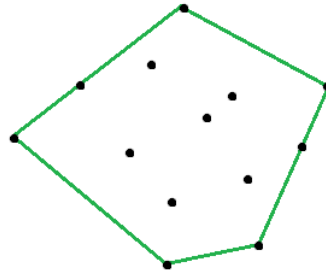


Convex Hulls

Natalia Bondarenko (Saratov State University)

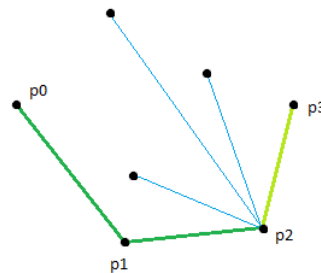
The **convex hull** of a given set of points is the smallest convex set that contains the given set.



2D Convex Hulls

Gift wrapping aka Jarvis march

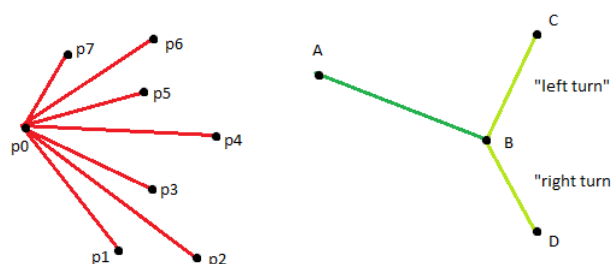
1. find the leftmost point p_0
2. at the step i , select the point p_{i+1} , such that all points are to the left of the line $p_i p_{i+1}$



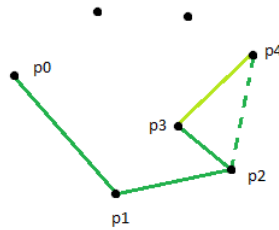
The time complexity is $O(nh)$, where n is the number of the given points, and h is the number of points on the convex hull. $O(n^2)$ in the worst case.

Graham scan

1. find the leftmost point p_0
2. sort the points p_1, p_2, \dots, p_{n-1} in increasing order of the angle they and the point p_0 make with the x -axis
3. add the point p_1 to the convex hull
4. at each step, determine, whether moving from the last two points of the convex hull to the next point in the sorted sequence is a “left turn” or a “right turn”



5. if a “left turn”, add the point to the convex hull
6. if a “right turn”, remove the last point from the convex hull, and keep doing this, while the last two points of the convex hull and the new point form a “right turn”, then add the new point



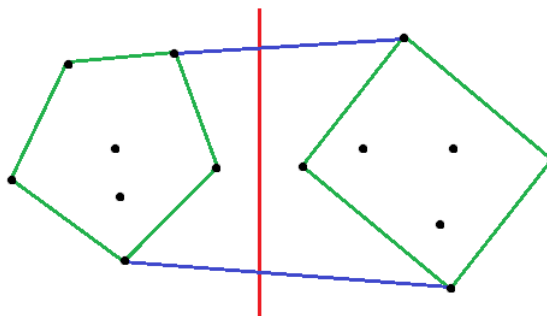
The time complexity is $O(n \log n)$ because of sorting. Other steps take $O(n)$ time.

Implementation details

- if you have duplicate points, discard them
- don't forget the cases, when the convex hull is a point or a segment
- do you need to include points lying on the edges of the convex hull?
- use the vector product to compare polar angles
- always remember, *devil in the details*

Divide and conquer

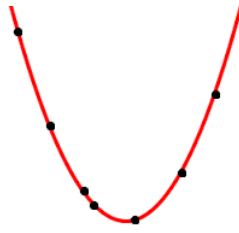
1. sort the points by x -coordinate and divide them into two equal parts by a vertical line
2. construct convex hulls for the both parts recursively
3. find “**bridge edges**”



$O(n \log n)$ complexity.

Minimal computational complexity

Consider points on a parabola, given in a random order.



The convex hull problem for them is equivalent to the sorting problem, so the minimal complexity is $O(n \log n)$. However, sometimes you can reduce it to $O(n)$, using the integer sorting, or if the points are sorted in advance.

There are **output-sensitive** algorithms with the complexity $O(n \log h)$.

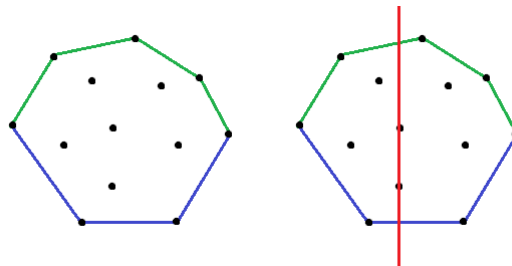
Kirkpatrick-Seidel algorithm

Kirkpatrick, David G.; Seidel, Raimund (1986). "The ultimate planar convex hull algorithm". SIAM Journal on Computing 15 (1): 287–299.

It reminds "divide and conquer" algorithm. The main idea: "marriage-before-conquest". Find the "bridge edges" in $O(n)$ time *before* doing recursion.

Suppose we can find "bridge edges" in $O(n)$ time.

1. find the leftmost and the rightmost points
2. separately construct upper-hull and lower-hull



3. to construct the upper-hull, find the median point by x -coordinate and draw a vertical line through it
 4. find a "bridge" over this line
 5. discard points under the "bridge", and construct left and right parts of the upper-hull recursively
- at the i -th level of recursion, there are 2^i subproblems
 - each subproblem has a size at most $\frac{n}{2^i}$
 - the total number of subproblems is at most h
 - the worst case: the complete binary tree of height $\log h$.

The total complexity is $O(n \log h)$.

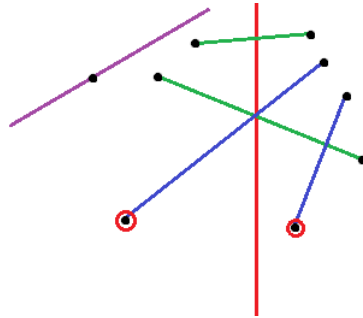
How to find a bridge?

1. split the points into pairs, let $x_1 \leq x_2$ in each pair
2. if $x_1 = x_2$, discard the point with the lower y -coordinate

3. for other pairs, determine the **slopes**:

$$k = \frac{y_1 - y_2}{x_1 - x_2}$$

4. find the median K of the set of k 's
5. draw a line with the slope K through every point, find the topmost line
6. check if it contains a bridge
7. otherwise you can discard $\frac{1}{4}$ -th of points



Indeed, let all the points on the topmost line with the slope K (the purple line in the figure) lie in the left half-plane with respect to the red line. Then the bridge has a smaller slope than K . Consider the segments with bigger slopes than K (blue ones in the figure). There are $1/2$ of such segments. Clearly, their lower points can not belong to the bridge. So we can discard these points (which are in the red circles). The case, when all the points on the topmost line lie in the right half-plane, is symmetrical.

In order to compute the time complexity of this method, one has to solve the recurrence

$$f(n) = f\left(\frac{3n}{4}\right) + O(n).$$

Chan's algorithm is another $O(n \log h)$ output-sensitive algorithm.

1. assume h is known a priori
2. divide the given set of points into $n/h + 1$ subsets with at most h points each
3. for each subset, compute a convex hull, using $O(n \log n)$ algorithm (it takes $O(n/h)O(h \log h) = O(n \log h)$ time)
4. use Jarvis march to join the precomputed convex hulls (h steps $\times n/h$ polygons at each step $\times O(\log h)$ operations of binary search $= O(n \log h)$)

What if we don't know h ?

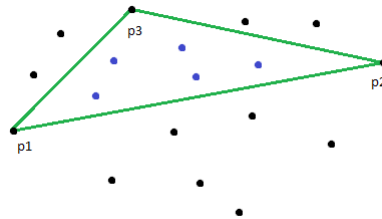
Try $h = 2^{2^t}$, $t = 0, 1, 2, \dots$

Then the total complexity is

$$\sum_{t=0}^{\lceil \log \log h \rceil} O(n \log 2^{2^t}) = O(n) \sum_{t=0}^{\lceil \log \log h \rceil} O(2^t) = O(n \log h).$$

QuickHull

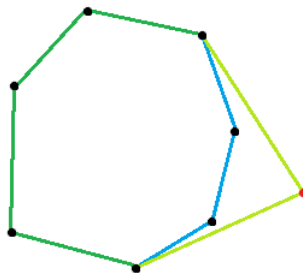
1. find the leftmost and the rightmost points, they belong to the convex hull
2. connect them by the base-line, and construct convex hulls for two subsets separated by the line recursively
3. determine the point with the maximum distance from the base-line, it belongs to the convex hull too
4. discard the points lying inside of the triangle
5. solve the problem recursively for new two edges of the triangle



The average case complexity is $O(n \log n)$, in the worst case it is $O(n^2)$.

Incremental algorithm

1. start with three points, forming a triangle
2. add other points one by one
3. if the point is outside the convex hull, delete all the edges that the new point **can see**, and connect the point with the remaining part of the convex hull



The time complexity is $O(n^2)$.

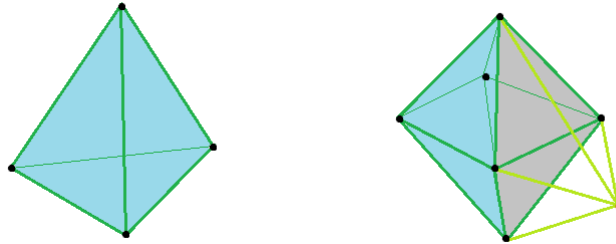
3D Convex Hulls

General facts

- a convex hull in 3D is a polyhedron
- its faces can be triangulated, so we will work with a set of triangles
- the number of edges and faces is $O(\text{Vertices})$, this is a consequence of the **Euler's formula** $\text{Vertices} - \text{Edges} + \text{Faces} = 2$

Incremental algorithm

1. start with a tetrahedron
2. add the points one by one
3. determine the (triangular) faces, that the new point **can see** (keep a plane equation $Ax + By + Cz + D$ with a certain orientation for each face)
4. if there are no such faces, the new point is inside the convex hull
5. otherwise you should delete some faces and add some new faces



6. when a face is being deleted, mark three corresponding edges
7. if an edge is marked twice, delete it
8. if an edge is marked only once, add a new triangular face, formed by this edge and the new point

The complexity is $O(n^2)$.

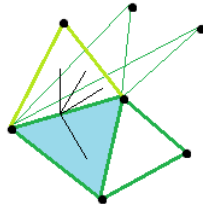
QuickHull

1. find four extreme points and construct a tetrahedron
2. the points inside the tetrahedron are discarded
3. other points are assigned to triangular faces, that they **can see**; each point is assigned to exactly one face
4. process the faces one by one, each of them has a point-list
5. find the most distant point in the face's point-list
6. this point can see other faces! delete all such faces
7. mark the edges of deleted faces
8. if the edge is marked twice, delete it
9. if the edge is marked only once, create a new face, formed by this edge and the new point
10. the points from the point-lists of the deleted faces should be reassigned to the new faces or discarded

The complexity is $O(n^2)$ in the worst case, $O(n \log n)$ in the average case.

Jarvis march

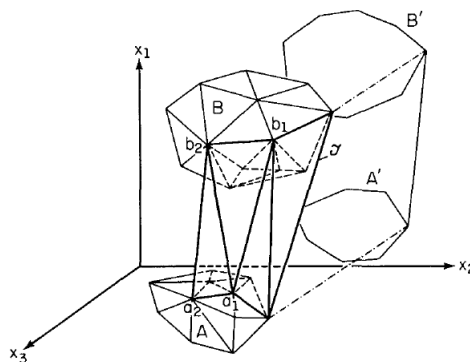
1. find an initial face
2. process its edges
3. the current edge form a plane with each of the rest points
4. choose such a point, that the angle between this plane and the current face is maximal
5. add a new face, and continue to process faces in breadth-first-search or in depth-first-search order



The complexity is $O(nh)$.

Divide and conquer (Preparata algorithm) works in the same way, as 2D divide-and-conquer. The most important question is: *how to merge two polyhedra in $O(n)$ time?*

1. project the polyhedra onto a plane and find a starting segment of the “cylinder”
2. try triangles, formed by the current segment and candidate points from the both polyhedra
3. every candidate point should be connected by an edge with one of the ends of the current segment
4. choose the plane, that forms the largest angle with the last constructed face of the “cylinder”



The figure (as well as the algorithm) is taken from the original research paper:
 Preparata F.P., Hong J.S. Convex Hulls of Finite Sets of Points in Two and Three Dimensions. Communications of the ACM, Volume 20, Issue 2, Feb. 1977, Pages 87-93.

The complexity of the algorithm is $O(n \log n)$.

Using this algorithm, Chan offers an $O(n \log h)$ output-sensitive algorithm in 3D.

Chan T.M. Optimal Output-Sensitive Convex Hull Algorithms in Two and Three dimensions. Discrete Comput. Geom. 16: 361-368 (1996).