6.897 Advanced Data Structures (Spring'05)

Prof. Erik Demaine TA: Mihai Pătraşcu

Abstracts of Final Projects

Bounds on the Independence Required for Cuckoo Hashing
Jeff Cohen and Daniel Kane

In Cuckoo Hashing, we achieve expected $O(1)$ amortized time per operation for dynamic hashing by using two hash functions and maintaining the property that each key is hashed to the value indicated by either of the hash functions. This result is known as long as the hash functions are chosen independently and have $O(\lg n)$-independence. In this paper, we show that only one of the hash functions must exhibit $O(\lg n)$-independence, and the other only needs to be 2-independent. We also show that 5-independence or less for both hash functions is insufficient. Furthermore, if the hash functions are not chosen independently of one another, but satisfy a condition that we call *joint-independence*, $\Omega(\lg n)$-joint-independence is both necessary and sufficient to guarantee that Cuckoo Hashing will work.

Spectral Lemma and Applications to Geometric Range Searching
Igor A. Ganichev

In this paper we state and prove a so-called Spectral Lemma that lower-bounds the size of a circuit needed to compute Ax for an integer $n \times n$ matrix A and $x \in \Re^n$. Spectral Lemma stands out from similar lower bound techniques because it proves that even if the circuit contains a significant number (usually on the order of $n/4$) of all-powerful gates that can compute any bivariate function, its size must still be big. Allowing all-powerful gates can be motivated by the fact that in practice the group over which we do additions and subtractions can be embedded into a ring or a field and new operations might be possible, or by the fact that we can make useful lookup tables in some cases.

The Spectral Lemma has been successfully applied to static geometric range searching problems, which are equivalent to computing Ax, where A is an incidence matrix and x is a weight vector. The lower bound of Spectral Lemma is in terms of the eigenvalues of A ($\Omega((k-2m)\lg \lambda_k)$, where $m \le k/2$ is the number of all-powerful gates, and λ_k is the k^{th} largest eigenvalue of A). It says that if A has high spectrum, then it is hard to compute. Therefore, applications of Spectral Lemma are just proofs of existence of a problem instance, whose incidence matrix has high spectrum. In the last section, we state two results that use Spectral Lemma and sketch a proof of one of them.

Both results and the Spectral Lemma are due to Bernard Chazelle.

Algorithms and Hardness for Dynamic Shortest Paths Problems
Tim Abbott and Yoyo Zhou

In this paper we present some recent results in the field of algorithms for the dynamic all-pairs shortest paths problem.

We explain some recent results of Roditty and Zwick. They obtain a fully dynamic algorithm for directed unweighted graphs with an $\widetilde{O}(m\sqrt{n})$ runtime. They also obtain a fully dynamic $\widetilde{O}(mn/t)$ ε-approximate algorithm, also for unweighted directed graphs. Both algorithms are randomized.

We also explain a recent result due to Roditty and Zwick, demonstrating that given even an efficient incremental or decremental algorithm for the single-source shortest paths problem, one can construct an efficient algorithm for the weighted all-pairs shortest paths problem. Combined with an $\Omega(mn)$ lower bound for the static all-pairs shortest paths problem for path comparison algorithms due to Karger et al., this provides a lower bound on the runtime of any partially dynamic single-source shortest paths structure based on classic path comparison algorithms such as Dijkistra's algorithm.

Finally, we explain an original result inspired by the previous result, its analogue for the dynamic $s-t$ minimum cut problem. Our work leaves open the possibility of efficient algorithms for maintaining the all-pairs $s-t$ minimum cut, in analogy to the shortest paths case.

Dynamic Ham-Sandwich Cuts of Convex Polygons in the Plane
Jelani Nelson and Vincent Yeung

In general, a *ham-sandwich cut* of two subsets S_1 and S_2 of the plane $\mathbb{R}^2$ is a line that simultaneously bisects both sets according to some measure m. If S_1 and S_2 are discrete sets of points, the measure m is usually the number of points; if S_1 and S_2 are regions, measure m can be area, perimeter, or the number of vertices (if S_1 and S_2 are polygonal).

A related problem, introduced by Megiddo, is that of finding a two-line partition. A *two-line partition* of a subset S of the plane is a pair of lines that partition the plane into four regions ("quadrants") each containing a quarter of the total measure, $\frac{1}{4}m(S)$. The (static) problems of finding a ham-sandwich cut or two-line partition for given sets S_1 and S_2 are well studied, with linear-time solutions for most variations. The connection between this problem and ham-sandwich cuts is that each line in the partition is a ham-sandwich cut with respect to the 2-coloring induced by the other line in the partition.

We provide an efficient data structure for dynamically maintaining a ham-sandwich cut of two non-overlapping convex polygons in the plane. The data structure supports queries for the ham-sandwich cut in $O(\log^3 n)$ worst-case time and insertions and deletions of vertices of the polygons in $O(\log n)$ worst-case time. We also show how this data structure can be used to answer queries for a two-line partition, also in $O(\log^3 n)$ worst-case time. In particular, if we use the vertex-count measure, the intersection of these two lines gives a point of Tukey depth $n/4$, which serves as an approximate Tukey median.

This work was joint with Timothy Abbott, Erik D. Demaine, Martin L. Demaine, Daniel Kane, and Stefan Langerman.

Data Structures for Planar Graphs
Pramook Khungurn

We survey results on dynamic data structures for planar embeddings, planarity testing, minimum spanning forest, connectivity, and shortest paths in plane and planar graphs. The best known upper bounds for the problems discussed are as follows:

- planar embedding: $O(\log n)$ amortized time per operation.
- incremental planarity testing: $O(\alpha(m, n))$ amortized time per operation.
- fully dynamic planary testing: $O(n^{1/2})$ amortized time per operation.
- fully dynamic connectivity: $O(\log^2 n)$ worst-case time per operation.
- fully dynamic 2-vertex-connectivity: $O(\log^2 n)$ worst-cast time per operation.
- decremental 2-edge-connectivity: $O(\log n)$ amortized time per operation.
- decremental biconnectivity and 3-edge-connectivity: $O(\log n)$ amortized time per operation with high probability.
- fully dynamic 3- and 4-edge-connectivity, and 3-vertex-connectiviy: $O(n^{1/2})$ amortized time per operation.
- shortest paths with nonnegative weights: $O(n^{2/3} \log^{7/3} n)$ amortized time per operation.
- shortest paths with nonnegative weights: $O(n^{4/5} \log^{13/5} n)$ amortized time per operation.

We also provide an in-depth coverage on a data structure for maintaining minimum spanning forest in plane graphs subjected operations that do not alter the embedding of the represented graph. The data structure achieves $O(\log n)$ amortized time per operation using $O(n)$ space.

Splay Trees and the Traversal Conjecture
Brian Jacokes

Splay trees were introduced as a way to maintain implicit balance in a binary search tree by rotating a queried element to the root during each access. Not only do splay trees achieve $O(\lg n)$ amortized time per operation for sufficiently many queries, but they also have several other remarkable properties such as optimality in a wide range of situations. Despite there being many results proved about splay trees, many conjectures still remain unsolved.

One of these, the traversal conjecture, is stated as follows: Let T_1 and T_2 be any two n-node binary search trees containing exactly the same items. Suppose we access the items in T_1 one after another using splaying, accessing them in the order they appear in T_2 in preorder. Then the total access time is $O(n)$.

Although special cases of this conjecture have been solved, no substantial progress toward a proof has been made. We wish to investigate this problem and determine possible methods for proving or refuting the conjecture. We are able to compute bounds on the worst-case number of rotations for small n, and attempt to extrapolate patterns which may yield insight into further directions for research.

Structures for Efficient File System-Scale Partial Persistence
Dan R. K. Ports and Austin T. Clements

A *persistent file system* stores every previous state of each file, allowing convenient access to the full state of the file system as it appeared at any point in the past. Achieving this convenient feature presents a challenging data structural problem because the amount of data involved is so large: it must use as little space as possible, and provide efficient operations for modifying the current state and accessing both current and past states. We formalize persistent file systems as a problem in data structures, and analyze it in the context of the external memory model. We begin by considering the design of our initial solution to this problem from the PersiFS_1 file system, which is based on a log of metadata changes and an indirection layer for storing file data. These "systems" data structures support the desired operations, but are not asymptotically efficient. Applying more advanced data structures, we refine the design into the next version, PersiFS_2. We use B^+-trees for file content indexing in order to improve the space efficiency of the system, and we present a novel partially-persistent B^+-tree design, which can be used to track changes to files with logarithmic modification and query cost. PersiFS_2 has been implemented as a working file system with these data structures, and our measurements indicate that the new file system data structure provides dramatically improved access time for previous revisions with a small increase in cost for modifications.

Deterministic and Randomized Bounds for H-freeness Testing
Mike Lieberman

Joint work with Anders Kaseorg.

This paper surveys the topic of testing the existence of a fixed subgraph in larger graphs. We focus on results relevant to testing for the existance of triangles. We consider deterministic, dynamic, and randomized algorithms.

Deterministically, there is a simple reduction to matrix multiplication. It is unknown if any deterministic algorithm can get closer to the theoretical $\Omega(n^2)$ lower bound. Dynamically, nothing nontrivial is known.

Much progress has been made with randomzied approximation algorithms. If the graph is "far away" from being free of the pertinent subgraph, then the algorithm should find one of the subgraphs with high probability. Interestingly, this problem is easier for dense graphs than non-dense graphs, and easier when searching for bipartite subgraphs than when searching for non-bipartite subgraphs.