Geometric Searching over the Rationals*

Bernard Chazelle

Department of Computer Science,
Princeton University, and NEC Research Institute
chazelle@cs.princeton.edu

Abstract. We revisit classical geometric search problems under the assumption of rational coordinates. Our main result is a tight bound for point separation, ie, to determine whether n given points lie on one side of a query line. We show that with polynomial storage the query time is $\Theta(\log b/\log\log b)$, where b is the bit length of the rationals used in specifying the line and the points. The lower bound holds in Yao's cell probe model with storage in $n^{O(1)}$ and word size in $b^{O(1)}$. By duality, this provides a tight lower bound on the complexity on the polygon point enclosure problem: given a polygon in the plane, is a query point in it?

1 Introduction

Preprocess n points in the plane, using $n^{O(1)}$ storage, so that one can quickly tell whether a query line passes entirely below or above the points. This *point separation* problem is dual to deciding whether a query point lies inside a convex polygon. As is well known, this can be done in $O(\log n)$ query time and $O(n)$ storage, which is optimal in the algebraic decision tree model [8,9]. This is suitable for infinite-precision computations [3,4,20], but it does not allow for bucketing or any form of hashing. Unfortunately, these happen to be essential devices in practice. In fact, the computational geometry literature is rife with examples of speed-ups derived from finite-precision encodings of point coordinates, eg, range searching on a grid [17], nearest neighbor searching [11,12], segment intersection [13], point location [16].

To prove lower bounds is usually difficult; even more so when hashing is allowed. Algebraic models are inadequate and one must turn to more general frameworks such as the cell probe model [18] or, in the case of range searching, the arithmetic model [7,19]. As a searching (rather than computing) problem, point separation lends itself naturally to the cell probe model and this is where we confine our discussion. Our main interest is in pinpointing what sort of query time can or cannot be achieved with polynomial storage. Note that some restriction on storage is essential since constant query time is trivially achieved with exponential space.

Let P be a set of n points in the plane, whose coordinates are rationals of the form p/q, where p and q are b-bit integers. A cell probe algorithm for point separation consists of a table of size n^c, with each cell holding up to b^d bits, for some arbitrarily large constants c, d. A query is answered by looking up a certain number of cells and outputting yes or

* This work was supported in part by NSF Grant CCR-96-23768, ARO Grant DAAH04-96-1-0181, NEC Research Institute, Ecole Polytechnique, and INRIA.

J. Nešetřil (Ed.): ESA'99, LNCS 1643, pp. 354–365, 1999.

no, depending on the information gathered. For lower bound purposes, the query time counts only the number of cells that are looked up during the computation.

Theorem 1. *Given any cell-probe algorithm for point separation, there exist an input of n points and a query line that require $\Omega(\log b/\log\log b)$ time. The lower bound is tight.*

The upper bound can be achieved on a standard unit-cost RAM. Take the convex hull of the points and, given the query line, search for the edges whose slopes are nearest that of the line. Following local examination of the relative heights of the line and edge endpoints, conclude whether there is point separation or not. This is elementary computational geometry and details can be skipped. The main point is that the problem reduces to predecessor searching with respect to slopes (rational numbers over $O(b)$ bits), which can be done optimally using a recent algorithm of Beame and Fich [2]. Their algorithm preprocesses n integers in $[0, N]$, so that the predecessor of any query integer can be found in $O(\log\log N/\log\log\log N)$ time, using $n^{O(1)}$ storage. By appropriate scaling and truncation, their scheme can be used for predecessor searching over the rationals, with the query time becoming $O(\log b/\log\log b)$, for rationals with $O(b)$-bit numerators and denominators.

2 The Complexity of Point Separation

The input consists of a set P of n points in R^2, which is encoded in a table T of size n^c, where c is an arbitrarily large constant. To simplify the notation we can replace c by $\max\{c, d\}$, and require that each cell should hold at most $w = b^c$ bits. A cell probe algorithm is characterized by a table assignment procedure (ie, a function mapping any P to an assignment of the table T to actual values) together with an infinite sequence of functions f_1, f_2, etc. Given a query ℓ (ie, a certain line in R^2), we evaluate the index $f_1(\ell)$ and look up the table entry $T[f_1(\ell)]$. If $T[f_1(\ell)]$ encodes whether ℓ separates the point set or not, we answer the query and terminate. Otherwise, we evaluate $f_2(\ell, T[f_1(\ell)])$ and look up the entry $T[f_2(\ell, T[f_1(\ell)])]$, and we iterate in this fashion until a cell probe finally reveals the desired answer. Note that such a framework is so general it easily encompasses every known solution to the point separation problem.

We use Miltersen's reformulation [15] of a cell probe algorithm as a communication complexity game between two players [14]. Alice chooses a set $\mathcal{L}_1$ of candidate queries (ie, a set of lines in the plane), while Bob decides on a collection $\mathcal{P}_1$ of n-point sets. Note that each pair $(\ell, P) \in \mathcal{L}_1 \times \mathcal{P}_1$ specifies a problem instance. Alice and Bob's task is then to exhibit a problem instance $(\ell, P) \in \mathcal{L}_1 \times \mathcal{P}_1$ that requires $\Omega(\log b/\log\log b)$ probes in T to answer. They do that by simulating each probe by a round in a communication complexity game.

The n^c possible values of the index $f_1(\ell)$ partition $\mathcal{L}_1$ into equivalence classes. Alice chooses one of them and sends to Bob the corresponding value of $f_1(\ell)$. Of all the possible 2^w assignments of the entry $T[f_1(\ell)]$ Bob chooses one of them and narrows down his candidate set $\mathcal{P}_1$ to the set $\mathcal{P}_2$ of point sets leading to that chosen value of $T[f_1(\ell)]$. Bob sends back to Alice his choice of $T[f_1(\ell)]$. Knowing ℓ and $T[f_1(\ell)]$, Alice chooses a value for $f_2(\ell, T[f_1(\ell)])$ and communicates it to Bob, etc. Each round

k produces a new pair $(\mathcal{L}_{k+1}, \mathcal{P}_{k+1})$ with the property that, for all queries in $\mathcal{L}_{k+1}$ and all point sets in $\mathcal{P}_{k+1}$, Bob and Alice exchange the same information during the first k rounds, which are thus unable to distinguish among any of the problem instances in $\mathcal{L}_{k+1} \times \mathcal{P}_{k+1}$.

We say a query line (resp. point set) is *active* at the beginning of round k if it belongs to $\mathcal{L}_k$ (resp. $\mathcal{P}_k$). The set $\mathcal{L}_k \times \mathcal{P}_k$ is called *unresolved* if it contains at least two problem instances (ℓ, P) and (ℓ', P') with different yes/no outcomes: in such a case, Bob and Alice need to proceed with round k, and the cost of the protocol (ie, the minimum number of rounds necessary) is at least k. We show that for some suitable $n = n(b)$, given any cell probe table assignment procedure, there exist a starting set $\mathcal{L}_1$ of query lines and a starting collection $\mathcal{P}_1$ of n-point sets in the plane that allow Bob and Alice to produce a nested sequence of unresolved sets

$$\mathcal{L}_1 \times \mathcal{P}_1 \supseteq \cdots \supseteq \mathcal{L}_t \times \mathcal{P}_t,$$

where $t = \Theta(\log b / \log\log b)$.

The protocol between Bob and Alice builds on our earlier work on approximate searching over the Hamming cube [5], which itself borrows ideas from the work of Ajtai [1] on predecessor searching. A protocol for predecessor queries of a similar flavor was recently devised independently by Beame and Fich [2].

2.1 Points and Lines

Let p_i denote the point (i, i^2), and given $i < j$, let $a_{ij} = \frac{1}{2}(i + j, i^2 + j^2)$ and $b_{ij} = ((i + j)/2, ij)$. Any of Bob's n-point sets P is of the form

$$P = \left\{ p_{i_1}, X_{i_1 i_2}, p_{i_2}, X_{i_2 i_3}, \ldots, p_{i_{s-1}}, X_{i_{s-1} i_s}, p_{i_s} \right\},$$

for some $i_1 < \cdots < i_s$, where $n = 2s - 1$ and X denotes the symbol a or b (not necessarily the same one throughout the sequence). Thus, P can be specified by an *index set* $I = I(P) = \{i_1, \ldots, i_s\}$ consisting of s distinct b-bit integers and a bit vector $\sigma = \sigma(P)$ of length $s - 1$ specifying the X's. For technical reasons, we require that all the integers of the index set I be even.

The starting query set $\mathcal{L}_1$ consists of the lines of the form, $y = 2kx - k^2$, for all odd b-bit integers k. Note that this is the equation of the line through p_k tangent to the parabola $y = x^2$. The number of bits needed to encode any point coordinate or line coefficient is $2b$ (and not b, a minor technicality). Note that the problem does not become suddenly easier with other representations such as $\alpha x + \beta y = 1$, and that for the purposes of our lower bound, all such representations are essentially equivalent. The following is immediate.

Lemma 2. *Let p_{i_j} and $p_{i_{j+1}}$ be two points of P and let ℓ be the line $y = 2kx - k^2$, where $i_j < k < i_{j+1}$. The line ℓ separates the point set P if and only if the symbol X in $X_{i_j i_{j+1}}$ is of type b.*

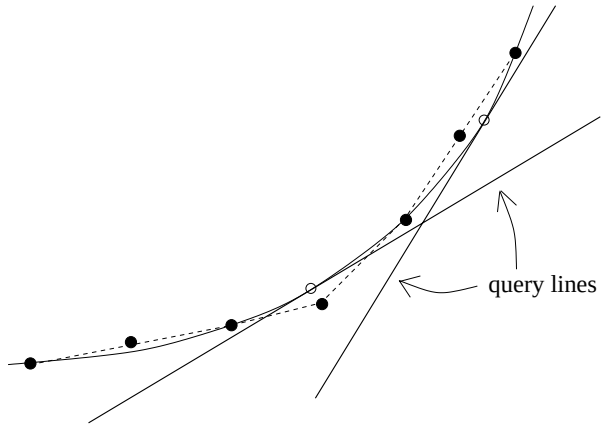

Fig. 1. A set P with $n = 7$ points and two queries with different answers.

2.2 A Hierarchy of Tree Contractions

Keeping control of Alice query lines is quite simple. The same cannot be said of Bob's point sets. Not only Bob's collections of point sets must be kept large but they must include point sets of all shape (but not size; remember that their size n is fixed). This variety is meant to make the algorithm's task more difficult. Some point sets must stretch widely with big gaps between consecutive points, while others must be confined to narrow intervals. For this reason, we cannot define point sets by picking points at random uniformly. Instead, we use a tree and a hierarchy of contractions of subtrees to define intervals from which we can specify the point sets.

Consider the perfect binary tree whose leaves (nodes of depth b) correspond to the integers 0 through $2^b - 1$, and let $\mathcal{T}_1$ denote its subtree of depth d^t sharing its root, where[1]

$$t = \left\lfloor \frac{\log b}{2 \log \log b} \right\rfloor \qquad \text{and} \qquad d = \lfloor c^2 \log b \rfloor \tag{1}$$

We assume throughout that the bit size b and the constant c are both suitably large. Note that b greatly exceeds d^t and so the tree $\mathcal{T}_1$ is well defined. Given a node v of the tree $\mathcal{T}_1$, let $\mathcal{T}_1(v)$ denote its subtree of depth d^{t-1} rooted at v. Contract *all* the edges of $\mathcal{T}_1$ except those whose (lower) incident node happens to be a leaf of $\mathcal{T}_1(v)$, for some node v of depth at most $d^t - d^{t-1}$ and divisible by d^{t-1}. This transforms the tree $\mathcal{T}_1$ into a smaller one, denoted $\mathcal{U}_1$, of depth d. Note that the depth-one subtree formed by an internal node v of $\mathcal{U}_1$ and its $2^{d^{t-1}}$ children forms a contraction of the tree $\mathcal{T}_1(v)$ (Fig.2).

Repeating this process leads to the construction of $\mathcal{U}_k$ for $1 < k \leq t$. Given an internal node v of $\mathcal{U}_{k-1}$, the depth-one tree formed by v and its children is associated with the subtree $\mathcal{T}_{k-1}(v)$, which now plays the role of $\mathcal{T}_1$ earlier, and is renamed $\mathcal{T}_k$. For any node $u \in \mathcal{T}_k$ of depth at most $d^{t-k+1} - d^{t-k}$ and divisible by d^{t-k}, let $\mathcal{T}_k(u)$ denote

[1] All logarithms are to the base two.

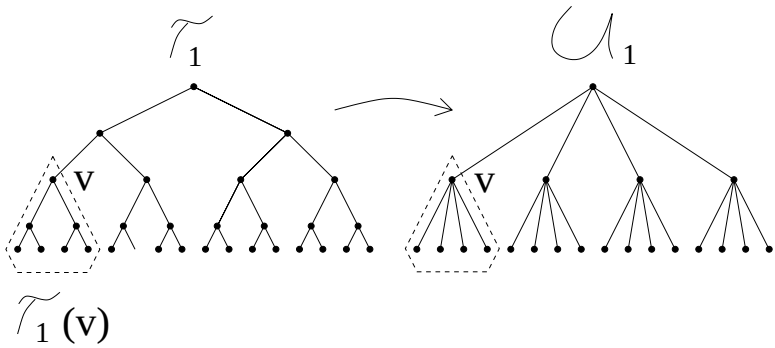

Fig. 2. The tree T_1 and its contraction into U_1.

the subtree of $\mathcal{T}_k$ of depth d^{t-k} rooted at u: as before, turn the leaves of $\mathcal{T}_k(u)$ into the children of u by contracting the relevant edges. This transforms $\mathcal{T}_k$ into the desired tree $\mathcal{U}_k$ of depth d.

The contraction process is the same for all $k < t$, but not for $k = t$. We simply make all the leaves of $\mathcal{T}_t$ into the children of the root and remove the other internal nodes, which produces a depth-one tree $\mathcal{T}_t$ with 2^d leaves. Although $\mathcal{T}_k$ is defined nondeterministically, it is always a perfectly balanced binary tree of depth d^{t-k+1}.

Lemma 3. *Any internal node of any $\mathcal{U}_k$ has exactly $2^{d^{t-k}}$ children if $k < t$, and 2^d children if $k = t$.*

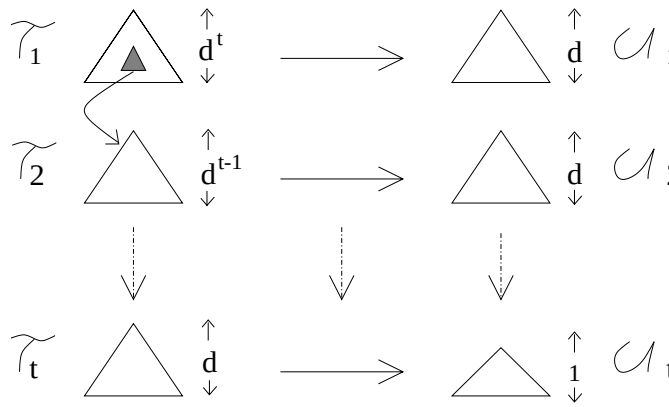

Fig. 3. The hierarchy of trees.

2.3 A Product Space Construction

We define any $\mathcal{P}_k$ by means of a distribution $\mathcal{D}_k$. We specify a lower bound on the probability that a random point set P_k drawn from $\mathcal{D}_k$ is active prior to round k, ie, belongs to $\mathcal{P}_k$.

• *Distribution* $\mathcal{D}_1$: A random P_1 is defined by picking a random index set I_1 (more on this below) and, independently, a random bit vector σ_1 uniformly distributed in $\{0,1\}^{s-1}$: I_1 is defined recursively in terms of $I_2, \ldots, I_t$. Each I_k is defined with respect to a certain tree $\mathcal{U}_k$. Any node v in any $\mathcal{U}_k$ is naturally associated with an interval of integers between 0 and $2^b - 1$ of size larger than any fixed constant (go back to the node v of $\mathcal{T}_1$ to which it corresponds to see why): call the smallest even integer in that interval the *mark point* of v. We define a random index set I_1 by setting $k = 1$ in the procedure below:

- For $k = t$, a random I_k (within some $\mathcal{T}_k$) is formed by the mark points of w^5 nodes selected at random, uniformly without replacement, among the leaves of the depth-one tree $\mathcal{U}_k$.
- For $k < t$, a random I_k (within some $\mathcal{T}_k$) is defined in two stages:
  [1] For each $j = 1, 2, \ldots, d-1$, choose w^5 nodes of $\mathcal{U}_k$ of depth j at random, uniformly without replacement, among the nodes of depth j that are not descendants of nodes chosen at lower depth ($< j$). The $(d-1)w^5$ nodes selected are said to be *picked by* I_k.
  [2] For each node v picked by I_k, recursively choose a random I_{k+1} within $\mathcal{T}_{k+1} = \mathcal{T}_k(v)$. The union of these $(d-1)w^5$ sets I_{k+1} forms a random I_k within $\mathcal{T}_k$.

Note that a random P_1 (drawn) from $\mathcal{D}_1$ is active with probability 1 since no information has been exchanged yet between Bob and Alice. We see by induction that a random I_k consists of $s = (d-1)^{t-k} w^{5(t-k+1)}$ integers. Setting $k = 1$, and using the fact that $n = 2s - 1$, we have the identity

$$n = 2(d-1)^{t-1} w^{5t} - 1. \tag{2}$$

• *Distribution* $\mathcal{D}_k$: We enforce the following

- point set invariant: For any $1 \le k \le t$, a random P_k from $\mathcal{D}_k$ is active with probability at least 2^{-w^2}.

By abuse of terminology, we say that $P_k \in \mathcal{D}_k$ if sampling from $\mathcal{D}_k$ produces P_k with nonzero probability. Once the probability of a point set is zero in some $\mathcal{D}_k$, it remains so in all subsequent distributions $\mathcal{D}_j$ ($j > k$), or put differently,

$$\mathcal{D}_1 \supseteq \cdots \supseteq \mathcal{D}_t.$$

Let $P_1 = (I_1, \sigma_1)$ be an input point set $\{p_{i_1}, X_{i_1 i_2}, \ldots, X_{i_{s-1} i_s}, p_{i_s}\}$ in $\mathcal{D}_1$. In the recursive construction of I_1, if v is a node of $\mathcal{U}_k$ picked by I_k in step [1], let $\{i_a, \ldots, i_b\}$ be the I_{k+1} defined recursively within $\mathcal{T}_{k+1} = \mathcal{T}_k(v)$. The set

$$P_{|v} \stackrel{\text{def}}{=} \left\{ p_{i_a}, X_{i_a i_{a+1}}, \ldots, X_{i_{b-1} i_b}, p_{i_b} \right\}$$

is called the v-*projection* of P_1. Similarly, one may also refer to the v-projection of any P_j ($j \leq k$), which might be empty. Obviously, it is possible to speak of a random $P_{|v}$ (with v fixed), independently of any P_1, as the point set formed by a random I_k and a uniformly distributed random bit vector σ_k of size $|I_k| - 1$. It is this distribution that will be understood in any further reference to a random $P_{|v}$.

Assume that we have already defined $\mathcal{D}_k$, for $k < t$. A distribution $\mathcal{D}_k$ is associated with a specific tree $\mathcal{T}_k$. To define $\mathcal{D}_{k+1}$, we must first choose a node v in $\mathcal{U}_k$ and make $\mathcal{T}_{k+1} = \mathcal{T}_k(v)$ our reference tree for $\mathcal{D}_{k+1}$. Any n-point set of $\mathcal{D}_k$ whose probability is not explicitly set below is assigned probability zero under $\mathcal{D}_{k+1}$. Consider each possible point set $P_{|v}$ in turn (for v fixed), and apply the following rule:

- If $P_{|v}$ is the v-projection of some P_k in $\mathcal{P}_k$, then take one[2] such P_k, and set its probability under $\mathcal{D}_{k+1}$ to be that of picking $P_{|v}$ randomly.
- Otherwise, take one $P_k \in \mathcal{D}_k$ whose v-projection is $P_{|v}$, and again set its probability under $\mathcal{D}_{k+1}$ to be that of picking $P_{|v}$ randomly.

During that round k, Bob reduces the collection of active point sets in $\mathcal{D}_{k+1}$ to form $\mathcal{P}_{k+1}$. To summarize, a random P_k is defined with reference to a specific tree $\mathcal{T}_k$. Note that the distribution $\mathcal{D}_k$ is isomorphic to that of a random $P_{|v}$, for fixed $v \in \mathcal{U}_{k-1}$, or equivalently, a random (I_k, σ_k), where σ_k is a uniformly distributed random bit vector of size $|I_k| - 1$.

2.4 Alice's Query Lines

As the game progresses, $\mathcal{L}_1$ decreases in size to produce the nested sequence $\mathcal{L}_1 \supseteq \cdots \supseteq \mathcal{L}_t$. Prior to round k, the currently active query set $\mathcal{L}_k$ is associated with the same reference tree $\mathcal{T}_k$ used to define a random P_k. As we observed in the last section, each node of $\mathcal{U}_k$ corresponds to a unique interval of integers in $[0, 2^b)$. By abuse of notation, we also let $\mathcal{L}_k$ designate the set of integers j defining the lines $y = 2jx - j^2$ in the set. We maintain the following:

- query invariant: For any $1 \leq k \leq t$, the fraction of the leaves in $\mathcal{U}_k$ whose intervals intersect $\mathcal{L}_k$ is at least $1/b$.

Lemma 4. *If $\mathcal{L}_t$ and $\mathcal{P}_t$ satisfy their respective invariant, then $\mathcal{L}_t \times \mathcal{P}_t$ is unresolved.*

Proof. Suppose that $\mathcal{L}_t$ satisfies the query invariant and that $\mathcal{L}_t \times \mathcal{P}_t$ is not unresolved: we show that $\mathcal{P}_t$ must then violate the point set invariant. For each leaf of $\mathcal{U}_t$ whose interval intersects $\mathcal{L}_t$, pick one $j_i \in \mathcal{L}_t$ in that interval. By Lemma 3 and the query invariant, this gives us a sequence $j_1 < \cdots < j_m$ of length

$$m \geq \frac{2^d}{b}\,. \tag{3}$$

Given $P_t \in \mathcal{D}_t$, we define the *spread* of P_t, denoted spread(P_t), as the number of intervals of the form $[j_i, j_{i+1}]$ $(0 \leq i \leq m)$ that intersect the index set $I(P_t)$ (Fig.4); for consistency we write $j_0 = 0$ and $j_{m+1} = 2^b - 1$. Suppose that the spread $|S|$ is defined

[2] It does not matter which one, but it has to be unique.

by some fixed set S of size less than w^4. Of the $m+1$ candidate intervals $[j_i, j_{i+1}]$, a random I_t must then avoid $m+1-|S|$ of them. Although such an interval may not always enclose a whole leaf interval, it does contain at least one mark point, and so the choice of I_t is confined to at most $2^d - m - 1 + |S|$ leaves of $\mathcal{U}_t$. Thus, the probability that the spread is defined by S is bounded by

$$\binom{2^d + |S| - m - 1}{w^5} \Big/ \binom{2^d}{w^5} \leq \left(1 - \frac{m - |S|}{2^d}\right)^{w^5}.$$

Summing over all S's of size less than w^4, it follows from (3) that

$$\text{Prob}\left[\text{spread}(P_t) < w^4\right] \leq \sum_{k<w^4} \binom{m+1}{k} \left(1 - \frac{1}{2b}\right)^{w^5} \leq 2^{-w^4}. \tag{4}$$

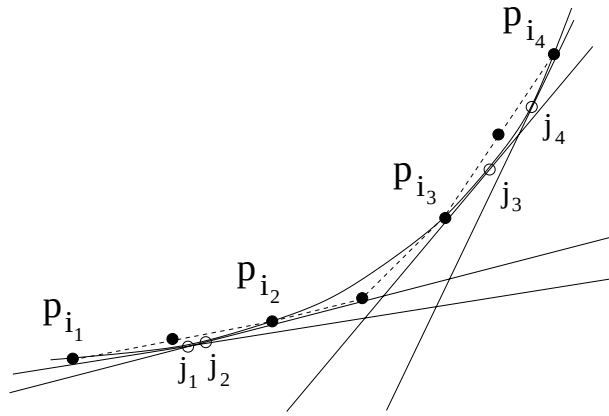

Fig. 4. A spread of 3 determined by $[j_0, j_1], [j_2, j_3], [j_4, j_5]$.

Suppose now that the spread is at least w^4. Then

$$P_t = \left\{ p_{i_1}, X_{i_1 i_2}, p_{i_2}, X_{i_2 i_3}, \ldots, p_{i_{s-1}}, X_{i_{s-1} i_s}, p_{i_s} \right\}$$

includes a subset P^* of at least $w^4 - 1$ points p_{i_j}, every one of which can be paired with a line $y = 2kx - k^2$ of $\mathcal{L}_t$, where $i_j < k < i_{j+1}$. Pick a random P_t from $\mathcal{D}_t$, and let Ξ denote the event: "all queries from $\mathcal{L}_t$ give the same answer yes/no with respect to point set P_t." By Lemma 2, the $X_{i_j, i_{j+1}}$'s are all of the form $a_{i_j, i_{j+1}}$ or all of the form $b_{i_j, i_{j+1}}$ (no mix). As we observed earlier, $\mathcal{D}_t$ is isomorphic to the distribution of a random (I_t, σ_t), where σ_t is a string of $w^5 - 1$ bits (drawn uniformly, independently). The constraint on the X's reduces the choice of a random P_t by a factor of at least 2^{w^4-2}, and hence,

$$\text{Prob}\left[\, \Xi \mid \text{spread}(P_t) \geq w^4 \right] \leq 2^{2-w^4}. \tag{5}$$

Putting together (4,5), we find

$$\begin{aligned}\mathrm{Prob}[\,\Xi\,] &= \mathrm{Prob}[\,\Xi \mid \text{spread}(P_t) < w^4\,] \cdot \mathrm{Prob}[\,\text{spread}(P_t) < w^4\,] \\ &\quad + \mathrm{Prob}[\,\Xi \mid \text{spread}(P_t) \geq w^4\,] \cdot \mathrm{Prob}[\,\text{spread}(P_t) \geq w^4\,] \\ &\leq 2^{-w^4} + 2^{2-w^4} < 2^{-w^2},\end{aligned}$$

which violates the point set invariant. □

During the k-th round, Alice chooses an index in Bob's table. As we discussed earlier, the set of n^c possible choices partitions her current query set $\mathcal{L}_k$ into as many equivalence classes. An internal node v of $\mathcal{U}_k$ is called *heavy* if one (or more) of these classes intersects the intervals associated with a fraction at least $1/b$ of the children of v. The following is a variant of a result of Ajtai [1].

Lemma 5. *The union of the intervals associated with the heavy nodes of $\mathcal{U}_k$ contains at least a fraction $1/2b$ of the leaves' intervals.*

Proof. Fix an equivalence class and color the nodes of $\mathcal{U}_k$ whose intervals intersect it. Mark every non-root colored node that is heavy with respect to the equivalence class. Then, mark every descendant in $\mathcal{U}_k$ of a marked node. Let N be the number of leaves in $\mathcal{U}_k$ and let N_j be the number of leaves of $\mathcal{U}_k$ whose depth-j ancestor in $\mathcal{U}_k$ is colored and unmarked (we include v as one of its ancestors). For $j > 1$, an unmarked, colored, depth-j node is the child of an unmarked, colored, depth-$(j-1)$ node that is not heavy for the chosen class, and so $N_j < N_{j-1}/b$. We have $N_1 \leq N$ and, for any $j > 0$,

$$N_j \leq \frac{N}{b^{j-1}}\,.$$

Repeating this argument for all the other equivalence classes, we find that all the unmarked, colored nodes (at a fixed depth $j > 0$) are ancestors of at most $n^c N/b^{j-1}$ leaves. This implies that the number of unmarked, colored leaves is at most $n^c N/b^{d-1} < N/2b$. (This follows from (1, 2).) The query invariant guarantees that at least N/b leaves of $\mathcal{U}_k$ are colored and so at least $N/2b$ are both colored and marked. It follows that the marked nodes whose parents are unmarked are themselves are ancestors of at least $N/2b$ leaves: all these nodes are heavy. □

Alice's strategy is to keep her active query sets as "entangled" as possible with Bob's point sets. Put differently, ideally the two should form a low-discrepancy set system [6] (at least in the one-way sense). The next result says that this is true on at least one level of $\mathcal{U}_k$, where many heavy nodes end up being picked by a random I_k.

Lemma 6. *For any $0 < k < t$, there is a depth j $(0 < j < d)$ such that, with probability at least 2^{-w^2-1}, a random P_k from $\mathcal{D}_k$ is active and its index set I_k picks at least w^3 heavy depth-j nodes in its associated $\mathcal{U}_k$.*

Proof. Recall that $\mathcal{D}_k$ is isomorphic to a random (I_k, σ_k). Fix σ_k once and for all. The heavy nodes of $\mathcal{U}_k$ are ancestors of at least a fraction $1/2b$ of the leaves (Lemma 5). It follows that, for some $0 < j < d$, at least a fraction $1/2bd$ of the nodes of depth j are heavy. Among these, I_k may pick only those that are not picked further up in the tree: this caveat rules out fewer than dw^5 candidate nodes, which by Lemma 3, represents a fraction at most $dw^5/2^d$ of all the nodes of depth j. So, it appears that among the set of depth-j nodes that may be picked by I_k, the fraction α of heavy ones satisfies

$$\alpha \geq \left(\frac{1}{2db} - dw^5 2^{-d}\right) \Big/ \left(1 - dw^5 2^{-d}\right) > \frac{1}{3db}\,.$$

The index set I_k picks w^5 depth-j nodes of $\mathcal{U}_k$ at random with no replacement. By Hoeffding's classical bounds [10], the probability that the number of heavy ones picked exceeds the lemma's target of w^3 is at least

$$1 - e^{-2w^5(\alpha - 1/w^2)^2} > 1 - 2^{-w^3}.$$

It follows from the point set invariant and the independence of I_k and σ_k that, with probability at least $2^{-w^2} - 2^{-w^3}$, a random P_k is active and its index set I_k picks at least w^3 heavy depth-j nodes in its associated $\mathcal{U}_k$. □

2.5 Probability Amplification

During the k-th round, Bob sends to Alice the contents of the cell $T[f_k(\ell, T[f_1(\ell)], \ldots)]$. The 2^w possible values partition the current collection $\mathcal{P}_k$ of active point sets into as many equivalence classes. We exploit the product nature of the distribution $\mathcal{D}_k$ to amplify the probability of being active by projecting the distribution on one of its factors.

Lemma 7. *For any $0 < k < t$, there exists a heavy node v of $\mathcal{U}_k$ such that, with probability at least $1/2$, a random P_{k+1} drawn from the distribution $\mathcal{D}_{k+1}$ associated with $\mathcal{T}_{k+1} = \mathcal{T}_k(v)$ belongs to $\mathcal{P}_k$.*

Proof. We refer to the depth j in Lemma 6. Let $p_{|S}$ denote the conditional probability that a random P_k from $\mathcal{D}_k$ belongs to $\mathcal{P}_k$, given that S is exactly the set of heavy nodes of depth j picked by I_k. Summing over all subsets S of heavy depth-j nodes of size at least b^3,

$$\sum_S \text{Prob}\,[\,S = \text{ set of heavy depth-}j\text{ nodes picked by } I_k\,] \cdot p_{|S}$$

is the sum, over all S, of the probability that $P_k \in \mathcal{P}_k$ and that S is precisely the set of heavy nodes of depth j picked by its index set I_k. By Lemma 6, this sum is at least 2^{-w^2-1}, and therefore $p_{|S} \geq 2^{-w^2-1}$, for some set S^* of at least w^3 heavy nodes of depth j.

Because a random P_k whose I_k picks v consists of a random (I_{k+1}, σ_{k+1}) drawn at node v independently of the rest of (I_k, σ_k), its v-projection has a distribution isomorphic to that of (I_{k+1}, σ_{k+1}), which is also $\mathcal{D}_{k+1}$. The same is true even if the distribution on

P_k is conditioned upon having S as the set of heavy depth-j nodes picked by I_k. If P_k belongs to $\mathcal{P}_k$ then its v-projection maps to a unique set $P_{k+1} \in \mathcal{D}_{k+1}$ also in $\mathcal{P}_k$.

Let $p_{|v}$ denote the probability that a random P_{k+1} drawn from the distribution $\mathcal{D}_{k+1}$ associated with $\mathcal{T}_{k+1} = \mathcal{T}_k(v)$ belongs to $\mathcal{P}_k$. It follows that

$$p_{|S} \;\leq \prod_{v \in S} p_{|v}.$$

Since $|S^*| \geq w^3$, it follows that

$$p_{|v} \geq \left(2^{-w^2-1}\right)^{1/|S\ |} \geq \frac{1}{2},$$

for some $v \in S^*$. □

Both query and point set invariants are trivially satisfied before round 1. Assume now that they hold at the opening of round $k < t$. Let v denote the node of $\mathcal{U}_k$ in Lemma 7. The n^c possible ways of indexing into the table T partition Alice's query set $\mathcal{L}_k$ into as many equivalence classes. Because v is heavy, the intervals associated with a fraction at least $1/b$ of its children intersect a particular equivalence class. Alice chooses such a class and the query lines in it as her new query set $\mathcal{L}_{k+1}$. The tree $\mathcal{U}_{k+1}$ is naturally derived from $\mathcal{T}_{k+1} = \mathcal{T}_k(v)$, and the query invariant is satisfied at the beginning of round $k+1$.

Upon receiving the index from Alice, Bob must choose the contents of the table entry while staying consistent with past choices. By Lemma 7, a random P_{k+1} from $\mathcal{D}_{k+1}$ (distribution associated with $\mathcal{T}_{k+1}$) is active at the beginning of round k with probability at least a half. There are 2^w choices for the table entry, and so for at least one of them, with probability at least $(1/2)2^{-w} > 2^{-w^2}$, a random point set from $\mathcal{D}_{k+1}$ is active at the beginning of round k and produces a table with that specific entry value. These point sets constitute the newly active collection $\mathcal{P}_{k+1}$, and the point set invariant still holds at the beginning of round $k+1$.

To show that t rounds are needed, we must prove that $\mathcal{L}_k \times \mathcal{P}_k$ is unresolved, for any $k \leq t$. In fact, because of the nesting structure of these products, it suffices to show that $\mathcal{L}_t \times \mathcal{P}_t$ is unresolved, which follows from Lemma 4. This proves the lower bound of Theorem 1. □

Acknowledgments

I wish to thank the anonymous reviewer for useful suggestions. I also thank Faith Fich for informing me of her recent work with Paul Beame, and in particular, their upper bound for predecessor searching in a bounded universe.

References

1. Ajtai, M. *A lower bound for finding predecessors in Yao's cell probe model*, Combinatorica, 8 (1988), 235–247.
2. Beame, P., Fich, F. *Optimal bounds for the predecessor problem*, Proc. 31st Annu. ACM Symp. Theory Comput. (1999), to appear.
3. Ben-Or, M. *Lower bounds for algebraic computation trees*, Proc. 15th Annu. ACM Symp. Theory Comput. (1983), 80–86.
4. Björner, A., Lovász, L., Yao, A.C. *Linear decision trees: Volume estimates and topological bounds*, Proc. 24th Annu. ACM Symp. Theory Comput. (1992), 170–177.
5. Chakrabarti, A., Chazelle, B., Gum, B., Lvov, A. *A good neighbor is hard to find*, Proc. 31st Annu. ACM Symp. Theory Comput. (1999), to appear.
6. Chazelle, B. *The Discrepancy Method: Randomness and Complexity*, Cambridge University Press, to appear.
7. Fredman, M.L. *A lower bound on the complexity of orthogonal range queries*, J. ACM, 28 (1981), 696–705.
8. Grigoriev, D., Karpinksi, M., Meyer auf der Heide, F., Smolensky, R. *A lower bound for randomized algebraic decision trees*, Computational Complexity, 6 (1997), 357–375.
9. Grigoriev, D., Karpinksi, M., Vorobjov, N. *Improved lower bound on testing membership to a polyhedron by algebraic decision trees*, Proc. 36th Annu. IEEE Symp. Foundat. Comput. Sci. (1995), 258–265.
10. Hoeffding, W. *Probability inequalities for sums of bounded random variables*, J. Amer. Stat. Assoc., 58 (1963), 13–30.
11. Karlsson, R.G. *Algorithms in a restricted universe*, Tech Report CS-84-50, Univ. Waterloo, Waterloo, ON, 1984.
12. Karlsson, R.G., Munro, J.I. *Proximity on a grid*, Proc. 2nd Symp. Theoret. Aspects of Comput. Sci., LNCS Springer, vol.182 (1985), 187–196.
13. Karlsson, R.G., Overmars, M.H. *Scanline algorithms on a grid*, BIT, 28 (1988), 227–241.
14. Kushilevitz, E., Nisan, N. *Communication Complexity*, Cambridge University Press, 1997
15. Miltersen, P.B. *Lower bounds for union-split-find related problems on random access machines*, Proc. 26th Annu. ACM Symp. Theory Comput. (1994), 625–634.
16. Müller, H. *Rasterized point location*, Proc. Workshop on Graph-Theoretic Concepts in Comput. Sci., Trauner Verlag, Linz (1985), 281–293.
17. Overmars, M.H. *Computational geometry on a grid: an overview*, ed. R. A. Earnshaw, Theoretical Foundations of Computer Graphics and CAD, NATO ASI, vol.F40, Springer-Verlag (1988), 167–184.
18. Yao, A.C. *Should tables be sorted?*, J. ACM, 28 (1981), 615–628.
19. Yao, A.C. *On the complexity of maintaining partial sums*, SIAM J. Comput. 14 (1985), 277–288.
20. Yao, A.C. *Decision tree complexity and Betti numbers*, Proc. 26th Annu. ACM Symp. Theory Comput. (1994), 615–624.