ALGORITHMS FOR 2D CONVEX HULLS

Given a finite point set $S \subseteq \mathbb{R}^d$, how could we find $\mathrm{conv}(S)$ efficiently? Here are some easy proposals:

needed
t if
s to leave
we could
t.

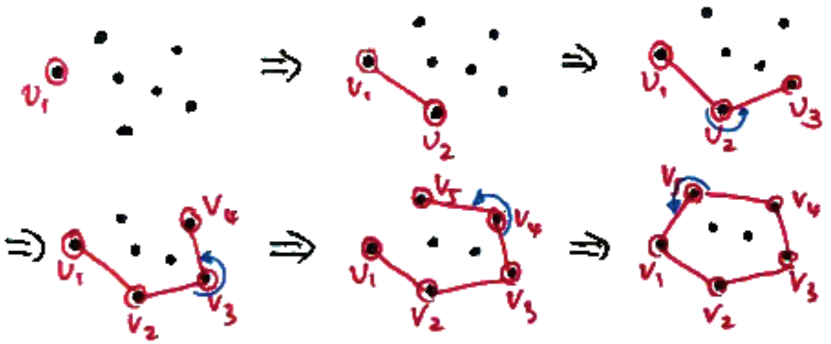
- For each $x \in S$ we test if it is a vertex of $\mathrm{conv}(x)$ by asking whether $x \overset{?}{\in} \mathrm{conv}(S \setminus \{x\})$. The test could be carried out by enquiry all subsets $R \in \binom{S\setminus\{x\}}{d+1}$ if $x \overset{?}{\in} \mathrm{conv}(R)$. The correctness is guaranteed by Carathéodory's theorem, and the running time is clearly polynomial. For the 2D case in particular, the running time is $O(n \cdot n^3) = O(n^4)$.
- Or we could make use of Lemma 11. In the 2D case, we test for each

pair $\{x,y\} \in \binom{S}{2}$ if the segment $\overline{xy}$ bounds $\mathrm{conv}(S)$. The test is carried out by asking if all other points lie on the same side of xy. This yields an $O(n^2 \cdot n) = O(n^3)$ algorithm.

But there exist much better algorithms for the 2D case, which we will introduce next.

Jarvis' Wrap

Intuition:



This should explain the name of the algorithm.

More formally, in the initial step we choose the leftmost point in S (which is guaranteed to bound $\mathrm{conv}(S)$). Then, in the wrapping step i, we choose a v_i such that all other vertices lie to the left of $\overrightarrow{v_{i-1}v_i}$.

In implementation, we could always choose the next v_i in $O(n)$ time as follows:

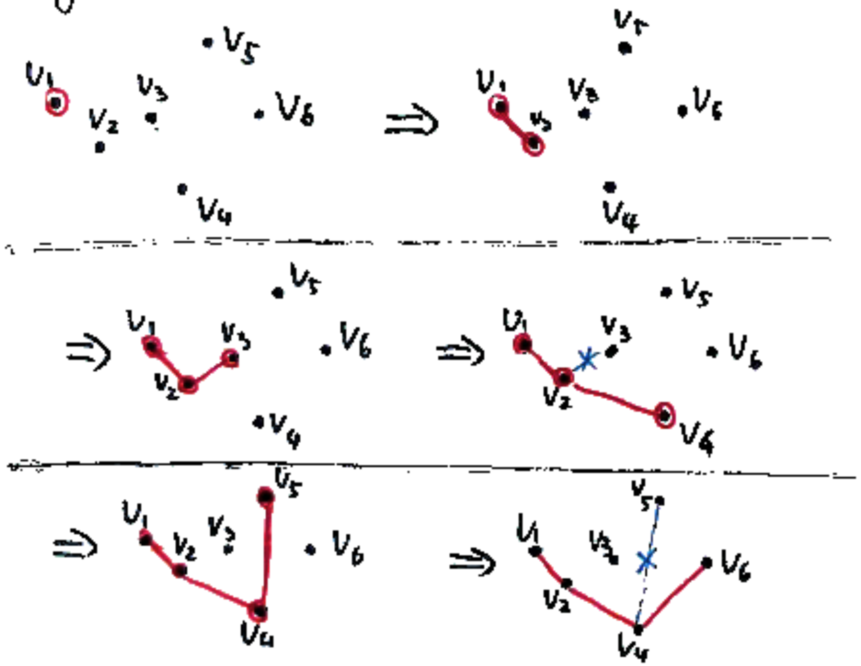
```
v_i := an arbitrary vertex except v_1, ..., v_{i-1}
for v ∈ S \ {v_1, ..., v_{i-1}} do
|   if v_{i-1} v_i v form a right turn
|   L   v_i := v
```

Since the ~~angle~~ segment $\overline{v_{i-1}v_i}$ always rotates clockwise ~~as the procedure~~ ~~chose~~ whenever there's something to the right, we would eventually locate the real v_i when the procedure finishes.

Clearly, Jarvis' Wrap takes time $O(n \cdot h)$ where $h :=$ length of convex hull boundary. In the worst case it's an $O(n^2)$ algorithm.

Graham's Scan

Idea: Sort the points as $v_1, \dots, v_n$ from left to right. In step i we maintain the lower half of $\mathrm{conv}(v_1, \dots, v_i)$, so after n steps we would have computed the lower half of $\mathrm{conv}(S)$. Then repeat the same procedure, this time maintaining the upper half. Glueing the two halves give the $\mathrm{conv}(S)$.



[Illustration for the lower half]

Formal description:

Let $v_1, \dots, v_n$ be the sorted points from left to right.

```
Stack.push(v1)
Stack.push(v2)
for i = 3 ... n do
    while stack[-2], stack[-1], vi
          form a right turn do
        Stack.pop()
    Stack.push(vi)
Output the vertices in stack
  in order.
```

Exercise. Prove the correctness of Graham's scan.

Note that each vertex could be pushed into/popped from the stack at most once, the scanning process takes $O(n)$ time. Combining with the $O(n \log n)$ time of sorting, the overall running time is $O(n \log n)$.

Chan's Algorithm

It can be shown that $\Theta(n\log n)$ is the lower bound for 2D convex hull algorithms, so Graham's scan is optimal in the worst case. Still, in some cases the ~~number~~ length h of convex hull boundary is significantly smaller than $\log n$, then Jarvis' wrap outperforms. Chan's algorithm cleverly achieves the best of both worlds.

On a high level, Chan's algorithm could be briefly described as "divide by Graham, wrap by Jarvis."

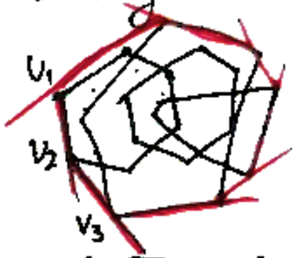
(1) "Divide by Graham".

In the following we assume an oracle tells us the true value of h is $\hat{h}$. We divide the point set S into $\frac{n}{\hat{h}}$ equally-sized sets, each having $\hat{h}$ points. We run Graham's algorithm on each of these sets, using total time $\frac{n}{\hat{h}} \cdot O(\hat{h}\log\hat{h}) = O(n\log\hat{h})$.

When we finish, we ~~has~~ obtain $n/\hat{h}$ many convex hulls, say $C_1, \dots, C_{n/\hat{h}}$.

(2) "Conquer by Jarvis"

Now we combine these hulls into a big hull, using a variant of Jarvis' wrap.



Still, we start from the leftmost vertex and search for the next vertex v_i: all points are to the left of $v_{i-1}v_i$. ~~Note~~ Observe that such v_i is exactly a right-tangent for some convex hull C_j, w.r.t. v_{i-1}. By a previous exercise, we could search for a right-tangent for a given convex hull in logarithm time, so in each wrapping step we only need to spend

$$\underbrace{\frac{n}{\hat{h}}}_{\#\text{convex hulls}} \cdot O(\underbrace{\log\hat{h}}_{\text{size of each hull}}) = O\left(\frac{n}{\hat{h}}\log\hat{h}\right)$$

time. There are $\hat{h}$ steps in total (because we magically predicted that the length of convex hull boundary is exactly $\hat{h}$), so we spend $O(n\log\hat{h})$ time in total.

(3) Guess h by doubly exponential search.

Now we remove the unrealistic assumption that h is known beforehand. Well, since we don't own supernatural power, let's do some moderate guesses. Our strategy is the so-called "doubly exponential search". We try the algorithm on

guess#	0	1	2	3	4	...
$\hat{h}$	2^1	2^2	2^4	2^8	2^{16}	...

and abort a guess if we found out that our guess is too small, i.e. not large enough to cover the length of convex hull boundary in (2).

Clearly, in guess #i we spend time

$$O(n\log\hat{h}_i) = O(n\log 2^{2^i}) = O(n2^i)$$

and we will reach the truth in $\lceil\log\log h\rceil \le \log\log h + 1$ guesses. So the overall running time writes

$$\sum_{i=0}^{\log\log h+1} O(n2^i) = O(n)\cdot\sum_{i=0}^{\log\log h+1} 2^i \le O(n)\cdot 2^{\log\log h+2} = O(n\log h).$$

Obviously, this takes the advantages of both Jarvis' wrap and Graham's scan.