Chapter 8

Lower Bound Techniques

We have discussed quite a few algorithms for geometric problems, including constructing convex hulls of finite sets of points in the plane, solving proximity problems (e.g., finding the closest pair and the farthest pair, and constructing Euclidean minimum spanning trees), finding the intersection of geometric objects, and searching in PSLGs. Most of these problems can be solved by brute force methods in time $O(n^2)$ or more. Our techniques (geometric sweeping, divide and conquer, prune and search, and reduction) give faster algorithms for solving these problems. Most of our algorithms run in linear time or in time $O(n \log n)$. For those linear time algorithms, we know that we have obtained asymptotically optimal solutions because even just reading the input for the problems takes linear time. For those $O(n \log n)$ time algorithms, however, a very natural question is whether we can further improve them, or, equivalently, are these algorithms the best possible?

This question brings us to an important, deep, and in general difficult branch in theoretical computer science, the study of lower bounds of problems. Here instead of *designing* a *single* efficient algorithm for a given problem, we want to *prove* that *any* algorithm solving the problem takes at least certain amount of time.

Let us look at the problem of constructing convex hulls. We have discussed the relationship between constructing convex hulls and sorting (see Section 7.1), we may have realized that an algorithm faster than $O(n \log n)$ for convex hull is impossible, since as we have seen in Algorithm Analysis that sorting n numbers requires at least $\Omega(n \log n)$ comparisons (see, for example, [2]), and since

$$\text{SORTING} \propto_n \text{CONVEX-HULL},$$

so the problem CONVEX-HULL is at least as hard as SORTING. However, we are not completely satisfied with this result because in the proof for the lower bound for SORTING, the underlying computational model is the comparison trees, which is too restricted and cannot even do multiplication! The above reduction from SORTING to CONVEX-HULL only shows that the problem CONVEX-HULL cannot be solved in time less than $\Omega(n \log n)$ on a computational model that cannot do multiplications. On the other hand, just computing the standard Euclidean distance metric requires quadratic polynomials, which are results of multiplications.

In this chapter, we will introduce a general technique for deriving lower bounds for geometric problems. We first look closely at the computational model that can do only comparisons, the *linear decision trees*, then extend the result on linear decision trees to a more powerful computational model, the *algebraic decision trees.* Lower bounds then are obtained on this model for most of the geometric problems we have discussed in the previous chapters. Combining these lower bound results and the algorithms we have developed for the problems, we conclude that most of those algorithms developed in the previous chapters are in fact optimal.

8.1 Preliminaries

Let us start with a brief review on geometry. We will denote by E^n the n-dimensional Euclidean space, and call it the *n-space.* Let S be a subset of the n-space E^n. The set S is *connected* if for any pair of points p and q of S, there is a path P adjoining p and q such that P is entirely contained in S. By the definition, a convex set in E^n is connected. Now consider a subset W of E^n that is not necessarily connected. A *connected component* of W is a maximal connected subset of W (i.e., no subset of W that is a strict superset of W can be connected). We will denote by $\#W$ the number of connected components of the set W.

A function $f(x_1, \cdots, x_n)$ is a *polynomial* if f is a sum of terms of the form $cx_1^{i_1}x_2^{i_2}\cdots x_n^{i_n}$, where c and all i_j's are constants, and all i_j's are nonnegative integers. The *degree* of the term $cx_1^{i_1}x_2^{i_2}\cdots x_n^{i_n}$ is defined to be the number $i_1 + \cdots + i_n$. The *degree* of a polynomial is the maximum of the degrees of its terms. The function f is a *linear polynomial* if f is a degree-1 polynomial. Let f_L be a linear polynomial. Then the equation $f_L(x_1, \cdots, x_n) = 0$ defines a hyperplane in the n-space E^n; the open inequality $f_L(x_1, \cdots, x_n) > 0$ (or $f_L(x_1, \cdots, x_n) < 0$) defines an *open halfspace* in E^n, with the hyperplane $f(x_1, \cdots, x_n) = 0$ being its boundary; and the closed inequality $f_L(x_1, \cdots, x_n) \geq 0$ (or $f_L(x_1, \cdots, x_n) \leq 0$) defines a *closed halfspace* in E^n, with the hyperplane $f(x_1, \cdots, x_n) = 0$ being its boundary. It is easy to see that hyperplanes, open halfspaces, and closed halfspaces are all convex sets in E^n.

Let S be the set of points $(x_1, \cdots, x_n)$ satisfying a sequence of relations:

$$\begin{aligned} f_i(x_1, \cdots, x_n) &= 0; \quad i = 1, \cdots, m_1, \\ g_j(x_1, \cdots, x_n) &> 0; \quad j = 1, \cdots, m_2, \\ h_k(x_1, \cdots, x_n) &\geq 0; \quad k = 1, \cdots, m_3, \end{aligned}$$

where all functions f_i, g_j, and h_k, with $i = 1, \cdots, m_1$, $j = 1, \cdots, m_2$, and $k = 1, \cdots, m_3$, are linear polynomials. Then the set S is the intersection of the hyperplanes $f_i = 0$, for $1 \leq i \leq m_1$, the open halfspaces $g_j > 0$, for $1 \leq j \leq m_2$, and the closed halfspaces $h_k \geq 0$, for $1 \leq k \leq m_3$. Since hyperplanes, open halfspaces, and closed halfspaces are all convex, by Theorem 3.1.1, the set S is also convex.

A problem is a *decision problem* if it has only two possible solutions, either the answer YES or the answer NO. Abstractly, a decision problem consists simply of a set of *instances* that contains a subset called the set of YES-*instances*. As we have studied in Algorithm Analysis, decision problems play a very important role in the study of NP-completeness theory. In practice, many general problems can be reduced to decision problems such that a general problem and the corresponding decision problem have the same or similar complexity.

There are certain problems where it is realistic to consider the number of branching instructions executed as the primary measure of complexity. In the case of sorting, for example, the outputs are identical to the inputs except for their orders. It thus becomes reasonable to consider a model in which all steps are two-way branches based on a "decision" that we should make when computation reaches that point.

The usual representation for a program of branches is a binary tree called a *decision tree*. Each non-leaf vertex represents a decision. The test represented by the root is made first, and "control" then passes to one of its children, depending on the outcome of the decision. In general, the control continues to pass from a vertex to one of its children, the choice in each case depending on the outcome of the decision at the vertex, until a leaf is reached. The desired output is available at the leaf reached. If the decision at each non-leaf vertex of a decision tree is a comparison of a polynomial of the input variables with the number 0, then the decision tree is called an *algebraic decision tree*.

The algebraic decision tree model may look much weaker than a real computer, but it is probably more powerful than what we would think. First of all, given a computer program, we can always represent the executions of the program on all inputs of a fixed length by a decision tree by "unwinding" loops in the program executions. Secondly, the operations a real computer can perform are essentially additions and branchings. All other operations are in fact done by microprograms that consist of those elementary operations. For example, the value of $\sin(x)$ for a number x is actually obtained by an approximation of the Taylor's extension of the function $\sin(x)$. Finally, we simply ignore the computation instructions and concentrate on only branching instructions because we are working on lower bounds of algorithms. If we can prove that for a problem instance, at least N branchings should be made, then of course, the number of total instructions, including computation instructions and branching instructions, is at least N.

Let us now give a less informal definition. We will concentrate on the decision tree model for decision problems.

Definition An *algebraic decision tree* for a decision problem Q is a binary tree that, on n variables $(x_1, \cdots, x_n)$, which are for all instances of length n of Q, has its vertices labeled with the statements as follows.

1. each non-leaf vertex v is labeled with a statement L of the form

   if $f(x_1, \cdots, x_n) \bowtie 0$ **then** goto w_1 **else** goto w_2,

   where $f(x_1, \cdots, x_n)$ is a polynomial of $(x_1, \cdots, x_n)$, $\bowtie$ is a comparison relation in the set $\{=, >, \geq\}$ (note this also covers the operations "$<$" and "$\leq$"), and w_1 and w_2 are the children of the vertex v in the tree;
2. each leaf vertex is labeled with a statement either YES or NO.

If all polynomials at non-leaf vertices of an algebraic decision tree are linear polynomials, then we call the tree a *linear decision tree*.

Let P be a decision problem whose instances are sets of real numbers. If we fix an input length n, and consider the instants of P that consist of n real numbers. Then P corresponds to a subset W of the n-space E^n such that a point $(x_1, \cdots, x_n) \in E^n$ is in W if and only if the answer of the problem P to the instance $(x_1, \cdots, x_n)$ is YES. Let T be an algebraic decision tree that "solves" the problem P in the following way: for any point $p = (x_1, \cdots, x_n) \in E^n$, the answer of P to the instance p is YES if and only if when we feed the root of the algebraic decision tree T with the input p, then eventually we are led to a YES-leaf v in the tree T by following the decisions made on the non-leaf vertices on the path from the root to the leaf v in the tree T. In this case, we also say that the algebraic decision tree T *accepts* the subset W in E^n.

8.2 Algebraic decision trees

The *depth* of a tree is the length of the longest path from the root to a leaf in the tree. It is easy to see that the depth of an algebraic decision tree corresponds to the worst case time complexity of the tree. Therefore, to derive a lower bound on the worst case time complexity of a problem P, it suffices to derive a lower bound on the depth of the algebraic decision trees that solve the problem P. In this section, we show a lower bound on the depth of an algebraic decision tree, assuming that we know the number of connected components of the corresponding subset in E^n the tree accepts.

We first observe the following simple lemma.

Lemma 8.2.1 *The depth of a binary tree with m leaves is at least $\lceil \log m \rceil$.*

Let P be a decision problem, and let W be the subset of E^n that corresponds to the YES-instances of the problem P in E^n. Thus, a point $p = (x_1, \cdots, x_n)$ in E^n is in the set W if and only if the solution of the problem P to the input p is YES. Let T be an algebraic decision tree that solves P, or equivalently, that accepts the subset W.

Suppose in some way that the number $\#W$ of the connected components of the set W is known. What can we say about the depth of the algebraic decision trees that accept W? We answer this question first for the linear decision tree model, then we extend the result to the generral algebraic decision tree model.

Theorem 8.2.2 *Let W be a subset of the n-space E^n, and let T be a linear decision tree of n variables that accepts the set W. Then the depth of T is at least $\lceil \log(\#W) \rceil$.*

Proof. Every path from the root to a leaf l in T corresponds to a sequence of conditions:

$$\begin{aligned} f_i(x_1, \cdots, x_n) = 0; &\quad i = 1, \cdots, m_1, \\ g_j(x_1, \cdots, x_n) > 0; &\quad j = 1, \cdots, m_2, \\ h_k(x_1, \cdots, x_n) \geq 0; &\quad k = 1, \cdots, m_3, \end{aligned} \tag{8.1}$$

which are the testings occurring on the path. Each of these functions is a linear polynomial since we assume that the tree T is a linear decision tree. If we feed the root with a point $p = (x_1, \cdots, x_n)$ in E^n, then the point p eventually goes to the leaf l if and only if the coordinates $(x_1, \cdots, x_n)$ of p satisfies all the conditions in (8.1). Therefore, the leaf l corresponds to a set S_l of points in E^n that satisfy all the conditions in (8.1). Thus, the set S_l is the intersection of the hyperplanes, the open halfspaces, and the closed halfspaces represented by these conditions. By the discussion we gave in the last section, we conclude that the set S_l is convex. Consequently, S_l is connected.

Now let l be a YES-leaf, then the corresponding set S_l is a subset of the set W. Since S_l is connected, by the definition of a connected component that a connected component of W is a maximal connected subset of W, S_l must be entirely contained in a single connected component of W. Therefore, each YES-leaf of the linear decision tree T only accepts points in a single connected component of W. Since W has $\#W$ connected components, and each point of W should be accepted by some YES-leaf of T, we conclude that the tree T contains at least $\#W$ YES-leaves. Consequently, the number of leaves of T is at least $\#W$. Now by Lemma 8.2.1, the depth of the linear decision tree T is at least $\lceil \log(\#W) \rceil$. $\square$

The linear decision tree model seems too restricted (people would never be happy if you tell them that their computers cannot do multiplications). It is desired to extend the result above for the linear decision tree model to the more general algebraic decision tree model. Let us see what is the obstacle to such an extension. Suppose that an algebraic decision tree T accepts a subset W of E^n. Each YES-leaf l of T accepts a subset S_l of the set W. The subset S_l is again the intersection of the subsets presented by the conditions appearing on the path from the root to the leaf l in the tree T. However, since the polynomials at the non-leaves of T are not necessarily linear polynomials, the set S_l may be *not* connected.[1] Therefore, each YES-leaf now can accept points from (possible many) different connected components of W. Suppose that each YES-leaf can accept points from at most c connected components, then the only thing we can conclude is that there are at least $\#W/c$ YES-leaves. Therefore, by Lemma 8.2.1 again, we conclude that the depth of T is at least $\lceil \log(\#W/c) \rceil$. In the case where the number c is of the same order as $\#W$, we will only obtain a trivial constant lower bound on the depth of the algebraic decision tree T.

[1]For example, in the n-space E^2, even a single condition with a non-linear polynomial $x^2 - y^2 \geq 1$ defines a non-connected area.

On the other hand, if the number c above is bounded by some constant, then $\lceil \log(\#W/c) \rceil$ will have the same order as $\lceil \log(\#W) \rceil$, thus again we obtain a nontrivial lower bound on the depth of the algebraic decision tree T. Therefore, we would like to know under what conditions the number c, i.e., the maximum number of connected components whose points can be accepted by a single leaf of an algebraic decision tree, can be bounded. Here is a condition:

Theorem 8.2.3 (Milnor-Thom) *Let W be the set of points in the n-space E^n defined by the conditions*

$$f_i(x_1, \cdots, x_n) = 0; \qquad i = 1, \cdots, h, \tag{8.2}$$

where all f_i are polynomials of degree bounded by d. Then the number $\#W$ of connected components of the set W is bounded by $d(2d-1)^{n-1}$, a number that is independent of the number of the conditions in (8.2).

The above theorem is a deep result in algebraic geometry. However, the idea of the theorem is fairly intuitive: a polynomial of small degree defines a subset of "simple shape" in a Euclidean space, and the intersection of "simple-shape" subsets in a Euclidean space cannot have a very complicated shape, that is, it cannot have many pieces of connected components.

Unfortunately, Milnor-Thom Theorem cannot be used directly to our algebraic decision trees: it only covers the case of equalities, while our algebraic decision trees also have inequalities. Thus, it is necessary to extend Milnor-Thom Theorem to cover inequalities.

Lemma 8.2.4 *Let W be the set of points in the n-space E^n defined by the following conditions:*

$$\begin{aligned} f_i(x_1, \cdots, x_n) &= 0; & i &= 1, \cdots, m_1, \\ g_j(x_1, \cdots, x_n) &> 0; & j &= 1, \cdots, m_2, \\ h_k(x_1, \cdots, x_n) &\geq 0; & k &= 1, \cdots, m_3, \end{aligned} \tag{8.3}$$

where all f_i, g_j, h_k are polynomials of degree bounded by d. Then the number of connected components of the set W is bounded by $d(2d-1)^{n+m_2+m_3-1}$.

Proof. Suppose that W has r distinct connected components C_i, $1 \leq i \leq r$. Arbitrarily pick a point $p_i = (x_1^{(i)}, \cdots, x_n^{(i)})$ from the connected component C_i, $1 \leq i \leq r$. Consider the rm_2 real numbers

$$g_j(x_1^{(i)}, \cdots, x_n^{(i)}); \qquad 1 \leq j \leq m_2, \ 1 \leq i \leq r.$$

Note that all these rm_2 real numbers are positive since all these points $p_i = (x_1^{(i)}, \cdots, x_n^{(i)})$, $1 \le i \le r$, are in W. Let $\epsilon > 0$ be the smallest real number in these rm_2 real numbers.

Consider the set W' in E^n defined by the following conditions:

$$\begin{aligned} f_i(x_1, \cdots, x_n) &= 0; & i &= 1, \cdots, m_1, \\ g_j(x_1, \cdots, x_n) - \epsilon &\ge 0; & j &= 1, \cdots, m_2 \\ h_k(x_1, \cdots, x_n) &\ge 0; & k &= 1, \cdots, m_3. \end{aligned} \tag{8.4}$$

We claim that the number of connected components of the set W' is at least as large as that of the set W. In fact, the set W' is a subset of the set W since a point satisfying the conditions in (8.4) obviously satisfies the conditions in (8.3). Thus, no two connected components of W can be "merged" into a single connected components of W'. Moreover, no connected components of W can completely disappear in W' – for each connected component C_i of W, at least the chosen point p_i satisfies all conditions in (8.4), by the definition of the number ϵ. Therefore, instead of bounding the number of connected components of the set W, which is defined by the equalities, the open inequalities, and the closed inequalities of (8.3), we can work on the bound of the number of connected components of the set W', which is defined by the equalities and the closed inequalities of (8.4).

The technique of converting a closed inequality into an equality is well-known in linear programming. For the set W' defined by the conditions in (8.4), we introduce $m_2 + m_3$ new variables y_j and z_k, $1 \le j \le m_2$, $1 \le k \le m_3$, and construct the following $m_1 + m_2 + m_3$ conditions with $n + m_2 + m_3$ variables x_i, y_j, z_k, $1 \le i \le n$, $1 \le j \le m_2$, $1 \le k \le m_3$:

$$\begin{aligned} f_i(x_1, \cdots, x_n) &= 0; & i &= 1, \cdots, m_1, \\ g_j(x_1, \cdots, x_n) - \epsilon - y_j^2 &= 0; & j &= 1, \cdots, m_2, \\ h_k(x_1, \cdots, x_n) - z_k^2 &= 0; & k &= 1, \cdots, m_3. \end{aligned} \tag{8.5}$$

Let W'' be the subset of the $(n+m_2+m_3)$-space $E^{n+m_2+m_3}$ that is defined by the conditions in (8.5). It is easy to see that the number $\#W''$ of connected components of W'' is the same as the number $\#W'$ of connected components of W', which is at least as large as the number $\#W$ of connected components of the set W. By Milnor-Thom Theorem, the number $\#W''$ is bounded by $d(2d-1)^{n+m_2+m_3-1}$. Therefore, the number $\#W$ is also bounded by $d(2d-1)^{n+m_2+m_3-1}$. This completes the proof of the lemma. $\Box$

Now similar as the proof for the case of linear decision trees, we can prove a lower bound on the depth of general algebraic decision trees.

Definition An algebraic decision tree is of *bounded-order* if its degree is bounded by a fixed integer d, which is independent of the input size n.

Theorem 8.2.5 (Ben-or's Theorem) *Let W be a subset of the n-space E^n. Then any bounded-order algebraic decision tree T that accepts W has depth at least $\Omega(\log \#W - n)$.*

PROOF. Suppose that T is an order d algebraic decision tree that accepts the set W, where d is a fixed constant independent of n. Let l be a YES-leaf of the tree T that is associated with the following conditions:

$$\begin{aligned} f_i(x_1, \cdots, x_n) &= 0; \quad i = 1, \cdots, m_1, \\ g_j(x_1, \cdots, x_n) &> 0; \quad j = 1, \cdots, m_2, \\ h_k(x_1, \cdots, x_n) &\geq 0; \quad k = 1, \cdots, m_3, \end{aligned} \tag{8.6}$$

where all f_i, g_j, h_k are polynomials of degree bounded by d.

Let W_l be the subset of W that is accepted by the YES-leaf l of T, that is, W_l is the subset of E^n defined by the conditions in (8.6). Since $m_1+m_2+m_3$ is the length of the path from the root of the algebraic decision tree T to the leaf l, $m_1 + m_2 + m_3$ is bounded by the depth h of the tree T.

By Lemma 8.2.4, the number of connected components of the set W_l is bounded by $d(2d-1)^{n+m_2+m_3-1}$, which is bounded by $d(2d-1)^{n+h-1}$, where h is the depth of the tree T. Now since the set W has $\#W$ connected components, and each point of W must be accepted by some YES-leaf of T, we conclude that the algebraic decision tree T has at least $\#W/(d(2d-1)^{n+h-1}$ leaves. By Lemma 8.2.1, the depth h of the tree T is at least

$$\log\left(\frac{\#W}{d(2d-1)^{n+h-1}}\right).$$

From this, we get

$$h \geq \log(\#W) - \log d - (n+h-1)\log(2d-1).$$

Therefore, the depth h of the algebraic decsion tree T is at least

$$\frac{1}{1+\log(2d-1)}(\log(\#W) - n\log(2d-1) + \log((2d-1)/d)).$$

When the number d is a fixed constant, we get $h = \Omega(\log(\#W) - n)$. $\square$

By Theorem 8.2.5, to derive a lower bound on the time complexity of a problem Q, i.e., a lower bound on the depth of the bounded-order algebraic decsion trees that solve the problem Q, we may consider the corresponding set W in the n-space E^n for all n, then compute the number of connected components of the set W. We will use this technique to derive non-trivial lower bounds for several problems.

8.3 Proving lower bounds directly

With the lower bounds on the depth of algebraic decision trees obtained in the last section, now we are ready to derive lower bounds for many problems. Some of these problems are combinatorial problems that are fundamental in combinatorics and closely related to the geometric problems we have studied. The problems we are going to develop lower bounds using Ben-or's Theorem directly include EXTREME-POINTS, ELEMENT-UNIQUENESS, UNIFORM-GAP, and SET-'S DISJOINTNESS.

The basic idea is as follows: given a decision problem Q, we formulate the YES-instances of Q with n parameters into a subset W of the n-space E^n. Then we derive a lower bound B on the number of connected components of the subset W. Now by Ben-or's Theorem, the logarithm of B gives a lower bound on the depth of bounded-order algebraic decision trees that solve the problem Q, that is in consequence a lower bound on the computational time of the algebraic decision trees (thus, the computational time of algorithms) solving the problem Q.

Recall that bounded-order algebraic decision trees are algebraic decision trees whose degree is bounded by a fixed constant that is independent of the input length.

8.3.1 Element uniqueness

We start with a simplest example, the problem of ELEMENT-UNIQUENESS, which is formally defined as follows.

ELEMENT-UNIQUENESS
Input: a set S of n real numbers.
Question: are all numbers in S distinct?

We derive a lower bound for the problem ELEMENT-UNIQUENESS by using Ben-or's Theorem (Theorem 8.2.5) directly.

Theorem 8.3.1 *Any bounded-order algebraic decision tree that solves the problem* ELEMENT-UNIQUENESS *runs in time* $\Omega(n \log n)$.

PROOF. Adopting the standard technique, we first consider the number of connected components of the following set in the n-space E^n:

$$W = \{(x_1, \cdots, x_n) \mid \text{all } x_i\text{'s are distinct}\}.$$

A point $p = (x_1, \cdots, x_n)$ in n-space E^n is a YES-instance of the problem ELEMENT-UNIQUENESS if and only if the point p belongs to the set W.

Fix a point $p = (x_1, \cdots, x_n)$ in n-space E^n in which all x_i's are distinct. Consider the $n!$ points in the n-space E^n obtained by permuting the numbers in $(x_1, \cdots, x_n)$:

$$\sigma(p) = (x_{\sigma(1)}, \cdots, x_{\sigma(n)}); \qquad \sigma \text{ is a permutation of } (1, \cdots, n).$$

Clearly, all these $n!$ points are in the set W. We claim that no two of these $n!$ points are in the same connected component of W. We prove this by contradiction. Suppose that σ and σ' are two different permutations of $(1, \cdots, n)$ and that the points $\sigma(p)$ and $\sigma'(p)$ are in the same connected component of W. Then there is a continuous path P in W connecting $\sigma(p)$ and $\sigma'(p)$. The path P can be written in its parametric form:

$$P(x) = (f_1(x), f_2(x), \ldots, f_n(x)), \quad 0 \le x \le 1,$$

where $f_i(x)$ are continuous functions, $1 \le i \le n$, such that $P(0) = \sigma(p)$ and $P(1) = \sigma'(p)$. That is, we have

$$f_i(0) = x_{\sigma(i)} \quad \text{and} \quad f_i(1) = x_{\sigma'(i)}, \quad \text{for } 1 \le i \le n.$$

Since σ and σ' are different permutations of $(1, \cdots, n)$, we can find an index k such that $x_{\sigma(k)}$ is the smallest number in $\sigma(p)$ such that $x_{\sigma(k)} \neq x_{\sigma'(k)}$. Note that this means $x_{\sigma(k)} < x_{\sigma'(k)}$. Suppose that $x_{\sigma(k)} = x_{\sigma'(h)}$ for some index $h \neq k$, then we also have $x_{\sigma(h)} > x_{\sigma(k)} = x_{\sigma'(h)}$. Thus, we have

$$x_{\sigma(k)} < x_{\sigma(h)} \quad \text{and} \quad x_{\sigma'(k)} > x_{\sigma(k)} = x_{\sigma'(h)}.$$

Since $f_k(x)$ and $f_h(x)$ are continuous functions with

$$f_k(0) = x_{\sigma(k)}, \quad f_k(1) = x_{\sigma'(k)}, \quad f_h(0) = x_{\sigma(h)}, \quad f_h(1) = x_{\sigma'(h)},$$

and from above we have

$$f_k(0) < f_h(0) \quad \text{and} \quad f_k(1) > f_h(1),$$

there must be a real number α in the interval $(0, 1)$ such that $f_k(\alpha) = f_h(\alpha)$. However, by our assumption, the point

$$P(\alpha) = (f_1(\alpha), f_2(\alpha), \ldots, f_n(\alpha))$$

is on the path P that is entirely in the set W, so all numbers in $P(\alpha)$ must be distinct, contradicting the fact $f_k(\alpha) = f_h(\alpha)$. This contradiction proves that there are no two permutations σ and σ' of $(1, 2, \ldots, n)$ such that the points $\sigma(p)$ and $\sigma'(p)$ are in the same connected component of the set W.

Thus the set W has at least $n!$ connected components, i.e., $\#W \geq n!$. Now since

$$n! = 1 \cdot 2 \cdot \cdots \cdot n > \frac{n}{2} \cdot (\frac{n}{2} + 1) \cdot \cdots \cdot n \geq (\frac{n}{2})^{\frac{n}{2}},$$

we have

$$\log(\#W) \geq \log(n!) \geq \log{(\frac{n}{2})}^{\frac{n}{2}} = \frac{n}{2} \log(\frac{n}{2}) = \Omega(n \log n).$$

By Ben-or's Theorem (Theorem 8.2.5), any bounded-order algebraic decision tree that solves the problem ELEMENT-UNIQUENESS runs in time at least

$$\Omega(\log(\#W) - n) = \Omega(n \log n).$$

This completes the proof of the theorem. $\square$

8.3.2 Uniform gap

Let $S = \{x_1, x_2, \ldots, x_n\}$ be a set of real numbers. We say that the set S has a *uniform gap* g if after sorting the set S into a non-decreasing sequence $x_{i_1} \leq x_{i_2} \leq \cdots \leq x_{i_n}$, we have $x_{i_{k+1}} - x_{i_k} = g$ for all k, $1 \leq k \leq n - 1$.

The UNIFORM-GAP problem is formally defined as follows.

UNIFORM-GAP
Input: a set S of n real numbers.
Question: does the set S have a uniform gap that is larger than 0?

Theorem 8.3.2 *Any bounded-order algebraic decision tree that solves the problem* UNIFORM-GAP *runs in time* $\Omega(n \log n)$.

PROOF. The proof is similar to the proof of Theorem 8.3.1.

Consider the following set in the n-space E^n:

$$W = \{(x_1, \cdots, x_n) \mid (x_1, \cdots, x_n) \text{ is a YES-instance of UNIFORM-GAP}\}.$$

Thus a point $(x_1, \cdots, x_n)$ in the n-space E^n is in the set W if and only if there is a permutation σ of $(1, 2, \cdots, n)$ such that $x_{\sigma(i+1)} - x_{\sigma(i)} = g > 0$ for all i, $1 \leq i \leq n-1$, where g is a fixed constant independent of the index i.

Fix a point $p = (x_1, x_2, \cdots, x_n)$ in W such that $x_{i+1} - x_i = g > 0$ for all $1 \leq i \leq n-1$. Consider the $n!$ points in the n-space E^n obtained by permuting $(x_1, x_2, \cdots, x_n)$:

$$\sigma(p) = (x_{\sigma(1)}, \cdots, x_{\sigma(n)}), \qquad \sigma \text{ is a permutation of } (1, 2, \cdots, n).$$

Clearly, all these $n!$ points are in the set W. We claim that no two of these $n!$ points are in the same connected component of W. Assume the contrary that σ and σ' are two different permutations of $(1, 2, \cdots, n)$ and that the points $\sigma(p)$ and $\sigma'(p)$ are in the same connected component of W. Then there is a continuous path

$$P(x) = (f_1(x), f_2(x), \ldots, f_n(x)), \quad 0 \leq x \leq 1,$$

in W connecting the two points $\sigma(p)$ and $\sigma'(p)$, such that $P(0) = \sigma(p)$ and $P(1) = \sigma'(p)$. That is, there are n continuous functions $f_i(x)$, $1 \leq i \leq n$, such that

$$f_i(0) = x_{\sigma(i)} \quad \text{and} \quad f_i(1) = x_{\sigma'(i)}, \quad \text{for } 1 \leq i \leq n.$$

Exactly the same as in the proof of Theorem 8.3.1, we can find two indices k and h such that

$$f_k(0) < f_h(0) \quad \text{and} \quad f_k(1) > f_h(1).$$

So there exists a real number α in the interval $(0, 1)$ such that $f_k(\alpha) = f_h(\alpha)$. But then the point $P(\alpha) = (f_1(\alpha), f_2(\alpha), \cdots, f_n(\alpha))$ on the path $P(x)$ would not be in the set W since the distance between the numbers $f_k(\alpha)$ and $f_h(\alpha)$ is 0 so the set $\{f_1(\alpha), f_2(\alpha), \ldots, f_n(\alpha)\}$ cannot have a positive unform gap. This contradiction proves that the set W has at least $n!$ connected components, i.e., $\#W \geq n!$. By Ben-or's Theorem (Theorem 8.2.5), any bounded-order algebraic decision tree that solves the problem UNIFORM-GAP runs in time

$$\Omega(\log(\#W) - n) = \Omega(n \log n)$$

This completes the proof of the theorem. $\square$

8.3.3 Set disjointness

The third problem we study is the following problem.

SET-DISJOINTNESS
Input: two sets X and Y of n real numbers.
Question: $X \cap Y = \emptyset$?

For an instance (X, Y) of the problem SET-DISJOINTNESS, where we assume $X = \{x_1, x_2, \cdots, x_n\}$ and $Y = \{y_1, y_2, \cdots, y_n\}$, we associate it with a point in the $(2n)$-space E^{2n}:

$$(x_1, y_1, x_2, y_2, \cdots, x_n, y_n).$$

This mapping gives a one-to-one correspondence between the points in E^{2n} and the instances of size n of the problem SET-DISJOINTNESS if we suppose that the elements of the sets X and Y are given in some order (we call (X, Y) an instance of size n if each of the sets X and Y contains n real numbers.) Let W be the subset of E^{2n} that corresponds to the YES-instances of size n of the problem SET-DISJOINTNESS. We prove that W has at least $n!$ connected components.

Fix two sets $X = (x_1, x_2, \cdots, x_n)$ and $Y = (y_1, y_2, \cdots, y_n)$ such that

$$x_1 > y_1 > x_2 > y_2 > \cdots > x_n > y_n.$$

Then (X, Y) is a YES-instance of the problem SET-DISJOINTNESS that corresponds to the point

$$p = (x_1, y_1, x_2, y_2, \cdots, x_n, y_n)$$

in the $(2n)$-space E^{2n}. Thus the point p is in the set W. Consider the $n!$ points in E^{2n} that are obtained by permuting the n components in p with even indices (i.e., the "y-elements" in p). That is, consider the $n!$ points

$$\sigma(p) = (x_1, y_{\sigma(1)}, x_2, y_{\sigma(2)}, \ldots, x_n, y_{\sigma(n)}),$$

where σ is a permutation of $(1, 2, \cdots, n)$.

Clearly, all these $n!$ points $\sigma(p)$ are in the set W. We prove that no two of these $n!$ points are in the same connected component of W. Assume the contrary that σ and σ' are two different permutations of $(1, 2, \cdots, n)$ and that the points $\sigma(p)$ and $\sigma'(p)$ are in the same connected component of W. Then there is a continuous path $P(x)$:

$$P(x) = (h_1(x), f_1(x), h_2(x), f_2(x), \ldots, h_n(x), f_n(x)), \qquad 0 \le x \le 1,$$

in W connecting the two points $\sigma(p)$ and $\sigma'(p)$, i.e., $P(0) = \sigma(p)$ and $P(1) = \sigma'(p)$, where all $h_i(t)$ and $f_i(t)$, $1 \le i \le n$, are continous functions. with

$$\begin{array}{ll} h_i(0) = h_i(1) = x_i; & \text{for } 1 \le i \le n \\ f_i(0) = y_{\sigma(i)} \quad \text{and} \quad f_i(1) = y_{\sigma'(i)}; & \text{for } 1 \le i \le n. \end{array}$$

Since σ and σ' are different permutations of $(1, 2, \cdots, n)$, we can find an index k such that $y_{\sigma(k)}$ is the smallest number in $(y_1, \cdots, y_n)$ such that $y_{\sigma(k)} \neq y_{\sigma'(k)}$. Since $y_{\sigma(k)}$ is the smallest such a number in $(y_1, y_2, \cdots, y_n)$, we have $y_{\sigma(k)} < y_{\sigma'(k)}$. By the definition of the point p, there must be an x_l in $(x_1, x_2, \cdots, x_n)$ such that

$$y_{\sigma(k)} < x_l < y_{\sigma'(k)}.$$

Now consider the function $F(t) = h_l(t) - f_k(t)$, we have

$$F(0) = h_l(0) - f_k(0) = x_l - y_{\sigma(k)} > 0$$

and

$$F(1) = h_l(1) - f_k(1) = x_l - y_{\sigma'(k)} < 0.$$

The function $F(t)$ is continuous because $h_l(t)$ and $f_k(t)$ are. Therefore, there is a real number α such that $0 < \alpha < 1$ and

$$F(\alpha) = h_l(\alpha) - f_k(\alpha) = 0$$

That is, $h_l(\alpha) = f_k(\alpha)$. However, by our assumption, the point

$$P(\alpha) = (h_1(\alpha), f_1(\alpha), h_2(\alpha), f_2(\alpha), \cdots, h_n(\alpha), f_n(\alpha))$$

on the path P is in the set W, so the sets $\{h_1(\alpha), h_2(\alpha), \ldots, h_1(\alpha)\}$ and $\{f_1(\alpha), f_2(\alpha), \ldots, f_1(\alpha)\}$ should be disjoint. In particular, the numbers $h_l(\alpha)$ and $f_k(\alpha)$ should be different. This contradiction proves that for each different permutation σ of $(1, 2, \ldots, n)$, the point $\sigma(p)$ is in a different connected component of the set W.

Thus the set W has at least $n!$ connected components, i.e., $\#W \ge n!$. By Ben-or's Theorem (Theorem 8.2.5), any bounded-order algebraic decision tree that solves the problem Set-Disjointness runs in time at least

$$\Omega(\log(\#W) - n) = \Omega(n \log n).$$

The above discussion gives the following theorem.

Theorem 8.3.3 *Any bounded-order algebraic decision tree that solves the problem* Set-Disjointness *runs in time* $\Omega(n \log n)$.

8.3.4 Extreme points

The above three problems are combinatorial problems. In this subsection, we derive a lower bound for a geometric problem, which is called EXTREME-POINTS problem that is closely related to the problem CONVEX-HULL. The proof is again similar to those given above, though slightly more complicated.

Definition Let S be a set of points in the plane E^2. A point $p \in S$ is an *extreme point* of S if p is on the boundary of the convex hull *CH(S)*, and p is not an interior point of any boundary edge of *CH(S)*.

The CONVEX-HULL problem is to find all extreme points of a given set S in the counterclockwise order with respect to some interior point of *CH(S)*. The following decision problem has an obvious relationship with the CONVEX-HULL problem.

EXTREME-POINTS
Input: a set S of n points in the plane E^2.
Question: are all points in S extreme points of S?

The EXTREME-POINTS problem seems "simpler" than the CONVEX-HULL problem since it is not required to check the counterclockwise order of the extreme points on the boundary of the convex hull *CH(S)*. We will see, however, that it takes the same amount of time to solve the EXTREME-POINTS problem as to solve the CONVEX-HULL problem.

A point p in the plane E^2 can be uniquely represented by an ordered pair of two real numbers $p = (x, y)$, where x and y are the x- and y- coordinates of p, respectively. Similarly, an ordered list of $2n$ points $(p_1, p_2, \cdots, p_{2n})$ in the plane E^2 can be uniquely represented by a tuple of $4n$ real numbers $(p_1, p_2, \cdots, p_{2n}) = (x_1, x_2, x_3, x_4, \cdots, x_{4n})$, where $p_i = (x_{2i-1}, x_{2i})$, for $1 \le i \le 2n$. Therefore, each $(2n)$-point instance $(p_1, p_2, \cdots, p_{2n})$ for the EXTREME-POINTS problem uniquely corresponds to a point in the $(4n)$-space E^{4n}. Conversely, any point $(x_1, x_2, \cdots, x_{4n})$ in the $(4n)$-space E^{4n} can be interpreted uniquely as a $(2n)$-point instance $(p_1, p_2, \cdots, p_{2n})$ for the EXTREME-POINTSS problem, if we let $p_i = (x_{2i-1}, x_{2i})$, for $1 \le i \le 2n$. As a result, the set of $(2n)$-point YES-instances of the EXTREME-POINTS problem is a subset of the $(4n)$-space E^{4n}. Note that a set $S = \{p_1, p_2, \cdots, p_{2n}\}$ of $2n$ points in the plane E^2 can correspond to up to $(2n)!$ different ordered lists, thus $(2n)!$ different points in the $(4n)$-space E^{4n}, if we consider all permutations of the $2n$ points in S. Thus any set of $2n$ points in the plane makes $(2n)!$ different instances for the EXTREME-POINTS problem.

Lemma 8.3.4 *Let W be the subset of the $(4n)$-space E^{4n} that corresponds to the set of $(2n)$-point* YES*-instances for the* EXTREME-POINTS *problem. Then W has at least $n!$ connected components.*

PROOF. We construct $n!$ points in the set W and prove that no two of these points are in the same connected component of the set W.

Let $I = (p_1, q_1, p_2, q_2, \cdots, p_n, q_n)$ be a counterclockwise sequence of $2n$ distinct extreme points of a convex polygon. Then $I \in E^{4n}$ is a point in the set W. Consider the following $n!$ different sequences of $(2n)$-point instances of the problem EXTREME-POINTS:

$$\begin{gathered}\sigma(I) = (p_1, q_{\sigma(1)}, p_2, q_{\sigma(2)}, \cdots, p_n, q_{\sigma(n)}); \\ \sigma \text{ is a permutation of } (1, \ldots, n),\end{gathered} \tag{8.7}$$

where each sequence $(q_{\sigma(1)}, q_{\sigma(2)}, \cdots, q_{\sigma(n)})$ is a permutation of the sequence $(q_1, q_2, \cdots, q_n)$. Each instance $\sigma(I)$ corresponds to a point in the $(4n)$-space E^{4n}. Since the instances $\sigma(I)$ share the same set of $2n$ points $\{p_1, q_1, \cdots, p_n, q_n\}$ in the plane E^2 with the instance I and I is a YES-instance of the problem EXTREME-POINTS, the instance $\sigma(I)$ for each σ of the $n!$ permutations of $(1, 2, \ldots, n)$ is also a YES-instance of the problem EXTREME-POINTS (i.e., every point in $\sigma(I)$ is an extreme point of the point set $\sigma(I)$). As a result, the instances $\sigma(I)$ for the $n!$ permutations of $(1, 2, \ldots, n)$ correspond to $n!$ points in the set W.

We prove that any pair of instances in (8.7) are in two different connected components of the set W. Suppose the contrary that there are two instances $\sigma(I)$ and $\sigma'(I)$ in (8.7) that are two points in E^{4n} and are in the same connected component of the set W. Then there is a continuous path $P(x) = (h_1(x), f_1(x), \ldots, h_n(x), f_n(x))$ in W that adjoins the two points $\sigma(I)$ and $\sigma'(I)$. More precisely, there are $2n$ continuous functions on the interval $[0, 1]$ that map the real numbers in $[0, 1]$ to E^2 (recall that each of the functions h_i and f_j gives a point in E^2):

$$h_1(x), f_1(x), h_2(x), f_2(x), \cdots, h_n(x), f_n(x),$$

such that $h_k(0) = h_k(1) = p_k$, and $f_k(0) = q_{\sigma(k)}$ and $f_k(1) = q_{\sigma'(k)}$, for $k = 1, 2, \cdots, n$, and, for all $x \in [0, 1]$, the point

$$P(x) = (h_1(x), f_1(x), h_2(x), f_2(x), \cdots, h_n(x), f_n(x))$$

is in the subset W of E^{4n}, and $P(0) = \sigma(I)$ and $P(1) = \sigma'(I)$.

Since $\sigma(I)$ and $\sigma'(I)$ are two different instances in (8.7), there must be an index k such that $q_{\sigma(k)}$ and $q_{\sigma'(k)}$ are different points in the set

$\{q_1, q_2, \cdots, q_n\}$. Pick k such that $\sigma(k)$ is the smallest index such that $q_{\sigma(k)} \neq q_{\sigma'(k)}$, and let $\sigma(k) = i$. Then $i < \sigma'(k) \leq n$. Consider the triangle $\triangle(p_i q_i p_{i+1}) = \triangle(p_i q_{\sigma(k)} p_{i+1})$, whose three points are three consecutive extreme points in the counterclockwise order of the convex polygon $I = (p_1, q_1, p_2, q_2, \cdots, p_n, q_n)$. Thus, the turn from p_i to $q_{\sigma(k)} = q_i$ to p_{i+1} is a left turn, so the signed area $\text{Area}(p_i q_{\sigma(k)} p_{i+1}) = \text{Area}(h_i(0) f_i(0) h_{i+1}(0))$ of the triangle $\triangle(p_i q_{\sigma(k)} p_{i+1})$ is positive. On the other hand, since we have $\sigma'(k) \neq i$, the turn from p_i to $q_{\sigma'(k)}$ to p_{i+1} is a right turn so the signed area $\text{Area}(p_i q_{\sigma'(k)} p_{i+1}) = \text{Area}(h_i(1) f_i(1) h_{i+1}(1))$ of the triangle $\triangle(p_i q_{\sigma'(k)} p_{i+1})$ is negative.

Since $h_i(x)$, $f_i(x)$, $h_{i+1}(x)$ are all continuous functions of the variable x, the signed area $\text{Area}(h_i(x) f_i(x) h_{i+1}(x))$ of the triangle $\triangle(h_1(x) f_i(x) h_2(x))$ is also a continuous function of the variable x. This, combined with the facts $\text{Area}(h_i(0) f_i(0) h_{i+1}(0)) > 0$ and $\text{Area}(h_i(1) f_i(1) h_{i+1}(1)) < 0$, implies that there is a real number $\alpha \in (0, 1)$ such that $\text{Area}(h_i(\alpha) f_i(\alpha) h_{i+1}(\alpha)) = 0$, i.e., the three points $h_i(\alpha)$, $f_i(\alpha)$, and $h_{i+1}(\alpha)$ are co-linear. That is, on the path $P(x)$, there is a point

$$P(\alpha) = (h_1(\alpha), f_1(\alpha), h_2(\alpha), f_2(\alpha), \cdots, h_n(\alpha), f_n(\alpha)),$$

where $0 < \alpha < 1$, in which three points $h_i(\alpha)$, $f_i(\alpha)$, and $h_{i+1}(\alpha)$ in the plane E^2 are co-linear. It is easy to see that for three co-linear points, at least one of the points cannot be an extreme point of the set $P(\alpha)$. Thus, the point $P(\alpha)$ in the $(4n)$-space E^{4n} is not a YES-instance of the EXTREME-POINT, thus is not in the set W, contradicting the fact that the entire path $P(x)$, with $0 \leq x \leq 1$, is contained in the set W.

The contradiction proves that no two points in (8.7) can be in the same connected component of the set W. Since there are $n!$ different points in (8.7), we conclude that the set W has at least $n!$ connected components. □

Now combining Lemma 8.3.4 with Theorem 8.2.5, we easily obtain a lower bound for the problem EXTREME-POINTS.

Theorem 8.3.5 *Any bounded-order algebraic decision tree that solves the problem* EXTREME-POINTS *runs in time* $\Omega(n \log n)$ *on an input of* n *points in the plane.*

PROOF. Since we are working on the worst-case time complexity, we only need to show that for some integers n, the theorem holds true.

Let $n = 2m$, and consider the EXTREME-POINTS problem for instances consisting of n points in E^2, which correspond to points in the $(4m)$-space

E^{4m}. By Lemma 8.3.4, the subset W of E^{4m} that corresponds to the YES-instances of EXTREME-POINTS on $2m$ points has at least $m!$ connected components, i.e., $\#W \geq m!$. By Theorem 8.2.5, the depth of a bounded-order algebraic decision tree solving EXTREME-POINTS is $\Omega(\log(\#W) - 4m)$. From

$$m! = 1 \cdot 2 \cdots m > \frac{m}{2} \cdot \left(\frac{m}{2} + 1\right)\left(\frac{m}{2} + 2\right) \cdots m > \left(\frac{m}{2}\right)^{\frac{m}{2}},$$

we derive

$$\log(\#W) \geq \log(m!) \geq \log\left(\left(\frac{m}{2}\right)^{\frac{m}{2}}\right) = \frac{m}{2} \log\left(\frac{m}{2}\right).$$

Now the depth of the algebraic decision tree is $\Omega(\log(\#W) - 4m) = \Omega(m \log m) = \Omega(n \log n)$. $\square$

8.4 Deriving lower bounds by reductions

The techniques used in the last section for deriving lower bounds on problems seem impressive. Such elegant techniques were developed and such deep mathematics results were used in deriving the lower bounds. It is not clear how these techniques can be generalized to deriving lower bounds for general geometric problems. Fortunately, we do not have to do this very often. For some geometric problems, the lower bounds can be derived by "reducing" the problems to some other problems for which the lower bounds are known.

Let us first review the concept of problem reductions. We say that a problem P can be *reduced* to a problem P' in time $O(t(n))$, expressed as $P \propto_{t(n)} P'$, if there is an algorithm $\mathcal{T}$ solving the problem P as follows:

1. for an instance x of size n for the problem P, convert x in time $O(t(n))$ into an instance x' to the problem P';
2. call a subroutine to solve the problem P' on instance x';
3. convert in time $O(t(n))$ the solution to the problem P' on instance x' into a solution to the problem P on instance x.

We have seen in Chapter 7 that the technique of reductions is very useful in designing efficient algorithms for geometric problems. In this section, we will study how to use this technique to derive lower bounds for geometric problems. The following theorem plays an important role in our discussion.

Theorem 8.4.1 *Assuming $Q \propto_n Q'$. If solving the problem Q takes time $\Omega(n \log n)$, then solving the problem Q' also takes time $\Omega(n \log n)$.*

PROOF. Suppose the contrary that the problem Q' can be solved in time $T(n) = o(n \log n)$. Note that for any constant $c > 0$, we have $(cn)\log(cn) = O(n \log n)$. Therefore, $T(cn) = o((cn)\log(O(cn))) = o(n \log n)$, i.e., $T(O(n)) = o(n \log n)$. By Lemma 7.0.1, then the problem Q can be solved in time $O(n+T(O(n))) = o(n \log n)$. But this would contradict the assumption that the problem Q takes time $\Omega(n \log n)$. This contraction shows that the problem Q' takes time $\Omega(n \log n)$. □

In other words, Theorem 8.4.1 claims that under the condition $Q \propto_n Q'$, the lower bound $\Omega(n \log n)$ on the time complexity of the problem Q gives a lower bound $\Omega(n \log n)$ on the time complexity of the problem Q'.

We first use Theorem 8.4.1 to derive a lower bounds for the CONVEX-HULL problem.

Theorem 8.4.2 *Any bounded-order algebraic decision tree that constructs the convex hull for a set of points in the plane takes time $\Omega(n \log n)$ on an input of n points in the plane.*

PROOF. By Theorem 8.3.5, any bounded-order algebraic decision tree that solves the problem EXTREME-POINTS runs in time $\Omega(n \log n)$. According to Theorem 8.4.1, it will suffice to prove the theorem by showing

$$\text{EXTREME-POINTS} \propto_n \text{CONVEX-HULL}.$$

We give this reduction by the following algorithm.

```
Algorithm Reduction-1
\\ Reduce Extreme-Points to Convex-Hull.
1. on an instance S of Extreme-Points, where S is a set of n
   points in the plane, pass S to Convex-Hull;
2. the solution of Convex-Hull to the set S is the convex hull
   CH(S) of S. Pass CH(S) back to Extreme-Points;
3. if CH(S) has n hull vertices, the answer of Extreme-Points
   to the instance S is YES; Otherwise, the answer is NO.
```

Since both Step 1 and Step 3 take at most time $O(n)$, the above algorithm is a linear time reduction of the problem EXTREME-POINTS to the problem CONVEX-HULL. □

Thus, constructing the convex hulls of a set of n points in the plane takes time $\Omega(n \log n)$. This result implies that many algorithms we have discussed before for constructions of convex hulls, including `Graham Scan`, `MergeHull`, and `Kirkpatrick-Seidel`, are optimal.

As we have discussed in the last chapter, the problem Convex-Hull can be reduced to the problem Sorting in time $O(n)$. By Theorem 8.4.1 and Theorem 8.4.2, we obtain

Theorem 8.4.3 *Any bounded-order algebraic decision tree sorting n real numbers runs in time $\Omega(n \log n)$.*

Theorem 8.4.3 is a result stronger than the one we got in Algorithm Analysis. In Algorithm Analysis, it is proved that a linear-decision tree model that sorts runs in time $\Omega(n \log n)$. On the other hand, Theorem 8.4.3 claims that even the computation model is allowed to do multiplication, it still needs $\Omega(n \log n)$ time to sort.

We have seen many proximity problems solvable in time $O(n \log n)$. Now we prove that these algorithms for the problems are optimal.

We start with two problems whose lower bounds are easily obtained from the problem Sorting: Euclidean Minimum-Spanning-Tree (Euclidean-MST) and Triangulation. For this, we first prove a simple lemma.

Lemma 8.4.4 *Let S be a set of n real numbers x_1, x_2, $\cdots$, x_n. If S is given in such a way that for each $1 \leq k \leq n$, the number x_k is companied by an index i_k such that x_{i_k} is the smallest number in S that is larger than x_k. Then the set S can be sorted in linear time.*

Proof. To sort the set S, we first scan the set S to find the minimum number x_{k_1} in S. Since x_{k_1} is companied by an index $k_2 = i_{k_1}$ such that x_{k_2} is the smallest number in S that is larger than x_{k_1}, x_{k_2} must be the second smallest number in S. Moreover, since we know the index k_2, we can get x_{k_2} and put it immediately after x_{k_1} in constant time. In general, suppose we have obtained x_{k_i} that is the i-th smallest number in S. Then since x_{k_i} is companied by an index $k_{i+1} = i_{k_i}$ such that $x_{k_{i+1}}$ is the smallest number in S that is larger than x_{k_i}, $x_{k_{i+1}}$ is the $(i+1)$-st smallest number in S, and we can get $x_{k_{i+1}}$ and put it immediately after x_{k_i} in constant time. It is clear that after $n-1$ such iterations, we reach the largest number in S and obtain a sorted list of the numbers in S. Since each iteration takes only constant time, we conclude that the set S is sorted in linear time. $\square$

We now consider the problem Euclidean-MST.

Lemma 8.4.5 SORTING *can be reduced to* EUCLIDEAN-MST *in linear time.*

PROOF. Given a set S of n real numbers $x_1, x_2, \cdots, x_n$, which is an instance of the problem SORTING, we construct an instance S' of the EUCLIDEAN-MST problem that is the set of n points in the plane:

$$(x_1, 0), (x_2, 0), \cdots, (x_n, 0).$$

Moreover, for each $1 \leq i \leq n$, we attach an index i to the point $(x_i, 0)$. It is easy to see that the (unique) solution to the EUCLIDEAN-MST problem on the instance S' is a chain T of $n-1$ segments in the plane, such that a segment $\overline{(x_i, 0)(x_j, 0)}$ is in T if and only if the number x_j is the smallest number in S that is larger than x_i.

Pass the chain T back to SORTING. For each segment $\overline{(x_i, 0)(x_j, 0)}$ in T, construct a pair (x_i, j) (remember that the index j is attached to the point $(x_j, 0)$). Using these pairs, by Lemma 8.4.4, we can construct the sorted list of S in linear time. This proves SORTING $\propto_n$ EUCLIDEAN-MST. □

This Lemma, together with Theorem 8.4.3 and Theorem 8.4.1 gives the following theorem.

Theorem 8.4.6 *Any bounded-order algebraic decision tree that constructs the Euclidean minimum spanning tree for a set of n points in the plane runs in time $\Omega(n \log n)$.*

Therefore, the algorithm presented in Section 7.4 that constructs the Euclidean minimum spanning tree for sets of points in the plane is optimal.

Now we consider the problem TRIANGULATION.

Lemma 8.4.7 SORTING *can be reduced to* TRIANGULATION *in linear time.*

PROOF. The proof is similar to that of Lemma 8.4.5. Given a set S of n real numbers $x_1, x_2, \cdots, x_n$, we construct a set S' of $n+1$ points in the plane

$$q = (x_1, 2), p_1 = (x_1, 0), p_2 = (x_2, 0), \cdots, p_n = (x_n, 0).$$

It is easy to see that the set S' has a unique triangulation that consists of the n segments $\overline{qp_i}$ for $1 \leq i \leq n$, plus the $n-1$ segments $\overline{p_i p_j}$ where the number x_j is the smallest number in S that is larger than x_i.

Now using the similar argument as the one we presented in the proof of Lemma 8.4.5, we conclude that we can construct the sorted list of S from the triangulation of S' in linear time. □

Theorem 8.4.8 *Any bounded-order algebraic decision tree that constructs the triangulation for a set of n points in the plane runs in time $\Omega(n \log n)$.*

Thus the problem Triangulation also has an optimal algorithm, which was presented in Section 7.3.

A simple generalization of the problem Triangulation is the problem Constrained-Triangulation, as introduced in Section 4.4, which on a given PSLG G, constructs a triangulation of the vertices of G that uses all edges in G. A lower bound for the Constrained-Triangulation can be obtained from the lower bound of Triangulation.

Theorem 8.4.9 *Any bounded-order algebraic decision tree solving the problem* Constrained-Triangulation *runs in time $\Omega(n \log n)$.*

Proof. It is easy to prove that

$$\text{Triangulation} \propto_n \text{Constrained-Triangulation}.$$

In fact, every instance of the problem Triangulation, which is a set S of n points in the plane, is an instance $G = (S, \phi)$ of the problem Constrained-Triangulation, in which the PSLG G has an empty set of edges.

Since the problem Triangulation has a lower bound $\Omega(n \log n)$, by Theorem 8.4.1, the problem Constrained-Triangulation has a lower bound $\Omega(n \log n)$. □

One may observe that the reductions we have used in Lemma 8.4.5 and Lemma 8.4.7 reduce the instances of Sorting to "degenerated" instances of the geometric problems Euclidean-MST and Triangulation, respectively, in which we allow three or more points to be co-linear, while in our developments of algorithms for the problems, we assumed non-degenerated instances. The degeneration can in fact be avoided if we use higher-degree curves instead of straight lines. For example, in the reduction from Sorting to Euclidean-MST, instead of mapping the real numbers in the instance $S = \{x_1, x_2, \ldots, x_n\}$ of Sorting to the points in the x-axis, we can map the numbers to the parabola $y = x^2$ by $x_i \to (x_i, x_i^2)$ (we can assume that all numbers in S are positive – otherwise, we add a sufficiently large constant to every number in S). It is not difficult to see that for this set of constructed points, no three points are co-linear and no four points are co-circular, i.e., the constructed instances are non-degenerated for the geometric problem. On the other hand, it is also easy to verify that the Euclidean minimum spanning tree of the point set is a polygonal path from which a sorted list of the numbers in S can be obtained in linear time.

To derive lower bounds for the problems CLOSEST-PAIR and ALL-NEAREST-NEIGHBORS, we use the lower bound for the problem ELEMENT-UNIQUENESS, as derived in the last section.

Theorem 8.4.10 *Any bounded-order algebraic decision tree finding the closest pair for a set of n points in the plane runs in time $\Omega(n \log n)$.*

PROOF. We prove ELEMENT-UNIQUENESS $\propto_n$ CLOSEST-PAIR.

For an instance $S = \{x_1, x_2, \ldots, x_n\}$ of ELEMENT-UNIQUENESS, which is a set of n real numbers, we construct an instance for CLOSEST-PAIR:

$$(x_1, 0), (x_2, 0), \cdots, (x_n, 0),$$

which is a set S' of n points in the plane. Clearly, all elements of S are distinct if and only if the closest pair in S' does not consist of two identical points. So the problem ELEMENT-UNIQUENESS is reducible to the problem CLOSEST-PAIR in linear time. Now the theorem follows from Theorem 8.3.1 and Theorem 8.4.1. □

Since it is straightforward to verify that

$$\text{CLOSEST-PAIR} \propto_n \text{ALL-NEAREST-NEIGHBORS},$$

by Theorem 8.4.10 and Theorem 8.4.1, we obtain the following theorem.

Theorem 8.4.11 *Any bounded-order algebraic decision tree finding the nearest neighbor for each point of a set of n points in the plane runs in time $\Omega(n \log n)$.*

Thus, the algorithms we derived in Section 7.2 for the problems CLOSEST-PAIR and ALL-NEAREST-NEIGHBORS are optimal.

To discuss the lower bound on the time complexity of the problem MAXIMUM-EMPTY-CIRCLE, we use the $\Omega(n \log n)$ lower bound for the problem UNIFORM-GAP derived in the last section.

Theorem 8.4.12 *Any bounded-order algebraic decision tree that constructs a maximum empty circle for a set of n planar points runs in time $\Omega(n \log n)$.*

PROOF. We show that

$$\text{UNIFORM-GAP} \propto_n \text{MAXIMUM-EMPTY-CIRCLE}.$$

On a set $S = \{x_1, x_2, \cdots, x_n\}$ of n real numbers, which is an instance of the problem UNIFORM-GAP, if all numbers in S are the same (this can be easily verified in linear time), then we can directly conclude that S is a NO-instance of UNIFORM-GAP. Otherwise, we let $g = (x_{\max} - x_{\min})/(n - 1)$, where $x_{\max}$ and $x_{\min}$ are the largest and the smallest numbers in S, respectively and construct a set S' of n points in the plane

$$S' = \{(x_1, 0), (x_2, 0), \cdots, (x_n, 0)\},$$

which is an instance of the problem MAXIMUM-EMPTY-CIRCLE. Note that the number g and the point set S' can be constructed from the given number set S in linear time.

Note that the diameter d of the maximum empty circle of S', which is part of the solution of MAXIMUM-EMPTY-CIRCLE to the instance S', is the largest difference between two consecutive numbers in a sorted sequence of the set S. It is easy to see that S is a YES-instance of UNIFORM-GAP if and only if $d = g$. Therefore, given the diameter d of the maximum empty circle of the point set S', a correct solution of UNIFORM-GAP to the instance S can be obtained directly. This proves that UNIFORM-GAP is reducible to MAXIMUM-EMPTY-CIRCLE in linear time.

By Theorems 8.3.2 and 8.4.1, we derive the lower bound $\Omega(n \log n)$ for the time complexity of the problem MAXIMUM-EMPTY-CIRCLE. □

Therefore, the algorithm in Section 7.5 for finding the maximum empty circle given a set of points in the plane is optimal.

We have presented an $O(n \log n)$ time algorithm for the FARTHEST-PAIR problem in Section 4.3. We now prove that this algorithm is optimal by showing a lower bound on the time complexity of the problem. For this, we make use of the $\Omega(n \log n)$ lower bound for the problem SET-DISJOINTNESS.

Theorem 8.4.13 *Any bounded-order algebraic decision tree that solves the problem* FARTHEST-PAIR *runs in time* $\Omega(n \log n)$.

PROOF. We prove SET-DISJOINTNESS $\propto_n$ FARTHEST-PAIR.

Given an instance $I = (X, Y)$ of the problem SET-DISJOINTNESS, where X and Y are sets of n real numbers, we transform I into an instance of FARTHEST-PAIR as follows. Without loss of generality, suppose that all numbers in X and Y are positive (otherwise, we scan the sets X and Y to find the smallest number z in $X \cup Y$, then add the number $-z + 1$ to each number in X and in Y). Now find the largest number $z_{\max}$ in $X \cup Y$. Convert each number x_i in the set X into a point on the unit circle in the plane that has a polar angle $x_i\pi/z_{\max}$, and convert each number y_j in the set Y into a point on the unit circle in the plane that has a polar angle $y_j\pi/z_{\max} + \pi$. Intuitively, we transform all numbers in the set X into points in the first and second quadrants of the unit circle in the plane, while transform all numbers in the set Y into points in the third and fourth quadrants of the unit circle. Such a transformation gives a set S of $2n$ points in the plane. It is easy to see that the diameter of S is 2 if and only if the intersection of X and Y is not empty. This proves that the problem SET-DISJOINTNESS can be reduced to the problem FARTHEST-PAIR in linear time.

By Theorem 8.3.3, the problem SET-DISJOINTNESS has a lower bound $\Omega(n \log n)$. By Theorem 8.4.1, the problem FARTHEST-PAIR also has a lower bound $\Omega(n \log n)$ on its time complexity. □

8.5 A remark on our model

We have successfully developed lower bounds for a number of geometric problems on the model of algebraic decision trees. These lower bound results show that the algorithms we developed for these problems are optimal, giving very satisfactory conclusions to the computational complexity for these problems.

In this section, we give an observation that is somehow surprising, showing that some of the lower bounds we developed can be broken. Consider the following algorithm for the problem UNIFORM-GAP: [2].

```
Algorithm Dubbs-III
Input: A set S = {x_1, x_2, ..., x_n} of n real numbers.
Question: does S have a uniform gap that is larger than 0?
\\  A below is an array A[1..n], which is initialized to empty

1. find the min number x_min and the max number x_max in S;
2. if (x_min = x_max) return("no");
3. g = (x_max - x_min)/(n-1);
4. for i = 1 to n do
4.1   k = (x_i - x_min)/g;
4.2   if (k is not an integer or A[k+1] is not empty)
          return("no");
4.3   else A[k+1] = x_i;
5.  return("yes").
```

[2]The author was informed of this algorithm by Roger B. Dubbs III.

This algorithm obviously runs in linear time. To see the correctness, first observe that if $x_{\min} = x_{\max}$ in step 2, then all numbers in S are the same and the uniform gap is 0 so S is a NO-instance and step 2 returns correctly. Otherwise, if a number x_i is placed in $A[k]$, then the value of x_i must be $x_{\min} + g(k-1)$. Moreover, no array element of A holds more than one numbers in S. As a result, if the algorithm returns at step 5, then every array element of A holds exactly one number from the set S, and these numbers are $x_{\min} + g \cdot (k-1)$, for $k = 1, 2, \cdots, n$. Therefore, the set S must be a YES-instance of the problem UNIFORM-GAP.

For the other direction, if the algorithm returns at step 4.2, then either S is not uniformly distributed (otherwise all values $(x_i - x_{\min})/g$ should be integral) or the set S contains two identical numbers. In either case, the set S cannot be a YES-instance of the problem UNIFORM-GAP.

This example brings up an interesting point: there are certain very common operations not included in the algebraic decision tree model that allow to do things that are not possible in the algebraic decision tree model. In the algorithm `Dubbs-III`, such an operation is the testing of integral for the number k in step 4.2. There are also other such operations, including the functions that take the floor or the ceiling of a real number. Other problems that have an $\Omega(n \log n)$ lower bound but can be solved in linear time with certain "non-algebraic" operations were also discovered. These examples imply that these non-algebraic operations, such as the floor and ceiling functions and the integral testing cannot be performed in constant time in the algebraic decision tree model.