A Method for Proving Lower Bounds For Certain Geometric Problems

Raimund Seidel

TR 84-592
February 1984

Department of Computer Science
Cornell University
Ithaca, New York 14853

A METHOD FOR PROVING LOWER BOUNDS FOR CERTAIN GEOMETRIC PROBLEMS

by

Raimund Seidel

Department of Computer Science
Cornell University
Ithaca NY. 14853

Abstract:

We prove lower bounds for a number of geometric problems. Our results show that certain types of additional input information cannot possibly permit faster solutions for these problems. The main idea in all our lower bound proofs is the use of the index of a point as an additional coordinate.

I. Introduction

Most of the research effort in the field of computational geometry goes into the design of algorithms for the efficient solution of geometric problems. Often several algorithms are known for particular problems and the question of comparing their respective efficiency arises. The so-called asymptotic worst case time complexity is one of the favorite measures for this comparison.

Whenever algorithms for a particular problem are compared and one of them is found to be best, the question naturally arises whether this algorithm can still be improved upon or whether the problem is intrinsically so difficult that there is no substantially better way of solving it in the given computational framework. This is the point where the computational geometer has to change his viewpoint entirely because instead of looking for (geometric) properties of the given problem that facilitate an easy and efficient solution he must search for "bad" properties that will force a certain amount of inefficiency on any algorithm that purports to solve the problem.

This lower bound question has been answered for a number of geometric problems such as the point uniqueness problem [B-O], different versions of the convex hull problem [Sha],[Yao], [K-S], the Voronoi Diagram [Sha],[Pas],[S82], certain intersection problems [Swa], and a number of searching problems [Fre1],[Fre2]. The lower bounds given in this paper are of a special kind. We look at problems for which in the general case a lower bound is known that is also matched by an optimal algorithm and we ask whether this lower bound is still applicable, if additional input information is included in the problem definition. It is well known, for instance, that the construction of the convex hull of a set S of n points in $\mathbf{R}^3$ requires $\Omega(n \log n)$ time in the worst case. But it is not clear whether this problem becomes easier if the order of the points in S with respect to each of the three coordinate directions is known. The answer to this question turns out to be negative. Similarly we show in this paper that if is not easier to find the convex hull of S even if the points of S are the vertices of a star-shaped polytope, or we prove that having n points in the plane arranged as a triangulation does in general not aid in locating every one of these points in a planar subdivision.

All the lower bounds in this paper are proved by reduction, i.e. we show that a fast solution for the problem of interest would yield a fast solution for some other problem and would thus contradict a known lower bound for this other problem. The unifying tool in all these proofs is the use of the index of a point as an additional coordinate.

As our underlying model of computation we assume the RAM with real arithmetic and unit cost per operation. Throughout the paper $x(p)$ and $y(p)$ denote the x and y-coordinate of a point p, respectively, and $\langle a,b \rangle$ denotes the ordinary scalar product between two vectors a and b.

II. Sorting Triangulated Points

It is well known that any algorithm that sorts a set $A = \{x_1, \ldots, x_n\}$ of n reals by the use of comparisons needs $\Omega(n \log n)$ time for some input set A. This implies immediately that the problem of sorting a set $S = \{p_1, \ldots, p_n\}$ of n points in the plane with respect to their x-coordinate has an $\Omega(n \log n)$ lower bound also. After all, any algorithm that sorts S in such a way could be used to sort the set of points $\{(x_1,0), \ldots, (x_n,0)\}$ and would thus produce the order of A. Now the question arises, what if additional information about the set S is available. What happens if we know a triangulation of S? By a *triangulation* $D(S)$ of S we mean a maximal set E of unordered pairs $\{p_i,p_j\} \subset S$, $i \neq j$, called *edges*, such that for each pair of edges $\{p_i,p_j\}$, $\{p_k,p_m\} \in E$ the open straight line segments joining p_i and p_j, and p_k and p_m, respectively, do not intersect. Figure 2.1 shows a point set and two of its triangulations.

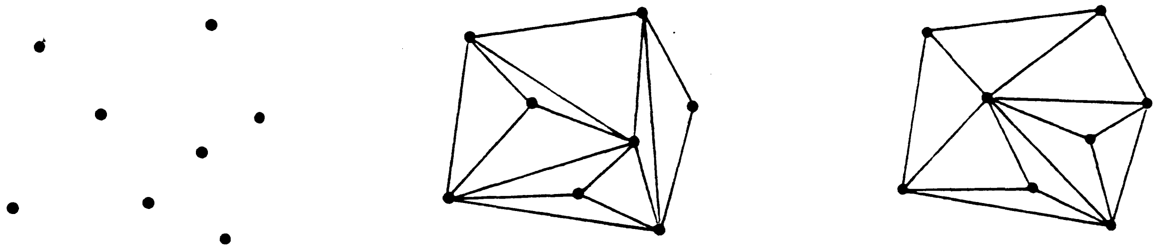

Figure 2.1

Thus a triangulation can be viewed as a planar graph whose vertices are the points of the given set. In the following we will therefore assume that triangulations are described by typical planar graph representations, such as cyclically ordered adjacency lists, or the quad edge data structure of [G-S].

A triangulation of a point set S obviously contains a fair amount of information about the structure of S. It has also been shown [Sha] that this information is sufficiently complex that in general $\Omega(n \log n)$ time is required to construct a triangulation for an n-set S in the plane. On the other hand, in the following we show that in general a triangulation contains so little information about S that sorting S by, say, the x-coordinate still takes time $\Omega(n \log n)$.

In order to prove this result we need the following lemma that will be important throughout the paper.

Lemma 1:

Let $A = \{x_1, \ldots, x_n\}$ be an (unordered) set of n real numbers.

In $O(n)$ time one can construct an ordered n-set $S = \{p_1, \ldots, p_n\} \subset \mathbf{R}^2$ along with a triangulation D of S, where for $1 \leq i \leq n$, $x(p_i) = x_i$, and for $1 \leq i < n$, $y(p_i) < y(p_{i+1})$.

Proof:

For $1 \leq i \leq n$ let $p_i = (x_i, i)$. Now clearly the p_i's have the two properties stated in the lemma.

Thus it only remains to show that one can construct a triangulation D of S in linear time. We will prove by induction that S can be triangulated in time proportional to $n + e$, where e is the number of edges in the triangulation. Note that for planar graphs with n vertices always $e \leq 3n-6$. Thus $n+e$ is $O(n)$.

The basis for our induction is obviously true since a triangulation of a constant number of points can be constructed in constant time.

So let $S' = \{p_1, \ldots, p_{n-1}\}$, let D$'$ be a triangulation of S', and let e' be the number of edges in D$'$. Assume inductively that D$'$ has been constructed in time proportional to $n-1+e'$.

As $y(p_n) > y(p)$ for $p \in S'$, p_n lies outside the convex hull of S' and thus there is only one way of extending D$'$ to a triangulation D of S, namely by introducing edges between p_n and a number of points in S' that lie on the convex hull (see Figure 2.2). Assume there are $h \geq 2$ such new edges. By the y-maximality of p_{n-1} in S' one of these edges must be incident to p_{n-1}. From this fact and from D$'$ the remaining edges can be determined at constant cost per new edge. Thus D$'$ can be extended to D in time proportional to a constant plus h. Using the inductive assumption this means that D was constructed in time proportional to $n+e'+h$. But as clearly $e=e'+h$, D was therefore constructed in time proportional to $n+e$, i.e. it was constructed in O(n) time.

Q.E.D.

Now we can state the first lower bound result of this paper.

Theorem 1:

Let S be a set of n points in the plane and let D be a triangulation of S.

Any algorithm that, given S and D as input, sorts the points of S with respect to their x-coordinates takes time $\Omega(n \log n)$ in the worst case.

Proof: (reduction to sorting)

Consider the following algorithm for sorting a set $T = \{x_1, \ldots, x_n\}$ of reals.

1. Use Lemma 1 to construct the set $S = \{(x_1,1), \ldots, (x_n,n)\}$ and a triangulation D thereof.
2. Use any algorithm to sort S by x-coordinate that also makes use of D. As for $1 \leq i \leq n$, $x((x_i,i)) = x_i$, this effectively also yields the sorted order of T.

As by Lemma 1 step 1 takes time O(n) and since sorting T requires time $\Omega(n\log n)$ in the worst case it is clear that step 2 must take time $\Omega(n \log n)$ in the worst case.

Q.E.D.

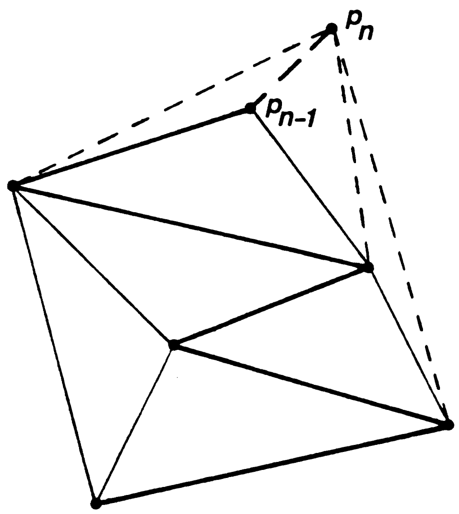

Figure 2.2

It is now worth asking whether some special classes of triangulations yield more information. One popular class of triangulations are the so-called Delaunay triangulations, which are the duals of Voronoi Diagrams (see [Sha]). A necessary and sufficient condition for a triangulation D of a set S to be Delaunay is that for any triangle T in D the open disc bounded by the circumscribed circle of T contains no point of S (see Figure 2.3). For a finite set S in the plane the Delaunay triangulation $\Delta(S)$ is unique[1].

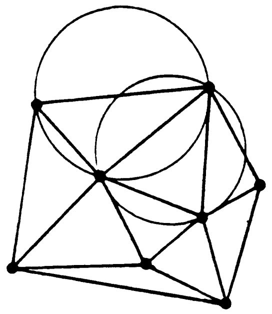

Figure 2.3

We now show a result analogous to Lemma 1.

Lemma 2:

Let $A = \{x_1, \ldots, x_n\}$ be a set of n reals.

In $O(n)$ time one can construct an n-set $S = \{p_1, \ldots, p_n\} \subset \mathbf{R}^2$ along with its Delauny triangulation $\Delta(S)$, where S has the property that for $1 \leq i \leq n$, $x(p_i) = x_i$, and for $1 \leq i < n$, $y(p_i) < y(p_{i+1})$.

Proof:

The inductive proof for this lemma is almost identical to the proof of Lemma 1. One only needs to be more judicious about the choice of y-coordinate for a point p_i.

For the base case of the induction observe that because of the distinctness assumption for the points in A it is possible to choose in constant time reals $y_1 < y_2 < y_3$ such that $p_i = (x_i, y_i)$, $i=1,2,3$, forms a triangle, which of course is the Delaunay triangulation of $\{p_1, p_2, p_3\}$.

Now assume inductively that we have found $y_1 < y_2 < \cdots < y_{n-1}$ and the Delaunay triangulation of $S' = \{p_1, \ldots, p_{n-1}\}$, where $p_i = (x_i, y_i)$, in time proportional to $n-1$ plus e', the number of edges in $\Delta(S')$. Also assume inductively that the real number

$$m' = \max\{ y(q) \mid q \in C, \text{where } C \text{ is the circumscribed disc of any triangle in } \Delta(S') \}$$

has been determined within that time bound.

Observe that $m' \geq y_{n-1}$. Now choose $y_n = m' + 1$, let $p_n = (x_n, y_n)$, and, as in the proof of Lemma 1, complete $\Delta(S')$ to a triangulation D of S in time proportional to h, the number of edges added. We claim that $D = \Delta(S)$.

To prove this we have to show that the circumscribed open disc of every triangle in D contains no points of S. This is obviously the case for all triangles that were already present in $\Delta(S')$, since the circumscribed disc of such a triangle cannot contain a point of S', as $\Delta(S')$ is a Delaunay triangulation, and since it cannot contain p_n by the choice of y_n.

[1]Strictly speaking, $\Delta(S)$ is only guaranteed to be a triangulation if no four points in S are co-circular. However, the "degenerate" cases where this is not true are not of interest here.

Thus it remains to show that the circumscribed open disc of the $h-1$ "new" triangles contain no point of S'. Let T be such a triangle, let s be the edge of the convex hull of S' that is a side of T, and let T' be the unique triangle in $\Delta(S')$ that has s as a side (see Fig. 2.4). Furthermore, let C and C' be the circumscribed discs of T and T' respectively.

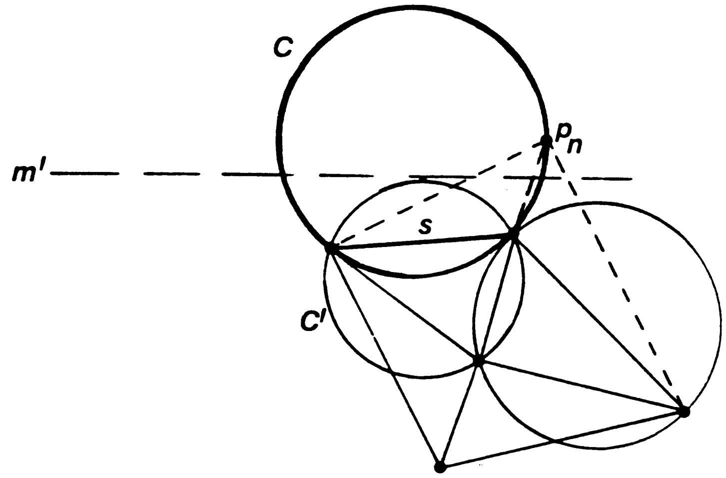

Figure 2.4

To see that C indeed contains no point of S' note that s cuts C into two parts. The part outside the convex hull of S' clearly cannot contain a point of S'. The other part must be contained in C' (since p_n not in C' !). But as T' is a triangle of the Delaunay triangulation $\Delta(S')$, C' does not contain a point of S', and hence C does not.

Finally note that clearly m' can be "updated" to a new y-maximum m in time proportional to h.

Q.E.D.

This lemma immediately yields a stronger version of Theorem 1.

Theorem 2:

Let S be a set of n points in the plane and let $\Delta(S)$ be its Delaunay triangulation.

Any algorithm that, when given S and $\Delta(S)$ as input, sorts the points of S with respect to their x-coordinates takes time $\Omega(n \log n)$ in the worst case.

Proof:

Identical to the proof of Theorem 1; just use Lemma 2 instead of Lemma 1.

We want to point out, however, that this proof takes advantage of the unit cost assumption in our RAM model of computation since there is no obvious upper bound for the y-coordinates of the points obtained in Lemma 2.

Q.E.D.

III. Preprocessed Planar Subdivision Searching

A classical 1-dimensional problem is the so-called dictionary problem: let $A = \{x_1, \ldots, x_n\}$ be a set in $\mathbf{R}$ with $x_1 < x_2 < \cdots < x_n$; for an arbitrary query number $z \in \mathbf{R}$ one is to determine whether there exists an i, $1 \leq i < n$, such that $x_i \leq z < x_{i+1}$, and in case it does, one is to report it. This problem is usually solved by binary search with $O(\log n)$ comparisons, and it also has been shown that $\Omega(\log n)$ comparisons are necessary even on fairly elaborate models of computation [Fri]. It is fairly easy to prove that solving the dictionary problem for an arbitrary set $B = \{z_1, \ldots, z_n\} \subset \mathbf{R}$ of n numbers requires time $\Omega(n \log n)$. The situation changes however if some additional information about B is available. If, for instance, B is sorted, i.e. $z_1 \leq z_2 \leq \cdots \leq z_n$, then these "batched" dictionary problems can be solved in $O(n)$ time by a simple merge procedure.

The dictionary problem can be reformulated as follows: for a given contiguous set of intervals on the real line and an arbitrary query number determine which interval (if any) contains that number. The 2-dimensional generalization of this problem is quite straightforward: for a given contiguous set of triangles in the plane and an arbitrary query point determine which triangle (if any) contains that point. This is of course the classical planar subdivision search problem in computational geometry and practical $O(\log n)$ time and $O(n)$ space solutions have been found by Kirkpatrick [K83] and by Edelsbrunner et al. [EGS]. It should be clear that the $\Omega(\log n)$ lower bound for a single query and the $\Omega(n \log n)$ lower bound for multiple queries of the 1-dimensional dictionary problem carry over to the 2-dimensional subdivision search problem since it is easy to express the 1-dimensional problem as an instance of the 2-dimensional one.

Now the question arises whether, like in the 1-dimensional case, the multiple query subdivision search problem can be solved in $o(n \log n)$ time if additional information is available about the set of query points Q. Is, for instance, an improvement possible, if Q is sorted by one coordinate, or if a triangulation of Q is available?

The following theorems anwer these questions in the negative.

Theorem 3:

Let $S \subset \mathbf{R}^2$ be a set of n points and let Δ be a triangulation of S. Let $Q \subset \mathbf{R}^2$ be a set of n query points and let D be a triangulation of Q.

The problem of determining for each point in Q which triangle (if any) of Δ contains it requires time $\Omega(n \log n)$ in the worst case.

Proof:

Let us assume we have some algorithm that solves the problem stated in the theorem in time $f(n)$ in the worst case. We will show that with an additional time expenditure of $O(n)$ this algorithm could be used to solve the multiple query dictionary problem. However, for that problem there is an $\Omega(n \log n)$ lower bound. Hence $f(n)$ must be $\Omega(n \log n)$.

So let $A = \{x_1, \ldots, x_n\}$ with $x_1 < x_2 < \cdots < x_n$ be a dictionary and let $B = \{z_1, \ldots, z_n\}$ be an arbitrary set of query points. Using Lemma 1 obtain from B in linear time a set $Q = \{q_1, \ldots, q_n\} \subset \mathbf{R}^2$ and a triangulation D of Q, where for $1 \leq j \leq n$, $x(q_j) = z_j$, and for $1 \leq j < n$, $y(q_j) < y(q_{j+1})$.

Obtain in linear time from A a set $S = \{p_1, \ldots, p_n, p_1', \ldots, p_n'\}$, with $p_i = (x_i, y_{\min})$ and $p_i' = (x_i, y_{\max})$ for $1 \leq i \leq n$, where $y_{\min} = y(q_1)-1$ and $y_{\max} = y(q_n)+1$. Next construct the triangulation Δ of S as depicted in Figure 3.1. This can be done in linear time since the set A is sorted. Now run the assumed algorithm to locate all points of Q in the triangulation Δ in time $f(n)$. It should be clear that some $z_j \in B$ is "contained" in the dictionary A if and only if the corresponding point $q_j \in Q$ lies in some triangle of Δ. Furthermore, knowing the triangle that contains q_j allows to determine in constant time where z_j is to be found in the dictionary, i.e. for which i, $x_i \leq z_j < x_{i+1}$. Thus we have solved the multiple query dictionary problem in time $O(n)+ f(n)$ and this completes the proof. Q.E.D.

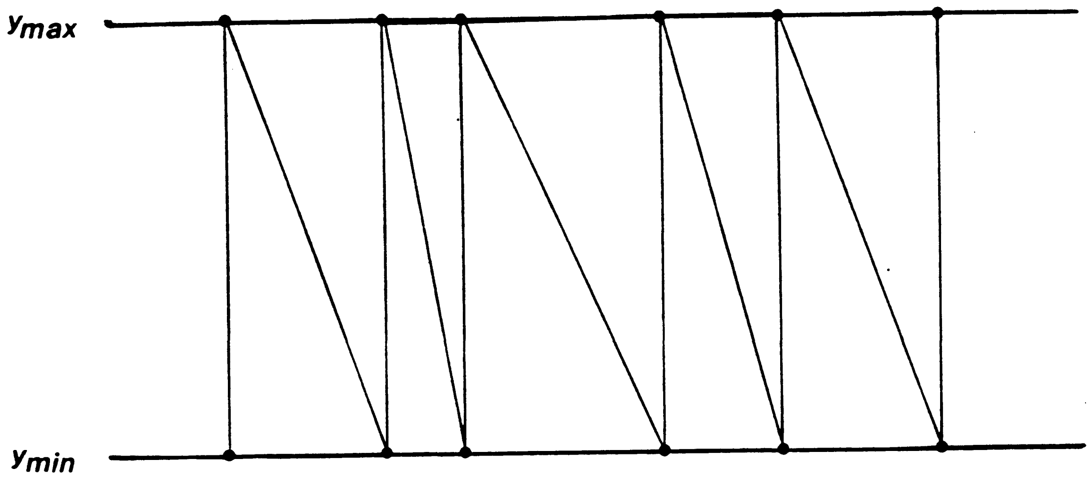

Figure 3.1

The preceding theorem can be strengthened in the following way.

Theorem 4:

Let $S \subset \mathbf{R}^2$ be a set of n points and let $\Delta = \Delta(S)$ be its Delaunay triangulation. Let $Q = \{q_1, \ldots, q_n\} \subset \mathbf{R}^2$ and let $D = \Delta(Q)$ be its Delaunay triangulation.

The problem of determining for each point in Q which triangle (if any) of Δ contains it requires time $\Omega(n \log n)$ in the worst case.

Proof:

The proof is basically identical to the proof of Theorem 3. Use Lemma 2 instead of Lemma 1 to obtain the set Q and its Delaunay triangulation.

The set S used in the proof of Theorem 3 is degenerate in the sense that for all i, $1 \leq i < n$, the four points p_i, p_{i+1}, p_i', and p_{i+1}' are co-circular, and the edges connecting p_i' and p_{i+1} should not really be in $\Delta(S)$. However, this problem can be circumvented by determining from A in linear time $\delta = \min_{1 \leq i < n} (x_{i+1} - x_i)$ and letting $p_i' = (x_i + \frac{1}{4}\delta, y_{max})$ for $1 \leq i \leq n$.

It can then be shown that the Delaunay triangulation of the set S defined this way has the same structure as the one of the original S is depicted in Figure 3.1. Furthermore, if $q_j \in Q$ is located in some particular triangle of $\Delta(S)$, then only one of two possible intervals of the dictionary A can contain the corresponding $x_j \in B$, and it can be determined in constant time which one does.

Q.E.D.

Before closing this section we want to point out something rather peculiar. We just showed that given two n-sets S and Q and their Delaunay triangulations $\Delta(S)$ and $\Delta(Q)$, generally $\Omega(n \log n)$ time is necessary to locate all points of Q in $\Delta(S)$. On the other hand Kirkpatrick [K79] has shown that given $\Delta(S)$ and $\Delta(Q)$ one can construct the Delaunay triangulation of $S \cup Q$ in *linear* time.

IV. Finding Convex Hulls of 3-Dimensional Sorted Sets

The convex hull problem is one of the best studied problems in computational geometry. For the 2-dimensional case, finding the smallest convex polygon that contains a given set of n points in $\mathbf{R}^2$, a number of different $O(n \log n)$ algorithms have been proposed ([Gra],[Sha], [P-H],[Pre],[K-S]), and in [Sha] it is shown that this problem[2] has an $\Omega(n \log n)$ lower bound matching the upper bound achieved by these algorithms.

For the 3-dimensional case, finding the smallest convex polytope that contains a given set of n points in $\mathbf{R}^3$, only four non-trivial algorithms are known ([C-K],[P-H],[S81],[Kal]). However, only the algorithm in [P-H] has $O(n \log n)$ worst case time complexity and is worst case optimal, whereas the other three algorithms have $O(n^2)$ worst case time complexity.

The fact that a fair number of optimal algorithms for the 2-dimensional problem are known, whereas there is only one single optimal algorithm for the 3-dimensional case indicates that intuitively the 3-dimensional problem is harder than the 2-dimensional one, although this difference in difficulty is not quantifiable by the usual means of asymptotic complexity measures. In this and the following section we give evidence that the 3-dimensional problem is indeed "harder" by showing that certain additional input information that allows linear time algorithms in the 2-dimensional case does not reduce the $\Omega(n \log n)$ complexity in the 3-dimensional case.

The first kind of additional input information we consider is the ordering of the input points with respect to some coordinate. If for a set $S = \{p_1, \ldots, p_n\} \subset \mathbf{R}^2$ it is guaranteed that $x(p_1) < x(p_2) < \cdots < x(p_n)$, then the planar convex hull of S can be constructed in $O(n)$ time by an incremental construction akin to the one used in the proof of Lemma 1. In contrast to this we show in the following that for a set $S \subset \mathbf{R}^3$ that is ordered in such a way there is still an $\Omega(n \log n)$ lower bound for constructing its 3-dimensional convex hull.

At first, though, we have to make more specific what we mean by constructing the convex hull of a point set $S \subset \mathbf{R}^3$. Such a convex hull is a 3-dimensional polytope whose vertices are points in S. Its boundary complex, i.e. the system of its vertices, edges, and faces, can be viewed as a planar graph. Thus for our purposes, an adequate description of the convex hull C of S has to identify which points of S are vertices of P and it also has to provide a good representation of the planar graph formed by the boundary complex of C, e.g. in form of cyclically ordered adjacency lists. This is the description that is to be produced by a 3-dimensional convex hull algorithm. All four algorithms mentioned above can be made to yield such a representation.

For the theorems in this section we need the following

Lemma 3:

Let S be a set of n points in $\mathbf{R}^3$ and assume an adequate representation representation of its convex hull C is available.
Let H be a plane in $\mathbf{R}^3$ and let S' be the orthogonal projection of S into H.

With this information the 2-dimensional convex hull C' of S' in H can be determined in $O(n)$ time.

Proof: (Outline)

Let H be the plane $\{ x \in \mathbf{R}^3 \mid \langle x, a \rangle = c \}$ for some $c \in \mathbf{R}$ and some non-zero $a \in \mathbf{R}^3$.
It should be clear that the boundary of C' is the projetion of some closed path P of edges and vertices of C. The criterion for an edge e of C to be in this path P is simple and can be tested in constant time: let N_e and N_e' be the outward normal vectors of the two faces of C that contain e, and let $s_e = \langle N_e, a \rangle$ and $s_e' = \langle N_e', a \rangle$; the edge e is in P iff one of s_e and s_e' is greater than zero and the other one is not.

[2]To be exact, the bound given in [Sha] is for the version of the convex hull problem, where one is to find the ordered sequence of vertices around the convex hull. A stronger lower bound for the weaker problem of just identifying the vertices of the convex hull is given in [K-S].

Thus all the edges of P can be identified in time proportional to the number of edges of C which is linear in n. Furthermore, it is easy to see that the cyclic order in which these edges appear on P can be determined in linear time also.

Finally let $V = \{v_0, \ldots, v_{k-1}\}$ be the cyclic sequence of vertices that appear on P and let $V' = \{v_0', \ldots, v_{k-1}'\}$ be their orthogonal projections into H. Remove in linear time from V' all v_i' that are collinear with $v'_{(i-1) \bmod k}$ and $v'_{(i+1) \bmod k}$. (These collinearities can appear if a face of C is normal to H.) The resulting sequence is the sequence of vertices around C'.

Q.E.D.

With this lemma the following result is easy:

Theorem 5:

Let $S = \{p_1, \ldots, p_n\}$ be a set of n points in $\mathbf{R}^3$ with $x(p_1) < x(p_2) < \cdots < x(p_n)$.

Any algorithm that finds the convex hull of such an ordered set requires time $\Omega(n \log n)$ in the worst case.

Proof:

Assume some algorithm that solves this problem in worst case time $f(n)$. We will use this algorithm to solve a 2-dimensional convex hull problem.

Let $T = \{(y_1,z_1), \ldots, (y_n,z_n)\}$ be a set of n points in $\mathbf{R}^2$. We construct the convex hull of T in the following way.

Form the set $S = \{ (i,y_i,z_i) \mid 1 \leq i \leq n \} \subset \mathbf{R}^3$ in linear time. Clearly S is sorted by x-coordinate. Thus we can apply our assumed algorithm to find C, the 3-dimensional convex hull of S in time $f(n)$.

Now let H be the y-z-plane and use the construction of Lemma 3 to find in linear time the 2-dimensional convex hull of the set $S' = \{ (0,y_i,z_i) \mid 1 \leq i \leq n \}$ which is the orthogonal projection of S into H.

Clearly S' can be identified with T and thus we have in effect constructed the 2-dimensional convex hull of T in time $O(n) + f(n)$. However, because of the $\Omega(n \log n)$ lower bound for the 2-dimensional convex hull problem $f(n)$ must therefore be $\Omega(n \log n)$.

Q.E.D.

Theorem 5 says that having a set in $\mathbf{R}^3$ sorted with respect to the x-coordinate does not really help in finding its convex hull. David Kirkpatrick has suggested a strengthening of this result.

Let a be a non-zero vector in $\mathbf{R}^3$. For a set $S = \{p_1, \ldots, p_n\} \subset \mathbf{R}^3$ we say we know the *order of S with respect to the direction a*, if we know a permutation π_a of the integers from 1 to n, such that $\langle a, p_{\pi_a(i)} \rangle \leq \langle a, p_{\pi_a(i+1)} \rangle$ for $1 \leq i < n$. The intuitive meaning of this definition is straightforward: if for some line L with direction a, l_i is the orthogonal projection of p_i onto L for $1 \leq i \leq n$, then when traversing L in direction a one encounters $l_{\pi_a(1)}, l_{\pi_a(2)}, \ldots, l_{\pi_a(n)}$ in this sequence.

Theorem 6:

Let k be a fixed positive integer, and let $A \subset \mathbf{R}^3$ be a set of k non-zero vectors in $\mathbf{R}^3$.
Let $S = \{p_1, \ldots, p_n\}$ be a set of n points in $\mathbf{R}^3$ for which the ordering is known with respect to each direction $a \in A$.

Any algorithm that finds the convex hull of S requires time $\Omega(n \log n)$ in the worst case, even if it has all these orderings available.

Proof:

Assume some algorithm that solves this problem in worst case time $f(n)$. The proof that $f(n) = \Omega(n \log n)$ is again by reduction to the 2-dimensional convex hull problem.

Let b be a vector in $\mathbf{R}^3$ such that $<b,a> \neq 0$ for all $a \in A$. It is easy to see that such a b must always exist and that it can be found in constant time if k is assumed to be a constant. Let $A^+ = \{ a \in A \mid <a,b> > 0 \}$ and let $A^- = \{ a \in A \mid <a,b> < 0 \}$.

Now let $T = \{q_1, \ldots, q_n\}$ be a set of n points in the plane $H = \{ x \in \mathbf{R}^3 \mid <x,b> = 0 \}$. We want to construct the 2-dimensional convex hull C of T in H.

We first construct a set $S = \{p_1, \ldots, p_n\} \subset \mathbf{R}^3$, where $p_i = q_i + \lambda_i b$ for $1 \leq i \leq n$. Obviously for any choice of the reals λ_i the orthogonal projection of S into H is T. Furthermore, we can choose the λ_i's such that S is sorted with respect to each direction $a \in A$. To be more precise, we will choose the λ_i's such that the index permutation π_a that gives the ordering of S with respect to some direction $a \in A$ is the identity permutation π_I if $a \in A^+$, and the reversal permutation π_R if $a \in A^-$, where π_I and π_R are defined by $\pi_I(i) = i$ and $\pi_R(i) = n+1-i$, for $1 \leq i \leq n$.

The reader may verify that this can be achieved by setting λ_1 to 1 and letting

$$\lambda_i = \max\left(\left\{ \frac{<p_{i-1}-q_i, a>}{<b,a>} \mid a \in A^+ \right\} \cup \left\{ \frac{<q_i - p_{i-1}, a>}{<b,a>} \mid a \in A^- \right\} \right)$$

for $i = 2, \ldots, n$.

As $A = A^+ \cup A^-$ contains only a constant number of elements all the λ_i's can be determined in $O(n)$ time and we have constructed the desired set S, whose projection into H is T and which is sorted with respect to each of the k directions in A.

Now apply the assumed algorithm to find the 3-dimensional convex hull of S in time $f(n)$. Subsequently, using the construction of Lemma 3, recover from this hull the 2-dimensional convex hull C of T in the plane H in time $O(n)$.

Thus we constructed C in time $O(n) + f(n)$. But because of the $\Omega(n \log n)$ lower bound for the 2-dimensional convex hull problem $f(n)$ must therefore be $\Omega(n \log n)$.

Q.E.D.

V. Convex Hulls of Simple and Star-Shaped Polytopes

Let us informally define a *simple polytope* to be a 3-dimensional solid which is not necessarily convex but does not have any "holes" and whose boundary consists of a finite number of planar pieces. A *star-shaped polytope* is a simple polytope P that contains a point z (called a *center* of P) such that for every $x \in$ P the straight line segment joining z and x is totally contained in P.

The results of this section were motivated by the attempt of several researchers to find linear time algorithms for constructing the convex hull of 3-dimensional star-shaped and simple polytopes. There was some encouraging evidence that such algorithms exist since the corresponding 2-dimensional problems do have linear time solutions. The second phase of Graham's algorithm [Gra] finds the convex hull of a star-shaped polygon in linear time and algorithms in [B-E] and [M-A] do so for simple polygons. (Several other algorithms have been published but their correctness has been challenged.)

A linear time convex hull algorithm for star-shaped polytopes would have had some nice applications (e.g. in the work presented in [Lit]) and furthermore it would have implied the existence of a linear time procedure for constructing the convex hull of the union of two convex polytopes.[3]

Before proving the main theorem of this section we need to introduce the notion of an upper hull. For a non-zero vector b and a finite set T in the plane we define the *upper hull* of T with respect to the direction b to be the ordered sequence of all the edges of the convex hull of T for which the outward normals form a positive inner product with b. This definition is made more intuitive in Figure 5.1.

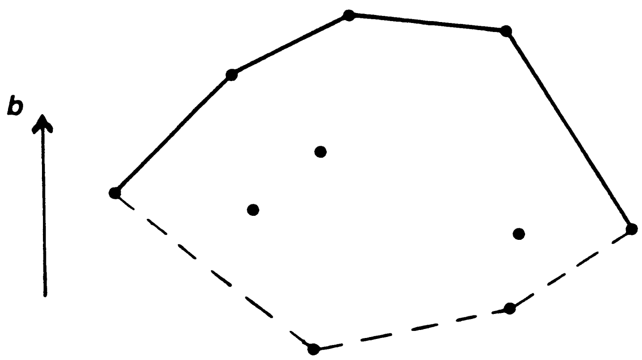

Figure 5.1

It should be clear that constructing the upper hull of a set T of n points with respect to some direction b requires $\Omega(n \log n)$ time in the worst case since this upper hull together with the upper hull with respect to $-b$ yields a complete description of the convex hull of T.

Finally, let us note that, like in the case of convex polytopes, we assume that a simple or star-shaped polytope is represented by a planar graph description of its boundary together with the coordinates of its vertices.

Theorem 7:
Any algorithm that constructs the convex hull of a star-shaped polytope with n vertices requires $\Omega(n \log n)$ time in the worst case.

[3]The problem whether there exists such a linear time procedure is still open in general and only solved for the case where the two polytopes are separated by a plane [P-H]. The claim in [B-S] that the solution in [P-H] generalizes to the non-separable case is as yet completely unsupported.

Proof:

Assume some algorithm that solves this problem in time $f(n)$. We prove that $f(n) = \Omega(n\log n)$ by reduction to the 2-dimensional upper hull problem.

Let $T = \{q_1, \ldots, q_{n-1}\}$ be a set of points in the x-z-plane with $q_i = (x_i, 0, z_i)$. We want to construct the upper hull of T with respect to the direction $b = (0,0,1)$.

Using the construction of Lemma 1 determine in $O(n)$ time a triangulation D of the set $T' = \{q_1', \ldots, q_{n-1}'\}$ in the x-y-plane, where $q_i' = (x_i, i, 0)$ for $1 \leq i < n$. Re-endow every point in T' with its "original" z-coordinate to yield the set $S = \{p_1, \ldots, p_{n-1}\}$, where $p_i = (x_i, i, z_i)$ for $1 \leq i < n$, and let S "inherit" the triangulation D of T'. This inherited triangulation can be interpreted as a piecewise planar surface F with the property that no planar piece is orthogonal to the x-y-plane, and every "vertical" line, i.e. a set of the form $\{ a + \lambda b \mid \lambda \in \mathbf{R} \}$, $a \in \mathbf{R}^3$, intersects F at most once.

This property ensures the existence of a sufficiently large μ such that the point $p_n = p_1 - \mu b$ lies "below" all the planes that contain a planar piece of F. Furthermore, this μ can be determined from D in $O(n)$ time since D has no more than $2n-7$ triangles and hence F consists of at most $2n-7$ planar pieces.

Next determine in $O(n)$ time the edges of D that correspond to the boundary of F (i.e. essentially the convex hull edges of T') and for each such edge e add the triangle spanned by e and p_n to the surface F. This yields a closed piecewise planar surface bounding a simple polytope P with vertex set $S' = \{p_1, \ldots, p_n\}$. As a matter of fact, P is star-shaped with center p_n.

Now use the assumed algorithm to find the 3-dimensional convex hull of P in time $f(n)$ and use Lemma 3 to obtain in linear time the projection of this hull onto the x-z-plane.

It is easy to see that this projection consists exactly of p_n and the vertices of the upper hull of the initial set T with respect to b. Thus in time $O(n) + f(n)$ we effectively constructed the upper hull of T. The $\Omega(n\log n)$ lower bound for the upper hull problem implies therefore that $f(n) = \Omega(n\log n)$.

Q.E.D.

Theorem 8:

Any algorithm that constructs the convex hull of a simple 3-dimensional polytope with n vertices requires $\Omega(n\log n)$ time in the worst case.

Proof:

This follows immediately from Theorem 7 since a star-shaped polytope is a special case of a simple polytope.

Q.E.D.

Acknowledgements:

I would like to thank John Gilbert for his careful reading of the manuscript and for his comments.

References:

[B-O] Ben-Or,M. "Lower Bounds for Algebraic Computation Trees." Proc. 15th Annual ACM Symposium on Theory of Computing, pp.80-86; 1983.

[B-E] Bhattacharya,B.K., El-Gindy,H. "A New Linear Convex Hull Algorithm for Simple Polygons." McGill Univ., Montreal, School of Computer Science, Report SOCS 81.29; 1981.

[B-S] Bentley,J.L., Shamos,M.I. "Divide and Conquer for Linear Expected Time." Information Processing Letters 7; pp.87-91; 1978.

[C-K] Chand,D.R., Kapur,S.S. "An Algorithm for Convex Polytopes." Journal of the ACM 17; pp.78-86; 1970.

[EGS] Edelsbrunner,H., Guibas,L.J., Stolfi,J. "Optimal Point Location in a Monotone Subdivision." Manuscript; 1983.

[Fre1] Fredman,M.L. "Inherent Complexity of Range Query Problems." Proc. of the 17th Annual Allerton Conf. on Communication, Control, and Computing; pp.231-240; 1979.

[Fre2] Fredman,M.L. "A Lower Bound on the Complexity of Orthogonal Range Queries." Journal of the ACM 28; pp.696-705; 1981.

[Fri] Friedman,N. "Some Results on the Effect of Arithmetics on Comparison Problems." Proc. 13th IEEE Symp. Switching and Automata Theory, pp.139-143; 1972.

[Gra] Graham,R.L. "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set." Information Processing Letters 1, pp.132-133; 1972.

[G-S] Guibas,L.J., Stolfi,J. "Primitives for the Manipulation of General Subdivisions and Computation of Voronoi Diagrams." Proc. 15th Annual ACM Symposium on Theory of Computing, pp.221-234; 1983.

[Kal] Kallay,M. "Convex Hull Algorithms for Higher Dimensions." Manuscript, Univ. of Oklahoma, Norman OK; 1981.

[K79] Kirkpatrick,D.G. "Efficient Computation of Continuous Skeletons." Proc. of 20th Annual IEEE Symposium on Foundations of Computer Science, pp.18-27; 1979.

[K83] Kirkpatrick,D.G. "Optimal Search in Planar Subdivisions." SIAM J. on Computing 12; pp.28-35; 1983.

[K-S] Kirkpatrick,D.G., Seidel,R. "The Ultimate Planar Convex Hull Algorithm?" Cornell Univ., CS Tech. Rep. 83-577; 1983.

[Lit] Little,J. "An Iterative Method fro Reconstructing Extended Gaussian Images." Proc. of the 1983 National Conf. on Artificial Intelligence; pp.247-250; 1983.

[M-A] McCallum,D., Avis,D. "A Linear Algorithm for Finding th Convex Hull of a Simple Polygon." Information Processing Letters 9; pp.201-206; 1979.

[Pas] Paschinger,I. "Konvexe Polytope und Dirichletsche Zellenkomplexe." Univ. Salzburg, Austria; Math. Inst. Arbeitsbericht 1-2; 1982.

[Pre] Preparata,F.P. "An Optimal Real Time Algorithm for Planar Convex Hulls." Communications of the ACM 22; pp.402-405; 1979.

[P-H] Preparata,F.P. Hong,S.J. "Convex Hulls of Finite Sets in Two and Three Dimensions." Communications of the ACM 20; pp.87-93; 1977.

[S81] Seidel,R. "A Convex Hull Algorithm Optimal for Point Sets in Even Dimensions." Univ. of British Columbia, Dept. of Computer Science, Rep. 81-14; 1981.

[S82] Seidel,R. "On the Size of Closest Point Voronoi Diagrams." Tech. Univ. Graz, Austria; Inst. fuer Informationsverarbeitung, Rep. F94; 1982.

[Swa] Swart,G. "An $\Omega(n^2)$ Lower Bound For Intersection Detection." Univ. of Washington, Dept. of Computer Science; Manuscript; 1983.

[Yao] Yao,A.C. "A Lower Bound to Finding Convex Hulls." Journal of the ACM 28; pp.780-787; 1981.