Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Sweep Line Algorithm for Convex Hull Revisited

Keivan Borna*[1]

[1]Faculty of Mathematics and Computer Science, Kharazmi University, Tehran, Iran.

ABSTRACT

Convex hull of some given points is the intersection of all convex sets containing them. It is used as primary structure in many other problems in computational geometry and other areas like image processing, model identification, geographical data systems, and triangular computation of a set of points and so on. Computing the convex hull of a set of point is one of the most fundamental and important problems of computational geometry. In this paper a new algorithm is presented for computing the convex hull of a set of random points in the plane by using a sweep-line strategy. The sweep-line is a horizontal line that is moved from top to bottom on a map of points. Our algorithm is optimal and has time complexity $O(nlogn)$ where n is the size of input.

Keyword: convex hull, sweep-line method, computational geometry, graham scan, time complexity, performance.

ARTICLE INFO

Article history:

Received 14, July 2018

Received in revised form 8, March 2019

Accepted 17 April 2019

Available online 01, June 2019

AMS subject Classification: 05C78.

*Corresponding author: K. Borna. Email: borna@khu.ac.ir

1 Abstract Continued

This algorithm is new and different from existing ones for constructing convex hull. Our algorithm is working on points on two different sides, which makes it different from Graham scan and performs better for a random set of points. Finally note that we have implemented our algorithm in C with two systems for taking input points: totally random and user specified points. The outputs of our implementations are presented in the paper.

2 Introduction

Computing the convex hull of a set of points is one of the most fundamental and important problems in computational geometry. Convex hull is used as primary structure in many other problems in computational geometry and other areas like image processing, model identification, geographical data systems, triangular computation of a set of points and etc. A subset A of the plane is called convex if the line segment is located totally in A whenever p, q are in A. The convex hull of a set of n points P is the smallest convex set that contains all points and it is shown by $CH(P)$. In more subtle word, in fact convex hull is intersection of all convex sets containing all n points. The sweep-line is a horizontal line that is moved from top to bottom on a map of points. The main purpose of this paper is to find the convex hull of a set of points in the plane by using a brand new sweep-line algorithm. In the rest of this paper we first review some preliminaries about convex hull and then present our proposed algorithm and its time complexity. The paper ends with implementation and some conclusions.

3 Related works

Many algorithms are discovered for computing convex hull of a set of points P. In this section we present a short review of such algorithms. Brute force algorithm [2] discovered by Anon. This running time of this algorithm is $O(n^2)$ and it works as follows: if we draw a connecting line for two points and if other points are standing in one side of

this line, then these two points shall be in the convex hull. In 1972, other algorithm discovered for drawing convex hull with running time $O(nlogn)$ by Graham [11]. This algorithm primarily finds a point p in the interior of $CH(P)$ and express each point in the polar coordinate and order all points based on their increasing angle. Then it sends spectrums from p to other points and starts to connect points to each other to emerge convex multilateral. Another algorithm is invented in 1970 by Chand and Kapur [6]. This algorithm is started to draw a line from lowest point and a horizontal line which is rotated in the counter-clockwise direction until it meets another point. It is running time $O(nh)$ where h is the number of points in the boundary of convex hull. In 1977, a divide and conquer algorithm was discovered by Preparata and Hong [17]. It divides points to two (almost) equal size sets and computes their convex hull and then incorporates these two hulls. This algorithm works totally in time $O(nlogn)$. One can refer to other algorithm by Jarvis 1973 [13], rapid hull algorithm by Eddy [8] in 1977 and, another with running time $O(nh)$ by Bykat [4] in 1978. In 1979, a monotone algorithm of order $O(nlogn)$ was given by Andrew [1]. In 1986, a compound algorithm before win was presented by Kirk Patrik and Seidel [14]. Overmars and Van Lee [16], computed a convex hull in time $O(nlogn)$ where the insertion and elimination of points is done simultaneously. Other results produced in convex hull by Hershberger and Suri [12], Chan [5], and Brodal and Jacob [3]. In 2016 the authors introduced TORCH (total-order convex-hull) [10] where the points were floating-point numbers. In 2018 some space-efficient implementations in the integer-arithmetic setting are studied [9]. In this paper we try to present a brand new and optimal algorithm for constructing convex-hull.

4 Convex hull and sweep-line method

In order to have a better insight to convex hull of a finite set P of n points in plane, imagine as some nails in a wall and their convex hull is a rubber drain that surrounds them all. This point of view leads to another definition of convex hull of P: singular convex multilateral that their topes are points of P and included other points P; see Fig. 1.

An algorithm that is remarked here is sweep plane and a line that is within it, is called

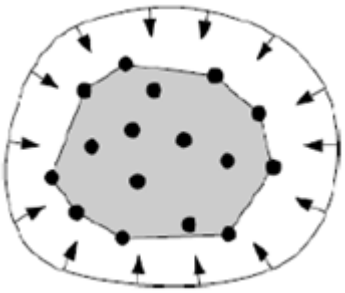

Figure 1: What is the convex hull of a set of points?

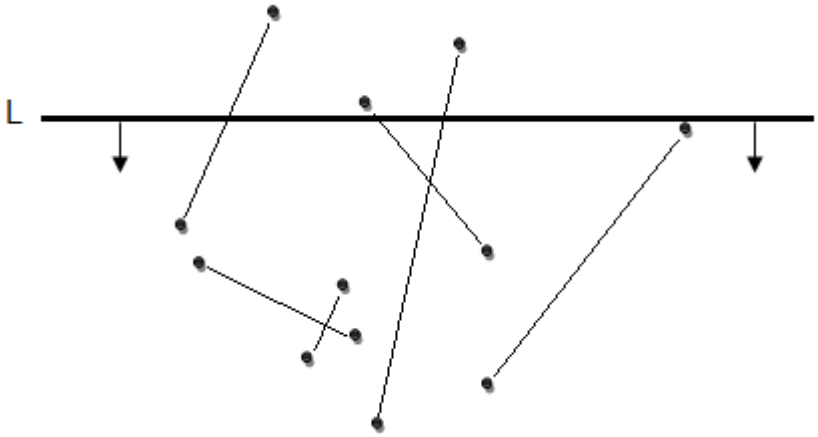

Figure 2: How the sweep line works in the line segment intersection problem

sweep line. Sweep line is a horizontal line that is moved along a map of objects from top to bottom and it traverses them. For instance, if our objects are line segments, their upper or lower end points and their intersection points are event points. Particularly, sweep-line is dropped by moving in whole space. Unexplored points are located under sweep line and explored points are located above this line [15]. Status of sweep line is a set of crossing line segments. While line is moved downward, status is produced. Upper and lower end points are called event points of sweep plane algorithm. It evaluates all events and finds all line segment intersections [7].

Here for the ease of reader and also for the sake of completeness, we review Graham Scan algorithm for constructing convex hull of a set of points. The first step is to find the bottommost point (i.e., the point with the lowest y-coordinate), say P. If there is more than one point with such property, the point with the lowest x-coordinate out of them will

be chosen. Note that here we spend $O(n)$ time, where n is the number of input points. In the next step, we sort the set of points in increasing order of the angle they and the point P make with the x-axis. If we use quicksort for sorting for example, we need to spend a further $O(nlogn)$ time which gives the total time complexity since in the sequel the overall complexity is smaller than that and the time to sort dominates the time to actually compute the convex hull. Now consider each of the points in the sorted array: For each point, Graham scan asks whether moving from the two previous points to this point is a left turn (has angle less that 180 degree) or a "right turn". If it is a "right turn", the second-to-last point is not part of the convex hull and should be removed. Note that as long as the set of the last three points is a "right turn" this process will be repeated. As soon as we encounter to a "left turn", the algorithm moves on to the next point in the array. Eventually this process will return to the point P that was the starting point. Finally we obtained the points on the convex hull in counterclockwise order.

5 Our Algorithm

Our purposed algorithm in this paper makes use of convex hull and sweep-line methodology. Our approach is similar to the sweep-line algorithm in the problem of finding intersections of some given line segments. That is, we can construct and update the convex hull as long as the sweep-line moves from top to bottom. For starters, take some random points in the plane for which no two of them have the same y coordinates. We then remove this extra assumption in Section 5. Let the sweep-line L stands above the highest point as in Fig. 3:
The sweep line moves down to reach to the first point that we call it PMAX; see Fig. 4: Now if we draw a vertical line passing through PMAX, the other points are located at left or right side of it. Our algorithm will use two lists for each side separately. We call them LL (resp. RL) for the list of points in left (resp. right) side of PMAX; see Fig. 5.
(1) It contacts to only one point in either sides: In this case we draw a line segment (as a part of convex hull edge) from the previously found point in that side to this point; see Fig. 6.
(2) It contacts to two points, one in left side and the other in right side: As in case (1),

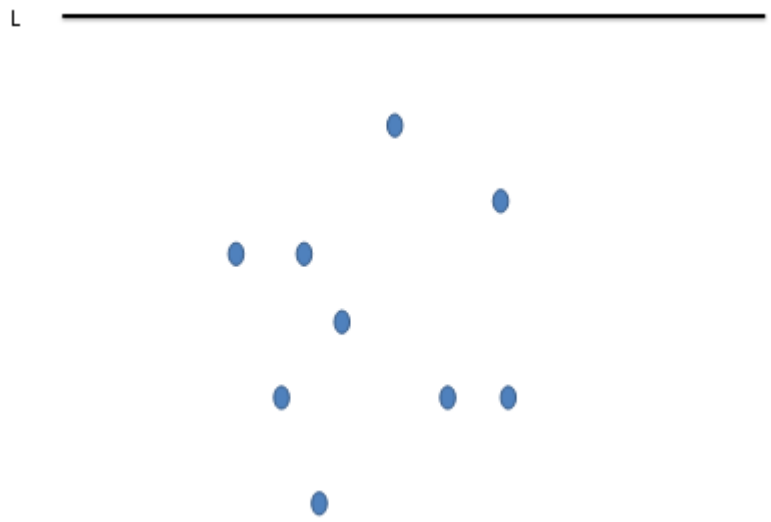

Figure 3: The sweep line stands above all points

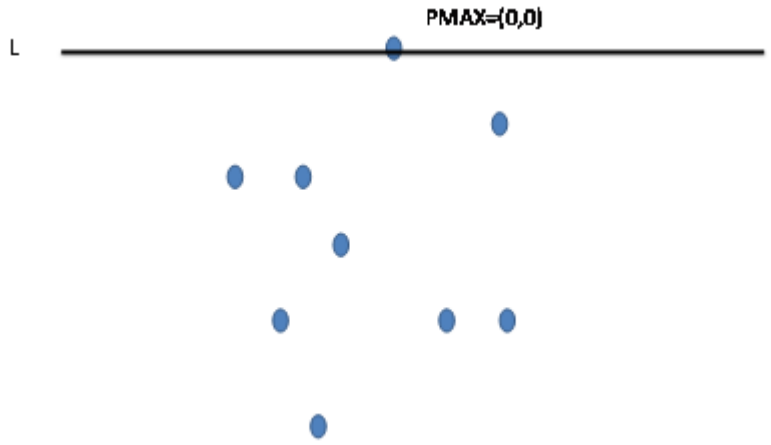

Figure 4: The sweep line moves down until it reaches to the first point (the topmost point)

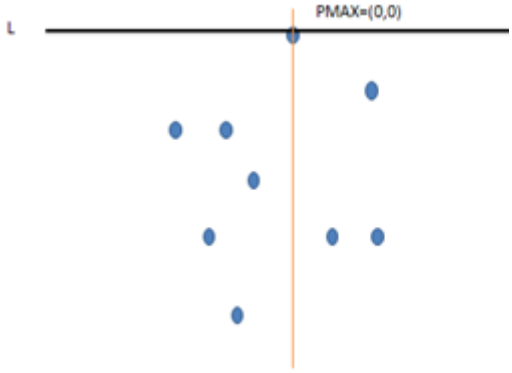

Figure 5: At PMAX, we draw a vertical line

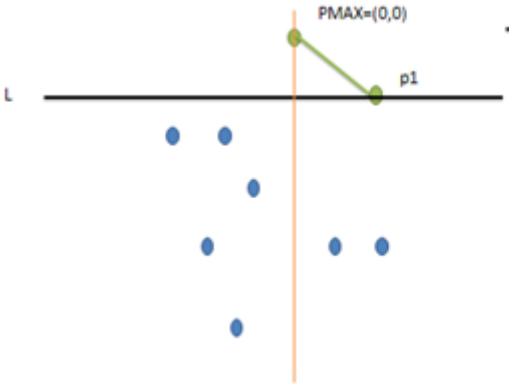

Figure 6: How our algorithm performs when the sweep line touches only one point in one side

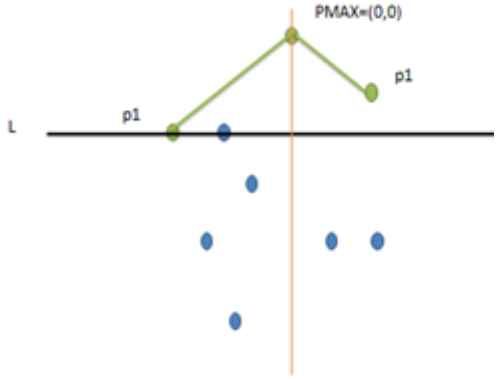

Figure 7: How our algorithm performs when the sweep line touches two points with the same y-coordinates in one side

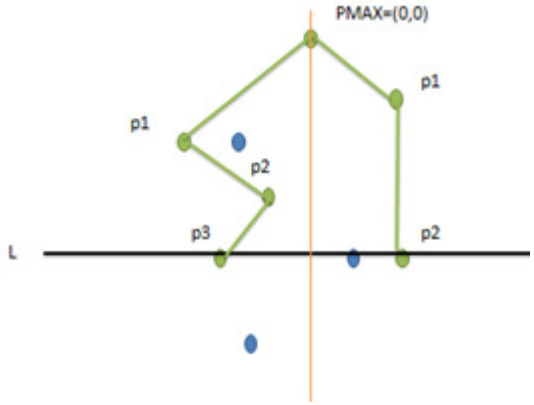

Figure 8: How our algorithm performs when the sweep line touches two (or more) points in two sides

Fig. 7, we draw two line segments from the last found point to these two points.

(3) It contacts to two points in one side: In this case if the points are in the right (resp. left) side, we keep the point with greater (resp. smaller) x coordinate and remove the latter; see Fig. 8 for example (its right side).

$$\begin{pmatrix} a_1 & b_1 & 0 \\ a_2 & b_2 & 0 \\ a_3 & b_3 & 0 \end{pmatrix}$$

Now if the result is 0, the three points are collinear; if it is positive, the three points constitute a "left turn" or counter-clockwise orientiation, otherwise a "right turn" or

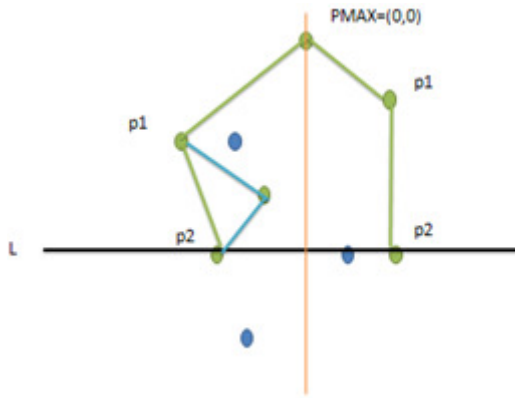

Figure 9: How our algorithm performs for right turns for the last three points in each side

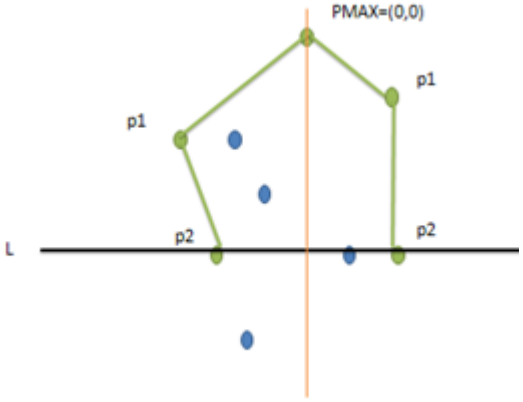

Figure 10: Continue moving downward

clockwise orientation.

We continue until the sweep-line touches all the points. Finally we draw a line segment from the last two found points in each side together to complete the convex hull; see Fig. 11.

Note that if at any stage the three points are collinear, we have two options, either to discard or to report it, since in some applications it is required to find all points on the boundary of the convex hull.

6 Degenerate cases

If there exist two points with the same y-coordinates, we first connect these two points to each other by a horizontal line and then use this line segment as one of the convex hull

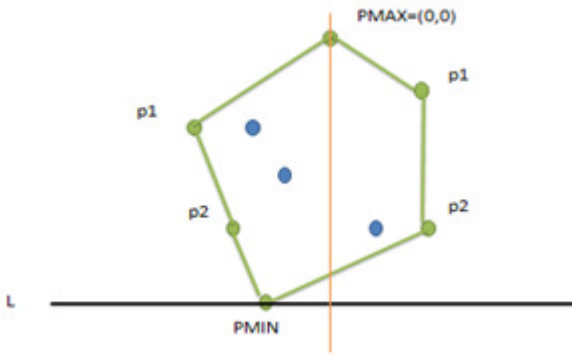

Figure 11: Reaching to the last point

edges and apply our proposed algorithm to the end points of this line segment.

7 Implementations

The pseudo code and some outputs of our algorithm are shown in this section:
Input: A set S of random points where no two of them have the same y-coordinates.
Output: The convex hull of S
Process:
0. **Find** the top most point of S, PMAX.
1. **Start** moving the sweep line L downward from PMAX.
2. **Create** two lists, LeftList (resp. RightList), consisting of points which are located below PMAX in the left (resp. right) side of a vertical line that is drawn from PMAX.
3. **Move** L downward until we reach another point, a new PMAX, draw a line segment from the old PMAX to new PMAX, go to step 2. Look carefully for right angles for the last three points in each side and remove the middle point in case.
4. If no other point is remained, the convex hull is computed.
Our algorithm has been implemented in C and in the following three outputs of it is presented. At left side of each figure, the starting points and the location of the sweep line are shown and in the right side, the convex hull is drawn. We use two systems for taking the input points, totally random and user specified points.

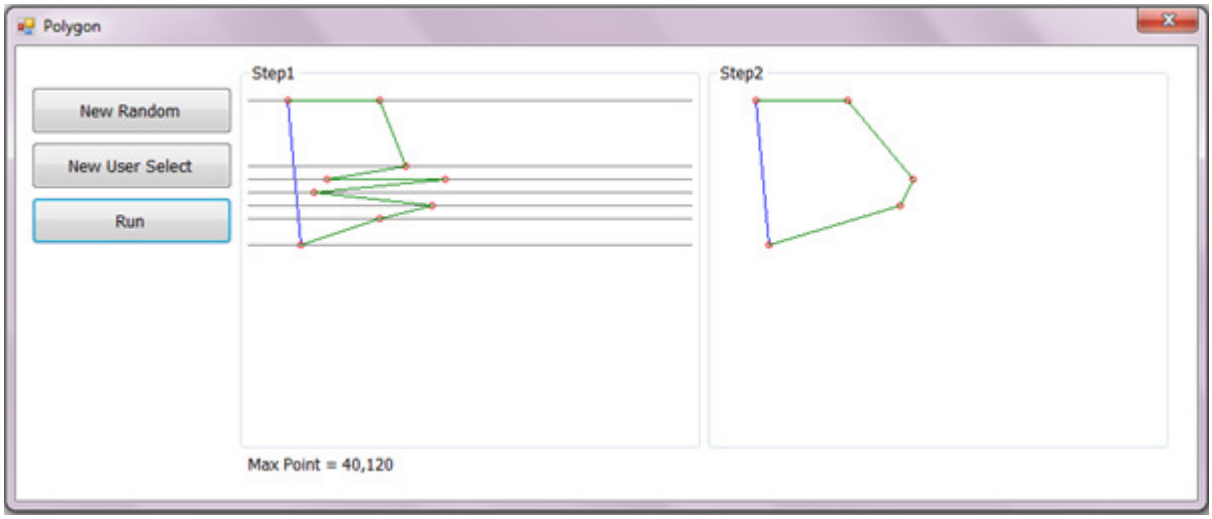

Figure 12: Demo 1 of our algorithm

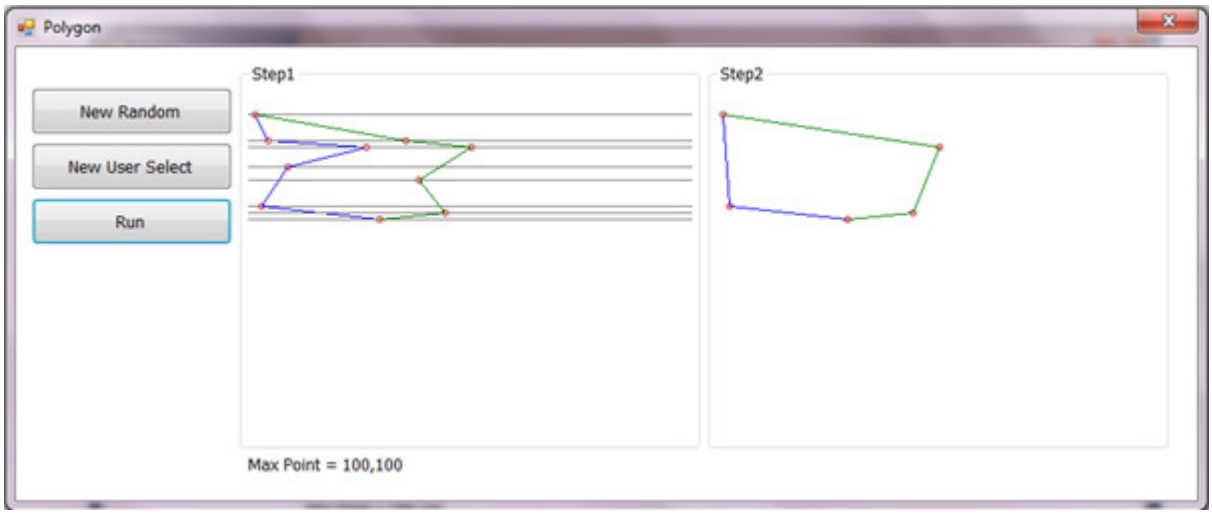

Figure 13: Demo 2 of our algorithm

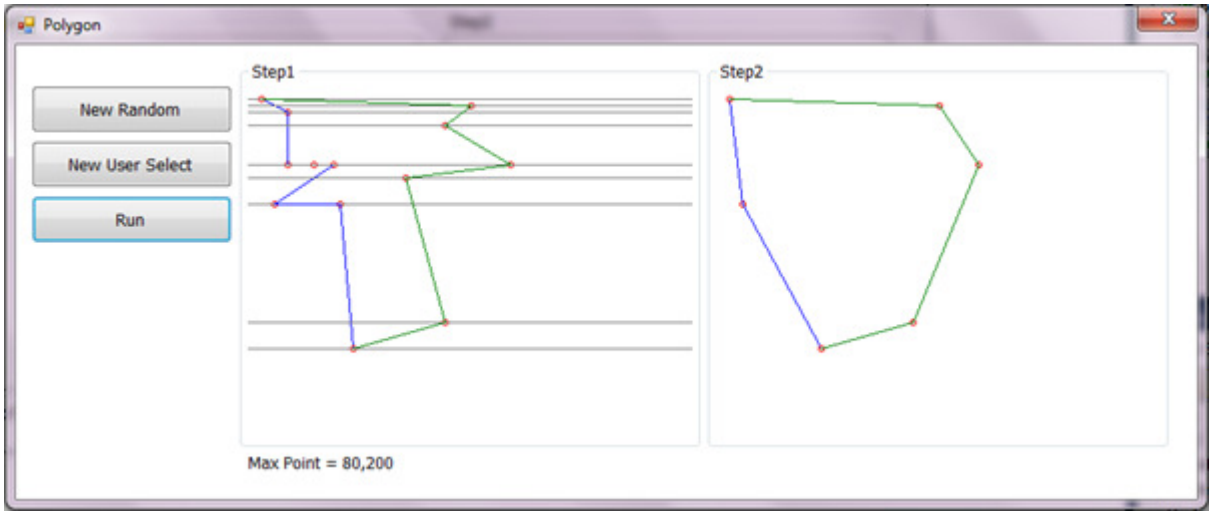

Figure 14: Demo 3 of our algorithm

8 Complexity analysis

$T(n) = n + T(n_1) + T(n_2)$, if $n_1 + n_2 \leq n$. Likewise quick sort, this has expected running time $O(nlogn)$, but worst-case time $O(n^2)$. Although it may seem that the time complexity of the loop is $O(n^2)$, because for each point it goes back to check if any of the previous points make a "right turn", it is actually $O(n)$, because each point is considered at most twice in some sense. Note that each point can appear only once as a point in a "left turn" because the algorithm advances to the next point after that), and as a point in a "right turn" (because the point is removed). If at each step almost all points are located at one side of PMAX, we get close to the worst-case. Since $T(n) = T(n-1) + n$ which sums to $T(n) = O(n^2)$.

But if points are distributed randomly, without any extra assumptions, this algorithm is going to be fast enough and runs optimally. In fact in this case $T(n) = 2T(n/2) + n$ which leads to $T(n) = O(nlogn)$.

9 Conclusion

This algorithm is new and different from existing ones for constructing convex hull, it has average case running time $O(nlogn)$. Our algorithm is working on points on two different sides, which makes it different from Graham scan and performs better for random points.

In fact we implemented two systems for taking the input points, totally random and user specified points.

References

[1] Andrew, A. M., Another efficient algorithm for convex hulls in two dimensions. Inform. Process. Lett., 9(1979) 216-219.

[2] Anon, W. P., The Dark Ages books, brute force algorithm, 2009.

[3] Brodal, G. S., Jacob, R., Dynamic planar convex hull, Proc. 43rd Annu. IEEE Sympos. Found. Comput. Sci., (2002) 617-626.

[4] Bykat, A., Convex hull of a finite set of points in two dimensions, Inform. Process. Lett., 7(1978) 296-298.

[5] Chan, T. M., Dynamic planar convex hull operations in near-logarithmaic amortized time, J. ACM, 48(2001), 1-12.

[6] Chand, D. R., Kapur, S. S, An algorithm for convex polytopes, J. ACM, 17(1970) 78-86.

[7] DeBerg, M., Cheong, O., Kreveld M. V. and Overmars, M., Computational Geometry (Algorithm and Applications), Third Edition, ACM Computing Classification (1998), Springer-Verlag Berlin Heidelberg.

[8] Eddy, W. F. , A new convex hull algorithm for planar sets, ACM Trans. Math. Softw., 3(1977) 398403 and 411-412.

[9] Gamby, A.N., Katajainen, J., Convex-Hull Algorithms: Implementation, Testing, and Experimentation, Algorithms, 11(2018) 1–27.

[10] Gomes, A.J.P., A total order heuristic-based convex hull algorithm for points in the plane, Comput. Aided Des., 70(2016) 153-160.

[11] Graham, R. L., An efficient algorithm for determining the convex hull of a finite planar set, Inform. Process. Lett., 1(1972) 132-133.

[12] Hershberger, J., Suri, S., Off-line maintenance of planar configurations, J. Algorithms, 21(1996) 453475.

[13] Jarvis, R. A., On the identification of the convex hull of a finite set of points in the plane, Inform. Process. Lett., 2(1973) 18-21.

[14] Kirkpatrick, D. G., Seidel, R., The ultimate planar convex hull algorithm?, SIAM J. Comput., 15(1986) 287-299.

[15] Kristensen, L. M., Mailund, T., A Generalised Sweep-Line Method for Safety Properties, Springer-Verlag Berlin Heidelberg (2002) 549-567.

[16] Overmars, M. H., Van Leeuwen, J., Maintenance of configurations in the plane, J. Comput. Syst. Sci., 23(1981) 166-204.

[17] Preparata, F. P., Hong, S. J., Convex hulls of finite sets of points in two and three dimensions, Commun. ACM, 20(1977) 87-93.