A Simple Randomized Sieve Algorithm for the Closest-Pair Problem

Samir Khuller [*][†]
University of Maryland

Yossi Matias [*][†]
University of Maryland
& Tel Aviv University

Abstract

We present a linear time randomized sieve algorithm for the closest-pair problem. The algorithm as well as its analysis are simple.

1. Introduction

The closest-pair problem can be found in almost any algorithms text-book as a basic problem in computational geometry (see e.g., [CLR90, PS86, Man89]). Deterministic algorithms that run in $O(n \log n)$ time are due to [Ben80, BS76, SH75, HNS88]. These algorithms are optimal in the algebraic decision-tree model of computation, where a matching lower bound of $\Omega(n \log n)$, even for the 1-dimensional closest-pair problem, is implied by a lower bound for element distinctness [Ben83]. This simply stated problem was used by Rabin [Rab76] in a classic paper, to illustrate the power of *randomization*, where he gave an algorithm that takes only $O(n)$ expected running time. Rabin uses a random sampling technique to decompose the problem into "small" subproblems for which the total cost of a brute force method is expected to be linear. His algorithm, although simple, has a somewhat complicated analysis.

In this paper, we present a novel approach using a sieve technique that yields a simple new algorithm. The algorithm takes linear expected time and has a simple analysis.

The *closest-pair* problem is defined as follows: given a set S of n points in $\Re^d$, for some constant $d > 0$, the task is to find the closest pair of points (in Euclidean distance).

The algorithm is described for the plane and can be easily extended to run in linear expected time for any fixed dimension.

[*]Partially supported by NSF grant CCR-8906949.
[†]Address: UMIACS, AV Williams Bldg., University of Maryland, College Park, MD 20742.

The more general ***all nearest neighbours*** problem, in which we are required to compute the closest neighbours for each point in the set S, has also been well studied [Ben80]. A randomized $O(n \log n)$ algorithm was given by [Cla83], and a deterministic $O(n \log n)$ algorithm was given by [V89].

2. The Sieve Closest-Pair Algorithm

Let S be the initial set of given points. We use $\delta(S)$ to denote the distance between points of the closest pair in S. We will assume that the points are not numbered, and we will number them as the algorithm proceeds. The algorithm consists of two stages. We first compute an approximation for $\delta(S)$; then, the approximation is used to compute $\delta(S)$. The main idea of the algorithm is to do a simple "filtering" process, in which points are deleted from the set. Let S_i be the set of remaining points at the start of iteration i (initially S_1 is S). The filtering continues as long as S_i is non-empty. In the filtering, the sizes of the sets S_i are shrinking rapidly. At the end of the process we get an approximation (up to a factor of 3) to the closest-pair distance. We then use a simple technique to find the closest pair.

The filtering process: Pick at random a point from S_i and call it x_i. The distance to the closest point from a point x, ***in the current set*** S_i is defined to be $d(x)$. Compute $d(x_i)$ by computing the distance from x_i to all points in S_i. To obtain S_{i+1} we delete from S_i all points x such that $d(x) \geq d(x_i)$. In this process, we may also delete some points x such that $d(x_i) > d(x) > d(x_i)/3$. However, all points x such that $d(x) \leq d(x_i)/3$ will remain in S_{i+1}. We stop the filtering process when the set S_i becomes empty. Let i^* be the smallest index such that $S_{i^*+1} = \emptyset$ and $S_{i^*} \neq \emptyset$. Let x_{i^*} be the point selected at random from S_{i^*} at iteration i^*. The closest-pair distance $\delta(S)$ is at most $d(x_{i^*})$ and at least $d(x_{i^*})/3$. Thus at the end of the filtering process we find an approximation to $\delta(S)$ within a factor of 3.

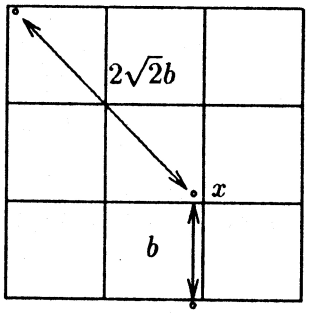

Figure 1: Neighborhood of a point x

It remains to show how to implement the above process, and how to use the resulting approximation $d(x_{i^*})$ to find the closest pair. These can be done easily by the notion of "neighborhood".

This notion was previously used in other algorithms for the closest-pair problem and for the more general all-nearest-neighbours problem. Consider a mesh of size b. The *neighborhood* of a point x, is the cell containing x plus the 8 neighboring cells (see Figure 1). Let $N(x)$ be the set of points in the neighborhood of x. The following facts can be easily verified.

(1) All points whose distance from x is at least $3b$ (actually $2\sqrt{2}b$) are not in $N(x)$.

(2) All points whose distance from x is at most b are in $N(x)$.

Note that points with distance from x between b and $2\sqrt{2}b$ may be either in $N(x)$ or not in $N(x)$.

Implementation of "filtering": Let x_i be a point selected at random from set S_i in iteration i. We build a mesh of size $b = d(x_i)/3$. A point x is deleted from S_i if and only if it is the only point in its neighborhood, (i.e., the only point in $N(x)$). This can be implemented in $O(|S_i|)$ expected number of steps and linear space, by using a perfect hash function [FKS84].

Lemma 2.1: *When the algorithm terminates ($S_{i^*+1} = \emptyset$), then $d(x_{i^*})/3 \leq \delta(S) \leq d(x_{i^*})$.*

Proof: Clearly, $\delta(S) \leq d(x_{i^*})$ (by definition). By Fact (1), in iteration i all points x with $d(x) \geq d(x_i)$ are deleted. Therefore, the sequence $\{d(x_i)\}$ is monotonically decreasing. Let j^* be the first iteration in which a point u of the closest-pair is deleted from S_{j^*}. By Fact (2) all points x with $d(x) \leq d(x_{j^*})/3$ are not deleted from S_{j^*}. Note that $d(u) = \delta(S)$ since both closest-pair points are in S_{j^*}. Therefore, $\delta(S) \geq d(x_{j^*})/3 > d(x_{i^*})/3$. □

Computing $\delta(S)$ from its approximation: We will use the following simple facts.

Consider a mesh of size b and assume that $b/3 \leq \delta(S) \leq b$. Then

(3) The neighborhood of each point in S contains at most a constant number of points.

(4) Each point of the closest pair is contained in the neighborhood of the other point.

We construct a mesh of size $d(x_{i^*})$ (the approximation of $\delta(S)$). For each non-empty cell we build a list of all points in this cell. This can be implemented in linear expected time and linear space by using a perfect hash function [FKS84]. Then, for each point we find the distance to its closest point in its neighborhood, if such a point exists. By Fact (3) this can be done in constant time per point, by using a brute force method (or other more efficient techniques). Fact (4) guarantees that for each point from the closest pair, the other point is in its neighborhood. Thus, the closest pair will be found by computing the minimum over the distances.

We now give the algorithm in more detail.

The Sieve Closest-Pair Algorithm:

Step 0. Initialize S_1 to be S, and $i = 1$.

Step 1. Pick at random a point x_i from S_i, and compute $d(x_i)$, the distance to the closest point in S_i.

Step 2. Construct a mesh of size $b = d(x_i)/3$; Define $X_i = \{x_j \mid N(x_j) = \{x_j\}\}$; $S_{i+1} = S_i - X_i$.

Step 3. If $S_{i+1} \neq \emptyset$ then $i = i + 1$, and goto Step 1.
Else (if $S_{i+1} = \emptyset$) let $i^* = i$. $(d(x_{i^*})/3 \leq \delta(S) \leq d(x_{i^*}).)$

Step 4. Construct a mesh of size $d(x_{i^*})$. For each point x in S, compute the distance to the closest point to x in $N(x)$ (if there is such a point). Find the closest pair by computing the minimum over all computed distances.

Analysis:

The only thing to show is that the filtering process takes expected linear time. As noted above the cost of iteration i is linear in the size of S_i. Let us first explain intuitively why the sizes of the sets S_i, $i = 1, 2, \ldots$, is expected to decrease at least geometrically. The key idea is to consider the sequence $\{d(x) : x \in S_i\}$ in non-decreasing order. When selecting at random x_i, all points x such that $d(x) \geq d(x_i)$ are deleted. Thus, on the average half of the points are deleted in each iteration, and hence a geometric decrease is expected. Formally we have

Lemma 2.2:

$$\mathbf{E}\left(\sum_{i=1}^{i^*} \mid S_i \mid\right) \leq 2n$$

Proof: Let s_i be the cardinality of S_i. We first show by induction that $\mathbf{E}(s_i) \leq \frac{n}{2^{i-1}}$ for $i \geq 1$. By the argument given above, $\mathbf{E}(s_{i+1}) \leq s_i/2$. Therefore,

$$\mathbf{E}(s_{i+1}) = \mathbf{E}(\mathbf{E}(s_{i+1})) \leq \mathbf{E}\left(\frac{s_i}{2}\right) = \frac{1}{2}\mathbf{E}(s_i).$$

Thus, by inductive hypothesis, $\mathbf{E}(s_{i+1}) \leq \frac{1}{2}\frac{n}{2^{i-1}} = \frac{n}{2^i}$.

By linearity of expectation,

$$\mathbf{E}\left(\sum_{i=1}^{i^*} s_i\right) = \sum_{i=1}^{i^*} \mathbf{E}(s_i) \leq \sum_{i=1}^{i^*} \frac{n}{2^{i-1}} \leq 2n.$$

□

We therefore have,

Theorem The sieve closest-pair algorithm solves the closest-pair problem in $O(n)$ expected time and $O(n)$ space.

Acknowledgments: We are grateful to Mike Atallah, Omer Berkman, Dave Mount, Micha Sharir, and Uzi Vishkin for helpful comments.

References

[Ben80] J. L. Bentley, "Multidimensional divide-and-conquer", *Comm. ACM 23*, pp. 214–229, (1980).

[Ben83] M. Ben-Or, "Lower bounds for algebraic computation trees", *15th Symp. on the Theory of Computing*, pp. 80–86, (1983).

[BS76] J. L. Bentley and I. Shamos, "Divide and conquer in multidimensional space", *8th Symp. on the Theory of Computing*, pp. 220–230, (1976).

[Cla83] K. L. Clarkson, "Fast algorithms for the all nearest neighbours problem", *24th Symp. on Foundations of Computer Science*, pp. 226–232, (1983).

[CLR90] T. Cormen, C. Leiserson and R. Rivest, "Introduction to Algorithms", McGraw Hill and The MIT Press, (1990).

[FKS84] M. L. Fredman, J. Komlós, and E. Szemerédi, "Storing a sparse table with $O(1)$ worst case access time", *J. of the Association for Computing Machinery*, 31, pp. 538–544, (1984).

[HNS88] K. Hinrichs, J. Nievergelt and P. Schorn, "Plane-sweep solves the closest pair problem elegantly", *Information Processing Letters*, 26, pp. 255-261, (1988).

[Man89] U. Manber, "Introduction to Algorithms: A Creative Approach", Addison Wesley, (1989).

[PS86] F. Preparata and M. Shamos, "Computational Geometry", Springer Verlag, (1986).

[Rab76] M. Rabin, "Probabilistic Algorithms", in *Algorithms and Complexity, Recent Results and New Directions*, Academic Press, pp. 21–39, (1976).

[SH75] M. I. Shamos and D. Hoey, "Closest-point problems", *16th Annual Symposium on Foundations of Computer Science*, pp. 151–162, (1975).

[V89] P. M. Vaidya, "An $O(n \log n)$ algorithm for the all-nearest-neighbors problem", *Discrete & Comput. Geometry*, 4, pp. 101–115, (1989).