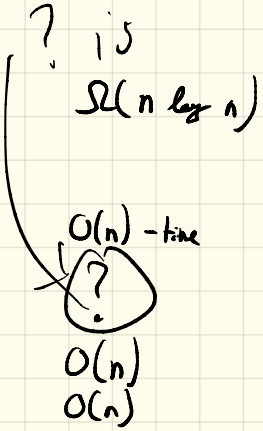
Output Sensitive Convex Hull Algo

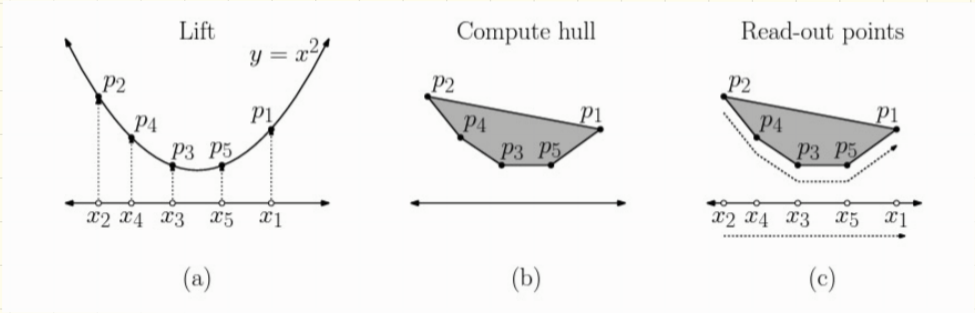
1/21

Can we do better than $O(n \log n)$ for convex hull
... in binary comparison (Real-RAM model) no!

Lower bound:
- given n numbers
  describe $O(n)$ time algo
  that sorts the nubers by using convex hull
  it's well known that sorting $\Omega(n \log n)$

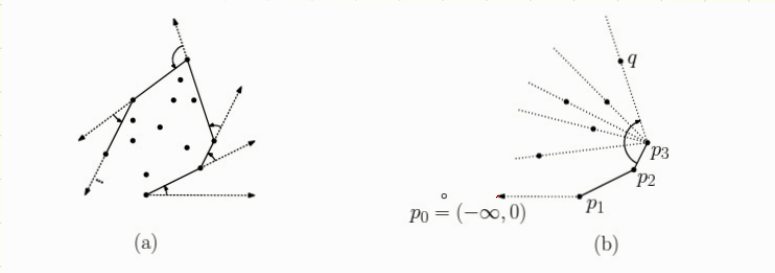


reduction
Given n numbers
take numbers → to points in 2D $x \mapsto (x, x^2)$
compute the convex hull of lifted points
  produce CCW ordered polygon
- find leftmost point in cyclic order
- read off the x-cord of points



Jarvis March (gift wrapping)

- assuming h verts on the hull $\rightarrow O(nh)$ time alg



1. find one point on the hull — smallest y-coord — $O(n)$
2. let p_k and p_{k-1} be last 2 verts of hull w.r.t. a q s.t. $\angle p_{k-1} p_k q$ is maximized — $O(n)$

run (2) h times $\Rightarrow O(hn)$ time to compute hull

Timothy Chan: $O(n \log h)$

Things to remember!

- $O(n \log h)$ we can only run graham scan on sets of size poly in h

$\log h^c = c \log h = O(\log h)$

Idea:

- make n/h sets all of size h
- run Graham's scan on each subset $O(h \log h)$ → give us "mini-hulls"

total time for the mini-hulls $O(\frac{n}{h} h \log h) = O(n \log h)$

Q: (a) how do we merge mini hulls

(b) how do we find h

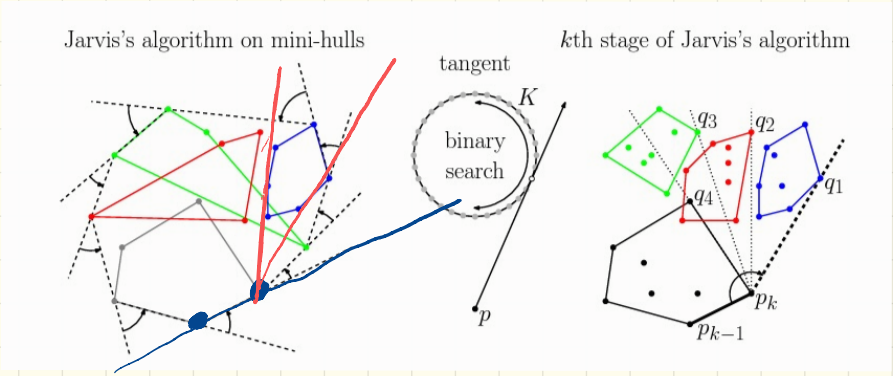
assumptions on h

let h^* be an estimate of h s.t. $h \le h^* \le h^2$

$\Rightarrow O(n \log h^*) = O(n \log h^2) = O(n \log h)$

$\Rightarrow$ break up our points into sets of size about h^*

# of points on mini-hull is $3, \ldots, h^*$



Lemma: Consider convex polygon K in the plane w/ m verts

K is ordered cyclically in an array

a point p external to K

we can find 2 supporting lines for K through p in $O(\log m)$ time

RestrictedHull(P, h^*) :

(1) Let $r \leftarrow \lceil n/h^* \rceil$.
(2) Partition P into disjoint subsets $P_1, P_2, \ldots, P_r$, each of size at most h^*.
(3) For $(i \leftarrow 1$ to $r)$
    compute Hull(P_i) using Graham's scan and store the vertices in an ordered array.
(4) Let $p_0 \leftarrow (-\infty, 0)$ and let p_1 be the bottommost point of P.
(5) For $(k \leftarrow 1$ to $h^*)$
    (a) For $(i \leftarrow 1$ to $r)$
        compute point tangent $q_i \in$ Hull(P_i), that is, the vertex of Hull(P_i) that maximizes the angle $\angle p_{k-1} p_k q_i$.
    (b) Let p_{k+1} be the point $q \in \{q_1, \ldots, q_r\}$ that maximizes the angle $\angle p_{k-1} p_k q$.
    (c) If $p_{k+1} \leftarrow p_1$ then return $\langle p_1, \ldots, p_k \rangle$ (success).
(6) (Unable to complete the hull after h^* iterations.) Return "Failure: h^* is too small."

Takeaway is that provide $h \le h^* \le h^2 \Rightarrow$ algo takes $O(n \log h^*)$ or it fails

How do we find h^*

try 1: $h^* = \{1, 2, 3, 4, \ldots, i\} \Rightarrow O(hn \log h)$ ✗

try 2: doubling $h^* = \{2, 4, 8, 16, \ldots, 2^i\} \Rightarrow O(n \log^2 h)$

try 3: repeatidly square $h^* = 2, 4, 16, \ldots, 2^{2^i}$

Consider our i^{th} guess

$h_i^* = 2^{2^i}$

$O(n \log h_i^*) = O(n \log 2^{2^i}) = O(n 2^i)$

$h_i^* > h$ if $i = \lceil \lg \lg h \rceil$ ← $\log_2$

$\Rightarrow$ running time is proportion to

$$T(n,h) = \sum_{i=1}^{\lg \lg h} n 2^i = n \sum_{i=1}^{\lg \lg h} 2^i \le n 2^{1+\lg \lg h} = n 2 2^{\lg \lg h} = 2n \lg h = O(n \lg h)$$

$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$

Putting it together

Hull(P) :

(1) $h^* \leftarrow 2$. $L \leftarrow$ fail.

(2) while ($L =$ fail)

(a) Let $h^* \leftarrow \min((h^*)^2, n)$.

(b) $L \leftarrow$ RestrictedHull(P, h^*).

(3) Return L.
