

ITK 279
Algorithms and Data Structures
Fall, 2008

Instructor: Dr. Mary Elaine Califf

Office: Old Union 106

Office Hours: MWF 11-11:30 and 12:30-1:50, and by appointment

Phone: 438-5203

Email: mecaliff@ilstu.edu (the best way to reach me in most cases)

Catalog Description:

Data structures, algorithms, mathematical foundations of computer science. Topics include lists, trees, graphs, sorting, searching, correctness, computational complexity, parallel algorithms.

Pre-requisites:

A C or better in ITK 179 and concurrent registration or prior completion of either MAT 160 or MAT 260. We will build on your knowledge of programming, object orientation, and elementary data structures from ITK 179. Discrete math covers much of the material of this course from a more mathematical and theoretical point of view, so you will be able to relate the work of this course to what you have done or are doing in your discrete mathematics course.

Extended Description:

The purpose of this course is for you to learn the rest of the basic knowledge you need to be a successful computer programmer in a variety of fields. It will provide you with additional tools in the form of more advanced data structures and algorithms. We'll also talk about how to compare different algorithms and data structures in order to choose the right one for a particular task. Finally, we'll talk about approaches to developing new algorithms for new problems. This is a challenging course for most, but it is also one of the most important courses in the computer science major.

Textbooks:

Weiss, Mark Allen. (2005). *Data Structures and Algorithm Analysis in C++*. 3rd Edition. Addison-Wesley.

Savitch, Walter. (2008). *Absolute C++*. 3rd Edition. Addison-Wesley.

Objectives:

Upon successful completion of ITK 279 you will be able to

1. Determine the computational complexity of an algorithm.
2. Analyze a problem and determine the best combination of data structure and algorithm to solve it.
3. Apply knowledge of various algorithms and data structures by writing programs that implement them.
4. Choose the most efficient algorithm for performing a task like sorting.
5. Explain what an NP-complete problem is.
6. Explain heuristic solutions for some of the common NP-complete problems like the traveling salesman problem.
7. Explain what dynamic programming and greedy algorithms are.
8. Compare and contrast several algorithms which perform the same task.

Language and Platform:

This course is taught in C++ on UNIX. You are strongly encouraged to spend some time trying to become comfortable with UNIX, as it's a very valuable job skill. You are not expected to have more than minimal prior exposure to UNIX, and you should feel free to come for help with the operating system as needed. The majority of you do not yet know C++. Even if you do know the language, we will be

looking at issues of memory management that are often overlooked in early courses. You will be expected to do some of the language learning on your own, but be sure to make use of the resources provided (including both textbooks and teacher).

Course Activities:

Examinations: There will be three examinations, including the final, which will be comprehensive. The second exam and the final will include take-home portions to allow you to apply the critical thinking and algorithm analysis developed in the course rather than just demonstrating that you can memorize the algorithms and data structures.

Papers: Writing is an essential part of this discipline, so you will write two papers in this course. The first will be a report on a timing experiment, allowing you to demonstrate your ability to describe algorithms, describe an experiment and its results, and analyze and explain the results. Because computer science is a discipline in a state of constant flux, you need to develop the ability to learn about new data structures and algorithms using the resources available outside the classroom, and you need the ability to explain an algorithm or data structure to your peers. Therefore, for your second paper, you will research an unfamiliar algorithm or data structure and write a paper describing that algorithm or data structure to your classmates

Programs: The center of this course is programming, so it will also include several programming assignments. The programming assignments will vary but will typically involve application of the data structures and algorithms covered in class to realistic types of problems. Some of them will involve writing multiple programs. None of the assignments are designed to be completed overnight. You will typically have 2-3 weeks to complete an assignment. You should start design work immediately and start programming **at least** a full week before the assignment is due. Remember that both the language and the platform are at least somewhat unfamiliar, and the assignments are larger than you have seen in previous courses.

Verbal directions: In the real world, many (most) of the directions you receive regarding projects will be verbal. This is your last “pure” programming class, so one of its purposes is to prepare you for programming in the real world. Therefore, some of the requirements for your programs will be given or clarified verbally. You are **expected** to follow these directions, not only those directions that are given in writing.

Pair Programming: In this course, you will have the opportunity to do programming in pairs. The idea here is that you will sit together at one computer, one person “driving” and the other watching, commenting, noting mistakes, and generally helping. Both people should spend time in each role. Note that you may only do one program with a given partner, so if you choose to do every program with a partner, you will have to have several different partners.

Homework: This course covers a lot of material, all of which is important to computer scientists whether headed to programming jobs or to graduate school. Much of the material is easy to follow when we work through it together on the board, but not so easy to retain and do on your own. Therefore, homework will be assigned regularly (almost every class period) to give you the opportunity to work through the material outside of class and make sure you really understand it. The in-class exam questions are usually very similar to homework problems. You may work on most written homework assignments in groups of up to four people. Turn in one copy of group homework with the names of all people who worked on it. Please take advantage of the group homework opportunity if you’re struggling in the course at all (or even if you’re not). Typically, people doing group homework as intended have scored higher on exams as well as homework.

Quizzes and Reading: Please read the assigned material **before** class and come prepared to participate in class discussion. I want to focus our class time on the material that you couldn’t get out of the book on

your own, and on working through examples. I can best do that if everyone in the class comes prepared. Participation will be factored into your classwork grade. Unannounced quizzes will be given periodically. They will cover the assigned reading as well as questions similar to the homework. The frequency of these quizzes will depend somewhat on the degree to which I can see that you've been doing your reading. I don't expect you to get everything out of reading the book – this is challenging material; but I do expect you to get something out of reading the book.

Course Grading Guidelines

A full distribution of grades from A to F is possible with A's being reserved for outstanding performance. A grade of C represents the minimal acceptable performance. The following rubric describes the levels of performance typically associated with each grade:

- A Outstanding performance. Student demonstrates solid conceptual understanding and insight. The student not only demonstrates mastery of the course content, but is able to make extensions or apply that knowledge to new situations. Assignments and tests are of excellent quality and the student contributes substantially to class discussions.
- B Good performance. Student demonstrates good understanding and mastery of the course content. Assignments and tests are of good quality but not exceptional. The student regularly contributes positively to the class discussion.
- C Adequate performance. Student demonstrates adequate understanding and mastery of course content but has difficulty extending or applying the knowledge to new situations. Assignments and tests are adequate. Student may not contribute as regularly to class discussions.
- D Inadequate performance. Student demonstrates inadequate understanding of course content. Assignments and/or tests are inadequate, but show effort. Student contributes little to class discussions.
- F Unacceptable performance. Student demonstrates little or no understanding of the course content. Assignments are not completed, are late, or of poor quality. The student does not contribute to class discussion. The student's work provides little evidence of effort.

Evaluation:

The various components of your grade will be weighted as follows:

Final:	15%
Midterms:	25%
Programs:	35%
Papers:	10%
Homework, quizzes, and participation:	15%

	100%

Important: Although programs are only 35% of your average, you must achieve a C average on programming assignments in order to receive a C in this course. This is because the ability to produce working and well-designed programs is a crucial part of your skills as a computer programmer.

Each assignment or test item will be graded in a holistic manner, based on a rubric. A general version of the rubric is provided below. (A rubric specific to programs will be provided with the first programming assignment. A specific rubric will also be provided for each paper.)

- A (5) **Outstanding performance.** Student demonstrates solid conceptual understanding and insight. All required components are clearly present. Material is well written, demonstrating coherent thoughts and reasoning as well as utilizes proper grammar and correct spelling. Programs not only work perfectly, but are well-designed and documented.

- B (4) **Good performance.** Student demonstrates good understanding and insight. All required components are present. Material is well written, demonstrating coherent thoughts and reasoning. Programs work at least on standard data and provided test cases and are well-designed and documented.
- C (3) **Adequate performance.** Student demonstrates adequate understanding and insight. Most required components are present. Material is written coherently, demonstrating adequate writing skills, but may contain numerous grammatical or spelling errors. Programs may contain a major execution error or several minor execution errors or may be poorly designed or documented.
- D (2) **Inadequate performance.** Student demonstrates inadequate understanding and insight. Required components are not present. Writing indicates little thought and reflection, or is of poor quality, making it difficult to read and understand. Programs run, but contain serious flaws.
- F (1 or 0) **Totally unacceptable performance.** Student demonstrates little to no understanding of the content. Work is not turned in, or most of the required components are missing. Writing indicates virtually no effort. Programs do not run or do not compile.

Grading scale:

- A: 4.5 or greater
- B: 3.75-4.49
- C: 3.0-3.74
- D: 2.0-2.99
- F: less than 2.0

Class Policies

Late work: Programs, papers, and other assignments completed outside class are due at the **beginning** of the class period. Graded homework will not be accepted late. You will receive two “late tickets” good for turning in a program or paper up to **5 days** late. Use of the late ticket will be explained in class. No assignment will be accepted late without a late ticket. Save your late tickets for real emergencies, don’t use them up on the first two programs. Unused late tickets will give a small boost (.05 each) to your final course average.

Time: This course requires a significant time commitment. Be prepared to spend several hours a week in the lab programming in addition to time spent studying the material. In other words, **do** actually study and expect to need time for your programs.

Academic honesty: Academic honesty is very important to me and to this university. You are expected to be aware of the student code, including the section on academic dishonesty (cheating and plagiarism). The minimum penalty for any form of academic dishonesty in this class will be a zero on the assignment in question. Do not work together on any project unless given explicit permission by me. Serious or repeated offenses will result in harsher penalties up to and including failure in the course. All cases of academic dishonesty will be reported to CRR as required by university policy and may result in disciplinary penalties as well as academic penalties.

Respecting class time: Class time is very important in this course. Although rare students are able to learn the course material from the textbook, the vast majority of you need the explanations and examples provided in class. Therefore, it is important that you help me to maximize the value of our class time in the following ways:

- 1) Read the assigned material before class. Note that quizzes can include questions over the assigned material.
- 2) Attend class regularly. Attendance is not included in the course grade, but participation is. It’s very difficult to participate when you are not present. It is also difficult to gain much from the explanations and examples I provide when you do not hear and see them. Note that you are

responsible for everything I say in class whether you're there or not. This includes verbal clarifications and additional specifications for your programming assignments.

- 3) Be on time. Class will begin promptly with the assignment for the following class period. If you must be late, slip in quietly, and remember to ask to see the assignment at the end of class.
- 4) Be respectful of class time. Pay attention and avoid providing interruptions such as cell phones ringing in the middle of class. I prefer a relaxed classroom environment, but it is important that all students be able to focus on the course material without distractions.

Web sites: I will be using two different web sites to provide course information to you via the web. A class calendar with assignments and links to the course slides along with some other course-related information can be found on the course web site at

<http://www.itkilstu.edu/faculty/mecalif/ITK279/Fall2008/index.htm>. I will also use Blackboard for electronic assignment submission, some course information, and access to your grades in the course. You can access Blackboard by going to <http://blackboard.ilstu.edu> and logging in. Your login ID is the same as your ULID, and your password is your general university password. There will be a link to Blackboard from the course web site.

Contacting me: If you need to contact me outside of office hours and class, your best bet is to send email. I read email multiple times a day, and have been known to answer questions late at night (though I do not promise to always be available then).

A comment about office hours: Most of you have been very successful in your previous programming courses. For most of you, this course will provide your first time getting really stuck on a program, your first failure to complete the program you started the night before, the first time that you didn't really **get** the concept in a programming course. In other words, it may be the first time you've really ever needed to go see the professor in an ITK course. **COME SEE ME.** Do **not** just get help from friends and each other. Your level of understanding and your grades will both improve if you come in when you first get stuck or have a question.

Any student needing to arrange a reasonable accommodation for a documented disability should contact Disability Concerns at 350 Fell Hall, 438-5853 (voice), 438-8620 (TDD).

Tentative Schedule

Week	Date	Topics
Week 1	Aug 18	Introduction C++
Week 2	Aug 25	C++ cont.
Week 3	Sep 1	Labor Day Dynamic Memory Recursion
Week 4	Sep 8	Program Performance Algorithm Analysis Lists, Stacks, Queues (review)
Week 5	Sep 15	Binary Trees (review) Priority Queues
Week 6	Sep 22	Sorting
Week 7	Sep 29	Sorting, cont.
Week 8	Oct 6	Binary Search Trees
Week 9	Oct 13	Hashing Disjoint Sets
Week 10	Oct 20	Graphs
Week 11	Oct 27	Graphs cont.
Week 12	Nov 3	Graphs cont. NP completeness
Week 13	Nov 10	Greedy Algorithms Divide and Conquer Dynamic Programming
Week 14	Nov 17	Randomized Algorithms Backtracking Red Black Trees
	Nov 24	Spring Break – NO SCHOOL
Week 15	Dec 1	Catch up and review

Final Exam Tuesday, December 9, 7:50 am