Geometric Algorithms – The Convex Hull Problem in 2 & 3 Dimensions

Rehman, M. Usaid   Ali, Syed Anus   Ozair, Faraz

Habib University

August 21, 2021

Outline

Outline

The Paradigm of Computational Geometry

- A sub-field of algorithm theory involving design and analysis of algorithms with geometric input and output.
- Mostly focuses on the discrete aspect of geometric problem solving.
- Primarily deals with straight or flat objects or simple curves.
- Computational geometry aims to provide basic geometric tools from which application areas can build algorithms.
- It also aims to provides theoretical analytical tools to analyze the performance of these algorithms.

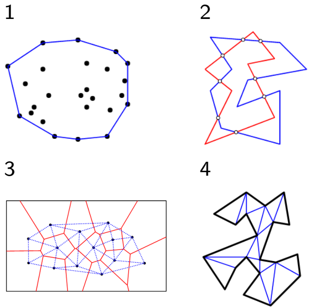
Outline

A Brief History & Timeline

- Geometry has been central to mathematics since the time of the Ancient Greeks and the Ancient Egyptians.
- Euclid's *Elements* had a profound influence on the study of geometry.
- Introduction of coordinates permitted an increase in computational power.
- The term "Computational Geometry" was coined first by Marvin Minsky in his book "Perceptrons" in 1969.
- Multiple application areas have been the incubation bed for this discipline.
- These problems include Euclidean traveling salesman, minimum spanning tree, linear programming etc.

Famous Problems in Computational Geometry

1. Convex Hulls
2. Intersections
3. Voronoi Diagrams and Delaunay Triangulations
4. Triangulation and Partitioning
5. Optimization and Liner Programming
6. Geometric Search Problems
   - Nearest-neighbor searching
   - Point location



Outline

Basics of Euclidean Geometry

- The main objects considered in computational geometry are sets of points in Euclidean space.
- Sets of points are finite, or at least *finitely specifiable*.
- By $\mathbb{R}^d$, we denote the *d-dimensional Euclidean space*.
- A d-tuple $(x_1, \ldots, x_d)$ denotes a point of $\mathbb{R}^d$ where $x_i \in \mathbb{R}$.
- A *polygon* in $\mathbb{R}^2$ is defined by a finite set of line segments where each extreme point is shared by exactly two segments called *edges*.
- A *polyhedron* in $\mathbb{R}^3$ is defined by a finite set of plane polygons such that every edge of a polygon is shared by only one other polygon.

Outline

What is a Convex Hull?

- A *convex set* is a subset of Euclidean space where given any two points in the set, the set contains the whole line segment joining them.
- A *convex hull* of a shape is the smallest convex set containing the shape.
- The convex hull can also be defined as the intersection of all convex sets of a given subset of Euclidean space.

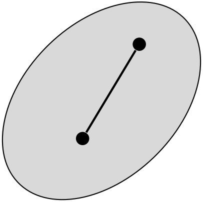


Figure: A convex set

Mixing Things

- Imagine you have the following mixtures of substances:

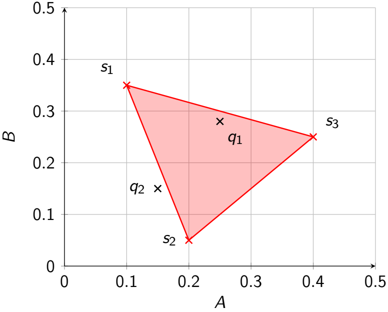
subst.	fract. A	fract. B
s_1	10%	35%
s_2	20%	5%
s_3	40%	55%

- Using these substances can we create two other substances s q_1 and q_2?

q_1	25%	28%
q_2	15%	15%

Mixing Things

- Let's plot these values:



- We can observe that a substance $q \in \mathbb{R}^2$ can be created using substances in $S \subset \mathbb{R}^2$ if and only if $q \in \mathrm{CH}(S)$.

Outline

What is the Convex Hull Problem?

- The convex hull problem is formally defined as:

  Given a set S of N points in $\mathbb{R}^d$, construct its convex hull $\mathrm{CH}(S)$.

- Another version of the convex hull problem is:

  Given a set S of N points in $\mathbb{R}^d$, identify those that are vertices of $\mathrm{conv}(S)$.

- Both versions of the problem have the same asymptotic hardness since one problem can be transformed to the other.

Convex Hull for Finite Set of Points

- The convex hull for a finite set of points in the plane produces a convex polygon.

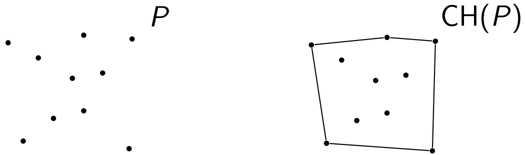


Figure: A point set and its convex hull.

- In 3 dimensions however, the convex hull is not a convex polygon, but instead a convex polyhedron – generalizes to a polytope in higher dimensions.

Outline

Lower Bound on Computational Complexity

Overview and Approach

- The convex hull problem outputs a polygon – a cyclic enumeration of the boundary vertices.
- We can say that we have to produce a sorting of the vertices. which has a lower bound of $\Omega(n \log n)$.
- To find lower bound of convex hull problem, we will reduce the sorting problem to the convex hull problem in $O(n)$ time, which implies:

Theorem

Any algorithm for the convex hull problem requires $\Omega(n \log n)$ time in the worst case.

Lower Bound on Computational Complexity

Reduction of Sorting to Convex Hull

- Let $X = \{x_1, \ldots, x_n\}$ be the n values we wish to sort.
- We can "lift" each of these points onto a parabola $y = x^2$, by mapping x_i to the point $p_i = (x_i, x_i^2)$.
- P is the resulting set of points – all of which lie on the convex hull.
- The sorted order of points along the lower hull is the same as the sorted order X.

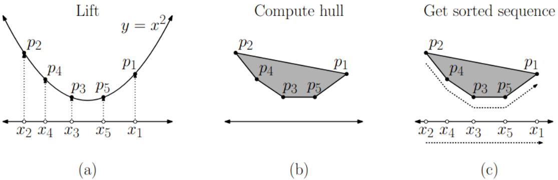


Figure: Reduction from sorting to convex hull.

Output-Sensitive Algorithms

- Some convex hull algorithms depend both on n input points, and h output points.
- Such algorithms are called *output-sensitive* algorithms.
- They may be asymptotically more efficient than $\Theta(n \log n)$ algorithms when $h = o(n)$.
- The lower bound for these algorithms is $\Omega(n \log h)$ in the planar case.
- Kirkpatrick and Seidel devised an algorithm that achieved this in 1986.
- In 1996, a much simpler algorithm was devised by Chan called Chan's algorithm.

Outline

The General Position Assumption

- Before we look at algorithms, let's look at an important assumption.
- We assume that the points are in *general position*.
- This means that degenerate configurations do not arise in input.
- Point sets in general position maintain general position if perturbed infinitesimally.
- Merely a convenience to avoid dealing with a lot of special cases during algorithm design.

Convex Hull Algorithms

- There are multiple algorithms that employ different techniques to compute the convex hull of a point set.
- Following are some of these algorithms with their time complexity:

Algorithm	**Time Complexity**
Jarvis March	$O(nh)$
Graham Scan	$O(n \log n)$
Quickhull	$O(n \log n)$
Andrew's Algorithm	$O(n \log n)$
Kirkpatrick-Seidel Algorithm	$O(n \log h)$
Chan's Algorithm	$O(n \log h)$

Table: Some algorithms and their complexities.

Jarvis March

Algorithm Description & Visualization

- Jarvis march a.k.a the gift wrapping algorithm, was published in 1973 by R. A. Jarvis.
- The algorithm begins with $i \leftarrow 0$, and a point p_0 known to be on the convex hull.
- Then it selects the point p_{i+1} such that all points are to the right of the line segment $p_i p_{i+1}$.
- This can be done in $O(n)$ time by comparing polar angles.
- Let $i \leftarrow i + 1$, and repeat until algorithm $p_h \leftarrow p_0$ again.
- This would yield the convex hull in h steps.
- Here is a link to an animation for Jarvis march.

Jarvis March

Performance & Time Complexity

- Jarvis march is among the least efficient algorithms for convex hull.
- The algorithm loops h times and within each iteration, it loops n times.
- Therefore, the time complexity is $O(nh)$.

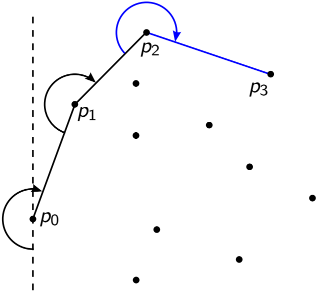


Figure: The Jarvis march procedure.

Graham Scan

Algorithm Description & Visualization

- Based on the *incremental construction* technique.
- First, select the lowest point in the point set P called point p_0.
- Sort all points cyclically w.r.t point p_0 based on polar angles.
- For each point, determine if travelling to that point constitutes a left turn or a right turn.
- If right turn: second-to-last point is not part of CH(P) but lies inside.
- Do the same for previously visited points until 'left-turn set' is encountered.
- Repeat until starting point is reached in a counter-clockwise fashion.
- Link to visualization.

Graham Scan

Pseudocode

Algorithm 1: GRAHAM SCAN

Input: A point set P

Output: The convex hull of P, CH(P)

let p_0 be the minimum and left-most point in P
let $\langle p_1, p_2, \ldots, p_n \rangle$ be the remaining points in P sorted by polar angle in counterclockwise order around p_0
let S be an empty stack
PUSH(p_0, s)
PUSH(p_1, S)
PUSH (p_2, S)
for $i \leftarrow 3$ *to* $i \leftarrow n$ **do**
    while *the angle formed by points TOP(S) NEXT_TO_TOP(S), and p_i make a non-left turn* **do**
        POP(S)
    PUSH(p_i, S)
return S

Graham Scan

Performance & Time Complexity

- Graham Scan is an efficient algorithm compared to Jarvis march.
- The sorting procedure is done in $O(n \log n)$ time.
- The loop is $O(n)$ and not $O(n^2)$.
- We visit each point at most twice – once as a left turn and once as a right turn.
- The time to sort dominates the time complexity of the algorithm.
- Therefore, Graham scan is $O(n \log n)$.

Quickhull Algorithm

The Philosophy Behind the Algorithm

- The Quickhull algorithm is a divide-and-conquer algorithm.
- It is based on the strategy of triangular expansion.
- It is also used for computing convex hull in higher dimensions due to its efficiency.
- Quickhull also uses the prune-and-search approach, where the input size is reduced by a constant factor at each step.
- Its working can be visualized here.

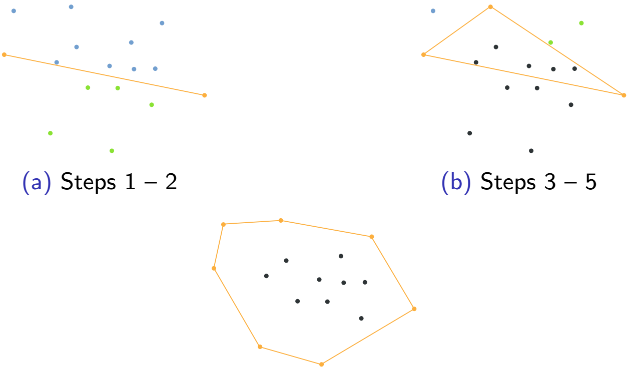
Quickhull Algorithm

Pseudocode for 2-Dimensional Quickhull

1. Find the points with minimum and maximum x-coordinates.
2. Use the line formed by the two points to divide the set into two subsets of points.
3. Determine the point, on one side of the line, with the maximum distance from the line — forms a triangle with those of the line.
4. The points lying inside of that triangle cannot be part of the convex hull and can be ignored in later steps.
5. Repeat the previous two steps on the lines formed by the triangle (recursion).
6. Keep on doing so until no points are left. The points selected constitute the convex hull.

Quickhull Algorithm

Visualization of Algorithm



(a) Steps 1 – 2

(b) Steps 3 – 5

(c) Step 6

Quickhull Algorithm

Performance & Time Complexity

- Finding the extreme points is an $O(n)$ operation.
- Quickhull divides into two sub-problems.
- In the best case, the partition size is well-balanced:

$$T(n) = 2T(n/2) + O(n)$$

$$T(n) = O(n \log n)$$

- Whereas, in the worst case this degenerates to $T(n) = nT(n-1) + cn$, which means the worst case is quadratic.

The 3D Convex Hull Problem

- Recall that in 3 dimensions, the convex hull is a polyhedron (3-polytope).
- In this case, we are concerned with the *facets* of a polytope.
- The convex polytope is the convex hull of all its vertices.
- More complicated problem which is harder to visualize and implement.

Figure: A 3-dimensional convex hull.

The Gift-Wrapping Algorithm

- The most famous and the simplest algorithm is the gift wrapping algorithm.
- It is the extension of the 2-dimensional Jarvis march algorithm.
- The steps of the algorithm are:
  1. Find a facet that is guaranteed to be on the convex hull.
  2. Repeat:
     1. Find an edge e of a face f that's on the CH, and such that the face on the other side of e has not been found.
     2. For all remaining points π, find the angle (e, π) with f.
     3. Find point π with the minimal angle; add face (e, π) to CH.
- The algorithm runs in $O(n \cdot F)$ time where F is the number of faces on the convex hull.

Other Approaches

- Approaches regarding 3-dimensional convex hulls mostly extend to N dimensions.
- The N-dimensional convex hull is an N-polytope.
- The gift-wrapping algorithm extends to N-dimensions.
- There are also other methods such as:
  - N-dimensional Quickhull
  - Chan's Algorithm
  - Divide-and-Conquer
  - Beneath-and-Beyond Method

References

M. I. Shamos & F. P. Preparata
Computational Geometry: An Introduction.
Springer New York, 1985.

M. de Berg (ed.)
Computational Geometry: Algorithms and Applications.
Springer, 2010

T.H. Cormen, C.E. Leiserson, R. Rivest, C. Stein
Introduction to Algorithms
MIT Press, 1989