CS 301 – Algorithms and Complexity
Course Syllabus and Calendar – Fall 2006

Professor Don Colton

Brigham Young University Hawaii

CS 301 serves a crucial role in the CS curriculum. It stands at the middle, as a keystone to your preparations in lower classes, and as a gatekeeper to the upper classes.

1 Course Overview

The study of algorithms is focused primarily on speed. One can always buy more memory or a bigger hard disk. It just costs money. One cannot buy time.

The issues with speed revolve around the question of how best to approach each problem. Array search is a wonderful example. The brute-force lookup method examines each item in a set and stops when the desired item is found. Binary search divides the set into two halves and decides in which half the target would be. Then it repeats this procedure until the set has just one item left. At one second per comparison, and with a set of one million items, the brute-force method would take 11.5 days to find that the target is not in the set. The binary search method would take only about 20 seconds. For a large enough set, it is clear to see that even the fastest computer using the brute-force method cannot win against an ordinary computer using the binary search method.

The study of algorithms examines the running time of various programs and looks at some important algorithmic discoveries, such as the divide-and-conquer method used by the binary search. Students will gain skills in both **algorithm analysis** and **algorithm design**, and probably gain a few surprising insights along the way.

Prerequisites: Object-Oriented Programming (CS 202) is a prerequisite. For this class we expect you to have programming maturity based on programming experience. OO Programming reflects a desired level of maturity.

Discrete Math II (Math 202/L) is a prerequisite. In the discrete math classes you will have learned about trees, graphs, and other data structures and algorithms that are common in Computer Science. When we refer to these same concepts in CS 301, we will expect you to understand them already, or to (re)learn them rapidly.

1.1 The Course

- **Course Number:** CS 301
- **Title:** Algorithms and Complexity
- **Course Description:** Algorithmic analysis, strategies, fundamental algorithms, distributed algorithms, basic computability. (Prerequisites: CS 202, Math 202/L.)
- **Textbook:** *Introduction to Algorithms, 2/e*, by: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. McGraw Hill, 2001. ISBN: 0-07-013151-1.
- **Class Time:** MWF 12:00–12:50 PM
- **Final Exam:** Wed 13 Dec, 11:00–2:00 PM
- **Classroom:** GCB 143

1.2 The Instructor

- **Instructor (me):** Don Colton
- **My email:** don@colton.byuh.edu
- **My Office:** GCB 130 B
- **Office Hours:** Daily 11:00 to 11:50 AM

1.3 My Office Hours

My office hours are shown above. You can contact me by email to make an appointment at another time. I also have an open-door policy: **If my door is open (even just a bit) feel free to knock and come in.** (My door is usually open.)

1.4 Subject to Change

It is possible that I will revise some aspects of the course as we go along. Any changes I make are likely to be to your advantage. If any of my changes seems unfair to you, let me know. I will try to correct it.

1.5 Special Needs

Brigham Young University Hawaii is committed to providing a working and learning atmosphere, which reasonably accommodates qualified persons with disabilities. If you have any disability that may impair your ability to complete this course successfully, please contact the students with Special Need Coordinator, Leilani A'una at 293-3518. Reasonable academic accommodations are reviewed for all students who have qualified documented disabilities. If you need assistance or if you feel you have been unlawfully discriminated against on the basis of disability, you may seek resolution through established grievance policy and procedures. You should contact the Human Resource Services at 780-8875.

1.6 Preventing Sexual Harassment

Title IX of the education amendments of 1972 prohibits sex discrimination against any participant in an educational program or activity that receives federal funds, including Federal loans and grants. Title IX also covers student-to-student sexual harassment. If you encounter unlawful sexual harassment or gender-based discrimination, please contact the Human Resource Services at 780-8875 (24 hours).

2 Course Calendar

Generally the lectures and discussion in class will precede the due dates for the various assignments (shown below).

Oct 07:	sSeq	REQ sequential search
Oct 07:	sBinary	REQ binary search
Oct 14:	sSelect	REQ selection sort
Oct 14:	sInsert	REQ insertion sort
Oct 14:	sBubble	REQ bubble sort
Oct 21:	sMerge	REQ merge sort
Oct 21:	sHeap	REQ heap sort
Oct 21:	sQuick	REQ quick sort
Oct 28:	stack	REQ stack
Oct 28:	queue	REQ queue
Nov 04:	hash1	REQ hash (single)
Nov 04:	hash2	REQ hash (double)
Nov 11:	pq	REQ priority queue
Nov 11:	bst	REQ binary search tree
Nov 18:	lcs1	longest common subseq
Nov 18:	huffman	Huffman Coding
Nov 25:	bfs	REQ breadth first search
Nov 25:	idfs	iterative depth first
Nov 25:	mst	minimum spanning tree
Dec 02:	vexed1	Vexed basics
Dec 02:	vexed2	Vexed max 4 solver
Dec 02:	lcs2	LCS list of options
Dec 08:	sRpt1	REQ sort report (basic)
Dec 08:	sRpt2	sort report (excellence)
Dec 13:	final	11-2, in class

3 Quizzes

Several quizzes have been developed to test your knowledge and skill. You must demonstrate mastery by earning a perfect score on each quiz. Each quiz is available online at quizgen.org. Before the quiz is given in class, you must practice outside of class until you earn a perfect score. Then the quiz will be given twice in class. I will keep your highest score. Once you pass a quiz perfectly, you do not need to take it again. The quizzes currently include Big Oh, Recurrence relations, Heapsort heapify, Quicksort partition, Double hash probe sequence, Longest common subsequence, Huffman coding, and Minimal spanning tree. Others are planned, including Shortest paths and Maximum flow.

4 Grading

Because of the nature of CS 301, as a prerequisite to almost all the classes that follow, there will not be any D grades. If you do not reach a sufficient level of demonstrated skill and performance, you will receive an F, not a D. That is to force you to retake the class until you do demonstrate the expected level of performance. You cannot earn a D in this class by attending and appearing to work hard.

To pass the class:

(1) you must do reasonably well at demonstrating how the studied algorithms function. That means, for example, that I can give you a list of numbers and tell you to heapify them. In all such cases we will do mini-quizzes in class to cover that material so you are familiar with it for the exam. Typically everyone does well enough in this category to pass the class. Most will probably do well enough for an A in this category.

(2) you must do reasonably well at demonstrating that you know the terminology used in the book and lectures. You show this by writing short paragraphs describing specific terminology, telling why it is significant, what it means, how it is used, or

something like that. Each major exam will have questions of this type.

(3) you must do reasonably well at completing the labs, and especially the ones assigned early in the semester. In particular, the labs labeled REQ (required): both search labs, all six sort labs, stack, queue, hash1, hash2, p1, bst, and bfs must be completed to pass the class. If you get that far, you can expect at least a C in the class, as far as labs are concerned. Additional labs including lcs1 and beyond will help me decide whether you earn an A, B, or C. This is the category that will be the most trouble for the most students.

(4) you must submit a reasonably good sort report. If you do not turn one in, you fail the class. The quality is negotiable, and will divide students into A, B, and C categories.

Aside from all that, grades will be computed on the basis of points earned generally as follows.

ceil	avail	req	category
100	4*42	0	daily classwork
375	375	375	required labs (must do all)
500	600	400	all programming labs
100	100	60	sort report
300	300	180	midt, final
1000		700	total

In each category, you must reach the "req" required minimum to avoid failing the class. In some categories (classwork and labs) there are more points available than you can keep (the ceiling). This gives you the option of skipping a certain number of assignments without it hurting your grade. For example, in the daily classwork category, at four points per day, after 25 days you have full credit in that category. In the programming labs category, you can skip 100 points of labs and still get full credit (500) for that category.

Grading Scale: I use the following grading scale for this class.

930+	A	900–929	A-	870–899	B+
830–869	B	800–829	B-	770–799	C+
730–769	C	700–729	C-	0–699	F

Final Exam Score: You must achieve a sufficient score on the final exam, as shown in this table. Your final grade will be the **lower** of your total-points grade (above) and the grade in this table based on your final-exam percentage.

83+	A	80–82	A-	77–79	B+
73–76	B	70–72	B-	67–69	C+
60–66	C	50–59	C-	0–49	F

Attendance: Attendance and in-class participation counts for 10% of your final grade. Attendance is worth 4 points per day: full credit for attending the full class period; partial credit for attending part of the class period. Missing and unnoticed persons get zeros.

Due to INS (immigration) and VA (veterans) requirements the Vice President for Student Life is supposed to be notified whenever a student misses four consecutive class days. I try to do this.

In class I follow a general "got questions?" teaching philosophy. It leaves the responsibility for learning with the people that are supposed to learn: the students. (I cannot learn for you.) Canned lectures can be fun and exciting, but frequently the relevant material is in the reading. Our class time will be focused on things you need to do the nearby assignments, or on explaining things that may not be sufficiently clear from the reading.

Reading: I am using *Introduction to Algorithms, second edition* by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. McGraw Hill, 2001. ISBN: 0-07-013151-1.

I will occasionally assign chapters to be read. The chapters will also be discussed in class.

Labs/GradeBot: Some of your time will be spent writing programs. They will be graded by my robotic grader, GradeBot. GradeBot is generally available 24 hours a day, seven days a week, to grade and return your lab assignments. This is currently done via web, "turnin," or email.

For grading, GradeBot is correct and authoritative. It is your boss. It is your client. It is your Drill Sergeant. There is always a particular correct behavior that it demands. You must make your program behave in exactly the way that GradeBot is requiring (including spelling errors, if any). Be sure to look at a sample "conversation" with GradeBot before you start writing your program.

If you discover a case where you believe that GradeBot is wrong, tell me about it. If you found an error in GradeBot, I generally reward you with some extra credit. Otherwise, you must assume GradeBot is right.

Sort Report: A term paper is required. It is called the Sort Report. After you have programmed several sorts, including insertion, selection, heap, merge, and quick, you will do empirical measurements of performance and write a report about it. Detailed instructions are provided.

5 Course Content

The CS 301 course covers the following CC2001 Knowledge Units. These are defined in Computing Curricula 2001, a joint project of IEEE-CS and ACM. The IEEE Computer Society and the Association for Computing Machinery are the two major professional societies in computer science.

AL1. Basic algorithmic analysis

- Asymptotic analysis of upper and average bounds
- Differences among best, average, and worst case
- Big O, little o, Ω (omega), and Θ (theta) notation
- Standard complexity classes
- Empirical measurements of performance
- Time and space tradeoffs in algorithms
- Using recurrence relations to analyze algorithms

AL2. Algorithmic strategies

- Brute-force algorithms
- Greedy algorithms
- Divide-and-conquer
- Backtracking
- Branch-and-bound
- Heuristics
- Pattern matching and string/text algorithms
- Numerical approximation algorithms

AL3. Fundamental computing algorithms

- Simple numerical algorithms
- Sequential and binary search algorithms
- Quadratic sorting algorithms (selection, insertion)
- $O(n \lg n)$ sorting alg (quick-, heap-, merge-)
- Hash tables, incl collision-avoidance strategies
- Binary search trees
- Representations of graphs (adjacency list, matrix)
- Depth- and breadth-first traversals
- Shortest-path algorithms (Dijkstra and Floyd)
- Transitive closure (Floyd's algorithm)
- Minimum spanning tree (Prim and Kruskal)
- Topological sort

AL4. Distributed algorithms

- Consensus and election
- Termination detection
- Fault tolerance
- Stabilization

AL5. Basic computability

- Tractable and intractable problems
- The halting problem