Finding an $o(n^2 \log n)$ algorithm is sometimes hard

Antonio Hernández Barrera*

Abstract

The best known algorithm for sorting $X + Y$ takes $O(n^2 \log n)$ but the question of how much computation time is really needed is open. We present here a collection of problems which are, at least, as difficult as sorting $X + Y$ as any improvement on them would imply an $o(n^2 \log n)$ algorithm for this problem of sorting.

1 Introduction

In the process of designing efficient algorithms, we are faced with the problem of trying to lower known time-bounds, or finding what the real lower bounds are for a given problem. It might be useful to know that a problem we are trying to solve is, at least, as difficult as another well-studied, *very* difficult one. Some classes of difficult problems have been found, where the NP-complete class is, maybe, the best known. Inside computational geometry, a large number of problems have been identified for which it is impossible to obtain subquadratic algorithms, unless one manages to improve the complexity of a very simple formulated problem known as "3SUM". 3SUM is defined as follows: given a set S of n integers, are there $a, b, c \epsilon S$ with $a + b + c = 0$? The class of 3SUM-hard problems has been of growing interest lately to the community of computational geometers and a paper surveying them can be found in [8]. There, the authors indicated as a future direction for research to find other interesting classes of related problems, if they exist.

In this abstract, we consider a collection of problems for which an $O(n^2 \log n)$ time complexity algorithm can be designed, but the question of whether an $o(n^2 \log n)$ solution exists is open. We show that sorting $X + Y$, our base problem, can be reduced to solve any of these problems. The computation time really needed to sort $X + Y$ has remained unknown for a long time in spite of all the effort done to find it. Hence, an $o(n^2 \log n)$ algorithm for one of these problems implies that we can construct an $o(n^2 \log n)$ algorithm for sorting $X + Y$. As in the 3SUM-hard class, the lower bound for sorting $X + Y$ will immediately carry over to the other problems, whose trivial lower bound is quadratic since the size of the output is quadratic in the worst case. And also like in the 3SUM-hard case the reductions do not imply that an $o(n^2 \log n)$ solution is impossible, but it suggests we better first try to improve the base problem. Our analysis covers computing the Minkowski sum, the x-sorting, enumerating distances, the polygon containment and other problems.

*Department of Mathematics, Faculty of Science, Hiroshima University

2 Preliminaries

2.1 Reductions

The idea of each proof in this abstract is to transform or reduce one problem into another so that if an $o(n^2 \log n)$ algorithm for the second problem is found then an $o(n^2 \log n)$ algorithm can be constructed for the first one. We say that problem P *reduces* to problem Q, $P \longrightarrow Q$, iff an $O(n^2) + T(n)$ algorithm for solving P exists, where T(n) is Q's complexity. P is equivalent to Q, $P \longleftrightarrow Q$, iff $P \longrightarrow Q$ and $Q \longrightarrow P$. It is clear that P is solved in less than $O(n^2 \log n)$ if Q is.

2.2 The base problem

We take as our base problem the problem known in the literature as sorting $X + Y$.

Sorting $X + Y$, where X and Y are the sets of real numbers $(x_i)_{1 \leq i \leq n}$ and $(y_j)_{1 \leq j \leq n}$, respectively, consists of sorting the n^2 sums $(x_i + y_j)_{1 \leq i,j \leq n}$. Using a standard sorting algorithm $X+Y$ can be sorted using $O(n^2 \log n)$ but, could it be done in less time, taking into consideration the particular structure of this set? This question has remained unanswered for more than 20 years. It was proven in 1976 in [9] that there exists a decision tree of depth $O(n^2)$ for sorting $X+Y$ but the proof was nonconstructive and it had to wait until 1990 ([10], see also [11]) for an algorithm which constructs such a quadratic depth tree. Although the algorithm indeed employs $O(n^2)$ comparisons to sort, it uses anyway $O(n^2 \log n)$ total time.

3 The reductions

3.1 Computing the Minkowski sum

Let A and B be two arbitrary sets in $\Re^d$ space. The Minkowski addition of A and B, denoted by $A \oplus B$, is defined as the set $\{a + b \mid a \epsilon A, b \epsilon B\}$, where "+"means vector sum.

We have this theorem:

Theorem 3.1 *Let P and Q be two polygons which are monotone with respect to the same line ℓ, that is, the intersection of every line orthogonal to ℓ and any of them is a connected interval, possibly empty. Let n be the total amount of edges of P and Q. Then*

$$\textit{Sorting } X + Y \longrightarrow \textit{Computing } P \oplus Q$$

Proof Let us construct rectilinear polygons P and Q as illustrated in Fig. 1. We generate the vertices of the staircase part in P and Q by taking them along the same line L as the figure shows: the intersection of a vertical line at x_i (y_j) with line L will be a vertex of P (Q) on the staircase. It is not difficult to see that by traversing the vertices on the lower staircase in $P \oplus Q$ we get the values of $X + Y$ sorted. □

It has been shown ([4]) that the sum $P \oplus Q$ of two monotone polygons can be computed in $O(n^2 \log n)$.

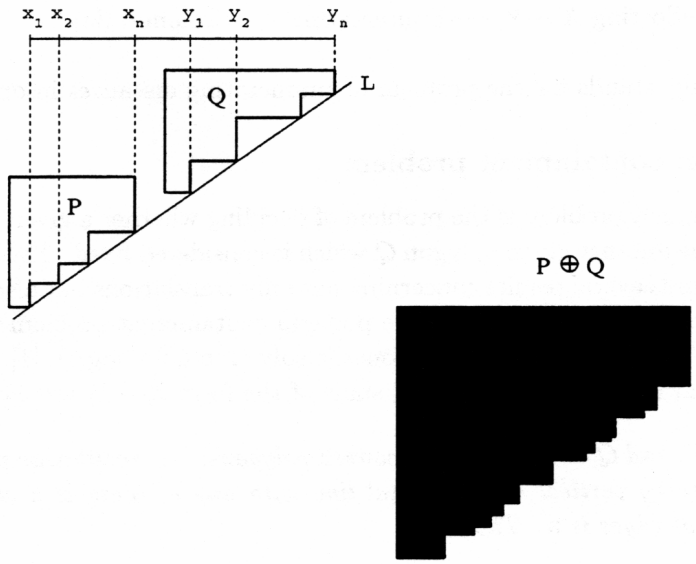

Figure 1: Generation of monotone polygons from X and Y

3.2 The x-sorting problem

The x-sorting problem is to sort the x-coordinates of the intersection points of n-planar lines. It has been mentioned that sorting $X+Y$ is a particular case of x-sorting lines (see [12]). Indeed, the authors proved the existence of a quadratic decision tree of depth $O(n^2)$ for x-sorting lines. Thus,

Theorem 3.2 *Sorting* $X+Y \longrightarrow x$*-sorting*

Clearly, if, instead of lines, we take line segments, the theorem still holds since x-sorting $\longrightarrow$ x-sorting segments. For both cases, lines or segments, the intersections can be reported (worst case) in quadratic time if no sorting is required, [7], [2], [5].

3.3 Enumerating distances

Consider this problem: given a finite set of n distinct points in the plane and a positive integer $k \leq \binom{n}{2}$, report the k smallest distances between n pairs of points. There is an $O(n \log n + k \log n)$ algorithm for this problem in [6]. Sorting $X+Y$ could be solved in less than $O(n^2 \log n)$ if this problem can be solved in $o(n^2 \log n)$ when k is $O(n^2)$.

Theorem 3.3 *Sort* $X+Y \longrightarrow$ *Enumerating*

Proof It is not difficult to see that:

Sorting $X+Y$ $\longrightarrow$Sorting $X-Y$ $\longrightarrow$Enumerating$_1$ $\longrightarrow$Enumerating

where Enumerating$_1$ stands for the problem of enumerating distances in one dimension. □

3.4 The polygon containment problem

The polygon containment problem is the problem of deciding whether a given polygon P can be translated to fit inside another given polygon Q which is considered fixed. The problem has been studied extensively and several results concerning not only translations but also rotations as well as different types of polygons are known. The polygon containment problem under translation when both polygons are rectilinearly convex ones is solved in $O(n^2 \log n)$, [1]. We proved in [3] that this problem and the problem of sorting sums of the form $X+Y$ are equivalent. That is,

Theorem 3.4 *Let P and Q be rectilinearly convex polygons, i.e. rectilinear polygons such that the intersection of every vertical or horizontal line with any of them is a connected interval, whose total number of edges is n. Then*

Sorting $X+Y$ $\longleftrightarrow$ PCP

Taking into consideration the close relation between the polygon containment problem and the motion planning problem we can also find versions of the latter such that sorting $X+Y$ reduces to them.

3.5 Sorting sums of consecutives numbers

The following problem is non geometric in nature but we considered it interesting enough, due to its close relation to sorting $X+Y$, to include it in our list: Let $a_1, a_2, ..., a_n$ be n numbers and let's define for $1 \leq i \leq j \leq n$

$$\sigma(i,j) = \sum_{k=i}^{j} a_k$$

Problem Sortσ
Sort the set of numbers $A^+ = \{\sigma(i,j),\ 1 \leq i \leq j \leq n\}$

Theorem 3.5 *Sorting $X+Y$ $\longleftrightarrow$ Sort σ*

Proof We proved this relation (see [3]) using the results in [10]. For the sake of completeness we outline the proof. The set A^+ can be represented as shown in Fig. 2 (a). With an appropriate definition of the values of x_i and y_j using the $a_1, a_2, ..., a_n$, the set $X+Y$ will be represented in the diamond shown in Fig 2 (b) while $Y-Y$ and $X-X$ will be represented in the left and right triangles, respectively. Sorting $X+Y$ $\longleftrightarrow$Sorting σ follows from here. □

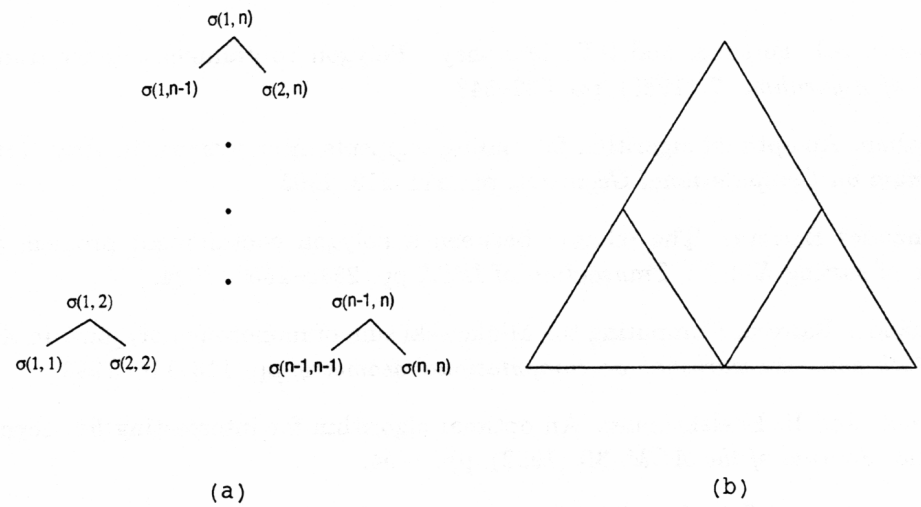

Figure 2: Representation of A^+ as a piramid.

3.6 Tracing the intersections of a ray

Suppose we are given a ray r emanating from a point p and a set of n points in the plane. Consider the problem of tracing the intersections of r with the arrangement formed by the $\binom{n}{2}$ lines passing through every two points. We can solve it by simply computing the intersection of r with every line and then sorting the intersections along r. In this way, we obtain an $O(n^2 \log n)$ time complexity, but can we do better? The problem is, at least, as hard as sorting $X + Y$ as the following theorem shows:

Theorem 3.6 *Sorting* $X + Y \longrightarrow$ *Tracing*

Proof Create for each x_i in X, a point $(x_i, 0)$ and for each y_j in Y, a point $(y_j, 2)$, i.e. the sequences X and Y will be placed along lines $y = 0$ and $y = 2$, respectively. Let p be the point $(-\infty, 1)$ and r the horizontal ray emanating from p toward $+\infty$. Notice that the intersection of r with line through points $(x_i, 0)$ and $(y_j, 2)$ has coordinates $(\frac{x_i+y_j}{2}, 1)$. □

4 Conclusions

We have presented here a collection of problems which are, at least, as difficult as sorting $X+Y$ as any improvement on them would imply an $o(n^2 \log n)$ algorithm for this problem of sorting. However, this does not imply that their lower bounds could not be higher than sorting $X+Y$'s. Finally, we think that there are more problems, indeed a class, which are related in a similar way.

References

[1] B.S. Baker, S.J. Fortune, and S.R. Mahaney. Polygon containment under translation. *Journal of Algorithms,* **7** (1986), pp. 532–548.

[2] I.J. Balaban. An optimal algorithm for finding segments intersections. In *Proc. 11th ACM Symposium on Computational Geometry*, pp. 211–219, 1995.

[3] A. Hernández Barrera. The relation between a polygon containment problem and the problem of sorting $X + Y$. *Transaction of IPSJ*, pp. 2545–2550, 1994.

[4] A. Hernández Barrera. Computing the Minkowski sum of monotone polygons. In *Abstracts of the 12th european workshop on computational geometry*, pp. 113–116, 1996.

[5] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM,* **39** (1992), pp. 1–54.

[6] M.T. Dickerson and R.L. Drysdale. Enumerating k distances for n points in the plane. In *Proceedings of the 7th Ann. Symposium Computational Geometry*, pp. 234–238, 1991.

[7] H. Edelsbrunner and L.J. Guibas. Topologically sweeping an arrangement. *Journal of Computer and System Sciences,* **38** (1989), pp. 165–194.

[8] A. Gajentaan and M.H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Computational geometry, Theory and Applications*, pp. 165–185, 1995.

[9] L.H. Harper, T.H. Paine, J.E. Savage, and E. Strauss. Sorting $X + Y$. *Communications of the ACM*, Vol. 18, No. 6, pp. 347–349, 1975.

[10] J.L. Lambert. Sorting the sums $(x_i + y_j)$ in $O(n^2)$ comparisons. In *STACS 90, Lecture Notes in Computer Science 415*, pp. 195–206. Springer-Verlag, 1990.

[11] J.L. Lambert. Sorting the sums $(x_i + y_j)$ in $O(n^2)$ comparisons. *Theoretical Computer Science*, Vol. 103, pp. 137–141, 1992.

[12] W. Steiger and I. Streinu. A pseudo-algorithmic separation of lines from pseudo-lines. *Information Processing Letters*, Vol. 53, No. 5, 1995.