A Simple Randomized Algorithm for All Nearest Neighbors

Soroush Ebadian* Hamid Zarrabi-Zadeh*

Abstract

Given a set $P$ of $n$ points in the plane, the all nearest neighbors problem asks for finding the closest point in $P$ for each point in the set. The following folklore algorithm is used for the problem in practice: pick a line in a random direction, project all points onto the line, and then search for the nearest neighbor of each point in a small vicinity of that point on the line. It is widely believed that the expected number of points needed to be checked by the algorithm in the vicinity of each point is $O(\sqrt{n})$ on average. We confirm this conjecture in affirmative by providing a careful analysis on the expected number of comparisons made by the algorithm. We also present a matching lower bound, showing that our analysis is essentially tight.

1 Introduction

The all nearest neighbors problem considers finding, for a set $P$ of $n$ points in the plane, the nearest neighbor of each point in $P$. This is a fundamental and well-studied problem in computational geometry, with various applications, e.g., in statistics, similarity search, and image processing.

Several $O(n \log n)$ time algorithms are available for the problem. In particular, it is well-known that the Delaunay triangulation of $P$ contains all edges connecting nearest neighbors. (See Figure 1.) Therefore, one can solve the all nearest neighbors problem in the plane in $O(n \log n)$ time using any of the optimal algorithms available for the Delaunay triangulation [2, 4]. In higher fixed dimensions, one can solve the problem in $O(n \log n)$ time using the algorithms of Clarkson [1] and Vaidya [6]. Both algorithms make use of spatial data partitioning trees, such as compressed quad-trees [5] and R-trees [3].

In this paper, we study an extremely simple randomized algorithm for the all nearest neighbors problem that uses no geometric data structure, and can be implemented in a few lines of code. It basically projects all the points onto a random line and searches for the nearest neighbor of each point in a small vicinity of that point on the line.

The main contribution of this paper is a careful and tight analysis of the expected runtime of this randomized algorithm. More precisely, we show that the expected number of comparisons made by the algorithm is $O(\alpha n \sqrt{n})$ in total, where $\alpha = \sqrt{\log \delta + 1}$, and $\delta$ is the ratio of the largest to smallest pairwise distance between the points and their nearest neighbors. In practice, $\alpha$ is upper bounded by a constant. For example, when input coordinates are represented by rational numbers with 64-bit integers, we have $\delta \leq 2^{128}$, and hence, $\sqrt{\log_2 \delta + 1}$ is at most 12.

The utter simplicity of the algorithm has made it a popular choice in cases where a fast implementation is preferable at the cost of slightly relaxing the optimal runtime. Due to its simplicity and removing the overhead of geometric data structures, the algorithm is even faster in practice compared to the other standard algorithms for the problem, such as Delaunay triangulations, when input data has only a few thousand points. Moreover, the algorithm finds the nearest neighbor of each point independently, after an initial step, which makes it highly flexible for parallel implementation.

2 Preliminaries

Let $p$ and $q$ be two points in the plane. We denote the Euclidean distance of $p$ and $q$ by $||pq||$. For a unit vector $u$ in the plane, we denote by $||pq||_u$ the projected distance between $p$ and $q$ along direction $u$. In other words, $||pq||_u = (p - q) \cdot u = ||pq|| \cos \theta$, where $\theta$ is the angle between $pq$ and $u$. Since $\cos \theta \leq 1$, we always have $||pq||_u \leq ||pq||$. 

*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. Email: ebadian@ce.sharif.edu, zarrabi@sharif.edu.
3 The Algorithm

In this section, we present the folklore randomized algorithm for the all nearest neighbors problem, and prove its correctness. The algorithm in its entirety is given in Algorithm 1. It takes as input a set \( P = \{p_1, \ldots, p_n\} \) of \( n \) points in the plane, and returns for each point \( p_i \) its nearest neighbor \( q_i \) in \( P \).

Algorithm 1 All Nearest Neighbors \((p_1, \ldots, p_n)\)

1: pick a random unit vector \( u \)
2: for \( i \) from 1 to \( n \) do
3: \( d_i \leftarrow \infty \)
4: for \( j \) in increasing order of \( \|p_i p_j\|_u \leq d_i \) do
5: if \( \|p_i p_j\| < d_i \) then
6: \( d_i \leftarrow \|p_i p_j\|, \quad q_i \leftarrow p_j \)
7: return \( q_1, \ldots, q_n \)

The algorithm works as follows. After picking a random unit vector, the algorithm processes each point \( p_i \) by checking the points in \( P \setminus \{p_i\} \) in their increasing projected distance to \( p_i \), while keeping the minimum Euclidean distance found so far in \( d_i \). The search for the nearest neighbor of \( p_i \) is terminated whenever we reach a point whose projected distance to \( p_i \) is more than \( d_i \).

An example of the execution of Algorithm 1 for a point \( p \) is illustrated in Figure 2. In this example, points are numbered in their increasing projected distance to \( p \). The algorithm stops when it reaches point \( p_5 \), whose projected distance to \( p \) is more than the best distance found so far, i.e., \( \|p p_2\| \).

![Figure 2: An example of the execution of Algorithm 1.](image)

To quickly iterate over the points in their increasing projected distance from a point \( p \), we perform a simple preprocess step as follows. We select a line \( \ell \) in direction \( u \), project each point \( p_i \in P \) to a point \( p'_i \) on \( \ell \), and sort the projected points along \( \ell \). Then, in the main loop for each point \( p_i \), we keep two pointers on \( \ell \) initially set to the points right before and after \( p'_i \) on \( \ell \), walking in opposite directions. At each step, we compare the distance of \( p'_j \) to the two projected points specified by the pointers, select whichever is smaller, and advance the corresponding pointer to the next one.

This way, iterating over each point \( p_j \) takes \( O(1) \) time in the algorithm.

The correctness of the algorithm is proved in the following lemma.

Lemma 1 For each point \( p_i \in P \), the algorithm correctly finds the nearest neighbor of \( p_i \).

Proof. Fix a point \( p_i \), and let \( q \) be the nearest point of \( p_i \) in \( P \). Suppose by way of contradiction that the inner loop of the algorithm terminates on a point \( p_j \), before reaching \( q \). Thus, \( \|p_i p_j\|_u < \|p_i q\|_u \). The inner loop terminates if \( \|p_i p_j\|_u > d_i \), where \( d_i \) is the distance between \( p_i \) and a previously-visited point \( p_k \). Therefore, we have \( \|p_i q\|_u > \|p_i p_j\|_u > d_i = \|p_i p_k\| \), which contradicts the fact that \( q \) is the closest point to \( p_i \).

4 The Analysis

Let \( P = \{p_1, \ldots, p_n\} \) be the set of input points. For \( 1 \leq i \leq n \), we denote by \( d_i \) the distance of \( p_i \) to its nearest neighbor in \( P \). Let \( P_l = \{p_j \in P \setminus \{p_i\} : \|p_i p_j\|_u \leq d_i \} \) be the set of points compared by the algorithm during the search for the nearest neighbor of \( p_i \).

Let \( X \) be a random variable indicating the total number of comparisons made by Algorithm 1. We can decompose \( X \) into \( n^2 \) indicator variables

\[
X_{i,j} = \begin{cases} 1 & \text{if } p_j \in P_i, \\ 0 & \text{otherwise.} \end{cases}
\]

Note that \( X_{i,i} = 0 \) for all \( i \), and \( X = \sum_{1 \leq i, j \leq n} X_{i,j} \).

Lemma 2 For all \( 1 \leq i, j \leq n, i \neq j \),

\[
\Pr \{X_{i,j} = 1\} \leq \frac{d_i}{\|p_i p_j\|_u}
\]

Proof. Fix two points \( p_i \) and \( p_j \) in \( P \). For all \( r \geq d_i \), let \( C_r \) be a circle of radius \( r \) centered at \( p_i \). Consider a strip \( S \) of width \( 2d_i \) enclosing \( C_{d_i} \) orthogonal to direction \( u \).

![Figure 3: An illustration of Lemma 2. The strip \( S \) is shown in gray, and \( C_r \cap S \) is shown by thick arcs.](image)
(see Figure 3). Note that for all points $p \in P$, we have $p \in P_i$ if and only if $p$ lies in $S$.

Let $A(r)$ denote the length of $C_r \cap S$, for all $r \geq d_i$. The curvature of the arcs in $C_r \cap S$ decreases as $r$ is increased, and hence, $A(r)$ is a decreasing function on $[d_i, \infty)$. Therefore, for all $r \geq d_i$, $A(r) \leq A(d_i) = 2\pi d_i$.

Since direction $u$ is chosen uniformly at random, the angle $\theta$ between $\overrightarrow{p_i p_j}$ and $u$ is uniformly chosen from the range $[0, 2\pi)$. In other words, $p_j$ lies uniformly at random on a circle $C_r$ with $r = ||p_i p_j||$. Therefore, the event $p_j$ lies in $S$ corresponds to the fraction $A(r)/2\pi r$ of the points on $C_r$. Hence,

$$\Pr\{X_{i,j} = 1\} = \frac{A(r)}{2\pi r} \leq \frac{2\pi d_i}{2\pi r} = \frac{d_i}{r},$$

which completes the proof. □

Let $E[X_{i,j}] = \sum_{n=1}^{\infty} E[X_{i,j}].$ An upper bound $B$ on $E[X_{i,j}]$ yields an upper bound $nB$ on $E[X]$, because $E[X] = \sum_{n=1}^{\infty} E[X_{i,j}]$. The rest of this section focuses on finding such an upper bound on $E[X_{i,j}]$.

**Lemma 3** For each $1 \leq j \leq n$, there is a permutation $\sigma$ of $\{1, \ldots, n\}$ such that

$$E[X_{i,j}] \leq 3 \sum_{i=1}^{n} \frac{d_{\sigma_i}}{\sqrt{\sum_{k=1}^{i-1} d_{\sigma_k}^2}}.$$

**Proof.** Let $p_{\sigma_1}, \ldots, p_{\sigma_n}$ be the points of $P$ ordered in their increasing distance from $p_j$. Note that $p_{\sigma_i} = p_j$.

For $1 \leq i \leq n$, let $C_i$ be a circle of radius $d_{\sigma_i}/2$ centered at $p_{\sigma_i}$. For any pair of points $p_{\sigma_i}$ and $p_{\sigma_j}$, $||p_{\sigma_i} p_{\sigma_j}|| \geq \max\{d_{\sigma_i}, d_{\sigma_j}\} \geq (d_{\sigma_i} + d_{\sigma_j})/2$. Therefore, all $C_i'$s are non-overlapping. Fix an index $2 \leq i \leq n$. Let $\ell = ||p_{\sigma_i} p_j||$, and $B_i = \{C_{\sigma_1}, \ldots, C_i\}$. Every circle in $B_i$ has radius at most $\ell/2$, and its center lies within distance $\ell$ to $p_j$. (See Figure 4.) Therefore, all circles in $B_i$ fit in a disk of radius $\frac{\ell}{2}$ centered at $p_j$. As the circles are non-overlapping, the area of $C$ must be at least as large as the total area of the circles in $B_i$. Therefore, $(\frac{\ell}{2})^2 \pi \geq \sum_{k=1}^{i} (\frac{d_{\sigma_k}}{2})^2 \pi$, and thus, $\ell = ||p_{\sigma_i} p_j|| \geq \sqrt{\sum_{k=1}^{i} d_{\sigma_k}^2}$. Now,

$$E[X_{i,j}] = \sum_{i=1}^{n} E[X_{i,j}] \leq \sum_{i=1}^{n} \frac{d_{\sigma_i}}{||p_{\sigma_i} p_j||} (\text{by Lemma 2})$$

$$= \sum_{i=2}^{n} \frac{d_{\sigma_i}}{||p_{\sigma_i} p_j||} \leq \sum_{i=2}^{n} \frac{3 \cdot d_{\sigma_i}}{\sqrt{\sum_{k=1}^{i-1} d_{\sigma_k}^2}},$$

which implies the lemma’s statement. □

Based on the upper bound proved in Lemma 3, we define the following function:

$$f(a_1, \ldots, a_n) = \frac{\sum_{i=1}^{n} a_i}{\sqrt{\sum_{j=1}^{n} a_j^2}},$$

where $a_1, \ldots, a_n$ is a sequence of real numbers. We prove some useful properties of $f$ in the next lemmas.

**Lemma 4** Let $A = \{a_1, \ldots, a_n\}$ be a set of positive real numbers, and $\sigma$ be a permutation of $A$. Then $f(\sigma)$ is maximized if $\sigma$ is a non-decreasing sequence.

**Proof.** Suppose by contradiction that $f$ is maximized by a permutation $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ which is not non-decreasing. Then, there exists an index $i$ such that $x = \sigma_i > \sigma_{i+1} = y$. Let $\pi = \{\sigma_1, \sigma_2, \ldots, \sigma_{i-1}, \sigma_i, \sigma_{i+1}, \ldots, \sigma_n\}$ be the ordering achieved by swapping $\sigma_i$ and $\sigma_{i+1}$. As $\sigma$ is an ordering that maximizes $f$, we have $f(\sigma) \geq f(\pi)$. Since the two permutations only differ in the $i$-th and $(i + 1)$-th term, by the definition of $f$, and by setting $s = x^2 + y^2 + \sum_{j=1}^{i-1} \sigma_j^2$, we have

$$\frac{x}{\sqrt{s - y^2}} + \frac{y}{\sqrt{s}} \geq \frac{x}{\sqrt{s}} + \frac{y}{\sqrt{s - x^2}},$$

which yields

$$x \cdot \left[\frac{1}{\sqrt{s - y^2}} - \frac{1}{\sqrt{s}}\right]^{-1} \geq y \cdot \left[\frac{1}{\sqrt{s - y^2}} - \frac{1}{\sqrt{s}}\right]^{-1}.$$

Since function $z \cdot \left[\frac{1}{\sqrt{s - y^2}} - \frac{1}{\sqrt{s}}\right]^{-1}$ is decreasing in the range $(0, s)$, the last inequality implies $x \leq y$. But this contradicts the fact that $x > y$. □

**Lemma 5** For any integer $n \geq 1$, and any real number $a \geq 0$,

$$\sum_{i=1}^{n} \frac{1}{\sqrt{a + i}} \leq 2\sqrt{n}.$$

**Proof.** Since $\frac{1}{\sqrt{s}}$ is a decreasing function on $(0, +\infty)$, for any real number $b > 1$, we have

$$\int_{x=b-1}^{b} \frac{1}{\sqrt{x}} > \int_{x=b-1}^{b} \frac{1}{\sqrt{b}} = \frac{1}{\sqrt{b}}.$$

Therefore,

$$\sum_{i=1}^{n} \frac{1}{\sqrt{a + i}} \leq \int_{x=a}^{a+n+1} \frac{1}{\sqrt{x}} dx = 2(\sqrt{a + n} - \sqrt{a}) \leq 2\sqrt{n},$$

Figure 4: A set of non-overlapping circles $\{C_1, \ldots, C_6\}$. 
where the last inequality follows from the fact that 
\[ \sqrt{x} + \sqrt{y} \leq \sqrt{x + y} \] for all \( x, y \geq 0 \).

\[ \square \]

**Lemma 6** Given real numbers \( a_1, \ldots, a_n \) with \( 1 \leq a_i \leq c \), for some constant \( c \geq 1 \),
\[ f(a_1, \ldots, a_n) \leq 2b\sqrt{n}\sqrt{\log_2 c + 1}. \]
for all \( b > 1 \).

**Proof.** Let \( \hat{a}_i = b^{\lfloor \log_2 a_i \rfloor} \). Since \( b > 1 \) and \( a_i \geq 1 \), we have \( \hat{a}_i \leq a_i < b \cdot \hat{a}_i \). Therefore,
\[ f(a_1, \ldots, a_n) = \sum_{i=1}^n \frac{a_i}{\sqrt{\sum_{j=1}^n a_j^2}} \leq \sum_{i=1}^n \frac{b \cdot \hat{a}_i}{\sqrt{\sum_{j=1}^n a_j^2}} = b \cdot f(\hat{a}_1, \ldots, \hat{a}_n). \]
By Lemma 4, \( f(\hat{a}_1, \ldots, \hat{a}_n) \) is maximized when \( \hat{a}_i \)’s are sorted non-decreasingly. Let \( s_i = \lfloor j : \lfloor \log_2 a_j \rfloor = i \rfloor \), for all \( i \in \{0, 1, \ldots, \lfloor \log_2 c \rfloor \} \). Then
\[ f(\hat{a}_1, \ldots, \hat{a}_n) \leq \sum_{i=0}^{\lfloor \log_2 c \rfloor} \frac{b^i}{\sqrt{\sum_{j=1}^{i} k \cdot 2^i + \sum_{j=0}^{i-1} s_j \cdot 2^j}} = \sum_{i=0}^{\lfloor \log_2 c \rfloor} \frac{1}{\sqrt{\sum_{j=1}^{i} k + \sum_{j=0}^{i-1} s_j \cdot 2^j}} \]
which by Lemma 5 is at most \( 2 \sum_{i=0}^{\lfloor \log_2 c \rfloor} \sqrt{s_i} \). This sum is maximized at equality. Hence, as \( \sum s_i = n \), we have
\[ \sum_{i=0}^{\lfloor \log_2 c \rfloor} \sqrt{s_i} \leq \sum_{i=0}^{\lfloor \log_2 c \rfloor} \sqrt{\frac{n}{\log_2 c} + 1} \leq \sqrt{n} \cdot \sqrt{\log_2 c + 1}. \]
Therefore,
\[ f(a_1, \ldots, a_n) \leq 2b\sqrt{n} \cdot \sqrt{\log_2 c + 1}. \]
\[ \square \]

### 4.1 Lower Bound

In this section, we show that the analysis presented in Section 4 is essentially tight by providing a lower bound example on which Algorithm 1 has a matching expected runtime. Our example is simply formed by the points of a \( \sqrt{n} \times \sqrt{n} \) square lattice. The nearest neighbor to each point in this lattice has distance exactly one, and hence, \( \delta = 1 \) in this case. The following theorem proves a lower bound of \( \Omega(n\sqrt{n}) \) on the expected runtime of the algorithm on this example, which matches the upper bound of \( O(n\sqrt{n}) \) proved in the previous section.

**Theorem 8** The expected runtime of Algorithm 1 on a \( \sqrt{n} \times \sqrt{n} \) square lattice is \( \Omega(n\sqrt{n}) \).

**Proof.** Let \( P = \{p_1, \ldots, p_n\} \) be the set of points on the lattice, and let \( u \) be the random unit vector chosen by the algorithm. The nearest neighbor to each point in \( P \) has distance one. Therefore, if \( X \) is a random variable indicating the size of the set \( \{ (p_i, p_j) : \|p_i - p_j\|_u \leq 1 \} \), then \( E[X] \) is a lower bound on the expected runtime of the algorithm.

We first claim that \( \Pr \{ \|p_ip_j\|_u \leq 1 \} \geq \frac{1}{\pi \|p_ip_j\|} \), for all \( 1 \leq i, j \leq n \). Fix two points \( p_i \) and \( p_j \). Let \( \ell \) be a line in direction \( u \) passing through \( p_i \), and let \( p_j' \) be the projection of \( p_j \) onto \( \ell \). Therefore, \( p_i, p_j, \) and \( p_j' \) form a right triangle. (See Figure 5.) Now, \( \|p_ip_j\|_u \leq 1 \) holds if and only if
\[ \angle p_ip_jp_j' = \arcsin\left( \frac{1}{\|p_ip_j\|} \right) \leq \arcsin\left( \frac{1}{\|p_ip_j\|} \right). \]
As \( \angle p_ip_jp_j' \) is chosen randomly, and \( \arcsin(x) > x \) for all \( 0 < x \leq 1 \), we have
\[ \Pr \{ \|p_ip_j\|_u \leq 1 \} \geq \frac{1}{\pi} \arcsin\left( \frac{1}{\|p_ip_j\|} \right) > \frac{1}{\pi \cdot \|p_ip_j\|}. \]
Every two points in the lattice have distance at most \( 2\sqrt{n} \). Therefore, \( \Pr \{ \|p_ip_j\|_u \leq 1 \} > \frac{2\sqrt{n}}{2\pi \sqrt{n}} \). Thus,
\[ E[X] > \frac{n(n-1)}{2\pi \sqrt{n}}, \]
and hence, the expected runtime of the algorithm is \( \Omega(n\sqrt{n}) \).
\[ \square \]

Figure 5: Projection of \( p_j \) on different lines specified by the random vector \( u \).
5 Conclusions

In this paper, we analyzed an extremely simple randomized algorithm for the all nearest neighbors problem. We proved that the algorithm has $O(\alpha n \sqrt{n})$ expected runtime, where $\alpha$ is a parameter of the input point set, usually bounded by a constant in practice.

Our analysis can be extended in a natural way to the case of general $L_p$ metric, yielding the same expected runtime. For higher $d$-dimensional space, we conjecture that the expected runtime of the algorithm is $O(n^{d-\frac{2}{2}} \text{poly}(\alpha))$. We can also extend the algorithm to report $k$ nearest neighbors of each point. While our analysis immediately implies an upper bound of $O(k \alpha \cdot n \sqrt{n})$ on the expected number of comparisons made by the algorithm, it is intriguing to obtain a tighter analysis for this variant of the problem.

Acknowledgments The authors would like to thank Michael Tikhomirov for suggesting the analysis of the algorithm studied in this paper as an open problem.

References