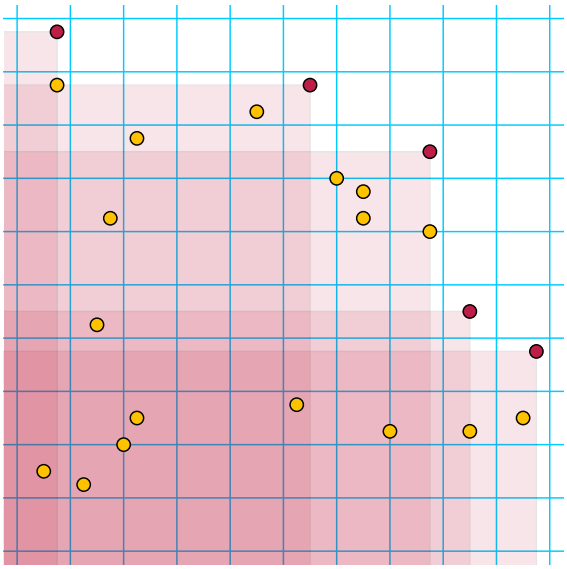
Maxima of a point set



The five red points are the maxima of the set of all the red and yellow points. The shaded regions show the points dominated by each of the five maxima.

In computational geometry, a point p in a finite set of points S is said to be *maximal* or *non-dominated* if there is no other point q in S whose coordinates are all greater than or equal to the corresponding coordinates of p. The **maxima of a point set** S are all the maximal points of S. The problem of finding all maximal points, sometimes called the **problem of the maxima** or **maxima set problem**, has been studied as a variant of the convex hull and orthogonal convex hull problems. It is equivalent to finding the Pareto frontier of a collection of points, and was called the **floating-currency problem** by Herbert Freeman based on an application involving comparing the relative wealth of individuals with different holdings of multiple currencies.[1]

1 Two dimensions

For points in two dimensions, this problem can be solved in time $O(n \log n)$ by an algorithm that performs the following steps:[1][2]

- Sort the points in one of the coordinate dimensions (the x-coordinate, say)
- For each point, in decreasing order by x-coordinate, test whether its y-coordinate is greater than the maximum y-coordinate of any previously processed point. (For the first point, this is vacuously true). If it is, output the point as one of the maximal points, and remember its y-coordinate as the greatest seen so far.

If the coordinates of the points are assumed to be integers, this can be sped up using integer sorting algorithms, to have the same asymptotic running time as the sorting algorithms.[3]

2 Three dimensions

For points in three dimensions, it is again possible to find the maximal points in time $O(n \log n)$ using an algorithm similar to the two-dimensional one that performs the following steps:

- Sort the points in one of the coordinate dimensions (the x-coordinate, say)
- For each point, in decreasing order by x-coordinate, test whether its projection onto the yz plane is maximal among the set of projections of the set of points processed so far. If it is, output the point as one of the maximal points, and remember its y-coordinate as the greatest seen so far.

This method reduces the problem of computing the maximal points of a static three-dimensional point set to one of maintaining the maximal points of a dynamic two-dimensional point set. The two-dimensional subproblem can be solved efficiently by using a balanced binary search tree to maintain the set of maxima of a dynamic point set. Using this data structure, it is possible to test whether a new point is dominated by the existing points, to find and remove the previously-undominated points that are dominated by a new point, and to add a new point to the set of maximal points, in logarithmic time per point. The number of search tree operations is linear over the course of the algorithm, so the total time is $O(n \log n)$.[1][2]

For points with integer coordinates the first part of the algorithm, sorting the points, can again be sped up by integer sorting. If the points are sorted separately by all three of their dimensions, the range of values of their coordinates can be reduced to the range from 1 to n without changing the relative order of any two coordinates and without changing the identities of the maximal points. After this reduction in the coordinate space, the problem

of maintaining a dynamic two-dimensional set of maximal points may be solved by using a van Emde Boas tree in place of the balanced binary search tree. These changes to the algorithm speed up its running time to $O(n \log \log n)$.[3]

3 Higher dimensions

In any dimension d the problem can be solved in time $O(dn^2)$ by testing all pairs of points for whether one dominates another, and reporting as output the points that are not dominated. However, when d is a constant greater than three, this can be improved to $O(n(\log n)^{d-3} \log \log n)$.[4] For point sets that are generated randomly, it is possible to solve the problem in linear time.[5][6]

4 References

[1] Preparata, Franco P.; Shamos, Michael Ian (1985), "Section 4.1.3: The problem of the maxima of a point set", *Computational Geometry: An Introduction*, Texts and Monographs in Computer Science, Springer-Verlag, pp. 157–166, ISBN 0-387-96131-3.

[2] Kung, H. T.; Luccio, F.; Preparata, F. P. (1975), "On finding the maxima of a set of vectors" (PDF), *Journal of the ACM*, **22** (4): 469–476, doi:10.1145/321906.321910, MR 0381379.

[3] Karlsson, Rolf G.; Overmars, Mark H. (1988), "Scanline algorithms on a grid", *BIT Numerical Mathematics*, **28** (2): 227–241, doi:10.1007/BF01934088, MR 938390.

[4] Gabow, Harold N.; Bentley, Jon Louis; Tarjan, Robert E. (1984), "Scaling and related techniques for geometry problems", *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing (STOC '84)*, New York, NY, USA: ACM, pp. 135–143, doi:10.1145/800057.808675, ISBN 0-89791-133-4.

[5] Bentley, Jon L.; Clarkson, Kenneth L.; Levine, David B. (1993), "Fast linear expected-time algorithms for computing maxima and convex hulls", *Algorithmica*, **9** (2): 168–183, doi:10.1007/BF01188711, MR 1202289.

[6] Dai, H. K.; Zhang, X. W. (2004), "Improved linear expected-time algorithms for computing maxima", in Farach-Colton, Martín, *LATIN 2004: Theoretical informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings*, Lecture Notes in Computer Science, **2976**, Berlin: Springer-Verlag, pp. 181–192, doi:10.1007/978-3-540-24698-5_22, MR 2095193.

5 Text and image sources, contributors, and licenses

5.1 Text

- **Maxima of a point set** *Source:* https://en.wikipedia.org/wiki/Maxima_of_a_point_set?oldid=725414343 *Contributors:* Bearcat, Akbg, Storkk, David Eppstein, Alvin Seville, Hardwigg, Midas02, BG19bot, Adam9007 and Warahul

5.2 Images

- **File:Maxima_of_a_point_set.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/84/Maxima_of_a_point_set.svg *License:* CC0 *Contributors:* Own work *Original artist:* David Eppstein

5.3 Content license

- Creative Commons Attribution-Share Alike 3.0