

# 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 order to determine all rectangles containing a query point  $q$ . In particular, we wish to examine the case in which rectangles are 5-sided and in  $\mathbb{R}^3$ . 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.*

At a high level, we will use interval trees to solve 5-sided queries in  $O(\lg^3 n + k)$  time, which is good for  $k \geq \lg^3 n$ . Otherwise, we'll use grids to break the structure into 4-sided queries which we solve with  $O(n \lg^* n)$  space and  $O(\lg n \lg^* n + k)$  query time using interval trees, shallow cuttings, and the following results:

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

## 1 5-Sided: Slow/Simple with 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  $\text{split}(v)$  which is the largest value in the left subtree of  $v$ , and  $\text{range}(v)$  which is  $(-\infty, \infty)$  at the root, and otherwise if  $\text{range}(v) = [x_\ell, x_r]$ , 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  $\text{range}(v)$ . 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

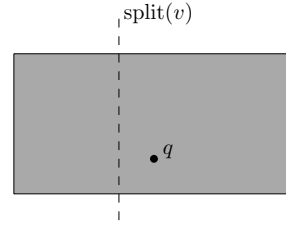


Figure 1: A 4-sided rectangle stored at  $v$  is reduced to 3-sided rectangles on either side of  $\text{split}(v)$  in an interval tree.

left of  $\text{split}(v)$  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  $\text{split}(v)$ . This effectively reduces the problem to a 3-sided query, for which we can store [Afs08] at each node (see Figure 1). 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 with Near-Linear Space

Given two points  $p, q \in \mathbb{R}^d$ , then  $p = (p_1, \dots, p_d)$  *dominates*  $q = (q_1, \dots, q_d)$  if  $p_i > q_i$  for all  $i \in \{1, \dots, d\}$ . 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  $P$ . First, project the rectangles onto the plane. Then build an interval tree on the rectangles as we did in Section 1, such that at each node  $v$ , our 4-sided projections are stored as 3-sided rectangles either left or right of  $\text{split}(v)$ .

At every node  $v$  of the interval tree, consider each 3-sided rectangle as a point, then we can take  $\lg^{(i)} n$ -level shallow cuttings  $R_i$  for all  $0 \leq i \leq \lg^* n$ .

**Local Structure** For each point  $p \in R_i$ , we build [Afs08] on all points that dominate  $p$ .

**Lemma 1** (3.1 in [Rah15]). *Given a point set  $P$  in  $\mathbb{R}^3$  and a strip  $\mathcal{R}$  of the plane such that the projection of all points onto the plane falls within  $\mathcal{R}$ , we can compute a subdivision  $\mathcal{A}$  of  $\mathcal{R}$  with  $|R|$  regions such that given a query point  $p$  in the plane, we can either find a point of  $P$  which dominates  $p$ , or no such point exists.*

**Global Structure** For each  $R_i$ , compute the Lemma 1 arrangement  $\mathcal{A}_i$ , then store all such projections in [Cha86].

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

**Space** The local structure for each shallow cutting  $R_i$  takes  $O(n)$  space, and there are  $\lg^* n$  such cuttings.

**Query Time** Querying the local structure takes  $O(\lg^* n)$  time and is performed at each of the  $O(\lg n)$  levels of the interval tree.

**Remark** You can instead build a data structure with  $O(n)$  space and  $O(\lg^{(c+1)} n + k)$  query time by only taking  $c \geq 2$  shallow cuttings.

**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 Together with Grids

We will use a grid technique adapted from [ABR00]. Let  $t = \lg^4 m$ , where  $m$  is the number of rectangles at the current level of recursion (initially  $n$ ). Project  $S$  onto the plane, then draw an orthogonal  $(2\sqrt{m/t}) \times (2\sqrt{m/t})$  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 which cross a grid line, then recurse on all other rectangles. 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  $\lg^3 n$  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 (see Figure 2).

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.

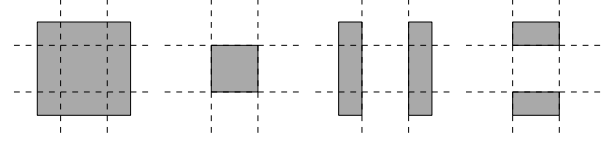


Figure 2: A rectangle (left) decomposes into an number of totally covered cells (middle-left), sides contained in adjacent vertical slabs (middle-right), and sides contained in adjacent horizontal slabs (right).

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.

**Space** The bottleneck is the structure from Theorem 3 which occupies  $O(n \lg^* n)$  space.

**Query Time** The bottleneck is the recursive case. Since we can find the grid cell containing  $q$  in  $O(\lg n)$  time and the size of the subproblem is  $\sqrt{nt}$ , we get  $Q(n) = Q(\sqrt{nt}) + O(\lg n)$ . With  $t = \lg^4 n$ , this solves to  $O(\lg n \lg^3 n + k)$  time, and this concludes the proof of Theorem 1.

**Remark** In the RAM model, we can find the grid cell containing  $q$  in  $O(1)$  time. This suggests we can achieve  $O(\lg \lg n + k)$  query time. Also, if Theorem 3 is not the time bottleneck, we can use the time-space trade-off to achieve  $O(n)$  space.

### 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.