Computational Geometry by David Mount

{Lecture 3: Convex Hulls in the Plane}

For any $d \geq 1$, let $\mathbb{R}^d$ denote real d-dimensional space, that is, the set of d-dimensional vectors over the real numbers.

Convexity. A set $K \subseteq \mathbb{R}^d$ is convex if given any points $p, q \in K$, the line segment $\overline{pq}$ is entirely contained within K.

Open/Closed: A set in $\mathbb{R}^d$ is said to be open if it does not include its boundary.
A set that includes its boundary is said to be closed.

Boundedness: A convex set is bounded if it can be enclosed with a sphere of a fixed radius. Otherwise, it is unbounded.

Convex body. A closed, bounded convex set is called a convex body.

Support line/hyperplane: An important property of any convex set K in the plane is that at every point p on the boundary of K, there exists at least one line l that passes through p such that K lies entirely in one of the closed halfplanes defined by l

Convex hull. The convex hull of any set P is the intersection of all convex sets that contains P, or more intuitively, the smallest convex set that contains P. We will denote this conv(P).

Convex Hull Problem

The (planar) convex hull problem is, given a set of n points P in the plane, output a representation of P's convex hull. The convex hull is a closed convex polygon, the simplest representation is a counterclockwise enumeration of the vertices of the convex hull

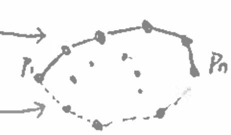
Graham's scan. $O(n \log n)$
(1970s)

incremental construction: add points in increasing order of x-coordinate.
(assume no duplicate x-coordinates)

represent the boundary of the convex hull as two polygon chains

— upper hull

— lower hull



Let $\langle p_1, \ldots, p_n \rangle$ denote sorted sequence (sort by x).

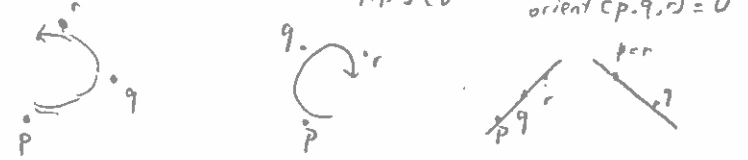
For i ranging from 1 to n. we will store the vertices of the current upper hull on a stack S, where the top of the stack corresponds to the most recently added point of P, and the bottom of the stack will be p_1.

Turning and orientations: "left-hand turn"

Given an ordered triple of points $\langle p, q, r \rangle$ in the plane, we say that they have
positive orientation if they define a counterclockwise oriented triangle.
negative orientation if they define a clockwise oriented triangle.
zero orientation if they are collinear

Note that orientation depends on the order in which the points are given

orient$(p,q,r) > 0$ orient$(p,q,r) < 0$ orient$(p,q,r) = 0$.



$$\text{Orient}(p,q,r) = \det \begin{pmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{pmatrix}$$

Orientation is formally defined as the sign of the determinant of the points given in homogeneous coordinates, that is, by prepending a 1 to each coordinate.

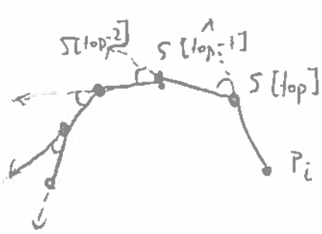
Given a sequence of three points p, q, r, we say that the sequence $\langle p, q, r \rangle$ makes a (strict) left-hand turn if Orient$(p,q,r) > 0$.

Graham's Scan

(1) Sort the points according to increasing order of their x-coordinates, denoted $\langle p_1, p_2, \ldots, p_n \rangle$

(2) push p_1 then p_2 onto S

(3) for $i \leftarrow 3, \ldots, n$ do
    (a) while ($|S| \geq 2$ and Orient(p_i, $S[top]$, $S[top-1]$) ≤ 0) pop S.
    (b) push p_i onto S.



Let P_i denote the subsequence consisting of the first i points.

Claim: After the insertion of the point p_i, the contents of S (from top to bottom) consist of the vertices of the upper hull of P_i in right to left order.

Running-time analysis:

Let d_i denote the number of points that are popped on processing p_i.
Because each orientation test takes $O(1)$ time the amount of time spent processing P_i is $O(d_i+1)$.
(The extra +1 is for the last point tested, which is not deleted).
Thus, the total running time is proportional to

$$\sum_{i=1}^{n} (d_i+1) = n + \sum_{i=1}^{n} d_i$$

To bound $\sum_i d_i$, observe that each of the n points is pushed onto the stack once.
Once a point is deleted it can never be deleted again. $\sum_i d_i \leq n$.

Thus after sorting, the total running time is $O(n)$
Since this is true for lower hull as well, to total time is $O(2n) = O(n)$.

Convex Hull by Divide-and-Conquer

The algorithm begins by sorting the points by their x-coordinate, in $O(n \log n)$ time. It splits the point set in half at its median x-coordinate, computes the upper hulls of the left and right sets recursively, and then merges the two upper hulls into a single upper hull. This latter process involves computing a line, called the upper tangent that is a line of support for both hulls.

$\boxed{\text{Divide-and-Conquer (Upper) Convex Hull}}$

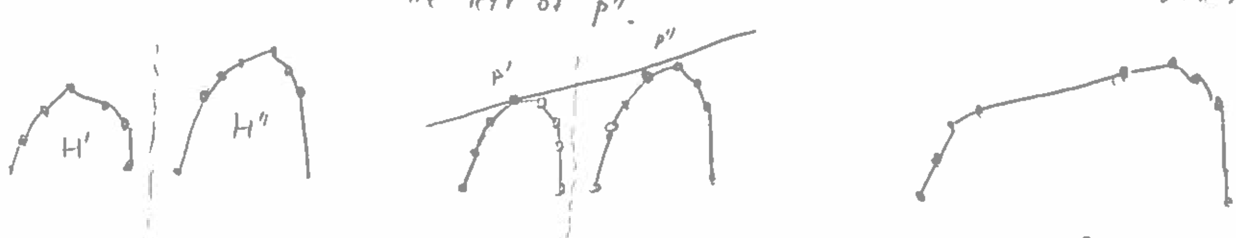
(1) If $|P| \leq 3$, then compute the upper hull by brute force in $O(1)$ time and return.

(2) Otherwise, partition the point set P into two sets P' and P'' of roughly equal sizes by a vertical line.

(3) Recursively compute upper convex hulls of P' and P'', denoted H' and H'', respectively.

(4) Compute the upper tangent $l = \overline{p'p''}$

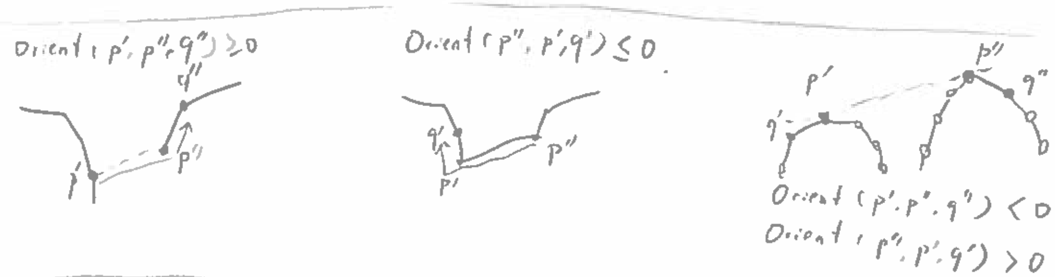
(5) Merge the two hulls into a single upper hull by discarding all the vertices of H' to the right of p' and the vertices of H' to the left of p''.



(a) (b) (c)

Upper Tangent (H', H''):

(1) Let p' be the rightmost point of H', and let q' be its predecessor.
(2) Let p'' be the leftmost point of H'', and let q'' be its successor
(3) Repeat the following until $\text{Orient}(p', p'', q'') < 0$ and $\text{Orient}(p'', p', q') > 0$:
  (a) while ($\text{Orient}(p', p'', q'') \geq 0$) advance p'' and q'' to their successors on H''
  (b) while ($\text{Orient}(p'', p', q') \leq 0$) advance p' and q' to their predecessors on H'
(4) return (p', p'').



The running time is $O(n)$.

Running-time analysis: Given an input of size n.

$$T(n) = \begin{cases} 1 & \text{if } n \leq 3 \\ n + 2T(n/2) & \text{otherwise} \end{cases}$$

$T(n) \in O(n \log n)$.

{ Lecture 4: Convex Hulls: Lower Bounds and Output Sensitivity }

Sorting is polytime reducible to convex hull $\longrightarrow$ lower bound $\Omega(n \log n)$

The reduction leaves open two questions:

1) What if we don't require that the points be enumerated in cyclic order, just that they all be identified? (even counting the number of points has an $\Omega(n \log n)$ lower bound)

2) What if all the points of P do not lie on the convex hull?

An algorithm whose running time is expressed as a function of both its input size and output size is called output sensitive.

Gift-Wrapping and Jarvis's March:
?

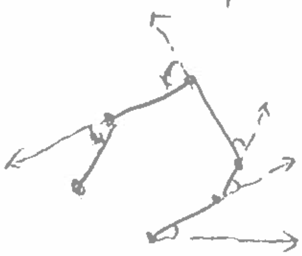
(can be seen as a variant on an $O(n^2)$ sorting algorithm called Selection Sort.

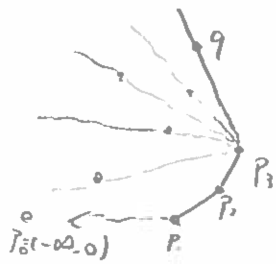
Jarvis's march:

Assuming that there are h vertices on the hull, this algorithm builds the hull in $O(nh)$ time by a process called "gift-wrapping".

It starts by adding one point of P that is guaranteed to be on the hull, say, the point with the smallest y-coordinate. It then repeatedly finds the "next" vertex on the hull in counterclockwise order.

Assuming that p_k and p_{k-1} were the last two points added to the hull, the algorithm finds the point $q \in P$ that maximizes $\angle p_{k-1} p_k q$. Clearly, we can find q in $O(n)$ time.



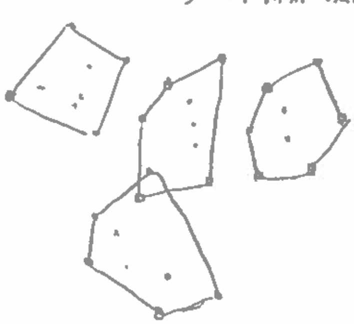


After repeating this h times, we will return back to the starting point and we are done. Thus, the overall running time is $O(nh)$. If $h = o(\log n)$ then this runs asymptotically faster than Graham's scan

☆ Chan's Algorithm: $O(n \log h)$

- It is a fast algorithm that is based on a combination of two slower algorithms Graham's and Jarvis's.
- It is based on "knowing" the final number of vertices on the convex hull. Since this number is not known, it adopts an interesting "guessing strategy" to determine its value (roughly). It is remarkable that the time to run the guess version is asymptotically as if you had known the number in advance.

Partition ($h^* = 8$) and mini-hulls



partition the set into n/h subsets, each of size h.

We can compute the convex hull of each subset in time $O(h \log h)$ by simply applying Graham's scan.

We call each of these a <u>mini-hull</u>

Total time. $O((\frac{n}{h}) \cdot h \log h) = O(n \log h)$

we had an estimate for h, call it h^*, where $h \le h^* \le h^2$

$O(n \log h^*) = O(n \log h^2) = O(n \log h)$

Merging the minis:

idea: run Jarvis's algorithm.

treat each mini-hull as if it is a "fat point".

At each step, compute the tangent lines of the current hull vertex to each of the mini-hulls including the mini-hull containing this vertex.

Among these tangents, we take the one that yields the smallest external angle.

Each mini-hull is a convex polygon

Lemma: Consider a convex polygon K in the plane and a point p that is external to K, such that the vertices of K are sorted in cyclic order in an array. Then the two tangents from p to K (more formally, the two supporting lines for K that pass through p) can each be computed in time $O(\log m)$, where m is the number of vertices in K.

$\boxed{\text{RestrictedHull}(P, h^*)}$:

(1) let $r \leftarrow \lceil n/h^* \rceil$.

(2) Partition P into disjoint subsets $P_1, P_2, \dots, P_r$, each of size at most h^*

(3) For $(i \leftarrow 1$ to $r)$

compute $\text{Hull}(P_i)$ using Graham's scan and store the vertices in an ordered array

(4) let $p_0 \leftarrow (-\infty, 0)$ and let p_1 be the bottommost point of P.

(5) For $(k \leftarrow 1$ to $h^*)$

(a) For $(i \leftarrow 1$ to $r)$

compute point tangent $q_i \in \text{Hull}(P_i)$, that is the vertex of $\text{Hull}(P_i)$ that maximizes the angle $\angle p_{k-1} p_k q_i$

(b) Let p_{k+1} be the point $q \in \{q_1, \dots, q_r\}$ that maximizes the angle $\angle p_{k-1} p_k q$

(c) If $p_{k+1} \leftarrow p_1$ then return $\langle p_1, \dots, p_k \rangle$ (success).

(6) (Unable to complete the hull after h^* iterations.) Return "Failure. h^* is too small."

Guessing the Hull's Size:

$h^* = 1,2,3,\dots,i \implies O(nh\log h)$

$h^* = 1,2,4,8,\dots,2^i \implies O(n\log^2 h)$ doubling search

$h^* = 2,4,16,64,\dots,2^{2^i} \implies O(n\log h)$ repeatedly squaring

we have $h \le h^* \le h^2$

$h_i^* = 2^{2^i}$ .. the ith trial takes time $O(n\log h_i^*) = O(n\log 2^{2^i}) = O(n2^i)$

we succeed as soon as $h_i^* \ge h$. $i = \lceil \lg\lg h \rceil$

Thus, the algorithm's total running time is

$$T(n,h) = \sum_{i=1}^{\lg\lg h} n2^i = n\sum_{i=1}^{\lg\lg h} 2^i$$

Note that $\sum_{i=0}^{k} 2^i = 2^{k+1}-1$

$$T(n,h) < n\cdot 2^{1+\lg\lg h} = 2n\cdot 2^{\lg\lg h} = 2n\lg h = O(n\log h)$$

$\boxed{\text{Hull}(P)}$

(1) $h^* \leftarrow 2$. $L \leftarrow$ fail

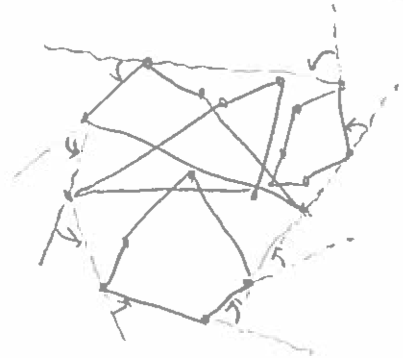
(2) while $(L = \text{fail})$

(a) Let $h^* \leftarrow \min((h^*)^2, n)$

(b) $L \leftarrow$ RestrictedHull(P, h^*)

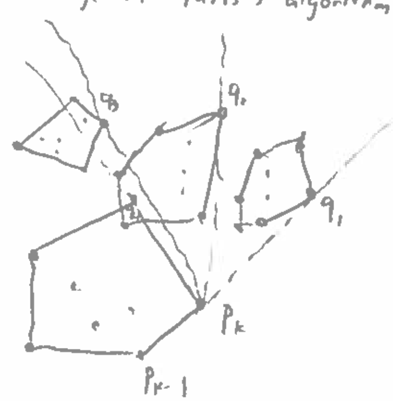
(3) Return L.

Jarvis's algorithm on mini-hulls



tangent

kth stage of Jaris's algorithm



Lower Bound.

We will give an $\Omega(n \log h)$ lower bound on the following simpler decision problem.

Convex Hull Size Verification Problem (CHSV):

Given a point set P and integer h, does the convex hull of P have h distinct vertices?

Assume the algorithm is allowed to compute any algebraic function of the input coordinates. The result is called an algebraic decision tree.

The input to the CHSV problem is a sequence of $2n = N$ real numbers. These numbers form a vector in real N-dimensional space, $(z_1, z_2, \dots, z_N) = \vec{z} \in \mathbb{R}^N$, which we call a configuration.

Each node of the decision tree is associated with a multivariate algebraic formula of degree at most d, where d is any fixed constant.

(Ben-Or) Theorem Let $Y \in \mathbb{R}^N$ be any set and let T be any d-th order algebraic decision tree that determines membership in W. If W has M disjoint connected components, then T must have height at least $\Omega((\log M) - N)$.

Multiset Size Verification Problem (MSV): Given a multiset of n real numbers and an integer k, confirm that the multiset has exactly k distinct elements.

Lemma: The MSV problem requires $\Omega(n \log k)$ steps in the worst-case in the d-th order algebraic decision tree.

Consider all the tuples $(z_1, \dots, z_n)$ with $z_1, \dots, z_k$ set to the distinct integers from 1 to k. $(k!)$ and $z_{k+1}, \dots, z_n$ each set to an arbitrary integer in the same range. (k^{n-k})

So there are at least $k!\, k^{n-k}$ different connected components.

$MSV \leq_p CHSV$.

Let $Z = (z_1, \dots, z_n)$ and k be an instance of MSV. We create a point set $\{p_1, \dots, p_n\}$ in the plane where $p_i = (z_i, z_i^2)$, and set $h = k$.