Computer Science
Dartmouth College

Computer Science 105
Algorithms (Graduate Level)

Winter 2005

Amit Chakrabarti

[Announcements | Administrative Basics | Handouts | Scribe Notes | Course Journal]

Course Description

In this course we shall advanced topics in the area of computer algorithms. Since Algorithms as a field of research is extremely vast, any choice of topics will necessarily cover only a tiny fraction of the field. The choices made for this course will lean towards relatively recent basic research work in algorithms in order to give computer science graduate students (the intended audience) a feel for what working in algorithms is like.

A strong mathematical background and a love of mathematical reasoning are expected and may be required frequently.

Announcements

- [Mar 9] Some solutions to HW2 have been posted. You are free to refer to them, and to everything else linked from this page, while working on your final exam.
- [Mar 7, 6pm] The **final exam is here**. Please read it and think a little about each problem before class on Wednesday. The exam is due Mar 14, 5pm sharp.
- [Mar 4] Play around with the applet at this website to gain some intuition for the KMP algorithm.
- [Feb 21, 6pm] A change has been made to clarify problem 3 in HW2. Please reload the homework in your browser.
- [Feb 19] Some solutions to HW1 have been posted. Please read and compare with your own solutions.
- [Feb 16] The second and final homework, Homework 2, is up on the website and is due Feb 28, before class.
- [Feb 3] Problem 2 in the homework has been spelled out to take care of different versions of Problem 11-4 in different printings of [CLRS]. Please make sure you're solving the correect problem in your homework!
- [Jan 26] Homework 1 is up on the website and is due Feb 7, before class.
- [Jan 3] You will be required to use LaTeX for your work in this course. Homework 0 tests your proficiency with LaTeX; please take it seriously and do it ASAP.

Administrative Basics

Lecture	Sudikoff 214 | 10 hour | MWF 10:00-11:05, X-hr Th 12:00-12:50
Instructor	Amit Chakrabarti | Sudikoff 107 | 6-1710 | Office hours: MW 11:30-12:30 and 15:00-16:00; also by appointment
Textbook	Required: none

Recommended:
"Introduction to Algorithms", by Cormen, Leiserson, Rivest and Stein, MIT Press.
"Randomized Algorithms", by Motwani and Raghavan, Cambridge University Press.

Prerequisites CS 25, CS 39

Work Scribing and preparing lecture notes for one week.
Class presentation on one of the topics in the list, or a related topic of the student's choosing.
Two homeworks.
One take-home final exam, due around March 15 (date to be confirmed later).

Handouts

Handouts are accessible only from within Dartmouth, for copyright reasons.

1. "Storing a Sparse Table with O(1) Worst Case Access Time". Fredman, Komlos, Szemeredi. JACM 31(3), 1984, pp. 538-544.
2. "Self-Adjusting Binary Search Trees". Sleator, Tarjan. JACM 32(3), 1985, pp. 652-686.
3. "A Simple Min-Cut Algorithm". Stoer, Wagner. JACM 44(4), 1997, pp. 585-591.
4. "A Randomized Linear-Time Algorithm to Find Minimum Spannning Trees". Karger, Klein, Tarjan. JACM 42(2), 1995, pp. 321-328.
5. "Linear-Time Pointer-Machine Algorithms for Least Common Ancestors, MST Verification, and Dominators". Buchsbaum, Kaplan, Rogers, Westbrook. STOC 1998.
6. Two sets of notes on maximum matching algorithms: Vempala's notes (detailed) and Tarjan's notes (very concise).
7. Khanh Do Ba's lecture notes for his lecture on the shortest superstring problem.

Scribe Notes

Here are the unedited scribe notes. I will try to edit and collate these some time soon.

- Probability theory and hashing (Sara Sinclair)
- Splay trees (Chien-Chung Huang)
- Min cut (Anne Loomis)
- Minimum spanning trees (Elizabeth Moseman)
- MST verification (Rajendra Magar)
- Maximum matching (Paritosh Kavathekar)
- Approximation algorithms: vertex cover, TSP, set cover and shortest superstring (Valika Wan and Khanh Do Ba)
- Weighted vertex cover and subset sum (Soumendra Nanda) [One | Two | Three]
- String matching (Ivelin Georgiev)

Course Journal, Homeworks

Week 1

- **Homework Zero:** Learn to typeset mathematical writing using LaTeX, by the first day of class (i.e., January 5). Then, typeset at least one page of one of these two PDF documents: Doc 1 or Doc 2. Submit your ".tex" and ".pdf" files by the second day of class (i.e., January 7).
- We reviewed the basics of random variables and expectation, then defined the concept of a family of universal hash functions. We bounded the expected search time, under a universal hashing scheme with linear probing to resolve collisions, to be 1 + (load factor).

Week 2

- We showed how to construct a simple number-theory-based family of universal hash functions and proved the universality of this family. We then discussed perfect hashing and showed how an easy-to-compute perfect hash function could be constructed if only we were allowed a quadratic-sized hash table.
- Using two-level hashing with universal hash functions, we showed how to build an overall $O(n)$-sized hash structure for storing n keys. The construction was randomized. We showed that it is expected to succeed within $O(n)$ time.
- Sara Sinclair showed how to improve upon the above construction and get the storage space down to $(n + o(n))$, for n keys.
- We discussed splay trees, described the splaying operation and the potential function used in the amortized analysis of the cost of splaying.

Week 3

- We completed the proof of the access lemma, which bounds the amortized cost of splaying. Using it, we proved the Balance Theorem (total access time bound) and the Static Optimality Theorem for splay trees. We had to skip the Static Finger Theorem for lack of time.
- Chien-Chung proved the Static Finger Theorem and the Long Splay Theorem. He described how to perform updates (join/split/insert/delete) on splay trees and also presented (without proof) the Snapshot Theorem. There are many more things about splay trees discussed in the paper but unfortunately our course is too short to do them all justice, so we must move on.
- After recalling the definition of a min cut and considering how it can be computed using $(n – 1)$ "parallel" max flow computations, we described Karger's contraction-based randomized algorithm. We analysed it to obtain a running time of $O(n^4 \log n)$ and a success probability of $(1 – n^{-c})$ for any positive constant c.

Week 4

- We described a recursive contraction-based algorithm for min cut that achieves a run time of $O(n^2 \log^3 n)$ and a success probability of $(1 – n^{-c})$ for any positive constant c. This, therefore, beats the best known max flow time bound as well as the best known deterministic time bound (to be described next).
- Anne Loomis described what is currently the fastest (and simplest) known deterministic min cut algorithm, due to Stoer and Wagner. She analyzed its running time, proved its correctness and showed us an example run.
- **Homework One:** Here it is. This is due on Mon Feb 7, before class. Please read *all* the instructions carefully before you start. Have fun solving!
- Minimum Spanning Tree (MST) was our new topic. We studied basic graph-theoretic properties of spanning trees, leading up to what we're calling the "Switch Theorem". We described the *Red Rule* and the *Blue Rule* for MSTs and proved the correctness of the Red Rule.

Week 5

- We proved the correctness of the Blue Rule. After considering the colourful history of MST algorithms, we described the three classic algorithms: Kruskal's, Prim's and Boruvka's. We proved their correctness using the Red Rule, Blue Rule and Blue Rule respectively.
- We defined the concept of F-heavy edges in a graph, with respect to a sub-forest F. Then we

described the Karger-Klein-Tarjan randomized MST (actually MSF, minimum spanning forest) algorithm and proved its correctness using the Red and Blue rules and induction.

- Elizabeth Moseman analyzed the worst case and the exected running times of the Karger-Klein-Tarjan algorithm. The latter turns out to be linear, provided we assume that MST verification can be done in linear time. Lizz also gave intuition for why the algorithm finishes in linear time with very high probability.
- We considered the MST verification problem, which makes up the last and most complicated piece of the Karger-Klein-Tarjan MST algorithm. We discussed how to perform m least common ancestor (LCA) computations in a rooted n-vertex l-leaf tree in time $O(m\alpha(m,l) + n)$. We then introduced the *microtree technique* and gave intuition for why it might help reduce the running time to $O(m + n)$. We shall eventually use LCA computations to do MST verification.

Week 6

- Rajendra Magar gave some details of the microtree technique. He then described an $O(m\alpha(m,l) + n)$-time MST verification algorithm very similar to last time's LCA algorithm. Finally, he gave a brief outline of how to reduce the running time to $O(m + n)$ using the microtree technique and further tricks.
- We considered the maximum matching problem. We defined augmenting paths and proved Berge's Theorem which gave us the nucleus of a polynomial time maximum matching algorithm.
- The carnival holiday ate up one class.

Week 7

- We considered a forest-growing algorithm for finding an augmenting path. It turned out that to make it work we had to assume that the input graph was bipartite. This led to a maximum matching algorithm for bipartite graphs.
- Paritosh Kavathekar proved a cycle-shrinking lemma which helped us modify the previous algorithm suitably to get an algorithm that worked on general graphs. This was, in essence, Edmond's famous "shrinking blossoms" algorithm.
- **Homework Two:** Here it is. This is due on Mon Feb 28, before class. As before, please read *all* the instructions carefully before you start. Do not forget to have fun solving these!
- We defined the notion of approximation algorithms (for minimization problems) and gave a simple 2-approximation algorithm for the NP-hard minimum vertex cover problem.

Week 8

- We considered approximation algorithms for the traveling salesman problem (TSP). We showed that finding a k-approximation, for any constant k, is NP-hard. Then we considered the more specialized *metric* TSP for which we gave a 2-approximation algorithm.
- Valika Wan introduced the set cover problem and gave a greedy algorithm with approximation ratio $H(d_{\max})$, where $d_{\max}$ is the size of the largest set in the instance and $H(k)$ is the k^{th} harmonic number. On a universe of size n, this is therefore a $(\log n + O(1))$-approximation algorithm.
- Khanh Do Ba introduced the shortest superstring problem and used a set cover approximation algorithm as a subroutine to obtain a $(2H_n)$-approximation. In fact he needed an approximation algorithm for *weighted* set cover, but noted that the algorithm presented by Valika handles that too.
- Linear programs are an extremely important tool in designing approximation algorithms. We introduced linear programs (LPs) and noted that there are known algorithms that solve LPs in polynomial (but not strongly polynomial) time. We formulated the vertex cover problem as in integer program (IP), relaxed this IP to an LP, and "rounded" the fractional optimal solution to the LP to a solution to the IP to get a 2-approximation.

Week 9

- We obtained an O(log *n*)-approximation for weighted set cover using the LP rounding technique. This time our rounding algorithm was randomized.
- Soumendra Nanda presented a PTAS for the subset sum problem.
- Our final problem is string matching. We saw the linear time Knuth-Morris-Pratt algorithm, which maintains a left pointer and a right pointer into the text to be searched and cleverly updates them to skip over unnecessary comparisons. The algorithm requires a preprocessing step which is itself a computationally interesting problem.

Week 10

- Ivelin Georgiev described the preprocessing step used in the KMP algorithm. This step computes, for every prefix *p* of the pattern, the longest proper prefix of *p* that is also a suffix of *p*. Ivelin showed how to do the whole thing in linear time. The proof of correctness is quite complicated.
- Here is the **final exam**.
- Final remarks.

Teaching *Home*