5-Sided Orthogonal Point Enclosure

Notes on Rahul 2015, by Simon Pratt

The primary intention of these notes is to answer: how does [Rah15] answer 3D 5-sided point enclosure queries?

Orthogonal Point Enclosure Queries (OPEQ) Preprocess a set S of n axes-parallel rectangles in $\mathbb{R}^3$ in order to determine all rectangles containing a query point q. In particular, we wish to consider the case in which a rectangle is 5-sided. To be precise, our rectangles are of the form $[x_1,x_2]\times[y_1,y_2]\times(-\infty,z]$. In these notes, we will prove the following:

Theorem 1 (5.1 in [Rah15]). *OPEQ on 5-sided rectangles can be answered using a structure of $O(n\lg^* n)$ size and $O(\lg n \lg\lg n + k)$ query time, where k is the size of the output.*

We will use the following results:

Problem	Query	Space	Reference
2D 4-side OPEQ	$O(\lg n+k)$	$O(n)$	[Cha86]
3D 3-side OPEQ	$O(\lg n+k)$	$O(n)$	[Afs08]

1 5-Sided: Simple and Slow in Linear Space

An *interval tree* is a binary tree whose leaves store the values of the endpoints of a set of intervals [Ede83]. At each node v, store split(v) which is the largest value in the left subtree of v, and range(v) which is $(-\infty,\infty)$ at the root, and otherwise if range(v) = $[x_\ell, x_r]$, then v's left child will have range $[x_\ell,\text{split}(v)]$, and symmetrically for the right child. An interval is stored at the node v of minimal height such that the interval is contained within range(v). If we additionally maintain lists that store the left/right endpoints of all intervals stored at v in non-decreasing/non-increasing order, then space remains $O(n)$, but we can answer a 1D OPEQ in $O(\lg n+k)$ time.

Given a set of 4-sided rectangles of the form $[x_1,x_2]\times(-\infty,y]\times(-\infty,z]$, we can build an interval tree of all rectangles' projection onto the x-axis. Observe that at node v, if the query point q is to the left of split(v) then for each rectangle r stored at v, r contains q iff $q\in[x_1,\infty)\times(\infty,y]\times(\infty,z]$, and similarly (but symmetrically) if q is to the right of split(v). This effectively reduces the problem to a 3-sided query, for which we can store [Afs08] at each node. Since we perform at most $O(\lg n)$ of these queries (one at each level of the interval tree), the total query time is $O(\lg^2 n+k)$ and space is still $O(n)$. Call this structure D_4. We can use the same technique to solve 5-sided queries in $O(\lg^3 n+k)$ time by storing D_4 structures at the nodes of the interval tree obtained by y-projection instead. This proves the following:

Theorem 2 (2.1 in [Rah15]). *OPEQ on 5-sided rectangles can be answered using a structure of $O(n)$ size and $O(\lg^3 n+k)$ query time, where k is the size of the output.*

2 4-Sided: Faster in Near-Linear Space

Given two points $p,q\in\mathbb{R}^d$, then $p=(p_1,\ldots,p_d)$ *dominates* $q=(q_1,\ldots,q_d)$ if $p_i>q_i$ for all $i\in\{1,\ldots,d\}$. Given a set of points P, R is a *t-level shallow cutting* of P if (i) $|R|=O(n/t)$, (ii) any point p that is dominated by at most t points of P dominates a point in R, and (iii) each point in R is dominated by $O(t)$ points in P.

Note that we can reduce dominance of a point set P in $\mathbb{R}^3$ to planar point location in orthogonal subdivision by projecting the orthants whose corners are at the points of P onto the plane, which we'll call the *orthant projection*. If we use the 4-sided to 3-sided reduction above, then consider each 3-sided rectangle as a point, then we can take $\lg^{(i)} n$-level shallow cuttings R_i for all $0\le i\le \lg^* n$. **Local Structure**: on each R_i, build the structure from [Afs08]. **Global Structure**: for each R_i, compute the orthant projection $\mathcal{A}_i$, then store all such projections in [Cha86].

To answer a query, first find all orthant projections which enclose the query point by querying the global structure, then for each such projection, query the local structure.

Theorem 3 (3.1 in [Rah15]). *OPEQ on 4-sided rectangles can be answered using a structure of $O(n\lg^* n)$ size and $O(\lg n\cdot\lg^* n+k)$ query time, where k is the size of the output.*

3 5-Sided: Putting it all Together

We will use a grid technique adapted from [ABR00]. Let $t = \log^4 n$. Project S onto the plane, then draw an orthogonal $(2\sqrt{n/t}) \times (2\sqrt{n/t})$ grid over the resulting rectangles such that each horizontal and vertical slab contains the projections of $\sqrt{nt}$ sides. We create a tree by creating a node to store all rectangles on which we don't recurse, and recursing on all rectangles completely contained within a slab with m, the number of rectangles in the current subproblem, replacing n wherever mentioned previously. Stop recursion when m reaches a constant. At each node of this tree: (i) *(slow)*: build the structure from Theorem 2, (ii) *(L_c)*: for each grid cell c, store at most $\log^3 n$ of the rectangles which completely cover c in decreasing order of z span, and (iii) *(side)*: build the structure from Theorem 3 on the (at most 4) sides cut from each rectangle by the grid.

To answer a query q, locate the grid cell c containing q and scan L_c, reporting rectangles until (a) we find a rectangle not containing q, or (b) we reach the end.

If (b), then $k \geq \lg^3 n$, thus querying the slow structure gives $O(k)$ query time. Otherwise, query the side structures then the recursive structures. This uses $O(n \lg^* n)$ space, querying this structure takes $O(\lg n \lg\lg n + k)$ time, and this concludes the proof of Theorem 1.

References

[ABR00] S. Alstrup, G. Brodal, and T. Rauhe. New data structures for orthogonal range searching. *FOCS*, 2000.

[Afs08] P. Afshani. On dominance reporting in 3d. *ESA*, 2008.

[Cha86] B. Chazelle. Filtering search: A new approach to query-answering. *SIAM J. Comp.*, 1986.

[Ede83] H. Edelsbrunner. A new approach to rectangle intersections part I. *Inter. J. Comp. Math.*, 1983.

[Rah15] S. Rahul. Improved bounds for orthogonal point enclosure query and point location in orthogonal subdivisions in $\mathbb{R}^3$. *SODA*, 2015.