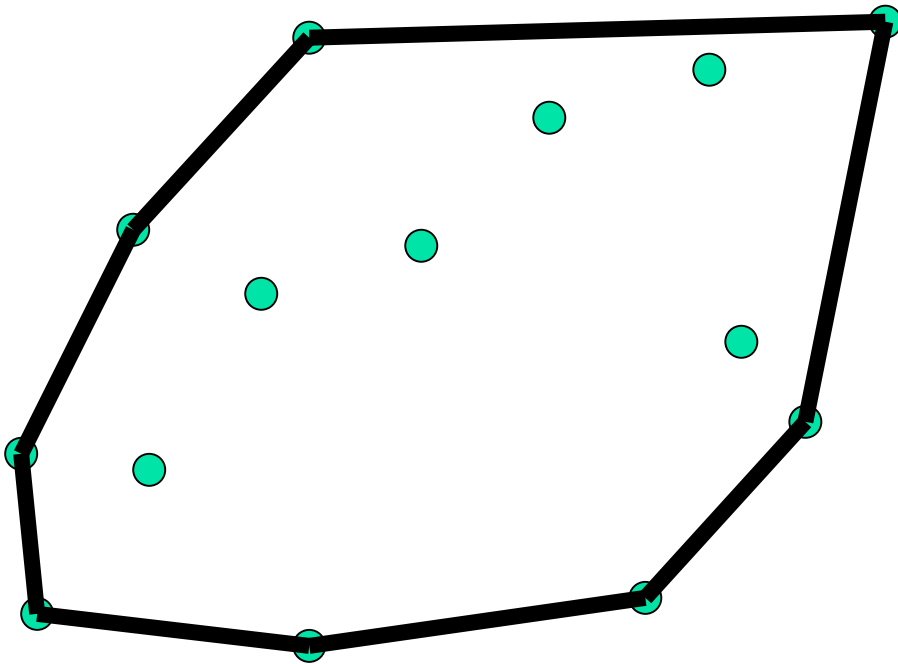


The convex hull



Proseminar by Sabrina Steffens

Date 21.06.2001

Contents

Definition of the convex hull

- Example for the necessity of the convex hull

Different algorithms

- **Jarvis's March** **03**
Running time **03**

- **Graham's Scan** **04**
Correctness of the Graham Scan **05**
Running time **05**

- **Interior Elimination** **05**
Advantages **06**
Running time **06**

Lower bound of the convex hull **06**

Attachments **07**

Sources **09**

Definition of the convex hull

The convex hull $CH(Q)$ of a finite set of points is the smallest convex polygon that contains all points. All points of the set have to be inside the hull or on its boundary. The convex hull can contain only three points, then it forms a large triangle, so that all the other points are in the interior of this triangle.

Convex means that all connections between all points of the set have to be part of the interior or the boundary of the hull.

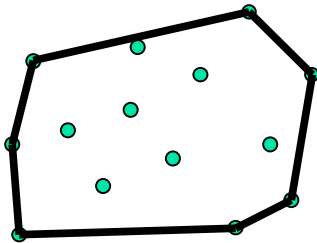
If a vertex is added or removed from the convex polygon, it still remains convex.

If three points form a triangle around a fourth point, consequently, that the fourth point isn't an vertex of the convex hull.

The convex hull can contain only a line segment, that means all the points appear on a single line. It would be the definition of the programmer which points are on the convex hull. You can decide that all points are on the boundary or just the end points.

If you insist of having all the points on the boundary it would be more work.

[GT] [PS] [S]



Example for the necessity of the convex hull

Assume there is a given field of trees and you have to build a fence around the trees.

The fence should be as tight and small as possible. What you can do is you can look at the trees as a set of points and if you compute the convex hull of these set you will automatically get the tightest fence. (Of course you wouldn't built it that way!)

Different algorithms

Jarvis's March

This algorithm is called Gift Wrapping or Package Wrapping Algorithm, too. It was invented by Jarvis in 1973. I will talk about Jarvis's March how it computes the convex hull in the plane. Jarvis's March can be parted in three steps:

1. First you have to find a starting point p_0 . This starting point has to be a vertex of the convex hull, otherwise the algorithms would not work correctly. Normally you use the lowest y- coordinate as point p_0 .
2. Pull a rope to the right of p_0 to an other point, you always take the least angle between the points. This step is called **the wrapping step**. If there are more points with the same angle, choose the one with the maximum distance.
3. Continue until the package or gift is wrapped. The important thing is, every wrapping step has to be on a left turn. This can be realized by taking an radial comparator, which tests if three current points take a left turn. Rational means that the comparator checks in rays if whether a point is or is not lower than another according to the starting point. On that way each point that is not part of the boundary of the convex hull will be eliminated. Jarvis's March will terminate when the starting point is reached.

[GT] [CLR] [S]

Running time

Let n be the number of points of the set and $v \leq n$ the number of the vertices of $CH(Q)$.

As you can divide the algorithm in different steps you can also divide the proof for the running time.

1. Finding the starting point p_0 takes $O(n)$ time, because Jarvis's March checks every point of the set, if it has a lower y - coordinate than the predecessor.
2. In every wrapping step the comparator has to check every angle from the current point to all the other points except the predecessor. So the running time of the wrapping step is $O(vn)$. In the **worst case** it can be a running time of $O(n^2)$. This happens if the number of points in the set is the number of vertices ($n = v$). In respect to this Jarvis's March is called an **output sensitive**. It means the running time depends on the size of the output. Consequently this algorithm is only effective for small sets.

[CLR] [GT]

The Graham Scan Algorithm

The Graham Scan Algorithm was invented in 1972 by R. L. Graham. It can be divided in three different steps as well:

1. You have to find a starting point under the same conditions you did it with Jarvis's March.
2. Now, the points have to be sorted by their angles in respect to the starting point. For this you use a sorting algorithm like mergesort. This is the main difference to Jarvis's March. You sort first and not in every iteration.
3. Now you start to scan through the sorted points beginning with the starting point p_0 . The algorithm is now pushing the first three points on a stack. Then it checks if the fourth point, takes a left turn together with the second and the third point. In case it does, the point will be pushed on the stack, too. If it does not, the third point will be popped from the stack and the fourth will be pushed on the stack, but only if it takes a left turn together with the first and the second point. On that way more than one point can be removed from the stack, as long as there is a left- turn again. That will continue until the starting point is reached again. Then the algorithm terminates. All points, that remain on the stack after termination, are the vertices on the convex hull in counterclockwise order.

[CLR] [GT] [S]

Correctness of the Graham Scan

If the Graham Scan is running on a set Q of points, $|Q| \geq 3$, then a point of Q is on the stack S at termination, iff it is a vertex of the convex hull of Q .

Proof:

First Case: Each point that is popped from the stack, is not a vertex on the convex hull.

What we already know from the definition is that a point that lies inside a triangle of three other points of the set is not a vertex of the convex hull. (also see Fig.01)

Second case: To show that each point on S is a vertex of $CH(Q)$ after termination.

Assume that all points of S are always vertices of the convex polygon. \Rightarrow All will be pushed on S and never removed from S , because of the comparison in respect to the left turn and least angle \Rightarrow All points on S are vertices of the convex hull after termination.

[CLR]

Running time

1. As for Jarvis's March finding the starting point takes $O(n)$ time.
2. Sorting points takes $O(n \log n)$ time, because the lower bound for sorting is $\Omega(n \log n)$.
3. Scanning through points, takes in the **worst case** $2n$. But this happens only if a point has to be removed from the stack. First a point has to be compared to an other point that takes $1n$ time. Secondly the point has to be removed from the stack, that takes $1n$ time, too.
4. In the worst case it takes $O(n \log n)$ time to get through the whole algorithm.

[GT]

Interior Elimination

This method is fast for deleting many points, in case you have a set with many points. You have to choose 4 points at the corners of the set. If you choose points which form a rectangle the test will be a little bit faster, if the points lie in the interior. The rectangle has to be parallel to the x- and y- axes. This would be the best case. But it's also possible to find a triangle, instead of a rectangle. Only the testing of the points would be a little harder. Then link the points together and delete everything that lies in the interior (middle) of the points. In the implementation there is just an easy test, if the current point lies in the rectangle. If it does it will be removed, if it doesn't it stays where it is.

So it's easy to understand why the points don't have to be vertices on the convex hull. This algorithm is only used to compute a smaller set than the original one. At this point there is of course no convex hull computed. If you want to compute one, you have to run another algorithm like Graham Scan. It will compute the convex hull of the smaller set, which will be the convex hull of the original set. (Fig.02)

[S]

Advantages

The Interior Elimination works good for randomly distributed points in a rectangle. The costs are low and the possible savings are high. You just have to run a simple test and most of the points are eliminated, during the run of the algorithm. To reduce the problem you should also take the interior elimination before the Graham Scan to eliminate as many points as possible. [S]

Running time

To eliminate a points in interior it takes $O(n)$ time, because you look at every point just once. Then you remove it or not. After that you don't care about it anymore and move on. No matter which algorithm used later (not with Jarvis's March!), the **worst case** would run in $O(n \log n)$ time. n because now the numbers of points is smaller than the original number n . [S]

Lower bound of the convex hull

Is it really true that finding the ordered $CH(A)$ of n points, requires $\Omega(n \log n)$ time? Suppose we have an unsorted set A and $(x_i, x_i^2) \in A$ while $x_i \geq 0$. You can see from Figure 3, that all points lie on the parabola, so that $y = x^2$. Assume that it is possible to compute the convex hull of the set A faster than $\Omega(n \log n)$. This means there is a lower bound than $\Omega(n \log n)$. Computing $CH(A)$ from Figure 03 would only mean, sorting the set A . \Rightarrow Sorting a set is faster than $\Omega(n \log n)$. This is a contradiction, because we already know from the lecture that sorting has a lower bound $\Omega(n \log n)$. \Rightarrow The lower bound for computing the convex hull of a set is $\Omega(n \log n)$.

[PS] [BKOS]

Attachments

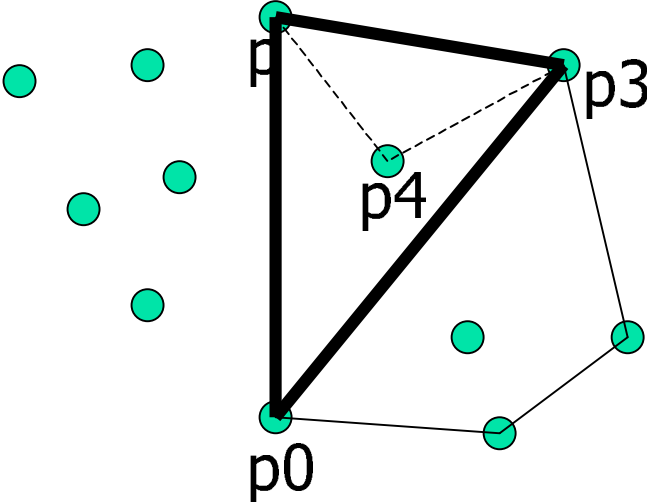


Figure 01

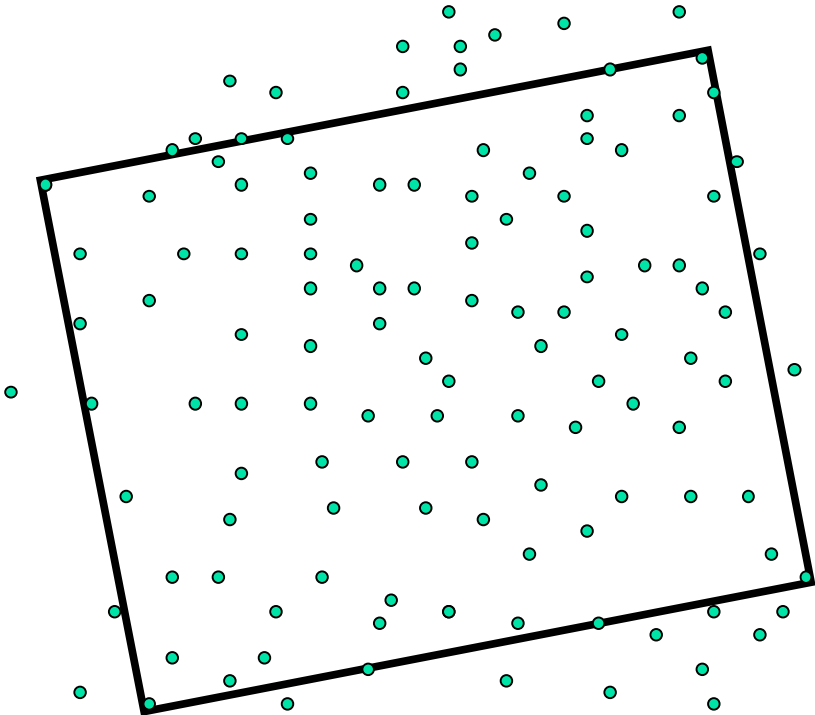


Figure 02

$A = \{(x_i, x_i^2)\}$ Reduction of sorting

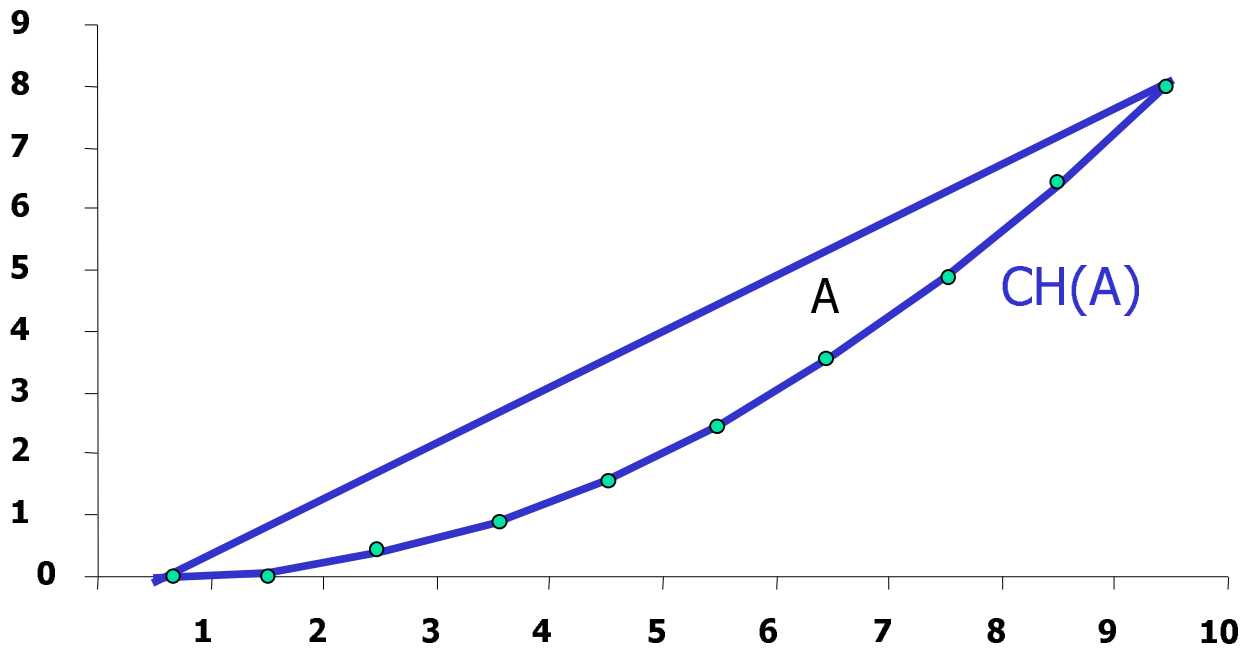


Figure 03

Sources:

1. M. de Berg; M. van Kreveld; M. Overmars; O. Schwarzkopf: Computational Geometry- Algorithms and Applications; Springer Verlag; Berlin, Heidelberg; 1997, 2000; s.E.; p. 13f. [Some really small addition to the theme.](#)
2. T.H. Cormen; C. E. Leiserson; R.L. Rivest: Introduction to Algorithms; The MIT- Press; Cambridge, Mass; 1990; p. 898- 907 [CLR] [This book helps a lot it's not to hard to understand. It's more than an introduction to the theme!](#)
3. M.T. Goodrich; R. Tamassia: Data Structures and Algorithms in Java; John Wiley & Sons, Inc.; 1998; § 15.2 [GT]
[This is the best book it has many details on the convex hull and really good example, the only bad thing is that is doesn't mention the interior elimination.](#)
4. F.P. Preparata; M. I. Shamos: Computational Geometry; Springer; New York; 1988; p. 99- 110 [PS]
[The book describes the matter in an more mathematik way, but it helped, and is not to hard to understand.](#)
5. R. Sedgewich: Algorithms; Addison- Wesley; 1988; p. 359- 372 [S]
[This book is hard to understand, especially the chapter about the interior elimination! Sometimes there could be more details.](#)