

## Chapter 3

# Approximate Nearest Neighbors

The main topic of this chapter is the following general algorithmic problem, called *nearest neighbor search*: Given a set  $P$  of  $n$  points in some metric space  $(X, \rho)$ , we would like to construct a data structure such that, for any point  $x \in X$  (called a *query point*), we can quickly find a nearest neighbor (with respect to the metric  $\rho$ ) of  $x$  in  $P$ , i.e., a point  $p^* \in P$  such that  $\rho(p^*, x) \leq \rho(p, x)$  for all  $p \in P$ . Of course, the point  $p^*$  with this property need not be uniquely defined, i.e., there can be several nearest neighbors, but we do not let that deter us and denote any of them by  $\text{NN}(x, P)$  (resolving ties arbitrarily, if necessary).

Obviously, there is a trivial solution to this task: Assuming that we have some way of computing the distance  $\rho(x, y)$  for any two points in  $X$ , we can simply store the input points  $P$  (in a list, say). Then, given a query point  $q$ , we can go through the list of points  $p \in P$  one by one, compute the distance  $\rho(p, q)$  for each of them. As we go along, we keep track of the minimum distance (and the minimizing point) computed so far. This procedure requires linear storage (assuming that we can store each point in constant space, otherwise there will be some correction factor) and a linear number of distance computations.

This kind of problem arises in numerous application areas, such as information retrieval, machine learning, computer vision etc., to drop just a few keywords. For a detailed survey of nearest neighbor search in various settings, including many references, we refer the reader to the article by Indyk [4]. Typically, the data sets  $P$  that arise in the applications are very large ( $n$  could be over a billion), and it is not very desirable to have to go through the entire data set for each query. Instead, we would like to preprocess  $P$  into some data structure that allows us to answer each query in substantially sublinear time. Ideally, we would like to achieve a query time that is logarithmic or polylogarithmic in  $n$ , while keeping the data structure “small”, ideally of size that is near-linear or polynomial (of low degree) in  $n$ .

### 3.1 ANN in Euclidean Space

We will restrict our attention to the case that  $X$  is some Euclidean space  $\ell_2^d$  (i.e.,  $\mathbb{R}^d$  with the Euclidean norm  $\|\cdot\|_2$ ). If  $d$  is small, the ideal can be more or less achieved. For instance,  $n$  points in the Euclidean plane  $\ell_2^2$  can be preprocessed into a data structure of size  $O(n)$  such that each query can be answered in time  $O(\log n)$ , see Indyk’s survey for the references.<sup>1</sup> However, also the dimension  $d$  is often quite large in the applications. For instance, the points in  $P$  could represent pictures in some database. If each picture has a resolution of  $1280 \times 854$  pixels, and for each pixel we have an essentially continuous range of colors, then we can think of each picture as a point in  $\mathbb{R}^{1280 \cdot 854} = \mathbb{R}^{1093120}$ . It is debatable if the Euclidean distance between these points is a good measure of similarity between the pictures, but at any rate, dimensions of several hundreds or thousands occur regularly in practice. Unfortunately, the known data structures and algorithms for nearest neighbor search have the property that either their storage requirements grow exponentially with the dimension  $d$  (for instance, there is a data structure with query time  $d^{O(1)} \log n$  but storage requirement  $n^{\Theta(d)}$ ); or the query time is very close to linear, i.e., not much better than what we get from the trivial solution. This problem is known as the “curse of dimensionality”. Again, we refer to [4] for more details.

One natural approach that has proved to be rather successful in avoiding this curse is to be a little bit less ambitious and look for *approximate nearest neighbors* instead. More precisely, given a finite set  $P$  of points in some metric space  $(X, \rho)$ , a query point  $x \in X$ , and a constant  $c \geq 1$ , we say that  $p \in P$  is a *c-approximate nearest neighbor* (with respect to the metric  $\rho$ ) of  $x$  in  $P$  if  $\rho(q, x) \leq c \cdot \rho(p, x)$  for all  $q \in P$ . We write  $p = c\text{-ANN}(x, P)$ , for short. Ideally, we would like to be able to choose the approximation factor close to 1, i.e.,  $c = 1 + \varepsilon$  for some small parameter  $\varepsilon > 0$  (which we can make smaller, but usually at the expense of query time and/or storage).

In this chapter, we will discuss a recent ANN data structure and algorithm by Ailon and Chazelle [1]:

**Theorem 3.1.** *Given a set  $P$  of  $n$  points in  $\ell_2^d$  and  $\varepsilon > 0$ , there exists a randomized data structure of size  $n^{O(\varepsilon^{-2})}$  that allows us, for every query point  $x \in \mathbb{R}^d$ , to compute a point  $p \in P$  such that with high probability (at least  $1 - 1/n$ , say), the computation takes  $O(d\varepsilon^{-2} \log n + \varepsilon^{-2} \log^2 n)$  steps and  $p = (1 + \varepsilon)\text{-ANN}(x, P)$ .*

**Remarks 3.2.** 1. *We will assume that the parameters  $n, d$ , and  $\varepsilon$  are such that the bound  $O(d\varepsilon^{-2} \log n)$  for the query time is less than  $dn$ , otherwise the naive*

---

<sup>1</sup>We remark that as usual in computational geometry, these storage requirements and running times refer to the so-called REAL RAM model of computation, where it is assumed that a single real number can require only a constant amount of storage and arithmetic operations on numbers can be executed in constant time.

algorithm of checking all the distances is faster and we do not have to bother to construct a complicated ANN data structure.

2. The statement “with high probability” in the theorem is with respect to the preprocessing, i.e., during the randomized construction of the data structure. If we are unlucky, the data structure might be bad for a query point  $x$  in the sense that the point  $p$  that we compute might not be a  $(1 + \varepsilon)$ -ANN of  $x$  and also the computation might take longer (but we like, we can always stop the computation after  $O(d\varepsilon^{-2} \log n)$  steps, since we already know that we have failed if it takes longer). In particular, since all of our random choices will take place during the construction of the data structure, a malicious adversary with access to the random bits used during that preprocessing could construct a specific query  $x$  for which our data structure fails its purpose.
3. We can increase the success probability of  $1 - 1/n$  to  $1 - n^{-C}$  for any constant  $C > 0$ , by increasing the implicit constants in the  $O$ -notation.

One of the main tools for the preprocessing is the following variant of the Johnson-Lindenstrauss Flattening Theorem.

**Theorem 3.3 (Flattening for  $\ell_2^d \rightarrow \ell_1^k$ ).** For  $\varepsilon > 0$ , any  $n$ -element subset of  $\ell_2^d$  can be randomly projected into  $\ell_1^k$  with distortion at most  $(1 + \varepsilon)$ , where  $k = O(\varepsilon^{-2} \log n)$ .

It will be more convenient for what follows to parametrize the distortion by  $\varepsilon/9$ . More precisely, the theorem states the following: There exist constants  $C, \varepsilon_0 > 0$  and  $d_0, n_0 \geq 1$  (implicit in the  $O$ -notation) with the following property: Given  $0 < \varepsilon < \varepsilon_0$  and integers  $d \geq d_0, n \geq n_0$ , and  $k \geq C\varepsilon^{-2} \log n$ , there is a random distribution of linear maps  $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^k$  such that the following holds: For  $y \in \mathbb{R}^d$ ,

$$\Pr[\underbrace{(1 - \varepsilon/9)\|x\|_2 \leq \|\Phi(y)\|_1 \leq (1 + \varepsilon/9)\|y\|_2}_{(*)}] \geq 1 - n^{-5}.$$

It follows that for  $Y \subset \mathbb{R}^d, |Y| = n$ , the probability that  $(*)$  holds for all  $y \in Y$  is at least  $1 - n^{-4}$ .

**Remark 3.4.** As we will see, the query time of the approximate nearest neighbor data structure is dominated by the time needed to compute the projection  $\Phi(x)$ . Since  $\Phi$  is a linear map  $\mathbb{R}^d \rightarrow \mathbb{R}^k$ , this means evaluating a  $(k \times d)$ -matrix on a  $d$ -dimensional vector. For the usual notions of random projections, the matrix will be dense (i.e., it will have few nonzero entries), which leads to the running time of  $dk = O(d\varepsilon^{-2} \log n)$ . Ailon and Chazelle in fact present a faster variant of the map  $\Phi$  such that, with high probability, only  $O(d \log d + \varepsilon^{-3} \log n)$  steps are required for the projection, which leads to a corresponding faster query time. We will not discuss this speedup.

## 3.2 Description of the ANN-Algorithm

We begin by listing

## The Main Ingredients of the ANN-Data Structure.

- All possible *combinatorial balls* of  $P$ , i.e., subsets  $Q \subseteq P$  of the form  $Q = P \cap B(p,r)$  for some  $p \in P$  and some radius  $r > 0$ . For each of these combinatorial balls, we store its diameter  $\text{diam}(Q)$  and its *effective radius*  $\max_{q \in Q} \|q - p\|_2$ . More precisely, for each point  $p \in P$ , we store a sorted list of the effective radii of the combinatorial balls centered at  $p$ , and for each radius, a pointer back to the ball.
- A set  $S$  of *possible search distances*; these will correspond to current “guesses” of the distance  $\|x - \text{NN}(x, P)\|_2$ .
- For each  $s \in S$ , a *discretization map*  $D^s : \mathbb{R}^d \rightarrow \mathbb{Z}^k$ , where  $k = O(\varepsilon^{-2} \log n)$  is the embedding dimension in the  $(\ell_2^d \rightarrow \ell_1^k)$ -Flattening Theorem 3.3. The map  $D_s$  will be composed of the random projection  $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^k$  from the Flattening Theorem and a *randomized rounding*  $\text{Rnd}_s : \mathbb{R}^k \rightarrow \mathbb{Z}^k$  depending on the parameter  $s$ . The discretization  $D^s$  will be *additive*, i.e.,  $D^s(u \pm v) = D^s(u) \pm D^s(v)$  for all  $u, v \in \mathbb{R}^d$ , and will, with high probability, rescale distances by a factor of roughly  $k/s$ . (This will be made more precise later.)
- For each  $s \in S$  and each  $p \in P$ , a *lookup table*  $T_{p,s}$ . The entries of  $T_{p,s}$  are indexed by integer vectors  $v \in \mathbb{Z}^k$  such that

$$\|v - D^s(p)\|_1 \leq 2(1 + \varepsilon/3)k.$$

For each such  $v$ , we store the the nearest  $\ell_1$ -neighbor of  $v$  among discretized points  $D^s(P)$ . More precisely, the entry  $T_{p,s}[v]$  is defined as the point  $q \in P$  that minimizes  $\|v - D^s(q)\|_1$ .

We postpone the detailed definition and analysis of these items (including estimating their complexity and making precise what “with high probability” means) until after the description of the algorithm.

The ANN-algorithms proceeds in two stages:

**Stage A: Finding an  $O(n)$ -ANN.** We chose a unit vector  $u \in \mathbb{S}^{d-1}$  uniformly at random. We precompute the numbers (1-dimensional projections)  $\langle u, p \rangle$  for  $p \in P$  and store them in sorted order. Given a query point  $x \in \mathbb{R}^d$ , we compute the number  $\langle u, x \rangle$  and compute the point  $p_0 \in P$  whose 1-dimensional projection is closest to that of  $x$ , i.e.,  $|\langle u, x \rangle - \langle u, p_0 \rangle| \leq |\langle u, x \rangle - \langle u, p \rangle|$  for all  $p \in P$ . Using binary search, we can find  $p_0$  in  $O(d + \log n)$  steps. We will show below that with constant probability at least  $1/2$ , the point  $p_0$  is an  $O(n)$ -ANN of  $x$  in  $P$ . By repeating Stage A  $O(\log n)$  times with independent choices of  $u$  and keeping the best of the outputs  $p_0$ , we can increase the success probability to  $1 - 1/(2n)$ .

**Stage B: From  $O(n)$  to  $(1 + \varepsilon)$  by Binary Search.** From now on, we assume that in Stage A, we have found  $p_0 = O(n)$ -ANN( $x, P$ ). Under this assumption, we can restrict our attention to the combinatorial ball  $Q := P \cap B(p_0, 2\|x - p_0\|_2)$ , because we have  $\|x - \text{NN}(x, P)\|_2 \leq \|x - p_0\|_2$  and thus, by the triangle inequality  $\|\text{NN}(x, P) - p_0\|_2 \leq \|x - \text{NN}(x, P)\|_2 + \|x - p_0\|_2 \leq 2\|x - p_0\|_2$ , i.e.,  $\text{NN}(x, P) \in Q$ .

We refine the approximation by binary search. Table 3.1 contains a precise description of this in pseudocode. At each step  $t$  of the binary search, we will work with a search distance  $s_t \in S$  and a point  $p_t \in Q$ , with the property that  $\|x - p_t\|_2 \leq 2s_t$  (we can think of  $s_t$  as our current guess of the distance from  $x$  to its nearest neighbor, and of  $p_t$  as the current candidate for the nearest neighbor). For the decision whether the search distance  $s_t$  is still too coarse and we should further decrease it, we will use the discretization  $D^{s_t}$ , and the lookup table  $T_{p_t, s_t}$ . The  $k/s_t$ -rescaling properties of the discretization will imply  $\|D^{s_t}(x) - D^{s_t}(p_t)\|_1 \leq 2(1 + \varepsilon/3)k$  with high probability, so there is indeed an entry  $q = T_{p_t, s_t}[D^{s_t}(x)]$  in the table.

### 3.3 Correctness of the Algorithm

We remark that all random choices for the ANN-algorithm are already made during the preprocessing, when constructing the data structure (the list of 1-dimensional projections  $\langle u, p \rangle$ , the discretization maps  $D^s$  and the lookup tables  $T_{p, s}$  for  $p \in P$  and  $s \in S$ ). In Lemma 3.19 below, we shall show that for any given  $x \in \mathbb{R}^d$ , with probability at least  $1 - 1/(2p)$ , all these random choices are favorable for us in the sense that the following holds:

**Assumption L (“L” for “Lucky”):** The point  $p_0$  produced in Stage A is an  $O(n)$ -ANN of  $x$  in  $P$ . Moreover, for every step  $t$  of the WHILE-loop the discretization  $D^{s_t}$  is *reliable* for all points  $p \in P$ , in the sense that

(i) *either*  $\|x - p\|_2 \leq \frac{s_t}{2}$  and  $\|D^{s_t}(x - p)\|_1 \leq \frac{1 + \varepsilon}{2}k$ ,

(ii) *or*  $\|x - p\|_2 > \frac{s_t}{2}$  and

$$(1 - \frac{\varepsilon}{3})\|x - p\|_2 \leq \frac{s_t}{k}\|D^{s_t}(x - p)\|_1 \leq (1 + \frac{\varepsilon}{3})\|x - p\|_2.$$

The goal of this section is to prove the following:

**Proposition 3.5.** *If  $\varepsilon \leq 1/3$  and if Assumption L holds, then the algorithm returns a  $(1 + \varepsilon)$ -ANN of  $x$  in  $P$ .*

We assume from now on and for the remainder of this section that  $\varepsilon \leq 1/3$  and that Assumption L holds.

```

ANN_StageB( $x, P, p_0$ ):
  (* Takes an  $O(n)$ -ANN  $p_0 \in P$  of  $x$  and,
  with high probability, returns a  $(1 + \varepsilon)$ -ANN *)
   $Q := P \cap B(p_0, 2\|x - p_0\|_2)$ 
   $\Delta := \text{diam}(Q)$ 
  IF  $\Delta \leq \frac{\varepsilon}{2}\|x - p_0\|$  THEN
    RETURN  $p_0$  (*  $p_0$  is already a  $(1 + \varepsilon)$ -ANN *)
  ELSE
    (* For all  $q \in Q$ , we have  $\Omega(\Delta/n) \leq \|x - q\|_2 \leq 6\varepsilon^{-1}\Delta$  *)
     $t := 0$  (* Counts the number of iterations of the WHILE-loop
    below for purposes of the exposition and later reference *)
     $s_0 := 6\varepsilon^{-1}\Delta$ 
    TOO_COARSE := TRUE
    WHILE TOO_COARSE
      (* Refine the search distance  $s_t$  by binary search
      and update  $p_t$ ; maintain  $\|p_t - x\|_2 \leq 2s_t$  *)
       $q := T_{p_t, s_t}[D^{s_t}(x)]$ 
      IF  $\|x - q\|_2 \leq s_t$  THEN
         $p_{t+1} := q$ 
         $s_{t+1} := s_t/2$ 
         $t := t + 1$ 
      ELSE
        TOO_COARSE := FALSE
    RETURN  $q$ 

```

Table 3.1: Stage B of the ANN-Algorithm.

We begin by showing that the proposition is true if the diameter of  $Q$  is small:

**Lemma 3.6.** *If  $\Delta = \text{diam}(Q) \leq \frac{\varepsilon}{2}\|x - p_0\|_2$  then  $p_0 = (1 + \varepsilon)$ -ANN( $x, P$ ).*

*Proof.* Let  $p^* = \text{NN}(x, P)$ . As remarked above, we have  $p^* \in Q$ . Therefore,  $\|p_0 - p^*\|_2 \leq \Delta \leq \frac{\varepsilon}{2}\|x - p_0\|_2$ , and hence  $\|x - p^*\|_2 \geq \|x - p_0\|_2 - \|p_0 - p^*\|_2 \geq (1 - \varepsilon/2)\|x - p_0\|_2$ . Thus,  $\|x - p_0\|_2 \leq \frac{1}{1 - \varepsilon/2}\|x - p^*\|_2 \leq (1 + \varepsilon)\|x - p^*\|_2$ , as desired (the last inequality holds for all  $\varepsilon \leq 1$ ).  $\square$

From now on, assume also that  $\Delta = \text{diam}(Q) > \frac{\varepsilon}{2}\|x - p_0\|_2$ .

Next, we observe that the comment after the first ELSE in Stage B of the algorithm is indeed correct:

**Lemma 3.7.** *For every  $q \in Q$ ,  $\Omega(\delta/n) \leq \|x - q\|_2 < \frac{6}{\varepsilon}\Delta$ .*

*Proof.* We have  $\Delta \leq 2\|x - p_0\|_2$  by definition of  $Q$ , and the assumption that  $p_0 = O(n)$ -ANN( $x, P$ ) implies  $\|x - q\|_2 \geq \Omega(1/n)\|x - p_0\|_2 = \Omega(\Delta/n)$  for all  $q \in P$ , hence in particular for all  $q \in Q$ . In order to see the second inequality, we note that for  $q \in Q$ ,  $\|x - q\|_2 \leq \|x - p_0\|_2 + \|p_0 - q\|_2 \leq 3\|x - p_0\|_2 < \frac{6}{\varepsilon}\Delta$ .  $\square$

**Lemma 3.8.** *For every iteration  $t$  of the WHILE-loop, we have  $\|x - p_t\|_2 \leq 2s_t$ .*

*Proof.* For  $t = 0$ , we have  $\|x - p_0\|_2 < \frac{2}{\varepsilon}\Delta < s_0$ . In later iterations, we only set  $s_{t+1} = s_t/2$  provided  $\|x - p_{t+1}\|_2 \leq s_t = 2s_{t+1}$ .  $\square$

**Lemma 3.9.** *Assume that the WHILE-loop terminates after iteration  $t$ , i.e., that  $q = T_{p_t, s_t}[D^{s_t}(x)]$  was the result of a lookup at iteration  $t$  and that  $\|x - q\|_2 > s_t$ . Then  $q = (1 + \varepsilon)$ -ANN( $x, P$ ).*

*Proof.* First, note that  $\|x - p_t\|_2 \leq s_t$  implies that  $\|D^{s_t}(x - p_t)\|_1 \leq 2(1 + \varepsilon/3)k$ , so there is indeed an entry  $q = T_{p_t, s_t}[D^{s_t}(x)]$ . Moreover, note that  $\|D^{s_t}(x - q)\|_1 \geq (1 - \varepsilon/3)k$ . Next, we claim that  $\|x - p\|_2 > s_t/2$  for all  $p \in P$ . For if  $\|x - p\|_2 \leq s_t/2$  for some  $p$ , then  $\|D^{s_t}(x - p)\|_1 \leq \frac{1+\varepsilon}{2}k < (1 - \varepsilon/3)k$ , contradicting the definition of  $q$ . Therefore, for all  $p \in P$ , we have  $\|x - p\|_2 \geq \frac{1}{1+\varepsilon/3}\|D^{s_t}(x - p)\|_1 \geq \frac{1}{1+\varepsilon/3}\|D^{s_t}(x - q)\|_1 \geq \frac{1-\varepsilon/3}{1+\varepsilon/3}\|x - q\|_2 \geq \frac{1}{1+\varepsilon}\|x - q\|_2$ , as desired.  $\square$

This completes the proof of correctness. For later purposes, we note the following consequence of the preceding lemmata:

**Lemma 3.10.** *The number of iterations of the WHILE loop in Stage B of the algorithm is at most  $O(\log(n/\varepsilon))$ .*

*Proof.* We only enter the WHILE loop if  $\Delta = \text{diam}(Q) > \frac{\varepsilon}{2}\|x - p_0\|_2$ , which implies  $\|x - \text{NN}(x, P)\|_2 \geq \Omega(\Delta/n)$ , by Lemma 3.7. By Lemma 3.8, we maintain the invariant that  $\|x - p_t\|_2 \leq 2s_t$ . Moreover, we halve the search distance  $s_t$  at every step. Since  $s_0 = 6\varepsilon^{-1}\Delta$ , and we only remain in the loop while there is a point  $q \in P$  with  $\|x - q\|_2 \leq s_t$ . Thus, we have  $\Omega(\Delta/n) \leq \|x - \text{NN}(x, P)\|_2 \leq s_t = 2^{-t} \cdot 6\varepsilon^{-1}\Delta$ , which implies the desired bound by cancelling out  $\Delta$  and taking logarithms.  $\square$

### 3.4 Analysis of the Data Structure

It remains to specify the set  $S$  of search radii and the discretization operator. Let  $t_{\max} = O(\log(n/\varepsilon))$  be the bound for the number of iteration of the WHILE loop. The set  $S$  consists of all search radii that appear as  $s_t$  during some iteration during some call of the algorithm:

$$S := \bigcup_{Q \text{ a combinatorial ball}} \{2^{-t} \cdot 6\varepsilon^{-1} \text{diam}(Q) : 0 \leq t \leq t_{\max}\}.$$

It follows that  $|S| \leq \binom{n}{2} \cdot t_{\max} = O(n^2 \log(n/\varepsilon))$ . For the discretization, we need the following tool from probability theory:

**Poisson Processes.** Before giving the formal definition, we mention a standard example of a Poisson process: Suppose we stand by the side of a busy highway and we count the number of cars that pass by within a given time interval. More precisely, we choose some arbitrary moment in time at which no car passes and we declare this to be time  $t = 0$ . For  $t > 0$ , we define  $\Psi^{(t)}$  to be the number of cars that pass by in the time interval  $(0, t]$ , while for  $t < 0$ , we set  $\Psi^{(t)}$  to be minus the number of cars that have passed in the time interval  $(t, 0]$ . Clearly, our count will depend on the density of traffic on the particular highway. The following definitions summarize the properties that we expect from the outcome of our counting:

**Definition 3.11.** Let  $\lambda > 0$  be a positive real number. An random variable  $Z$  that takes values in the set  $\mathbb{N}_0$  of nonnegative integers is said to have Poisson distribution with parameter  $\lambda$  if

$$\Pr[Z = n] = e^{-\lambda} \frac{\lambda^n}{n!}.$$

**Lemma 3.12.** 1. If  $Z$  is a Poisson random variable with parameter  $\lambda$  then  $\mathbf{E}[Z] = \lambda$ .

2. If  $Z_1, \dots, Z_k$  are independent Poisson with parameters  $\lambda_1, \dots, \lambda_k$ , respectively, then  $Z_1 + \dots + Z_k$  is Poisson with parameter  $\lambda_1 + \dots + \lambda_k$ .

The proof of this lemma is Exercise 18

**Definition 3.13.** Consider a family  $\Psi = (\Psi(t))_{t \in \mathbb{R}}$  of integer-valued random variables, indexed by real numbers.  $\Psi$  is called a (two-sided) Poisson process with rate  $\lambda > 0$  if the following properties hold:

1.  $\Psi(0) = 0$
2. For  $a < b$ , the difference  $\Psi(b) - \Psi(a)$  has the Poisson distribution with parameter  $\lambda(b - a)$ .
3. For any finite number of numbers  $a_1 < b_1 < a_2 < b_2 < \dots < a_m < b_m$ , the differences  $\Psi(b_i) - \Psi(a_i)$ ,  $i = 1, \dots, m$  are independent (the results of our counting for disjoint time intervals are independent).

One can show that the car counting example given at the beginning is indeed a Poisson process, provided we assume that the waiting times between the passage of the  $i$ -th car and the  $(i + 1)$ -th one is exponentially distributed (this is the usual mathematical model for waiting times); see Chung [3], Chapter 7.

**The Discretization.** For our discretization, we proceed as follows: For each  $s \in S$ , we consider  $k$  independent Poisson processes  $\Psi_1^s, \dots, \Psi_k^s$ , each with rate  $k/s$ . Given  $x \in \mathbb{R}^d$ , we define the discretization  $D^s(x) \in \mathbb{Z}^k$  coordinatewise by

$$D_i^s(x) := \Psi_i^s(\Phi_i(x)),$$

where  $\Phi = (\Phi_1, \dots, \Phi_k) : \mathbb{R}^d \rightarrow \mathbb{R}^k$  is the projection map in the  $\ell_2^d \rightarrow \ell_1^k$  Flattening Theorem.

The discretization assigns to each point  $x \in \mathbb{R}^d$  an integer point  $D^s(x) \in \mathbb{Z}^k$ , which we can think of as a rounding of the projection  $\Phi(x) \in \mathbb{R}^k$ . The crucial property is how this rounding behaves with respect to distances.

**Lemma 3.14.** *For  $x, y \in \mathbb{R}^d$ , the distance  $\|D^s(x) - D^s(y)\|_1$  has Poisson distribution with expected value  $\frac{k}{s}\|\Phi(x) - \Phi(y)\|_1$ .*

*Proof.* We have  $\|D^s(x) - D^s(y)\|_1 = \sum_{i=1}^k |\Psi_i^s(\Phi_i(x)) - \Psi_i^s(\Phi_i(y))|$ . By the properties of the Poisson distribution, each summand is a Poisson random variable with parameter  $\frac{k}{s}|\Phi_i(x) - \Phi_i(y)|$ , and since these summands are independent, it follows from Lemma 3.12 that the sum is a Poisson random variable with parameter  $\frac{k}{s} \sum_{i=1}^k |\Phi_i(x) - \Phi_i(y)| = \frac{k}{s}\|\Phi(x) - \Phi(y)\|_1$   $\square$

Moreover, when estimating the probability of success for our randomized algorithm, we will use the fact that Poisson variables are tightly concentrated around their expectation (see [2]. Appendix A, for the proof):

**Theorem 3.15.** *Let  $Z$  have Poisson distribution with parameter  $\lambda$ . Then, for all  $\delta > 0$ ,*

$$\begin{aligned} \Pr[Z \leq (1 - \delta)\lambda] &\leq e^{-\delta^2\lambda/2}, \text{ and} \\ \Pr[Z \geq (1 + \delta)\lambda] &\leq (e^\delta(1 + \delta)^{-(1+\delta)})^\lambda \end{aligned}$$

For our data structure, we precompute and store the projections  $\Phi(p)$  of all points  $p \in P$ . Moreover, for each  $p \in P$  and  $s \in S$ , we compute the integer point  $D^s(p)$  and we build the lookup tables  $T_{p,s}$ : For each integer vector  $v \in \mathbb{Z}^k$  with  $\|v - D^s(p)\|_1 \leq 2(1 + \varepsilon/3)k$ , we compute the point  $q \in P$  that minimizes  $\|v - D^s(q)\|_1$ , and define the entry  $T_{p,s}$  to be (a pointer to) this point  $q$ ; for this computation, we use the trivial nearest neighbor algorithm in  $\ell_1^k$ ; we can afford to do this, since we aim to prove a bound of  $n^{O(\varepsilon^{-2})}$  for the size of the data structure (and the time to compute it), and the factor of  $O(kn)$  steps to compute the  $\ell_1$ -NN of each  $v$  gets absorbed into the upper bound, by increasing the implicit constant in the exponent  $O(\varepsilon^2)$ .

**Lemma 3.16.** *The combined size of all lookup tables  $T_{p,s}$  is at most  $n^{O(\varepsilon^{-2})}$ .*

*Proof.* For any given  $p$  and  $s$ , the number of integer vectors  $v \in \mathbb{Z}^k$  with  $\|v - D^s(p)\|_1 \leq 2(1 + \varepsilon/3)k$  is at most  $2^k \binom{2(1+\varepsilon/3)k+k}{k} = 2^{O(k)} = n^{O(\varepsilon^2)}$ , see Exercise 19. Moreover, there are  $n$  points  $p$  and at most  $|S| = O(n^2 \log(n/\varepsilon))$  many different

values of  $s$ , and these factors get absorbed into the implicit constant in the exponent.  $\square$

### 3.5 Analysis of the Query Time

We begin by observing that Stage A of the algorithm takes  $O(d + \log n)$  steps ( $d$  for computing the projection  $\langle u, x \rangle$ , and  $\log n$  for finding the value closest to it in the sorted list  $\langle u, p \rangle$ ,  $p \in P$ ). When repeating this  $O(\log n)$  times, we get a time requirement of  $O(d \log n + \log^2 n)$ .

Once we have computed  $p_0$ , we can find  $Q$  and  $\Delta$  in time  $O(\log n)$  by binary search: In the list of effective radii of balls centered at  $p_0$ , we find the largest value smaller than  $2\|x - p_0\|_2$ .

For Stage B, we first compute the projection  $\Phi(x)$ ; this takes time  $O(dk) = O(d\varepsilon^{-2} \log n)$  (this is the size of the projection matrix). It remains to analyze the time complexity of the WHILE loop. By Lemma 3.10, the number of iterations is at most  $O(\log(n/\varepsilon))$ . At each iteration step  $t$  of the WHILE loop, the projection  $\Phi(x)$  is used to compute the discretizations  $D^{st}(x)$ ; ignoring the implementation details of the Poisson process, computing the discretization takes constant time for each coordinate, i.e., time  $O(k)$ . Moreover, as we proved in Section 3.4, each table  $T_{p_t, s_t}$  contains  $2^{O(k)}$  entries. If the table is implemented as a weight-balanced radix tree, the lookup time is  $O(k)$ ; we omit the details.

Hence, we see that the WHILE loop takes  $O(k \log(n/\varepsilon)) = O(\varepsilon^{-2}(\log n) \log(n/\varepsilon))$  steps. By the remark after Theorem 3.1, we have  $\varepsilon > n^{-O(n)}$ , hence  $\log(n/\varepsilon) = O(\log n)$ . Thus, Stage B takes  $O(d\varepsilon^{-2} \log n + \varepsilon^{-2} \log^2 n)$  steps, which dominates the query time. This proves the running time assertion in Theorem 3.1

### 3.6 The Probability of Being Lucky

**Lemma 3.17.** *Let  $p_0$  be the point found in Stage A of the algorithm (this point depends on the random choice of the projection vector  $u$ ). Then*

$$\mathbf{E}[\|x - p_0\|_2] = O(n\|x - \text{NN}(x, P)\|_2).$$

*Proof.* For  $p \in P$ , let  $\chi_p$  be the indicator variable of the event  $|\langle u, p \rangle - \langle u, x \rangle| \leq |\langle u, \text{NN}(x, P) \rangle - \langle u, x \rangle|$  (i.e.,  $\chi_p$  is 1 if this event occurs and zero otherwise). By Exercise 20,  $\mathbf{E}[\chi_p] = O(\|x - \text{NN}(x, P)\|_2 / \|x - p\|_2)$ . Moreover,  $\|x - p_0\|_2 \leq \sum_{p \in P} \chi_p \|x - p\|_2$ . Therefore, by linearity of expectation,  $\mathbf{E}[\|x - p_0\|_2] \leq \sum_{p \in P} \|x - p\|_2 \mathbf{E}[\chi_p] = O(n\|x - \text{NN}(x, P)\|_2)$ .  $\square$

**Corollary 3.18.** *If we repeat Stage A  $O(\log n)$  times (with independent choices of  $u$ ) then the point  $p_0$  is an  $O(n)$ -ANN of  $x$  in  $P$  with probability at least  $1 - \frac{1}{2n}$ .*

*Proof.* By choosing the implicit constant in the  $O(n)$ -approximation factor sufficiently large, it follows immediately from the preceding lemma and Markov's inequality, that for one random  $u$ , the probability that  $p_0$  is an  $O(n)$ -ANN of  $x$  in  $P$  is at least  $1/2$ . By repeating Stage A  $O(\log n)$  times with independent choices of  $u$ , the failure probability becomes smaller than  $(1/2)^{O(\log n)} \leq 1/(2n)$ .  $\square$

**Lemma 3.19.** *For any given query point  $x$ , with probability at least  $1 - 1/(2n)$ , the following is true for all  $s \in S$  and all  $p \in P$ :*

1. If  $\|x - p\|_2 \leq s/2$  then  $\|D^s(x - p)\|_1 \leq \frac{1+\varepsilon}{2}k$ ;
2. if  $\|x - p\|_2 > s/2$  then

$$\left(1 - \frac{\varepsilon}{3}\right)\|x - p\|_2 \leq \frac{s}{k}\|D^s(x - p)\|_1 \leq \left(1 + \frac{\varepsilon}{3}\right)\|x - p\|_2.$$

*Proof.* Fix  $x$ . For a given  $p$  and  $s$ , by Lemma 3.14, the random variable  $\|D^s(x - p)\|_1 = \|D^s(x) - D^s(p)\|_1$  has Poisson distribution with expected value  $\frac{k}{s}\|\Phi(x) - \Phi(p)\|_1$ . Both parts of the assertions now follow from the low-distortion properties of the projection map  $\Phi$  (Theorem 3.3, applied to the set of difference vectors  $Y = \{x - p : p \in P\}$ ) and the concentration bounds for Poisson random variables (Theorem 3.15): With probability at least  $1 - n^{-5}$ , we have  $(1 - \varepsilon/9)\|x - p\|_2 \leq \|\Phi(x - p)\|_1 \leq (1 + \varepsilon/9)\|x - p\|_2$ , and so  $(1 - \varepsilon/9)\frac{k}{s}\|x - p\|_2 \leq \mathbf{E}[\|D^s(x - p)\|_1] \leq (1 + \varepsilon/9)\frac{k}{s}\|x - p\|_2$ . Thus, if we assume for instance that  $\|x - p\|_2 > s/2$ , then with probability at least  $1 - n^{-5}$ , the expectation of  $\|D^s(x - p)\|_1$  is at least  $\frac{1-\varepsilon/9}{2}k$ . Conditioning on this event, the conditional probability that  $D^s(x - p) < \frac{1-\varepsilon/3}{2}k$  is bounded by  $e^{-\varepsilon^2 k/162}$ . If we chose the embedding dimension  $k$  to be  $C\varepsilon^{-2} \log n$  for a sufficiently large constant  $C$ , then this probability is also less than  $n^{-5}$ , say. Thus, the probability that  $\|x - p\|_2 > s/2$  and  $D^s(x - p) < \frac{1-\varepsilon/3}{2}k$  is at most  $O(n^{-5})$ . The other two parts of the estimate are proved similarly. Thus, we get a failure probability of  $O(n^{-5})$  for any particular  $p$  and  $s$ . Since there are at most  $n$  choices of  $p$  and at most  $O(n^2 \log n)$  choices of  $s$ , the total failure probability over all  $p$  and  $s$  is at most  $O\left(\frac{n^3 \log n}{n^5}\right) < 1/(2n)$ .  $\square$

### 3.7 Exercises

**Exercise 17.** Let  $x, y \in \mathbb{R}^d$  with  $0 < \|x\|_2 \leq \|y\|_2$ . For a random unit vector  $v \in \mathbf{S}^{d-1}$  (according to the uniform distribution on the sphere), show that  $\Pr[|\langle v, x \rangle| \geq |\langle v, y \rangle|] \leq O\left(\frac{\|x\|_2}{\|y\|_2}\right)$ . (The implicit constant is independent of  $d$ .) (Hint: You may use the following fact: If  $U$  is a fixed  $k$ -dimensional linear subspace of  $\mathbb{R}^d$  and if  $p$  is the orthogonal projection onto  $U$ , then  $p(v) \neq 0$  with probability 1, and  $p(v)/\|p(v)\|_2$  is a random vector from the  $(k - 1)$ -dimensional unit sphere in  $U$ . Use this fact to reduce the problem to a 2-dimensional one.)

**Exercise 18.** Prove Lemma 3.12.

**Exercise 19.** Let  $k \geq 1$  be integer and  $r > 0$ . Show that the number of integer points  $v \in \mathbb{Z}^k$  such that  $\|v\|_1 \leq r$  is at most  $2^k \binom{r+k}{k}$ .

**Exercise 20.** Let  $x, y \in \mathbb{R}^d$  with  $0 < \|x\|_2 \leq \|y\|_2$ . For a random unit vector  $v \in \mathbf{S}^{d-1}$  (according to the uniform distribution on the sphere), show that  $\Pr[|\langle v, x \rangle| \geq |\langle v, y \rangle|] \leq O\left(\frac{\|x\|_2}{\|y\|_2}\right)$ . (The implicit constant is independent of  $d$ .) (Hint: You may use the following fact: If  $U$  is a fixed  $k$ -dimensional linear subspace of  $\mathbb{R}^d$  and if  $p$  is the orthogonal projection onto  $U$ , then  $p(v) \neq 0$  with probability 1, and  $p(v)/\|p(v)\|_2$  is a random vector from the  $(k-1)$ -dimensional unit sphere in  $U$ . Use this fact to reduce the problem to a 2-dimensional one.)

# Bibliography

- [1] N. Ailon and B. Chazelle. Approximate nearest neighbors and the Fast Johnson-Lindenstrauss Transform. In *Proceedings of STOC'06, to appear*, 2006.
- [2] N. Alon and J. H. Spencer. *The probabilistic method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York, second edition, 2000. With an appendix on the life and work of Paul Erdős.
- [3] K. L. Chung. *Elementary probability theory with stochastic processes*. Springer-Verlag, New York, third edition, 1979. Undergraduate Texts in Mathematics.
- [4] P. Indyk. Nearest neighbors in high-dimensional spaces. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, chapter 39*. CRC Press, 2004. 2nd edition.