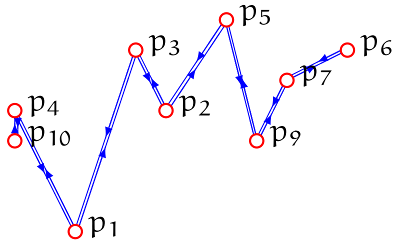
2. Chan's Algorithm

Lecture on Thursday 24th September, 2009 by Michael Hoffmann <hoffmann@inf.ethz.ch>

2.1 Graham Scan (Successive Local Repair)

Sort points lexicographically and remove duplicates: $(p_1, \dots, p_n)$.



$p_{10}\, p_4\, p_1\, p_3\, p_2\, p_5\, p_9\, p_7\, p_6\, p_7\, p_9\, p_5\, p_2\, p_3\, p_1\, p_4\, p_{10}$

As long as there is a (consecutive) triple (p, q, r) s.t. q is left of or on the directed line $\overrightarrow{pr}$, remove q from the sequence.

Theorem 2.1 *The convex hull of a set* $P \subset \mathbb{R}^2$ *of* n *points can be computed using* $O(n \log n)$ *geometric operations.*

Proof.

1. Sorting and removal of duplicate points: $O(n \log n)$.
2. At begin: $2n - 2$ points; at the end: h points. $\Rightarrow$ $2n - h - 2$ shortcuts/positive orientation tests. In addition at most $2n - 2$ negative tests. Altogether at most $4n - h - 4$ orientation tests.

In total $O(n \log n)$ operations. Note that the number of orientation tests is linear only, but $O(n \log n)$ lexicographic comparisons are needed. □

There are many variations of this algorithm, the basic idea is due to Graham [4].

2.2 Lower Bound

Theorem 2.2 $\Omega(n \log n)$ *geometric operations are needed to construct the convex hull of* n *points in* $\mathbb{R}^2$ *(in the algebraic computation tree model).*

Proof. Reduction from sorting (for which it is known that $\Omega(n \log n)$ comparisons are needed in the algebraic computation tree model). Given n real numbers $x_1, \dots, x_n$, construct a set $P = \{p_i \mid 1 \leq i \leq n\}$ of n points in $\mathbb{R}^2$ by setting $p_i = (x_i, x_i^2)$. This construction can be regarded as embedding the numbers into $\mathbb{R}^2$ along the x-axis and then projecting the resulting points vertically onto the unit parabola. The order in which the points appear along the lower convex hull of P corresponds to the sorted order of the x_i. Therefore, if we could construct the convex hull in $o(n \log n)$ time, we could also sort in $o(n \log n)$ time. □

Clearly this simple reduction does not work for the Extremal Points problem. But using a more involved construction one can show that $\Omega(n \log n)$ is also a lower bound for the number of operations needed to compute the set of extremal points only. This was first shown by Avis [1] for linear computation trees, then by Yao [5] for quadratic computation trees, and finally by Ben-Or [2] for general algebraic computation trees.

In fact, the argument is based on a lower bound of $\Omega(n \log n)$ operations for *Element Uniqueness*: Given n real numbers, are any two of them equal? At first glance, this problem appears a lot easier than sorting, but apparently it is not, at least in this model of computation.

2.3 Jarvis' Wrap and Graham Scan in C++

Jarvis' Wrap.

`p[0..N)` contains a sequence of points.
`p_start` point with smallest x-coordinate.
`q_next` some *other* point in `p[0..N)`.

```
int h = 0;
Point_2 q_now = p_start;
do {
  q[h] = q_now;
  h = h + 1;

  for (int i = 0; i < N; i = i + 1)
    if (rightturn_2(q_now, q_next, p[i]))
      q_next = p[i];

  q_now = q_next;
  q_next = p_start;
} while (q_now != p_start);
```

`q[0,h)` describes a convex polygon bounding the convex hull of `p[0..N)`.

Graham Scan.

`p[0..N)` lexicographically sorted sequence of pairwise distinct points, $N \geq 2$.

```
q[0] = p[0];
int h = 0;
```

```
// Lower convex hull (left to right):
for (int i = 1; i < N; i = 1 + 1) {
  while (h>0 && rightturn_2(q[h-1], q[h], p[i]))
    h = h - 1;
  h = h + 1;
  q[h] = p[i];
}

// Upper convex hull (right to left):
for (int i = N-2; i >= 0; i = i - 1) {
  while (rightturn_2(q[h-1], q[h], p[i]))
    h = h - 1;
  h = h + 1;
  q[h] = p[i];
}
```

`q[0,h)` describes a convex polygon bounding the convex hull of `p[0..N)`.

2.4 Chan's Algorithm

Given matching upper and lower bounds we may be tempted to consider the algorithmic complexity of the planar convex hull problem settled. However, this is not really the case: Recall that the lower bound is a worst case bound. For instance, the Jarvis' Wrap runs in $O(nh)$ time an thus beats the $\Omega(n \log n)$ bound in case that $h = o(\log n)$. The question remains whether one can achieve both output dependence and optimal worst case performance at the same time. Indeed, Chan [3] presented an algorithm to achieve this runtime by cleverly combining the "best of" Jarvis' Wrap and Graham Scan. Let us look at this algorithm in detail.

Divide. *Input:* a set $P \subset \mathbb{R}^2$ of n points and a number $H \in \{1, \ldots, n\}$.

1. Divide P into $k = \lceil n/H \rceil$ sets $P_1, \ldots, P_k$ with $|P_i| \leq H$.
2. Construct $\text{conv}(P_i)$ for all i, $1 \leq i \leq k$.
3. Construct H vertices of $\text{conv}(P)$. (*conquer*)

Analysis. Step 1 takes $O(n)$ time. Step 2 can be handled using Graham Scan in $O(H \log H)$ time for any single P_i, that is, $O(n \log H)$ time in total.

Conquer.

1. Find the lexicographically smallest point in $\text{conv}(P_i)$ for all i, $1 \leq i \leq k$.

2. Starting from the lexicographically smallest point of P find the first H points of $\text{conv}(P)$ oriented counterclockwise (simultaneous Jarvis' Wrap on the sequences $\text{conv}(P_i)$).

Determine in every step the points of tangency from the current point of $\text{conv}(P)$ to $\text{conv}(P_i)$, $1 \leq i \leq k$, using binary search.

Analysis. Step 1 takes $O(n)$ time. Step 2 consists of at most H wrap steps. Each wrap needs to find the minimum among k candidates where each candidate is computed by a binary searches on at most H elements. This amounts to $O(Hk \log H) = O(n \log H)$ time for Step 2.

Remark. Using a more clever search strategy instead of many binary searches one can handle the conquer phase in $O(n)$ time. However, this is irrelevant as far as the asymptotic runtime is concerned, given that already the divide step takes $O(n \log H)$ time.

Searching for h. While the runtime bound for $H = h$ is exactly what we were heading for, it looks like in order to actually run the algorithm we would have to know h, which—in general—we do not. Fortunately we can circumvent this problem rather easily, by applying what is called a *doubly exponential search.* It works as follows.

Call the algorithm from above iteratively with parameter $H = \min\{2^{2^t}, n\}$, for $t = 0, \ldots$, until the conquer step finds all extremal points of P (i.e., the wrap returns to its starting point).

Analysis: Let 2^{2^s} be the last parameter for which the algorithm is called. Since the previous call with $H = 2^{2^{s-1}}$ did not find all extremal points, we know that $2^{2^{s-1}} < h$, that is, $2^{s-1} < \log h$, where h is the number of extremal points of P. The total runtime is therefore at most

$$\sum_{i=0}^{s} cn \log 2^{2^i} = \sum_{i=0}^{s} cn2^i = cn(2^{s+1} - 1) < 4cn \log h = O(n \log h).$$

In summary, we obtain the following theorem.

Theorem 2.3 *The convex hull of a set* $P \subset \mathbb{R}^2$ *of* n *points can be computed using* $O(n \log h)$ *geometric operations, where* h *is the number of convex hull vertices.*

Questions

1. *How is convexity defined? What is the convex hull of a set in* $\mathbb{R}^d$*?* Give at least three possible definitions.

2. *What does it mean to compute the convex hull of a set of points in* $\mathbb{R}^2$*?* Discuss input and expected output and possible degeneracies.

3. *How can the convex hull of a set of* n *points in* $\mathbb{R}^2$ *be computed efficiently?* Describe and analyze (incl. proofs) Jarvis' Wrap, Successive Local Repair, and Chan's Algorithm.

4. *Is there a linear time algorithm to compute the convex hull of* n *points in* $\mathbb{R}^2$*?* Prove the lower bound and define/explain the model in which it holds.

5. *Which geometric primitive operations are needed to compute the convex hull of* n *points in* $\mathbb{R}^2$*?* Explain the two predicates and how to compute them.

References

[1] D. Avis, Comments on a lower bound for convex hull determination, *Inform. Process. Lett.* **11** (1980), 126.

[2] M. Ben-Or, Lower bounds for algebraic computation trees, in: *Proc. 15th Annu. ACM Sympos. Theory Comput.*, 1983, 80–86.

[3] Timothy M. Chan, Optimal Output-Sensitive Convex Hull Algorithms in Two and Three Dimensions, *Discrete Comput. Geom.* **16** (1996), 361–368.

[4] R. L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, *Inform. Process. Lett.* **1** (1972), 132–133.

[5] A. C. Yao, A lower bound to finding convex hulls, *J. ACM* **28** (1981), 780–787.