A Simple Analysis of Rabin's Algorithm for Finding Closest Pairs*

Bahareh Banyassady† Wolfgang Mulzer†

Abstract

The closest-pair problem is one of the most basic topics in computational geometry: given a set $P \subset \mathbb{R}^2$ of n points in the plane, find two distinct points $p, q \in P$ that minimize the Euclidean distance $d(p, q)$, among all pairs of points in P. In the algebraic decision tree model, this problem can be solved optimally in $\Theta(n \log n)$ time.

However, already in 1976, Rabin observed in a seminal work that, using the floor function and randomization, this can be improved to $O(n)$ *expected* time. We provide a new and simplified analysis of Rabin's algorithm that is intended to make this result more accessible to the modern reader.

1 Introduction

The closest-pair problem in computational geometry is defined as follows: given a set $P \subset \mathbb{R}^d$ of n points in d dimensions, find two distinct points $p, q \in P$ that minimize the Euclidean distance $d(p, q)$, among all pairs of points in P. As was already observed in the 1970s, this problem can be solved in the plane in $O(n \log n)$ time by computing the Delaunay triangulation of P [11, 13, 14]. For any fixed $d \geq 2$, the classic divide-and-conquer algorithm by Bentley and Shamos also achieves $O(n \log n)$ time [2, 5]. This running time is asymptotically optimal in the algebraic decision tree model of computation [1, 11].

However, this is far from the whole story. Once we leave the confines of the algebraic decision tree model, faster algorithms are possible. For example, in the *transdichotomous* model, where the input may be manipulated at the bit-level, we can compute planar Delaunay triangulations, and hence also the planar closest pair, in $O(n \log \log n)$ expected time [3].

More famously, if we admit the *floor-function* $x \mapsto \lfloor x \rfloor$ into our model, there are randomized algorithms that can compute the closest pair in *linear* expected time. This was shown first by Rabin, in a famous paper that is often considered the starting point for the study of randomized algorithms [12]. Indeed, it has been claimed that Rabin's algorithm is one of the first randomized algorithms in theoretical computer science [15].

*Supported by DFG project MU/3501-2.

†Institut für Informatik, Freie Universität Berlin, Germany
{bahareh, mulzer}@inf.fu-berlin.de

Since then, a lot more work has been done on the closest pair problem: Dietzfelbinger *et al.* [6] describe how to implement rigorously the hashing-based data structure that was left open in Rabin's original algorithm. They also provide a detailed analysis of Rabin's algorithm that shows that bounded independence suffices to obtain the desired expected running time. Khuller and Matias [9] describe an alternative, sieve based approach to closest pairs, and Golin *et al.* [7] give a very simple randomized algorithm that uses the randomized incremental construction paradigm and that can be found in several textbooks [8, 10]. Finally, Chan [4] presents a randomized framework for geometric optimization problems that also leads to a new randomized linear-time algorithm for the closest pair problem. The survey by Smid [15] contains a much more comprehensive treatment of these results.

Despite the amount of activity on the closest pair problem, the presentation of Rabin's original algorithm has remained untouched for more than 40 years [12]. We give a new description and analysis of this algorithm in today's terms. We hope that this simplified presentation will make Rabin's algorithm more accessible for modern students of computational geometry, and it may lead to new insights on the closest pair problem. Here, we focus on the planar case, although all the arguments hold for d dimensions, when $d \in \mathbb{N}$ is a constant.

2 Preliminaries

Let $P \subset \mathbb{R}^2$ be a set of n points in the plane such that all $\binom{n}{2}$ interpoint distances in P are pairwise distinct. Furthermore, we assume that P lies completely in the upper right quadrant, i.e., that all points in P have positive coordinates.

For $i \in \mathbb{Z}$, we define the *ith grid* $\mathcal{G}_i$ as the subdivision of the plane into square grid *cells* of diameter 2^i. The cells have pairwise disjoint interiors, side length $2^i/\sqrt{2}$, and they cover the whole plane. The grid $\mathcal{G}_i$ is aligned such that the origin appears as a corner of four adjacent grid cells. The *neighborhood* of a grid cell $\sigma \in \mathcal{G}_i$ consists of the 7×7 square of grid cells centered at σ. Two cells $\sigma, \tau \in \mathcal{G}_i$ are *neighboring* if τ lies in the neighborhood of σ (and hence σ lies in the neighborhood of τ). We define the identifier of a cell $\sigma \in \mathcal{G}_i$ as a pair from $\mathbb{Z} \times \mathbb{Z}$, indicating the column and row for σ. The cell whose lower left corner is the

origin is identified as $(0,0)$. As it is standard in randomized algorithms for the closest pair problem, we assume that there is an operation **findGridCell**(i,p), for $i \in \mathbb{Z}$ and $p \in \mathbb{R}^2$, that returns the identifier of the cell σ in $\mathcal{G}_i$, that contains the point p [7, 9, 12, 15]. Using the floor function, **findGridCell** can be implemented in $O(1)$ time.

A *cell dictionary* is a data structure for storing cells that are in a grid [7]. It supports the following operations:

- **create**(i): create an empty cell dictionary D for $\mathcal{G}_i$.
- **insert**(D,p): insert the point p into D.
- **lookup**(D,σ): Suppose that D is a cell dictionary for $\mathcal{G}_i$ and σ is a cell in $\mathcal{G}_i$. Then, the lookup operation reports the set of all points stored in D that lie in σ. It returns $\emptyset$, if σ contains no points.

As explained by Golin *et al.* [7], the cell dictionary can be implemented using the hashing-based techniques of Dietzfelbinger *et al.* [6], so that the expected time for **create** and **insert** is $O(1)$ and the expected time for **lookup** is $O(1+k)$, where k is the output size. Alternatively, an implementation based on binary search trees achieves a worst-case running time of $O(1)$ for **create**, $O(\log n)$ for **insert**, and $O(\log n + k)$ for **lookup**, where again k denotes the output size. In our analysis, we will separately count the operations on the cell dictionary and the remaining computational steps.

3 The Algorithm

We now describe our version of Rabin's algorithm. Let $P \subset \mathbb{R}^2$ be the n input points in the plane, and set $k = \lfloor \log n \rfloor - 1$. We compute a *random gradation* $P = P_0 \supset P_1 \supset P_2 \supset \cdots \supset P_k$ of P, where for $i = 1, \ldots, k$, the set P_i is a random subset of P_{i-1} with exactly $|P_i| = \lfloor n/2^i \rfloor$ elements. In particular, we have $|P_k| = O(1)$.

The algorithm proceeds in *rounds*. The rounds are numbered from $k+1$ to 1, beginning with round $k+1$. For $i = k+1, \ldots, 1$, the goal of round i is to compute a cell dictionary D_{i-1} that stores all points from P_{i-1} such that:

(A) each cell in D_{i-1} contains at most one point from P_{i-1}; and

(B) let $p, q \in P_{i-1}$ be the two points that constitute the closest pair in P_{i-1}. Then, the cells in D_{i-1} that contain p and q are neighboring.

Since $|P_k| = O(1)$, this can be easily achieved in round $k+1$: by checking all pairs in P_k, we compute the closest pair distance δ_k for P_k. Then, we set $j = \lceil \log \delta_k \rceil - 1$, and we create a cell dictionary D_k for the grid $\mathcal{G}_j$. We insert all points of P_k into D_k. Since the diameter of the cells of $\mathcal{G}_j$ is $2^j \in [\delta_k/2, \delta_k)$, each cell in D_k contains at most one point of P_k. Furthermore, since the cells in $\mathcal{G}_j$ have side length at least $\delta_k/2\sqrt{2}$, the cells for the closest pair in P_k are neighboring.

In round i, $i = k, \ldots, 1$, the algorithm has the cell dictionary D_i from the previous round available, and it constructs the dictionary D_{i-1} for round i as follows: first, we insert all points from P_{i-1} into D_i. Then, for each non-empty cell σ in D_i, we find the set Q_σ of points inside σ. We use a brute-force algorithm to compute the closest pair distance δ_σ for Q_σ, and we set $\delta_{i-1} = \min_{\sigma \in D_i} \delta_\sigma$. Next, we set $j = \lceil \log \delta_{i-1} \rceil - 1$, and we create a cell dictionary D_{i-1} for $\mathcal{G}_j$. Then, we insert all points from P_{i-1} into D_{i-1}. By construction, the diameter of the cells of D_{i-1} is $2^j \in [\delta_{i-1}/2, \delta_{i-1})$, and so, each cell of D_{i-1} contains at most one point of P_{i-1}. Furthermore, since the cells in D_{i-1} have side length at least $\delta_{i-1}/2\sqrt{2}$, the cells for the closest pair in P_{i-1} must be neighboring (we note that the closest pair distance could be much less than δ_{i-1}, since we only check the distances inside each cell of D_i to compute δ_{i-1}). The following lemma summarizes the running time of round i.

Lemma 1 *Let $i \in \{1, \ldots, k\}$. In round i, the algorithm performs $O(|P_{i-1}|)$ cell dictionary operations, and the additional work is proportional to*

$$\sum_{\sigma \in D_i} |Q_\sigma|^2,$$

where the sum is over all non-empty cells σ stored in D_i, and Q_σ is defined as $P_{i-1} \cap \sigma$.

Once we have the cell dictionary D_0 for P at hand, we can compute the closest pair of P with $O(n)$ cell dictionary operations and $O(n)$ additional work: we simply check the neighborhood of each non-empty cell in D_0, and we find the closest pair among all points that reside in these cells. Since each cell of D_0 contains at most one point, and since the closest pair must be in neighboring cells, this gives the closest pair of P in the desired time.

4 Analysis

We now analyse our version of Rabin's algorithm. For $i \in \{1, \ldots, k\}$, let

$$Z_i = \sum_{\sigma \in D_i} |Q_\sigma|^2$$

be the random variable that represents the amount of work in round i, excluding the time for the cell dictionary operations. We will show that $\mathbf{E}[Z_i] = O(|P_{i-1}|)$, for $i = 1, \ldots, k$.

For this, we fix an $i \in \{1, \ldots, k\}$ and a subset $Q \subseteq P$ with $|Q| = \lfloor n/2^{i-1} \rfloor$. First, we rewrite Z_i in a slightly different way.

Lemma 2 *Let $i \in \{1, \ldots, k\}$ and $Q \subseteq P$ with $|Q| = \lfloor n/2^{i-1} \rfloor$ be given. For $p \in Q$, let X_p denote the number of points from Q that are in the same cell of D_i as p (including p). Then,*

$$\mathbf{E}[Z_i \mid P_{i-1} = Q] = \sum_{p \in Q} \mathbf{E}[X_p \mid P_{i-1} = Q].$$

Proof. This can be seen through a simple application of the double-counting principle. We have

$$\begin{aligned}
&\mathbf{E}[Z_i \mid P_{i-1} = Q] \\
&= \mathbf{E}\Big[\sum_{\sigma \in D_i} |Q_\sigma|^2 \;\Big|\; P_{i-1} = Q\Big] \\
&= \mathbf{E}\Big[\sum_{\sigma \in D_i} \sum_{p \in \sigma \cap Q} |Q_\sigma| \;\Big|\; P_{i-1} = Q\Big] \\
&= \mathbf{E}\Big[\sum_{p \in Q} \sum_{\substack{\sigma \in D_i \\ p \in \sigma}} |Q_\sigma| \;\Big|\; P_{i-1} = Q\Big] \\
&= \mathbf{E}\Big[\sum_{p \in Q} X_p \;\Big|\; P_{i-1} = Q\Big],
\end{aligned}$$

as claimed. □

Next, we bound the expectation of X_p.

Lemma 3 *Let $i \in \{1, \ldots, k\}$ and $Q \subseteq P$ with $|Q| = \lfloor n/2^{i-1} \rfloor$ be given. Let $p \in Q$, and let X_p denote the number of points from Q that are in the same cell of D_i as p (including p). Then,*

$$\mathbf{E}[X_p \mid P_{i-1} = Q] = O(1).$$

Proof. Set $q = |Q|$, and let $r \in \{1, \ldots, q\}$. We know that $q \geq 2$ (since $|P_{i-1}| \geq 2$). First, we show that

$$\Pr[X_p \geq r \mid P_{i-1} = Q] \leq 2r\Big(\frac{2}{3}\Big)^{r-1}. \qquad (1)$$

For this, consider the event that $X_p \geq r$. This means that, the cell $\sigma \in D_i$ that contains p must contain at least r points from Q. How can this happen? For $j \in \mathbb{Z}$, let τ_j be the cell of $\mathcal{G}_j$ that contains p, and let $Q_j = Q \cap \tau_j$. Obviously, for j small enough, we have $Q_j = \{p\}$, for j large enough, we have $Q_j = Q$ (since we assumed that all coordinates in P are positive), and as j increases, Q_j grows monotonically. Let j^* be the smallest index such that τ_{j^*} has at least r point of Q , $|Q_{j^*}| \geq r$. And let $R \subseteq Q_{j^*}$ be an arbitrary subset with $|R| = r$. Now, if the cell $\sigma \in D_i$ with $p \in \sigma$ contains at least r points from Q, due to the definition of j^*, the grid cell τ_{j^*} is a subcell of the grid cell σ, and so, σ contains all of R. Furthermore, since σ appears in D_i, by the invariant it must be the case that σ contains at most one point from P_i; see Figure 1. Thus, $|P_i \cap R| \leq 1$, which implies:

$$\Pr[X_p \geq r \mid P_{i-1} = Q] \leq \Pr[|P_i \cap R| \leq 1 \mid P_{i-1} = Q].$$

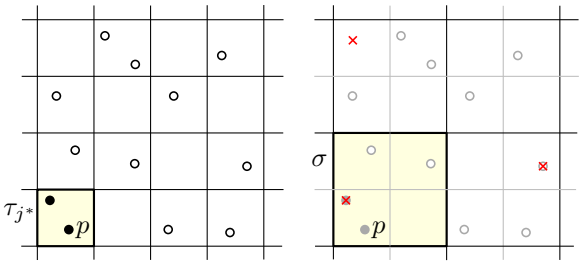

Figure 1: Left: The grid $\mathcal{G}_{j^*}$ and the set Q of points are shown. For $q = 2$ and the point p, the cell τ_{j^*} is specified. The points of the set $R \subseteq Q_{j^*}$, are filled with black color. Right: The cell σ of D_i is specified. The red crosses show the points of P_i. The cell σ contains all the points of R, thus $|P_i \cap R| = 1$.

Given $P_{i-1} = Q$, we have that P_i is a random subset of $\lfloor q/2 \rfloor$ points from Q. Since R is also a subset of Q, the desired probability is easily bounded. If $r > \lceil q/2 \rceil$, then the intersection of P_i and R is not empty, in another words $\Pr[|P_i \cap R| = 0 \mid P_{i-1} = Q] = 0$. Otherwise, if $r \leq \lceil q/2 \rceil$, we have

$$\begin{aligned}
&\Pr[|P_i \cap R| = 0 \mid P_{i-1} = Q] \\
&= \binom{q-r}{\lfloor q/2 \rfloor} \Big/ \binom{q}{\lfloor q/2 \rfloor} \\
&= \frac{(q-r)!}{\lfloor q/2 \rfloor! \, (\lceil q/2 \rceil - r)!} \frac{\lfloor q/2 \rfloor! \, \lceil q/2 \rceil!}{q!} \\
&= \prod_{k=0}^{r-1} \frac{\lceil q/2 \rceil - k}{q-k} \leq \left(\frac{\lceil q/2 \rceil}{q}\right)^r \leq \Big(\frac{2}{3}\Big)^r,
\end{aligned}$$

since $q \geq 2$. Moreover, if we have $r > \lceil q/2 \rceil + 1$, then $\Pr[|P_i \cap R| = 1 \mid P_{i-1} = Q] = 0$. Otherwise, if $r \leq \lceil q/2 \rceil + 1$, we have

$$\begin{aligned}
&\Pr[|P_i \cap R| = 1 \mid P_{i-1} = Q] \\
&= r \binom{q-r}{\lfloor q/2 \rfloor - 1} \Big/ \binom{q}{\lfloor q/2 \rfloor} \\
&= r \frac{(q-r)!}{(\lfloor q/2 \rfloor - 1)! \, (\lceil q/2 \rceil - r + 1)!} \frac{\lfloor q/2 \rfloor! \, \lceil q/2 \rceil!}{q!} \\
&= r \frac{\lfloor q/2 \rfloor}{q - r + 1} \prod_{k=0}^{r-2} \frac{\lceil q/2 \rceil - k}{q-k} \\
&\leq r \left(\frac{\lceil q/2 \rceil}{q}\right)^{r-1} \leq r\Big(\frac{2}{3}\Big)^{r-1},
\end{aligned}$$

since $q \geq 2$ and $\lfloor q/2 \rfloor / (q - r + 1) \leq 1$. Now, (1) follows, since

$$\begin{aligned}
\Pr[X_p \geq r \mid P_{i-1} = Q] &\leq \Pr[|P_i \cap R| \leq 1 \mid P_{i-1} = Q] \\
&= \Pr[|P_i \cap R| = 0 \mid P_{i-1} = Q] \\
&\quad + \Pr[|P_i \cap R| = 1 \mid P_{i-1} = Q] \\
&\leq \Big(\frac{2}{3}\Big)^r + r\Big(\frac{2}{3}\Big)^{r-1} \\
&\leq 2r\Big(\frac{2}{3}\Big)^{r-1}.
\end{aligned}$$

Now, we have

$$\mathbf{E}[X_p \mid P_{i-1} = Q] \leq \sum_{r=1}^{q} \Pr[X_p \geq r \mid P_{i-1} = Q] \leq \sum_{r=1}^{\infty} 2r\Big(\frac{2}{3}\Big)^{r-1} = O(1),$$

as claimed. □

Lemma 4 *For $i \in \{1, \ldots, k\}$, $\mathbf{E}[Z_i] = O(|P_{i-1}|)$.*

Proof. Fix $Q \subseteq P$ with $|Q| = \lfloor n/2^{i-1} \rfloor$. By Lemma 2 and Lemma 3, we have

$$\mathbf{E}[Z_i \mid P_{i-1} = Q] = \sum_{p \in Q} \mathbf{E}[X_p \mid P_{i-1} = Q] = \sum_{p \in Q} O(1) = O(|Q|) = O(|P_{i-1}|).$$

Thus, using the law of total probability

$$\begin{aligned} E[Z_i] &= \sum_{\substack{Q \subseteq P \\ |Q| = \lfloor n/2^{i-1} \rfloor}} \Pr[P_{i-1} = Q]\mathbf{E}[Z_i \mid P_{i-1} = Q] \\ &= O(|P_{i-1}|) \sum_{\substack{Q \subseteq P \\ |Q| = \lfloor n/2^{i-1} \rfloor}} \Pr[P_{i-1} = Q] \\ &= O(|P_{i-1}|), \end{aligned}$$

as claimed. □

The following theorem summarizes the analysis:

Theorem 5 *The algorithm from Section 3 computes the closest pair for P in expected time $O(n)$.*

Proof. We already argued correctness in Section 3. As mentioned above, using randomization and the floor function, we can implement all operations of the cell dictionary in $O(1)$ expected time [6, 7]. Thus, by Lemma 1, the total expected time for the cell dictionary operations is:

$$\sum_{i=1}^{k} O\left(|P_{i-1}|\right) = O\left(\sum_{i=0}^{k-1} \frac{n}{2^i}\right) = O(n).$$

Similarly, by Lemma 1 and Lemma 4, the total expected time for the remaining steps is:

$$\sum_{i=1}^{k} \mathbf{E}[Z_i] = O\left(\sum_{i=1}^{k} |P_{i-1}|\right) = O(n).$$

We also argued that, having the cell dictionary D_0 for P, we can compute the closest pair of P in $O(n)$ time. This concludes the analysis. □

References

[1] M. Ben-Or. Lower bounds for algebraic computation trees (preliminary report). In *Proc. 15th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 80–86, 1983.

[2] J. L. Bentley and M. I. Shamos. Divide-and-conquer in multidimensional space. In *Proc. 8th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 220–230, 1976.

[3] K. Buchin and W. Mulzer. Delaunay triangulations in $o(\mathrm{sort}(n))$ time and more. *J. ACM*, 58(2):6, 2011.

[4] T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22(4):547–567, 1999.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, third edition, 2009.

[6] M. Dietzfelbinger, T. Hagerup, J. Katajainen, and M. Penttonen. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms*, 25(1):19–51, 1997.

[7] M. Golin, R. Raman, C. Schwarz, and M. Smid. Simple randomized algorithms for closest pair problems. *Nordic J. Comput.*, 2(1):3–27, 1995.

[8] S. Har-Peled. *Geometric approximation algorithms*. Mathematical Surveys and Monographs. American Mathematical Society, vol 173, 2011.

[9] S. Khuller and Y. Matias. A simple randomized sieve algorithm for the closest-pair problem. *Inform. and Comput.*, 118(1):34–37, 1995.

[10] J. M. Kleinberg and É. Tardos. *Algorithm design*. Addison-Wesley, 2006.

[11] F. P. Preparata and M. I. Shamos. *Computational geometry*. Springer-Verlag, 1985.

[12] M. O. Rabin. Probabilistic algorithms. In *Algorithms and Complexity: New Directions and Recent Results*, pages 21–40. Academic Press, 1976.

[13] M. I. Shamos. Geometric complexity. In *Proc. 7th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 224–233, 1975.

[14] M. I. Shamos and D. Hoey. Closest-point problems. In *Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 151–162, 1975.

[15] M. Smid. Closest-point problems in computational geometry. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 877–935. Elsevier, 2000.