

CSCI 431: Algorithm Analysis

(Winter 2006)

Professor: Keith Schubert
Office: JB 348
Phone: 880-5328
Email: schubert@csci.csusb.edu
Web: www.csci.csusb.edu/schubert
Prerequisites: CSCI 330, Math 372

Text: Kleinberg & Tardos,
Algorithm Design
Class: MW 12:00-1:50 PM, JB 146
Office: MW 2:00-4:00 PM
and by appointment

1 Objective:

In this graduate course modern concepts and techniques in the design of high performance

1. To provide computer science students with the tools to analyze algorithms
2. To develop skills to design algorithms
3. To teach the student how to recognize and demonstrate difficult problems
4. To develop in students an appreciation for the need to carefully analyze algorithms.

2 Prerequisite Knowledge

I expect that you are familiar with the basics of data structures, and have seen complexity briefly before.

3 Grading:

Your grade is determined from three basic areas: homework (20%), midterm (35%), and a final (45%). Homework and labs have been weighted heavily since they tend to be better for students. If the final is better than your homework grade, I will shift 10% from homework to final (thus homework 10%, final 55%).

The grading scheme is listed below.

Ltr	Percent	Ltr	Percent
A	94 - 100	C	73 - 76
A-	90 - 94	C-	70 - 73
B+	86 - 90	D+	66 - 70
B	83 - 86	D	63 - 66
B-	80 - 83	D-	60 - 63
C+	76 - 80	F	00 - 60

4 Reading

The reading assignments listed cover the material that will be discussed in class. You are to read it in

advance for each class (except the first, which would not be possible). There are two main reasons for reading the material ahead of time.

1. Reading the text before class gives you the basic ideas, so the deeper truths can be covered in class.
2. Even the best of textbooks will cause most people to get confused on some aspects. If you read the book before the class, your confusion will get cleared up and make the course easier for you.

I know that reading before class is not frequently done, but I encourage you to do it. You are here to get a top quality education, but in order to get it you must do more than just show up. Graduate students must be motivated to learn and succeed. Don't sell yourself short.

5 Getting Help

Everything always seems easier in class. The goal of this course is to prepare you to understand and design digital logic, not to frustrate or confuse you. You will not know what is hard or confusing until you try though. When you hit that problem that you can't figure out, don't get frustrated, get help. You are highly encouraged to take advantage of office hours. Office hours are the premiere assistance methodology of this class. You are also encouraged to discuss problems and methods with each other. Study groups can be very helpful. Do not cheat yourself though by getting solutions and not understanding! All work must be your own. You can discuss and help, but may not copy someone else's work, or allow your work to be copied. That is plagiarism and is treated very severely.

Schedule

Date	Topic	Reading	Homework Due
1/9	Introduction and Stable Matching	1	
1/11	Complexity, Tractability, and Order of Growth	2.1-4	Ch 1: 8
1/16	Martin Luther King Holiday	-	-
1/18	Understanding Complexity and Asymptotic Notation	2.4-5	
1/23	Graphs: Basics, Searching	3	Ch 2: 2, 6
1/25	Graphs: Backtracking	3	
1/30	Greedy Algorithms: scheduling and caching	4.1-3	Ch 3: 7
2/1	Greedy Algorithms: Dijkstra and Kruskal	4.4-6	
2/6	Greedy Algorithms: Clustering and Huffman Codes	4.7-8	Ch 4: 13, 24
2/8	Midterm	1-4	
2/13	Divide and Conquer: Mergesort and Recurrence Relations	5.1-3	
2/15	Divide and Conquer: Closest Points and Multiplication	5.4-5	
2/20	Divide and Conquer: Convolution and FFT	5.6	Quicksort Problem
2/22	Dynamic Programming: weighted scheduling & Least Squares	6.1-3	
2/27	Dynamic Programming: Knapsacks and RNA	6.4-5	Ch 6: 1, 8
3/1	Dynamic Programming: Sequence Alignment	6.6-7	
3/6	Dynamic Programming: Shortest Distance	6.8-10	
3/8	NP: Reducability, SAT, 3-SAT	8.1-3	Ch 6: 18, 24
3/13	NP: NP-Complete, Sequencing, Partitioning	8.4-6	
3/15	NP: Coloring, Numerical Problems, Co-NP, Taxonomy	8.7-10	
3/20	Approximation Algorithms	11	Ch 8: 11, 37
3/22	Final (12:00 pm)	1-6, 8, 11	

6 Homework:

This course is a theory course, which means that how you get the answer is as important (if not more important) than what the answer is. *Given that your work is the most important part, no credit will be given to homework which just gives the results or that the work is not decipherable.* All homework must be neat, organized, and show all the steps. Assembly programs must be typed. Homework must have

Name
Date
Assignment #
CSCI 431

in the upper right corner of the first page. Multiple page homework must be stapled. Homework is due at the beginning of the class meeting on the due date. Late homework will be not accepted.

Students are encouraged to discuss class material, *but the work must be done individually.* The homework and all other graded work should reflect the effort of the individual who receives credit for it. Cheating will not be tolerated. The student may never copy other student's work, nor allow others to copy one's own work. If two assignments look exces-

sively similar, and are not narrow enough to justify the similarity, automatically a grade of zero results, with the likely referral to appropriate university bodies. According to the current CSUSB Bulletin, the offending student may receive penalties up to and including expulsion. Again, students are allowed and encouraged to discuss the material related to assignments, but when it comes to actually doing the assignment work it is to be done individually.

7 Midterm and Final

The midterm and final are closed book, but I allow 2 pages of 8.5×11 front and back of notes in the midterm and 4 pages for the final (theoretically two from the midterm and two new). You should bring paper to write on and a pen or pencil. You cannot use a calculator. The final is

Wednesday March 22 at 12:00 P.M.

8 Other Information:

The student is responsible for all material covered in class, and also for all announcements made therein.

9 Quicksort Problem

Assume you have an unsorted list with n items, that you want to sort from smallest to largest. You want to use a divide and conquer algorithm as outlined below.

```
Data: a[:], i, j
Result: a[i:j] sorted
if  $i < j$  then
  p=i;
  for  $k=i+1:j$  do
    if  $a[k] < a[i]$  then
      p++;
      swap(a[h],a[k]);
    end
  end
  swap(a[i],a[k]);
  quicksort(a,i,p-1);
  quicksort(a,p+1,j);
end
```

Algorithm 1: Quicksort

What is the expected running time for quicksort? What is the worst case running time? When does the worst case happen? Code up quicksort in C++ and have it sort a random vectors of $n=[10,100,1000,10000]$ items. How does the running time you measured compare with your predictions of the expected running times? Now have your algorithm sort a worst case situation. How do the times compare?

Is bubble sort ever faster? If so characterize when it happens. If not prove that it can't happen.

```
Data: a[:]
Result: a[:] sorted
 $n = |a|$ ;
for  $i=0:n-2$  do
  NumberSwaps=0;
  for  $j=0:n-2-i$  do
    if  $a[j+1] < a[j]$  then
      swap(a[j],a[j+1]);
      NumberSwaps++;
    end
  end
  if  $NumberSwaps=0$  then
    break;
  end
end
```

Algorithm 2: Bubble Sort