I/O-Efficient Dynamic Planar Range Skyline Queries

Problem Definition

We study the problem of maintaining a planar point set \( P \subseteq \mathbb{R}^2 \) under the modifications of insertion and deletion. Given two points \( p, q \in \mathbb{R}^2 \) we say that \( p \) dominates \( q \) if \( x(p) \geq x(q) \land y(p) \geq y(q) \), i.e. all coordinates of \( p \) are larger than those of \( q \). The maximal points of \( S \subseteq P \) are all the points of \( S \) that are not dominated by any point in \( S \).

We want to support modifications of insertion and deletion. Given two points \( q \) and \( q' \), the modification of insertion deletes \( q \) and the modification of deletion adds \( q \) to the point set. The problem of maintaining a planar point set \( P \) under the modifications of insertion and deletion.

The I/O Model

In the I/O model we have a memory capable of holding \( M \) elements and a disk of infinite size. When we read or write to the disk we can read or write \( B \) elements at a time, and the cost is 1 I/O, the cost is the same if we read or write less than \( B \) elements at a time, hence we want to batch reads and writes together.

Previous / Our Results

Our result is the first dynamic orthogonal skyline reporting data structure in the I/O model, all previous results are either not fully dynamic or are in a different model.

| Reference | Model       | Insert / Delete | Query        |
|-----------|-------------|-----------------|--------------|
| [BT 2011] | Pointer Machine | \( O(\log n) \) | \( O(\log n + t) \) |
| [BT 2011] | RAM         | \( O(\log n / \log \log n) \) | \( O(\log n / \log \log n + t) \) |
| [KTT 2012] | I/O         | \( O(\log^2 w n) \) | \( O(\log_{\log w} n + t/16) \) |

I/O-CPQA

The I/O efficient Catanable Priority Queue with Attrition (I/O-CPQA) supports the following operations in \( O(1) \) I/O's worst-case and \( O(1/B) \) I/O's amortized:

- InsertAndAttrite(\( e, Q \)) – appends \( e \) to the end of \( Q \) and deletes all attrited elements in \( Q \) that are smaller than \( e \), i.e. returns \( Q' = \{ e \in Q \mid e < e' \} \cup \{ e' \} \).
- DeleteMin(\( Q \)) – deletes the minimal element \( e = \min(Q) \) from \( Q \) and returns \( e \) and \( Q \).
- ConcatenateAndAttrite(\( Q_1, Q_2 \)) – Deletes all attrited elements in \( Q_1 \) that are smaller than \( e = \min(Q_1) \) and prepends the non-attrited elements onto \( Q_2 \), i.e. returns \( Q = \{ e' \in Q_1 \mid e' < e \} \cup Q_2 \).

A record consists of a buffer of \([b;4b] \) elements and a pointer to another I/O.

When we delete the minimum element from the I/O-CPQA we delete from \( C \), if this violates (*) then we take records out of \( B \) and put them into \( C \), else if \( k_C > 1 \) we merge \( D_{C,1} \) and \( D_{C,2} \) else we put the first record \( v \) of \( D_v \) into \( C \) and we put the I/O-CPQA \( Q' \) that \( v \) points to into \( Q \).

Future Work

It is still an open problem whatever it is possible to obtain bounds of \( O(\log n) \) for updates and \( O(\log n + \log B) \) for queries, or it is possible to show a lower bound. The cost in our bounds comes from the need to load \( B \) I/O-CPQAs and concatenate them in \( O(1) \) I/Os, which are only possible to do if the buffer size of each record is \( B^{1/2} \).

References

[KTT 2012] Kejlberg-Rasmussen, Tsakalidis, Tsigkas – I/O-Efficient Dynamic Planar Range Skyline Queries. Submitted to SODA 2013.
[BT 2011] Brodal, Tsakalidis – Dynamic Planar Range Maxima Queries. ALP 2011.