# Chan's optimal output- sensitive convex hull algorithm

CS60064: Computational Geometry: Spring 2023
Instructor: Sudebkumar Prasant Pal
Teaching Assistants: Prosenjit Kundu and Sandipan Bera

January 14, 2023

Given a set $P$ of $n$ points in the plane, we wish to compute the convex hull of $P$. The convex hull of a set of points in the plane is the smallest convex polygon containing the points.

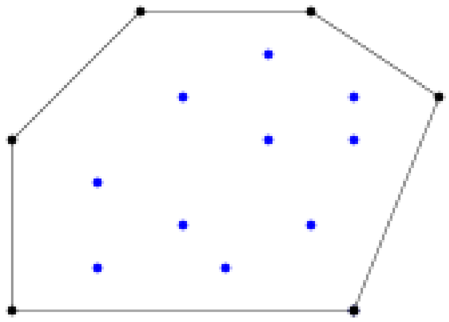

Figure 1: A set of points (blue) and its convex hull (black)

Chan's algorithm finds out the convex hull in $O(n \log h)$ time, where $h$ is the number of vertices on the hull. It uses Graham's Scan and Jarvis's March for finding the convex hull.

We must run Graham's scan on less than $n$ points so that we get an overall complexity of $O(n \log h$. We choose a sets of size $m$ and hope that $h \leq m \leq h^2$, though we do not know $h$ apriori. So, our guesses for $m$ can start from 2 and grow towards the unknown $h$, or evencross $h$ towards say a maximum of $h^2$. Since, there are $m$ points in a group, the number of groups is $\lceil \frac{n}{m} \rceil$. Lets denote it by $r$. Using Graham's Scan on each group takes $O(m \log m)$ time. So, total time for Graham's Scan on $r$ groups is

$O(rm \log m) = O(n \log m)$.

Now, we need to run Jarvis's March for merging the $r$ hulls into a single hull. We know that the time required for computing the tangents between a point and convex $m$-gon is $O(\log m)$. For finding the next hull vertex, we need to find tangents to each of the $r$ hulls. We need to find $h$ hull vertices. Hence, the time complexity for Jarvis's March step becomes $O(hr \log m) = O((hn/m) \log m)$. Combining the two steps, we get a time complexity of $O((n + (hn/m)) \log m)$. If $h \leq m \leq h^2$, the time complexity is $O(n \log h)$. However, we do not know $h$, so we try many values for $m$, increasing $m$ gradually.

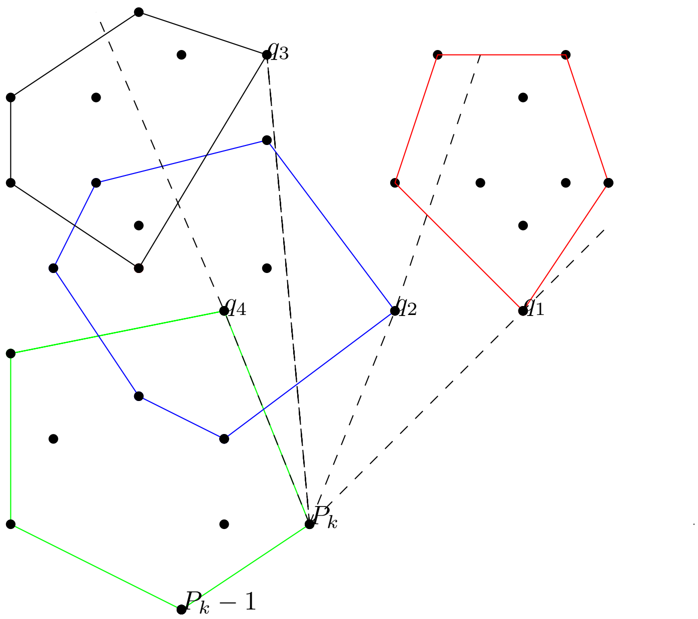


Figure 2: Modified Jarvis' march

**PartialHull(P; m)** :

(1) Let $r = \lceil (n/m) \rceil$. Partition $P$ into disjoint subsets $P(1), P(2), ...P(r)$, each of size at most $m$.

(2) For $i = 1$ to $r$ do: (a) $ComputeHull(P(i))$ using Graham's scan and store the vertices in an ordered array.

(3) Let $p(0) = (-Inf; 0)$ and let $p(1)$ be the bottommost point of $P$.

(4) For $k = 1$ to $m$ do:

(a) For $i = 1$ to $r$ do: Compute point $q$ in $P(i)$ that maximizes the angle $p(k-1)p(k)q$

(b) Let $p(k+1)$ be the point $q$ in $q(1), q(2), ...q(r)$, that maximizes the angle $p(k-1)p(k)q$.

(c) If $p(k+1) = p(1)$ then return $p(1), p(2), ...p(k)$.

(5) Else return '$m$ was too small, try again'.

We do not know the value of $h$. If we try $m = 1, 2, 3, ...,$, then time complexity becomes $O(nh \log h)$ which is too slow. Instead, we can use *doubling search* and try $m = 1, 2, 4, 8, ...2^t$ until it succeeds. This results in a time complexity of $O(n \log^2 h)$ which is again slow. We can try $m = 2, 4, 16, 256, ..., 2^{2^t}$.

In this case, we will find the correct value of $m$ when $2^{2^t} \geq h$. In total, we need to try $t = \lceil \log \log h \rceil$ different values of $m$. So, the running time is at most a multiple of

$\sum_{t=1}^{\log \log h} n2^t = n \sum_{t=1}^{\log \log h} 2^t \leq n2^{1+\log \log h} = 2n2^{\log \log h} =$
$2n \log h = O(n \log h)$