December 2012

MADALGO seminar by Casper Kejlberg-Rasmussen, Aarhus University

I/O-Efficient Planar Range Skyline and Attrition Priority Queues

Abstract:

In the planar range skyline reporting problem, the goal is to store a set *P* of *n* 2D points in a structure such that, given a rectangle Q = [a_1, a_2] \times [b_1, b_2], the maxima (a.k.a. skyline) of *P* \cap *Q* can be reported efficiently. *Q* is 3-sided if one of its edges is grounded, giving rise to two variants: top-open (b_2 = \infty) and left-open (a_1 = -\infty) queries.

This paper presents comprehensive results in external memory under the O(n/B) space budget (*B* is the block size), covering both the static and dynamic settings:

- For static *P*, we give structures that support a top-open query in O(\log_B n + k/B), O(\log\log_B U + k/B), and O(1 + k/B) I/Os when the universe is R^2, a *U* x *U* grid, and the rank space [O(n)]^2, respectively (where k is the number of points reported). The query complexity is optimal in all cases.
- We show that the left-open case is harder, such that any linear-size structure must incur \Omega((n/B)^\epsilon + k/B) I/Os to answer a query. In fact, this case turns out to be just as difficult as the general 4-sided queries, for which we provide a static structure with the optimal query cost O((n/B)^\epsilon + k/B). Interestingly, these lower and upper bounds coincide with those of orthogonal range reporting in R^2, i.e., the skyline requirement does not alter the problem difficulty at all.
- For dynamic *P*, we present a fully dynamic structure that supports a top-open query in O(\log_{2B^\epsilon} (n/B) + k/B^{1-\epsilon}) I/Os, and an insertion/deletion in O(\log_{2B^\epsilon}(n/B)) I/Os, where \epsilon can be any parameter satisfying 0 \le \epsilon \le 1. This result also leads to a dynamic structure for 4-sided queries with the optimal O((n/B)^\epsilon + k/B) query time, and O(\log (n/B)) amortized update time.

As a contribution of independent interest, we propose an I/O-efficient version of the fundamental structure priority queue with attrition (PQA). Our PQA supports FindMin, DeleteMin, and InsertAndAttrite all in O(1) worst-case I/Os, and O(1/B) amortized I/Os per operation. Furthermore, it allows the additional CatenateAndAttrite operation that merges two PQAs in O(1) worst-case and O(1/B) amortized I/Os. The last operation is a non-trivial extension to the classic PQA of Sundar, even in internal memory. The new PQA is a crucial component of our dynamic structure for range skyline reporting.

Joint work with Yufei Tao, Konstantinos Tsakalidis, Kostas Tsichlas, and Jeonghun Yoon