I/O-Efficient Dynamic Planar Range Skyline Queries

Problem Definition
We study the problem of maintaining a planar point set P ⊆ ℝ² under the modifications of insertion and deletion. Given two points p,q ∈ ℝ² we say that p dominates q iff x(p) ≥ x(q) ∧ y(p) ≥ y(q), i.e. all coordinates of p are larger than those of q. The maximal points of S ⊆ P are all the points of S that are not dominated by any point in S.

We want to support 3-sided orthogonal skyline reporting queries for the point set P, i.e. reporting all maximal points in S ∩ P where S=[x₁;x₂] X [y₁;∞].

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.

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(Q,e) – appends e to the end of Q and deletes all attrited elements in Q that are smaller than e, i.e. returns Q’={e’ ∈ Q | e’ < e} U {e}.
- DeleteMin(Q) – deletes the minimum element e = min(Q) from Q and returns e and Q\{e\}.
- ConcatenateAndAttrite(Q₁,Q₂) – Delets all attrited elements in Q₁ that are smaller than e= min(Q₂) and prepends the non-attrited elements onto Q₂, i.e. returns Q’={e’ ∈ Q₁ | e’ < e} U Q₂.

A record consists of a buffer of [b;4b] elements and a pointer to another I/O-CPQA Q, where max(b) < min(q). An I/O-CPQA Q consists of 2+k_i BO functional dequeues C,B,D₁, …, D_q of records. All records in the dequeue fulfills max(p) < min(q) where record p precedes record q in the same dequeue. Also max(C) < min(B) < min(D₁) and min(D_q) is the smallest element in all of the dirty dequeues D. The essential size invariant to guarantee the bounds is:

\[
|C(Q)| + \sum_{i=1}^{k_q} |D_i(Q)| + k_q - 1
\]

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_q > 1 we merge D_q−1 and D_q else we put the first record v of D_q into C and we put the I/O-CPQA Q’ that v points into Q, see the figure below.

Future Work
It is still an open problem whatever it is possible to obtain bounds of O(log² n) for updates and O(log² n + 1/B) for queries, or it is possible to show a lower bound. The ε in our bounds comes because we 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⁺⁺.

References