Computer Science 547 – Discrete Mathematics
Prof. Steven Skiena
Spring 1999

Proposals Due: March 17, 1999
Projects Due: May 4, 1999

The project will involve some concentrated work in one topic related to discrete mathematics or the design and analysis of algorithms. Projects can fall into one of three categories - survey papers, implementations, and original research. Below I list possible projects of all three types, but don't feel restricted by these subjects. If you can convince me that what you want to do is interesting, you can do it, however I strongly discourage projects derived from your own thesis research.

How much work should a project be? It is difficult to make a blanket statement, since the projects fall into different classes. For survey papers, 10 pages should be sufficient to cover a topic in some depth. Implementations vary in length according to the nature of the problem and subtlity of the algorithm, but 1000-1500 lines is a ballpark figure. Original work will be even more variable - evidence must be given that you have obtained a good understanding of the problem and have made some (not necessarily much) progress towards solving it. If you wish, you may work in groups, but the work expected will increase linearly with the number of people involved. If you have any questions about what is required, please come and talk to me.

A one page proposal on what you intend to do must be approved by me before the deadline above. The last week of the semester everyone will give a short talk about what they did.

Below, I have listed some ideas for projects that I am interested in; consider them only as suggestions. I strongly encourage everyone to consider projects in the original work category, for they are more likely to be challenging and interesting. Some are more difficult then others; some might be expanded into Master's theses or projects if you get interested enough in them. For many of them, I can give references or starting points. Before starting your project, clear your choice of topic by talking to me, whether or not it is on the list.

1 Open Problems

The most enjoyable type of project is doing some semi-original research. Is there some idea related to algorithms or discrete mathematics which you would like to work on? The following are some problems I have at least briefly played around with and find interesting. For all of them I have some ideas or references to get you started so please come by to talk to me to find out what I know before starting.

I have starred what I think are the best projects on the list, but use your judgement. A good project is one where you can identify an interesting direction to pursue. Can you find a simplification or generalization of the problem which you do know how to solve? Can you write a program or construct examples to get some insight into the problem? Are there papers you can read to get a better understanding of what is known? If after working on it for a day or two you feel there is nothing you can do, don't be afraid to change topics. To give yourself this luxury, start before you submit your proposal.

Other sources of accessible problems include Graham, Knuth, and Patashnik's *Concrete Math*, Richard Guy's *Unsolved Problems in Number Theory*, Croft, Falconer, and Guy's *Unsolved Problems in Geometry*, Klee and Wagon's *Old and New Unsolved Problems*, and Skiena's *Implementing Discrete Mathematics*. I have these and references/ideas for other problems, so feel free to come talk with me.

Many open problems are open because nobody has bothered to look at them, so don't be afraid to try them. Each time in the past which I have run a course like this, one project was eventually published, at least as a technical report and sometimes as a journal paper.

1. *Reconstructing Partitions* (*) – Given an (unknown) set partition of items, efficiently reconstruct it using questions/probes of the form "how many components are represented by this given subset". At least $n-1$ are necessary and $O(n \log\log n)$ sufficient, but can you prove a linear bound? See Grebinski and Kucherov, SODA 1999.

2. *Unscrambling address lines* – There are two interesting numerical functions in Broder et.al. in SODA 1999. Do some analysis of them, ie. find new upper and lower bounds or closed forms on them.

3. *Data structures for routing trucks* – Truck dispatch systems can be modeled by looking for the nearest directed segment in a data base of sequemnts. Can you give an efficient data structure for such queries?

4. *Frequency Assignment on Hexagonal Lattice* (*) – The following problem arises in cellular phone networks. Given a hexagonal grid of n cells, which of which is potentially partition into three 120 degree sectors. Each sector is colored a different collor red, white, or blue. Futher, each sector is free to rotate but not reverse. Can these disks always be spun so no sectors of identical colors touch? Can you give an algorithm to find an assignmetn if one exists?

5. *Second Hamiltonian Cycles* – Does every 4-regular graph with one Hamiltonian cycle also have a second Hamiltonian cycle? Prove special cases or give a counter-example.

6. *Computing the Edit Distance Between Two Graphs* (*) – Given graphs G and H, where $|G| = |H| = n$, find a permutation of the vertices of G which minimizes the Hamming distance between G and H; i.e. the smallest number of edges which must be inserted to or deleted from G to leave a graph isomorphic to H. (1) Can you solve the problem exactly for two trees? An unpublished reference by Edmonds and Matula is given in Garey and Johnson, which means it is likely either easy or wrong. (2) Can we approximate the solution for planar graphs? Perhaps such tools as graph matching and the planar separator theorem can help with this. It is hard for planar graphs.

7. *Sorting Permutations Using Editing Commands* (*) – Suppose we are given a file of arbitrary permutations of k-subsets on the alphabet $1, \ldots, n$, separated by dashes. Using a text editor, we can swap pairs of consecutive numbers whereever they occur in the permutation, e.g. 1,$s/5-3/3-5/ How many commands $f(n,k)$ are necessary and sufficient to ensure that each line is ordered at the end?

8. *Separating Red/Blue Points with an Axis-Oriented Cross* – Consider a set of n points, some red, some blue. Does there exist a cross (i.e. a pair of horizontal and vertical lines) such that all the points in each quadrant is monochromatic? Give an efficient algorithm to compute this cross.

   Carlos Serra of UPC presented an $O(n \log n)$ algorithm for red-blue wedge separability; our problem is a special case of this one. There is a matching $\Theta(n \log n)$ lower bound for this problem, however the lower bound breaks down for the axis-oriented case. Can this be computed in linear time, or is there a higher lower bound?

9. *File system renaming* (*) – We have developed an algorithm, based on bipartite matching, for shortening long filenames without creating collisions. How can we improve the quality of the filenames we are getting by better selection criteria, and how well does it work on large file systems? Can the algorithm be extended to give "unambiguously" lengthened names?

10. *Maintaining dynamic set partitions* (*) – Each test or feature in a classification system defines a set partition on a class of objects. Adding new features refines the classification, while deleting old features may result in merging previously distinguished classes. Give an efficient data structure for maintaining the partition of $(1, \ldots, n)$ after k partition insertions/deletions. The best I can do is $O(n \lg k)$ per update, but can you do better?

11. *Reconstructing strings from substrings with errors* (*) – The practicality of Interactive Sequencing by Hybridization (ISBH) depends on generalizing. reconstruction algorithms without error into algorithm which can handle error. How much error can this method tolerate?

12. *Clause partitioning for logic programming optimization* (*) – Dave Warren and I discussed an interesting problem of how to break up clauses into smaller clauses, which was combinatorial in nature. We both would like to see a student pursue this further.

13. *Constructing Decision Trees* - We have given an $O(\lg n)$ approximation algorithm for constructing minimum height decision trees for classification, when all items have equal probability. Show how to build good decision trees (in the expected sense) when the nodes can have different probabilities.

14. *Exponential Length Cycles in Graphs* (*) – Erdos and Gyarfas ask: Let H be a graph, every vertex of which has degree at least three. Is it true that H contains a cycle of length 2^n for some n? "We were fairly sure that this is false but could not find a counterexample". Can you?

15. *Tiling a matrix with dominos* (*) – Suppose we are given an $n \times n$ grid, with n horizontal dominos (row i has a domino $r_i \times 1$) and n vertical dominos (column i has a domino $1 \times c_i$). What is the complexity of seeing whether there is a packing that satisfies all constraints (from Frank Wagner via Sandor Fekete)

16. *Meshing independent observers* (*) – Consider the problem of identifying the correct answer key on n exams each of length l on an alphabet a. Each of the n students have varying skills, and so we must identify which students to trust most and adjust accordingly.

    Equivalently, think of this as a time series of l points, being observed by n non-uniformly imperfect observers. How accurately can we reconstruct the signal without knowing how good the observers are in advance? The correct solution depends upon what our model is of how the observers behave.

17. *Finding small graphs of large girth* – Can you find a new record-setting cage? Check out Norman Biggs, Constructions for Cubic Graphs with Large Girth A1 of Volume 5(1), Electronic Journal of Combinatorics, http://www.combinatorics.org for current records.

18. *Inducing formats of unstructured data* – Can you scan a file and partition it into chunks to ignore or process regularly – such as mail headers, text blocks, program code, time series data / tables, etc?

19. *Counting Monotone Functions* – How many boolean monotone functions – i.e. how many ways to order the 2^n subsets of n elements by dominance?

20. *Partitioning strategies for weighted coupon collector approximation* – Shiyong Lu and I identified upper and lower bounds on the weighted coupon collector problem using partitionig strategies. But what partiioning strategies yield the best bounds – can you prove it?

21. *Suffix trees with permuted character order?* – To reduce space, does it pay to consider suffix trees with arbitrary probes, instead of representing strings sequentially? This is based on the idea used in unification factoring, see Dawson, et.al.

22. *Pattern Matching with Repeated Phrases* – In chanting Hebrew prayers, it is common to stop in the middle, and repeat earlier words and phrases. Suppose we are given a source text of length n, and a chanted text of length m. What is the complexity of verifying that the chanted text is a complete and proper rendition of the source text?

23. *Scheduling jobs on tape* (*) – Suppose we are given a set of n jobs, each of which has a fetch time and a processing time associated with it. No job can processed until it has been completely fetched. What order should we process the jobs to minimumize total elapsed time? (See More, Muthukrishnan, and Shriver, PODS 99)

24. *Geometric Toy Construction* – Consider the following "sorting toy" for little kids. There are different 2d or 3d shapes, and there are also little openings cut from rigid material that match the shapes. The interesting thing is that the shapes chosen are such that one shape can be navigated through only one opening. A naive attempt to design such a toy would probably result in shapes that, when properly oriented or jiggled around, could be made to fit through several openings. This defeats the purpose of the toy. What algorithmic issues are there in designing such a toy? For example, given a set of shapes, find the largest subset which can be employed...

25. *Stable topological sorting* – Is there a fast "stable" topological sorting algorithm, that finds the linear extension which is closest to a given initial order?

26. *Optimal list merging* – Suppose you have two sorted lists of lengths m and n, respectively. The standard merge algorithm sorts these using $m + n - 1$ comparisons in the worst-case. However, if $m/n \geq 2$, then a better algorithm exists. It is also known that if $m/n \leq 3/2$ (give or take), then the standard algorithm is optimal. Even cases as small as $m = 12$, $n = 7$ haven't been solved. Can you do so?

27. *Bichromatic Cycles* – Given a directed graph G , can we decide in polynomial time whether the node-set of G can be coloured by red and blue so that every directed circuit contains at least one red and at least one blue node? Or is this known to be an NP-complete problem?

28. *Allowable Super-Hamiltonian Paths in Complete Graphs* – Given a complete n-vertex graph where each vertex has a count associated with it, does there exist a walk which visites each vertex exactly the required amount of times. Is this polynomial or not to decide?

29. *Laser Scanning Patterns* – From Kevin Hunter's dissertation came the following problem: Given a polygonal (even rectangular) area, design the optimal laser scanning pattern which cuts the region into pieces each of diameter at most d. Appropriate cost measures are (1) minimum travel distance, (2) minumum number of cuts, (3) minimum length of intersetion of cuts and region The solution is not obvious even in the case of a unit square region whiere $d = \sqrt{2} - \epsilon$ which I think is nick's in two corners for (3)

2 Survey Papers

Perhaps there is some topic applicable to your research? If not, consider those below. The paper must have some significant "meat" on it, i.e. display an understanding of some hard technical material. Depending upon the topic you may read several papers or only a few. But my interest is finding some depth - if you read but didn't have to think there is something wrong. Because previous years term papers have either been weak or plagarized, I would discourage working on the unstarred topics.

1. *Answer key for "The Algorithm Design Manual"* – With several classes using it as a textbook, an answer key is needed for my new book. I have some material for you to start from.

2. *A=B* (*) – A new book by Wilf called *A=B* presents an automated way to prove binomial coefficient identities. Read the book and explain how this works.

3. *Random and Extremal Graph Theory* (*) – These are two interest areas in graph theory, each of which has books written about it (by Bollobas). Pick either random or extremal graphs and tell me something interesting about them.

4. *Hardness of Approximation Algorithms* - The most exciting new work in algorithms concerns proving that many problems are NP-complete to approximate, let alone solve exactly. Try to understand what is going on - how do you prove a problem is hard to approximate?

3 Algorithm Evaluation and Implementation

The experimental analysis of algorithms is becoming a significantly more serious research area. Doing quality experimental work is challenging and time consuming, but can make an excellent project. It is important to test your implementations on a wide variety of classes of input and compare fairly. Good examples of experimental analysis include papers by Bentley and Johnson on TSPs, Moret and Shapiro on MST, and Goldberg on network flow.

The algorithm (or set of algorithms) must be substantial enough to compete with the other projects. Correctness and cleanness of implementation are the goals of the project. Testing for correctness is required, as well of benchmarking the performance. I may ask to see a demonstration of your program to try and break it.

1. *Chinese Remainder-Based Arithmetic* (*) – Design and implement a high precision arithmetic package based on the Chinese remainder theorem. What operations can be done efficiently, and how does the speed compare to other packages?

2. *Pruning TSP search by crossing edge tests* (*) – Bentley's April and May 1999 Dr. Dobb's Journal column will feature fast exact Euclidean TSP solutions. His codes do not feature pruning based on the fact that the optimal Euclidean TSP is not self-intersecting. Can such a ehuristic be used to speed up the search?

3. *An Internet Gambling Game* (*) – Consider the following internet gambling game. You can bet on either red or black, and you win if fewer dollars are bet on your color than the other one. Obviously this can be generalized to more colors. The house wins the difference between the bets on the two colors, which should be $O(\sqrt(n))$ expected. Analyzie this game for more colors, and build an WWW implementation which keeps track of what people do.

4. *Guessing people-random numbers* (*) – Write a program to predict the results of people trying to guess random bits.

5. *Implementation evaluation* – I have been collecting implementations of many fundamental algorithms, and would be very interested in careful studies of these implementations to determine which are best under which conditions. Part of this job is getting fluent with the different instance generators available to generate appropriate data.

6. *Algorithm implementation* (*) – There are also a number of combinatorial algorithms which I know that there is a need to have a good implementation of, in particular: (1) finding an optimal Chinese postman tour, (2) heuristics for feedback arc/vertex set, and (3) heuristics for set and vertex cover / packing. My main interest is getting these written in C++, although choice of programming language is certainly negotiable.

7. *Backtracking/search engine* – A general backtracking / heuristic search engine would make it easy to develop programs for solving a variety of NP-complete problems. The ultimate such engine should have a simple state-space based interface capable of iterating over permutations, subsets, and set partitions, and such features as search tree size estimation, visualization, and support of threads. Make a start towards this engine, and apply it to one or more problems such as coloring and vertex/set cover.

8. *More Efficient Combinatorica* (*) – One of the bottlenecks in my system Combinatorica is that adjacency matrices are used as the default representation instead of edge lists. Can you produce a sparse graph version? It wouldn't be as hard as it sounds, and would be a good way to learn Mathematica programming.

9. *Teacher/Student Guide to Combinatorica* - My system Combinatorica has been fairly widely used for education even though there is no teachers or student guide. Become familiar enough with the system to write such a guide, with examples and problems.

10. *3D Bin-packing* – a LI moving company seeks a program to optimize packing boxes into trucks, given length-width-height dimensions, box weights, restrictions on orientations (this side up) and whether a box must go on top (do not stack). They need fast turnaround, so heuristics will likely be more useful than exhaustive search. They expect that good solutions can save them one man-hour per day. See: Li and Cheng, "Heuristic Algorithms for On-Line Packing in Three Dimensions", Journal of Algorithms, 13, 1992.

11. *Course scheduling* (*) – the fundamental problems of course scheduling reduce to graph coloring, and are NP-complete, but build a system capable of handling a variety of constraints and producing decent schedules.