To gain experience in working with large software systems, including the specification, development, and documentation of such systems, the use (or reuse) of existing modules in the development of new systems, and the employment of a variety of software testing techniques for verifying systems.
- C. To provide students with a working knowledge of a variety of search and information retrieval structures, including trees, heaps, and hash tables.
- D. To study a variety of internal sorting techniques.
- E. To learn to choose the most appropriate data structures and algorithms for solving particular problems, taking into consideration the run-time complexity, space complexity, and conceptual complexity of each choice.

### Course Outcomes

1. Students will have experience implementing and reusing complex data structures.
2. Students will be competent in documenting the specification, design, and verification of software systems.
3. Students will be minimally competent in analysis of algorithms.
4. Students will be experienced in implementing and testing large software systems.

### Prerequisites by Topic

Fluency in an object-oriented language.  
Knowledge and experience with list, stack, and queue data structures.  
Exposure to introductory concepts related to the analysis of algorithms.

### Major Topics Covered in the Course

1. Preliminaries (4 hours)
  - Abstract Data Types
  - Mathematical background
  - Recursion
  - Proof Techniques

2. Algorithms(4 hours)
  1. Case analysis
  2. Asymptotic analysis
  3. Space bounds
3. Trees(12 hours)
  - B. Binary Trees
  - C. Binary Search Trees
  - D. Heaps and Priority Queues
  - E. General Trees
4. Sorting(10 hours)
  - Insertion sort
  - Divide and Conquer – Quicksort
  - Merging sorted sequences – Mergesort
  - Lower bound for sorting by comparison of keys
  - Heapsort
5. Searching(5 hours)
  1. Searching Arrays and Self-Organizing Lists
  2. Dynamic Sets and Searching
  3. Hashing
  4. Union-find
- A. Graph Algorithms (3 hours)
- B. Tests(2 hours)
- C. Final Exam(2 hours)

**Laboratory projects (specify number of weeks on each)**

- B. Convert a given postfix expression to the equivalent infix expression inserting parenthesis only as needed. The solution must utilize a stack of trees. A complete documentation package is required as part of the assignment. (2.5 weeks)
- A. Conduct a comparative study of sorting techniques (heap sort, quicksort, and mergesort) on ordered, reverse ordered, nearly ordered, and randomly ordered lists of data. Students will be given a driver to generate the data and must develop all other components for the sorting routines. A short technical paper that summarized the results of the experiment and includes tables illustrating the results must be submitted along with the software system. (4 weeks)
- B. Conduct a comparative study of collision resolution techniques (linear addressing, quadratic probing, and chaining) on hash tables with different loading factors. Students must develop this entire system from scratch. A complete documentation package is required for this system. (4 weeks)
- C. Implementation of a graph algorithm. Either Dykstra's, Prim's or the Biconnectivity algorithm is implementation. (2 weeks)

**E. Oral and Written Communications**

Every student is required to submit at least \_\_\_\_\_ written reports (not including exams, tests, quizzes, or commented programs) of typically \_\_\_\_\_ pages and to make \_\_\_\_\_ oral presentations of typically \_\_\_\_\_ minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

**F. Social and Ethical Issues**

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

**Theoretical Content**

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Analysis of Algorithms (5 class periods)  
Graph Theory (3 class periods)

**Problem Analysis**

Please describe the analysis experiences common to all course sections.

**Solution Design**

Please describe the design experiences common to all course sections.

Students are required to design and implement solutions to the Laboratory Projects listed above. Students design C++ classes to implement the data structures covered in the material and reuse these implementations throughout the semester.