Projects for Analysis of Algorithms

Andreas Klappenecker

Project 1 (Cliques in Graphs)**.** The Hamming distance $\mathrm{dist}(u, v)$ between two binary vectors $v = (v_1, \ldots, v_n)$ and $w = (w_1, \ldots, w_n)$ is the number of indices k such that $v_k \neq w_k$. A fundamental question in coding theory is to determine the number

$$A(n, d) = \max |\{S \subset \{0,1\}^n \mid \mathrm{dist}(u, v) \geq d \text{ for all distinct } u, v \in S\}|,$$

the maximal number of binary vectors of length n that one can find such that any two distinct vectors have a Hamming distance $\geq d$. For example, $A(5, 4) = 2$.

The Hamming graph $H(n, d) = (V, E)$ is the graph with 2^n vertices V given by binary strings of length n. We have $(u, v) \in E$ if and only if $\mathrm{dist}(u, v) \geq d$.

The number $A(n, d)$ coincides with the size of a maximal clique in $H(n, d)$. Find an implement "efficient" algorithms to compute the maximal clique in the Hamming graph (but note that the problem to compute maximal cliques is NP hard).

Background http://www.research.att.com/~njas/doc/pace2.ps

Challenge http://www.research.att.com/~njas/doc/graphs.html

Project 2 (Minimum Spanning Trees)**.** Finding the minimum spanning tree of an undirected connected graph (V, E) with weight function $w\colon E \to \mathbf{R}$ is an old problem in computer science. We have discussed the algorithms by Kruskal and by Prim. The asymptotically fastest algorithm known to date is due to Chazelle. It is an improvement of Borůvka's algorithm, the first algorithm that was developed for this problem. Extend the Boost graph library by Borůvka's and Chazelle's algorithms. Compare the performance of the algorithms by Borůvka, Chazelle, Kruska, and Prim.

Note that you should have some experience with C++ to attempt this project. You can also decide to implement the algorithms without integration into the Boost graph library, but the result will be less interesting.

Background Nesetril, Milková, and Nesetrilová on Borůvka's algorithm
http://citeseer.ist.psu.edu/nesetril00otakar.html
Chazelle's algorithm
http://www.cs.princeton.edu/~chazelle/pubs/mst.pdf

Challenge Measure and document the performance of the implementations for graphs that occur in interesting applications. Knuth's Stanford graphbase is contains numerous interesting examples.

Project 3 (The Traveling Salesman Problem, DIMACS Challenge)**.** The traveling salesman problem asks to determine the shortest tour for a salesman to visit each city of a given list. Finding the best possible tour is NP-hard, but there exist surprisingly good heuristics. Take the DIMACS challenge, implement several heuristics, and compare your results with the state of the art.

Background http://www.research.att.com/~dsj/chtsp/

Challenge http://www.research.att.com/~dsj/chtsp/download.html

Project 4 (Probabilistic Algorithms for k-SAT)**.** k-SAT is a canonical NP-complete problem. Recall that the SAT problem asks to determine whether or not a set of clauses in conjunctive normal form is satisfiable. The problem is called k-SAT if each clause has at most k literals.

Recently some progress has been made in designing randomized algorithms for k-SAT. Two randomized algorithms are given below. Both of them are extremely simple (a few lines of pseudo-code) and run in exponential time. The first one is based on a backtrack search procedure and the second one is based on a random walk. The second algorithm is superior *asymptotically*. You task is to determine whether or not the same is true *empirically*. Implement both algorithms and compare their performance in solving 3-SAT instances. Two sets of instances are provided. The first set consists of benchmark 3-SAT instances, and the second one consists of randomly generated 3-SAT instances with *unique solutions*. Start with small instances and report which is the largest instance that is solvable by your program in a certain amount of time (you decide the time limit). Describe your implementations and explain your findings in a way that is understandable to your classmates.

Background

- Algorithm 1: R. Paturi, P. Pudlak, and F. Zane: Satisfiability coding lemma. *Proceedings of the 38th IEEE Symposium on the Foundations of Computer Science*, 1997, pages 566-574.
- Algorithm 2: Uwe Schöning: A Probabilistic Algorithm for k -SAT Based on Limited Local Search and Restart. *Algorithmica* 32(4): 615-623 (2002)

3-SAT Challenge Instances

- Benchmark instances:
  http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/benchm.html
- Program that randomly generates unique solution 3-SAT instances :
  http://www.is.titech.ac.jp/~watanabe/gensat/a1/

Project 5 (Quadratic Sieve)**.** The quadratic sieve is one of the first algorithms to factor integers into primes in subexponential time. Implement the standard quadratic sieve algorithm by Pomerance and take great care to realize the linear-algebra part of the algorithm efficiently. In the basic quadratic sieve, one chooses integers x near $\sqrt{n}$ to search for values $x^2 - n$ that are B-smooth, *i.e.*, that has all factors in the range $[1, B]$. As the numbers x deviate from $\sqrt{n}$, the B-smooth numbers thin out rapidly. One way to get around this problem by choosing multiple polynomials, a method that has been suggested by Davis, Holdridge, and Montgomery. Implement the multiple-polynomial quadratic sieve as well. Find the right trade-offs between the size of the factor base, the number of polynomials, and fine-tune the implementations.

Background R. Crandall, C. Pomerance: *Prime Numbers – A Computational Perspective*, Springer, 2001; Chapter 6.
C. Pomerance, *The Quadratic Sieve Factoring Algorithm.* In: Advances in Cryptology: Proceedings of EUROCRYPT 84 (Ed. Th. Beth, N. Cot, and I. Ingemarsson). pp. 169-182, Springer, 1985.

Challenge Factor integers such as
RSA-100 =
1522605027922533360535618378132637429718068114961\3
8068865790849458012296325895289765400035069200613 9
(100 digits, checksum = 294805)

Project 6 (Matrix Multiplication)**.** The multiplication of $n \times n$ matrices takes $O(n^3)$ with the standard multiplication algorithm. We can improve the time complexity of matrix multiplication to $O(n^\omega)$, with $\omega = \log 7 / \log 2$, using Strassen's well-known divide-and-conquer algorithm. Coppersmith and Winograd showed that there exists a matrix multiplication algorithm that improves the exponent to $\omega \leq 2.38$, but the algorithm has a prohibitively large constant and is apparently not used in practice. Your task is to find and implement matrix multiplication algorithms that have a good performance for $n \times n$ matrices of, say, length $10 \leq n \leq 1000$.

Background V. Strassen, *Gaussian Elimination is not Optimal*, Numer. Math. 13, p. 354-356, 1969
D. Coppersmith, S. Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Comput. 9, p. 251-280, 1990

Challenge What is the largest n so that you can multiply two $n \times n$ matrices within 1 second, 10 seconds, or 1 minute on a platform of your choice?

Project 7 (Shortest Vector)**.** Suppose that $a_1, \ldots, a_n$ are n linearly independent vectors in $\mathbf{Q}^n$. The lattice L generated by $a_1, \ldots, a_n$ is the set of all integer linear combinations of these vectors, $L = \{\lambda_1 a_1 + \cdots + \lambda_n a_n \,|\, \lambda_k \in \mathbf{Z} \text{ for all } 1 \leq k \leq n\}$. The goal is to find the shortest nonzero vector in L with respect to the Euclidean norm. In dimensions $n = 1$ and 2, the problem can be solved in polynomial time. In higher dimensions, one can find approximate solutions using the LLL algorithm.

Background D. Micciancio and S. Goldwasser, *Complexity of Lattice Problems: A Cryptographic Perspective*, Kluwer, 2002
V.V. Vazirani, *Approximation Algorithms*, Springer, 2001; see Chapter 27.

Challenge TBD

Project 8 (Network Reliability)**.** Given a connected, undirected graph $G = (V, E)$, with failure probability p_e specified for each edge $e \in E$, compute the probability that the graph G becomes disconnected. Put differently, the graph becomes disconnected if all edges in some cut $(C, \overline{C})$, with $C \subset V$, fail.

The project should find and implement algorithms which estimate the probability that the graph becomes disconnected. If each edge has the same failure probability, then this is essentially a counting problem. An exact calculation of the probability that the network partitions is a herculean task, since this problem is ♯P-complete. Fortunately, there exist fully polynomial randomized approximation schemes.

Background Some basic references are:
V.V. Vazirani, *Approximation Algorithms*, Springer, 2001; see Chapter 28.
http://theory.lcs.mit.edu/~karger/Papers/reliability-sirev.ps

Challenge Obtain the data of real-world networks and calculate the probability that the network partitions due to edge failures.