Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
فارسی
Русский
Українська
Tiếng Việt
Edit links

Chan's algorithm

From Wikipedia, the free encyclopedia

In computational geometry, **Chan's algorithm**,[1] named after Timothy M. Chan, is an optimal output-sensitive algorithm to compute the convex hull of a set *P* of *n* points, in 2- or 3-dimensional space. The algorithm takes O(*n* log *h*) time, where *h* is the number of vertices of the output (the convex hull). In the planar case, the algorithm combines an O(*n* log *n*) algorithm (Graham scan, for example) with Jarvis march, in order to obtain an optimal O(*n* log *h*) time. Chan's algorithm is notable because it is much simpler than the Kirkpatrick–Seidel algorithm, and it naturally extends to 3-dimensional space. This paradigm[2] has been independently developed by Frank Nielsen in his Ph. D. thesis.[3]

Algorithm [edit]

Initially, we assume that the value of *h* is known and make a parameter *m*=*h*. This assumption is not realistic, but we remove it later. The algorithm starts by arbitrarily partitioning *P* into at most 1+*n*/*m* subsets Q with at most *m* points each. Then, it computes the convex hull of each subset Q using an O(*n* log *n*) algorithm. Note that, as there are O(*n*/*m*) subsets of O(*m*) points each, this phase takes O(*n*/*m*)O(*m* log *m*) = O(*n* log *m*) time.

The second phase consists of executing Jarvis's march and using the precomputed convex hulls to speed up the execution. At each step in Jarvis's march, we have a point p_i in the convex hull, and need to find a point $p_{i+1} = f(p_i,P)$ such that all other points of *P* are to the right of the line $p_i\, p_{i+1}$. If we know the convex hull of a set Q of *m* points, then we can compute $f(p_i,Q)$ in O(log *m*) time, by using binary search. We can compute $f(p_i,Q)$ for all the O(*n*/*m*) subsets *Q* in O(*n*/*m* log *m*) time. Then, we can determine $f(p_i,P)$ using the same technique as normally used in Jarvis's march, but only considering the points that are $f(p_i,Q)$ for some subset Q. As Jarvis's march repeats this process O(*h*) times, the second phase also takes O(*n* log *m*) time, and if *m*=*h*, O(*n* log *h*) time.

By running the two phases described above, we can compute the convex hull of *n* points in O(*n* log *h*) time, assuming that we know the value of *h*. If we make *m*<*h*, we can abort the execution after *m*+1 steps, therefore spending only O(*n* log *m*) time (but not computing the convex hull). We can initially set *m* as a small constant (we use 2 for our analysis, but in practice numbers around 5 may work better), and increase the value of *m* until *m*>*h*, in which case we obtain the convex hull as a result.

If we increase the value of *m* too slowly, we may need to repeat the steps mentioned before too many times, and the execution time will be large. On the other hand, if we increase the value of *m* too quickly, we risk making *m* much larger than *h*, also increasing the execution time. Chan's algorithm squares the value of *m* at each iteration, and makes sure that *m* is never larger than *n*. In other words, at iteration *t* (starting at 0), we have $m = \min(n, 2^{2^t})$. The total running time of the algorithm is

$$\sum_{t=0}^{\lceil \log\log h \rceil} O\left(n \log(2^{2^t})\right) = O(n) \sum_{t=0}^{\lceil \log\log h \rceil} O(2^t) = O\left(n \cdot 2^{1+\lceil \log\log h \rceil}\right) = O(n \log h).$$

To generalize this construction for the 3-dimensional case, an O(*n* log *n*) algorithm to compute the 3-dimensional convex hull should be used instead of Graham scan, and a 3-dimensional version of Jarvis's march needs to be used. The time complexity remains O(*n* log *h*).

Implementation [edit]

Chan's paper contains several suggestions that may improve the practical performance of the algorithm, for example:

- When computing the convex hulls of the subsets, eliminate the points that are not in the convex hull from consideration in subsequent executions.
- The convex hulls of larger point sets can be obtained by merging previously calculated convex hulls, instead of recomputing from scratch.

References [edit]

1. ^ Timothy M. Chan. "Optimal output-sensitive convex hull algorithms in two and three dimensions". *Discrete and*

Computational Geometry, Vol. 16, pp.361–368. 1996.
2. **^** Frank Nielsen. "Grouping and Querying: A Paradigm to Get Output-Sensitive Algorithms". *Discrete and Computational Geometry*, LNCS 1763, pp. 250–257, 2000.
3. **^** Frank Nielsen. "Adaptive Computational Geometry". Ph. D thesis, INRIA, 1996.

Categories: Convex hull algorithms