Catalog Description

An introduction to the formal techniques that support the design and analysis of algorithms, focusing on both the underlying mathematical theory and the practical considerations of efficiency. Topics include asymptotic complexity bounds, techniques of analysis, algorithmic strategies, advanced data structures, graph theory and other selected topics. Coursework includes object-oriented programming in C++ and covers templates, STL, and exception handling. Prerequisite: CS 300 and MATH 240 each with a minimum grade of C. Offered alternate years. 4 credits.

Student Learning Outcomes:

**AL/Basic Analysis**

1. Explain what is meant by “best”, “expected”, and “worst” case behavior of an algorithm. [Familiarity]
2. In the context of specific algorithms, identify the characteristics of data and/or other conditions or assumptions that lead to different behaviors. [Assessment]
3. Determine informally the time and space complexity of simple algorithms. [Usage]
4. State the formal definition of big O. [Familiarity]
5. List and contrast standard complexity classes. [Familiarity]

Run algorithms on input of various sizes and compare performance. [Assessment]

6. Use big O notation formally to give asymptotic upper bounds on time and space complexity of algorithms. [Usage]
7. Use big O notation formally to give expected case bounds on time complexity of algorithms. [Usage]
8. Explain the use of big omega, big theta, and little o notation to describe the amount of work done by an algorithm. [Familiarity]

9. Use recurrence relations to determine the time complexity of recursively defined algorithms. [Usage]
10. Solve elementary recurrence relations, e.g., using some form of a Master Theorem. [Usage]

**AL/Algorithmic Strategies**

1. For each of the strategies (brute-force, greedy, divide-and-conquer, recursive backtracking, and dynamic programming), identify a practical example to which it would apply. [Familiarity]
2. Use a greedy approach to solve an appropriate problem and determine if the greedy rule chosen leads to an optimal solution. [Assessment]
3. Use a divide-and-conquer algorithm to solve an appropriate problem. [Usage]
4. Use recursive backtracking to solve a problem such as navigating a maze. [Usage]
5. Use dynamic programming to solve an appropriate problem. [Usage]
6. Determine an appropriate algorithmic approach to a problem. [Assessment]
7. Describe various heuristic problem-solving methods. [Familiarity]
8. Use a heuristic approach to solve an appropriate problem. [Usage]
9. Describe the trade-offs between brute force and heuristic strategies. [Assessment]
10. Describe how a branch-and-bound approach may be used to improve the performance of a heuristic

**AL/Fundamental Data Structures and Algorithms**

1. Implement basic numerical algorithms. [Usage]
2. Implement simple search algorithms and explain the differences in their time complexities. [Assessment]
3. Be able to implement common quadratic and O(N log N) sorting algorithms. [Usage]
4. Explain how tree balance affects the efficiency of various binary search tree operations. [Familiarity]
5. Solve problems using fundamental graph algorithms, including depth-first and breadth-first search. [Usage]
6. Demonstrate the ability to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in a particular context. [Assessment]
7. Describe the heap property and the use of heaps as an implementation of priority queues. [Familiarity]
8. Solve problems using graph algorithms, including single-source and all-pairs shortest paths, and at least one minimum spanning tree algorithm. [Usage]
9. Trace and/or implement a string-matching algorithm. [Usage]
Topics:

- Asymptotic analysis of upper and average complexity bounds
- Identifying differences among best, average, and worst case behaviors
- Big O, little o, omega, and theta notation
- Standard complexity classes
- Empirical measurements of performance
- Time and space tradeoffs in algorithms
- Using recurrence relations to analyze recursive algorithms
- Master Method for solving recurrence relations
- Brute-force algorithms
- Greedy algorithms
- Divide-and-conquer
- Backtracking
- Branch-and-bound
- Heuristics
- Pattern matching and string/text algorithms
- Numerical approximation algorithms
- Simple numerical algorithms
- Sequential and binary search algorithms
- Quadratic sorting algorithms (selection, insertion)
- O(N log N) sorting algorithms (Quicksort, heapsort, mergesort)
- Representations of graphs (adjacency list, adjacency matrix)
- Depth- and breadth-first traversals
- Shortest-path algorithms (Dijkstra’s and Floyd’s algorithms)
- Transitive closure (Floyd’s algorithm)
- Minimum spanning tree (Prim’s and Kruskal’s algorithms)

The above topics and student learning outcomes were copied with permission from the recommendations found at:  http://www.acm.org/education/curricula-recommendations (CS2013, CS2008, and CC 2001)

Instructor Details

<table>
<thead>
<tr>
<th>Professor</th>
<th>Chris Lane</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td><a href="mailto:lanecad@pacificu.edu">lanecad@pacificu.edu</a></td>
</tr>
<tr>
<td>Office</td>
<td>Price 211</td>
</tr>
<tr>
<td>Phone</td>
<td>(503) 352-1494</td>
</tr>
</tbody>
</table>
| Office Hours | M:10:30 - 11:30 AM  
              | W: 1:00 -2:00 PM    
              | F: 10:30 – 12:00 PM 
              | or by appointment  |
## Course Details

<table>
<thead>
<tr>
<th>Course Title</th>
<th>CS 380 Algorithm Design and Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisites:</td>
<td>CS 300 Data Structures with a grade of C or better and MATH 240 Discrete Mathematics with a grade of C or better</td>
</tr>
<tr>
<td>Required for:</td>
<td>Computer Science Major</td>
</tr>
<tr>
<td>Meeting Times:</td>
<td>MWF 9:15am – 10:20am</td>
</tr>
<tr>
<td>Location:</td>
<td>SCOT 204</td>
</tr>
<tr>
<td>Textbooks:</td>
<td>Introduction to Algorithms by Cormen, Leiserson, Rivest, and Stein, 3rd Edition</td>
</tr>
<tr>
<td>Software:</td>
<td>Microsoft Visual Studio 2015 Community with Github plugin Dark GDK</td>
</tr>
<tr>
<td>Course Website:</td>
<td><a href="http://zeus.cs.pacificu.edu/lanec/cs380s17">http://zeus.cs.pacificu.edu/lanec/cs380s17</a></td>
</tr>
</tbody>
</table>

## Course Assessment

### Grade Distribution:

<table>
<thead>
<tr>
<th>Assignments/Homework</th>
<th>40%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unscheduled Quizzes (open-note)</td>
<td>5%</td>
</tr>
<tr>
<td>3 Exams</td>
<td>35%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>20%</td>
</tr>
</tbody>
</table>

### Programming Projects Grading:

<table>
<thead>
<tr>
<th>Successful Execution</th>
<th>70%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceptable structure, style, design, documentation, and efficiency. You must follow the C++ Coding Standards, version 4.0</td>
<td>30%</td>
</tr>
</tbody>
</table>

### Percent Breakdown:

<table>
<thead>
<tr>
<th>Grade</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>92-100%</td>
</tr>
<tr>
<td>A-</td>
<td>90-92%</td>
</tr>
<tr>
<td>B+</td>
<td>88-90%</td>
</tr>
<tr>
<td>B</td>
<td>82-88%</td>
</tr>
<tr>
<td>B-</td>
<td>80-82%</td>
</tr>
<tr>
<td>C+</td>
<td>78-80%</td>
</tr>
<tr>
<td>C</td>
<td>72-78%</td>
</tr>
<tr>
<td>C-</td>
<td>70-72%</td>
</tr>
<tr>
<td>D+</td>
<td>68-70%</td>
</tr>
<tr>
<td>D</td>
<td>60-68%</td>
</tr>
<tr>
<td>F</td>
<td>0-60%</td>
</tr>
</tbody>
</table>

## Important Dates

*Tentative dates for Exams:*
- Exam 1: Friday, February 24th, 2017
- Exam 2: Monday, March 20th, 2017
- Exam 3: Friday, April 14th, 2017

*Dates of no class:*
- Spring Break: Saturday, March 25th, 2017 to Sunday, April 2nd, 2017
- Senior Projects Day: April 26th, 2017
Course Policies

1. **Attendance**: Attendance at every class is critical to your success in this course. I expect you to be on time and ready to go at 9:15am and that you stay until the end of class. You will not be allowed into the classroom once I close the door and start teaching. Any missed lecture is your responsibility to make up; just remember that if you fall behind, it will be very difficult for you to catch up.

   - I reserve the right to raise or lower your grade based on class participation and attendance. Specifically, I may lower your grade or may officially withdraw you from the course through the tenth week of the semester for poor attendance or participation. Further, your final grade may be lowered by 1/3 of your final course grade for each day (or portion thereof) of class missed. Please notify me PRIOR to class if you must miss class for any reason.

2. **Programming Assignments**: All assignments are to be programmed in C++ in Visual Studio 2015. Both the electronic copy and hardcopy of your assignments are due at 9:15am on the day that they are due, unless specified otherwise.

   - The hardcopy must be placed under the instructor’s desk by 9:15am on the day following the assignment due date. If the hardcopy uses more than one sheet, then all sheets must be stapled. The code must be printed in color and double sided. Failure to submit a hardcopy of the assignment will result in a loss of 30% of the assignment points.

   - The electronic copy must be placed in the CS 380 Drop Box on Grace by 9:15am OR be available in your GitHub repository on the day the assignment is due. Failure to submit an electronic copy will result in a loss of 70% of the assignment points.

   - A program that does not successfully compile or produces no output loses 70% of the assignment grade.

   - No Late Assignments will be accepted.

   - All code in any form generated from this course becomes the intellectual property of Pacific University. You may not share this code with anyone without obtaining written permission from Pacific University.

3. **Homework Assignments**: All homework must be provided to the instructor by 9:15am on the day it is due. The instructor will specify if the homework is to be turned in with a hard copy or electronic copy or both.

4. No early or late exams/final will be given. No incompletes will be given.

5. **Academic Dishonesty**: Pacific University has no tolerance for academic dishonesty. It is university policy that all acts of academic dishonesty be reported to the Associate Dean. Forms of academic dishonesty include, but are not limited to, plagiarism, fabrication, cheating, tampering with grades, forging signatures, and using electronic information resources in violation of acceptable use policies. Please consult the Academic Conduct Policies in the A&S Catalog for more details.

   - For programming assignments, plagiarism takes the form of, *but is not limited to* copying code from someone else, whether copying files, glancing at someone else’s code, typing from someone else's notes or typing while they dictate. The source can be a classmate, former student, website,
program listing found in the trash, or anything else. Furthermore, plagiarism even on a small part of the program is cheating.

- Note that in this class, you will be translating pseudo code from the book directly into source code. Performing this translation yourself with no outside help does not constitute plagiarism. **However, you must cite, in comments in your source code, from where each function you implement has been translated. Further, you must not be looking at implementations of these algorithms on the Web to use in your translation.**

- You should also note that aiding someone else’s cheating also constitutes cheating. You should never leave your code where someone else could have access to it, such as staying logged onto a machine or placing solutions in the recycling bin where another student may take it.

- Sanctions that may be imposed for academic dishonesty are:
  
  o First offense for cheating: zero on the assignment/exam and 12 percentage points subtracted from your course total.
  
  o Second offense for cheating of any kind: ‘F’ in the course

6. Neither computer failure, software failure, nor lack of computer access are accepted as excuses for late programs; therefore, start work on the programs as soon as they are assigned, and don't put them off until the last minute. Further, corruption of programs due to bad disk media is also not accepted as an excuse for late programs; therefore, always keep a current backup of all programs on a separate disk. Please note that the Computer Science departmental servers are not backed up.

7. You may be asked to leave the class if you are causing a distraction e.g. cell phone ringing, talking, etc.

8. If you have a complaint regarding a grade on an assignment or exam, write a one-paragraph description of why you feel the grade is incorrect and deliver it to the instructor within five working days of when the graded material was returned to you. I will not consider any grade changes later than five working days after the graded material was returned.

9. **Learning Support Services for Students with Disabilities:** If you have documented challenges that will impede your learning in any way, please contact our LSS office in Scott Hall (ext.2107). The Director will meet with students, review the documentation of their disabilities, and discuss the services that Pacific offers and any appropriate ADA accommodations for specific courses.