CS 516 -3
Computational Geometry & Graph Drawing
(Spring 2013)

Last Class…

- Continued with (motivating) example: finding near neighbours (within fixed distance)
- Another (motivating) example from graph drawing: bar-visibility graphs

Reading

MountNotes Chapters 7,8
Chapters 9,10, 6

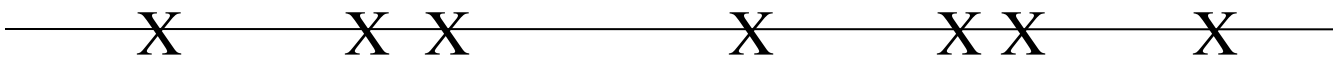
Today…

- Brief review of Asst0
- Convex hulls, halfspace intersections and duality
- Two equivalences concerning convex hulls:
    - Structural equivalence (via duality) with (origin-containing) half-space intersection
    - Algorithmic equivalence (via reducibility argument) with sorting

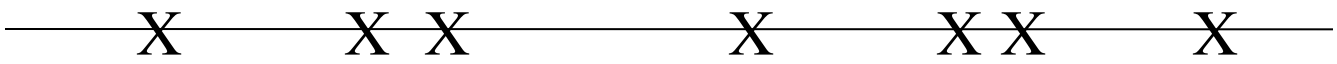
Today…

- Brief review of Asst0
- Convex hulls, halfspace intersections and duality
- Two equivalences concerning convex hulls:
  - Structural equivalence (via duality) with (origin-containing) half-space intersection
  - Algorithmic equivalence (via reducibility argument) with sorting

Assignment 0

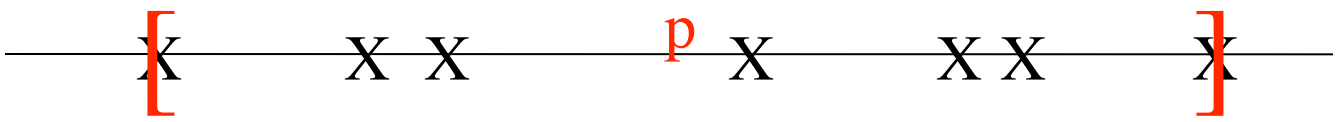


Assignment 0



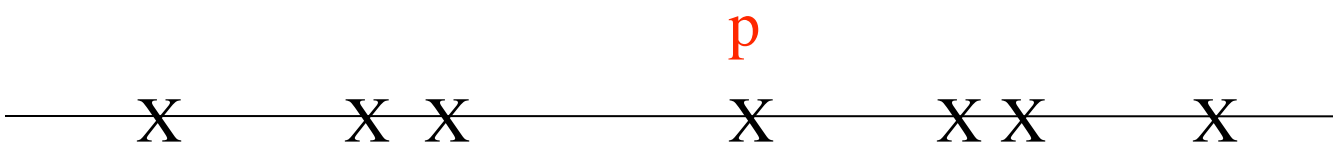
1(a) $\min_p \max_{q \text{ in } S} |q-p|$

Assignment 0



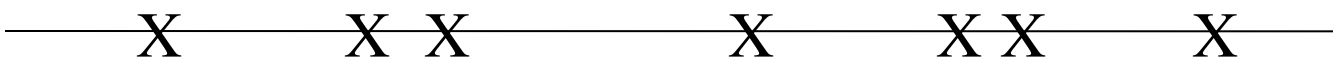
1(a) $\min_p \max_{q \text{ in } S} |q-p|$ (1-center)

Assignment 0



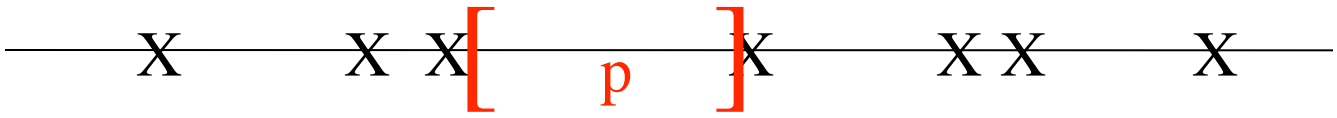
1(b) $\min_p \Sigma_{q \text{ in } S} |q-p|$ (1-median)

Assignment 0



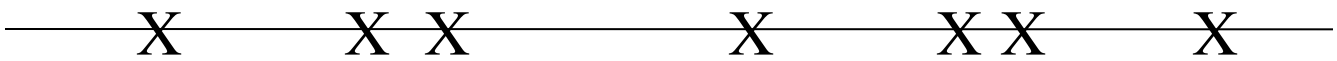
1(c) $\max_p \min_{q \text{ in } S} |q-p|$

Assignment 0



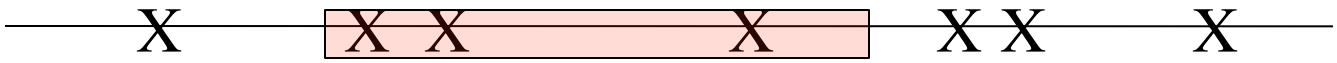
1(c) $\max_p \min_{q \text{ in } S} |q-p|$

Assignment 0



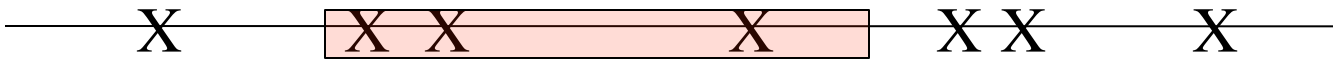
1(d)

Assignment 0



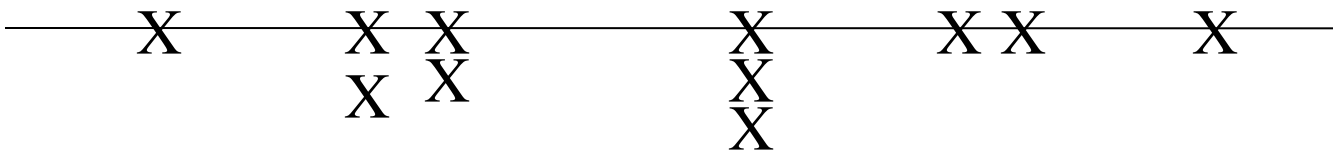
1(d)

Assignment 0



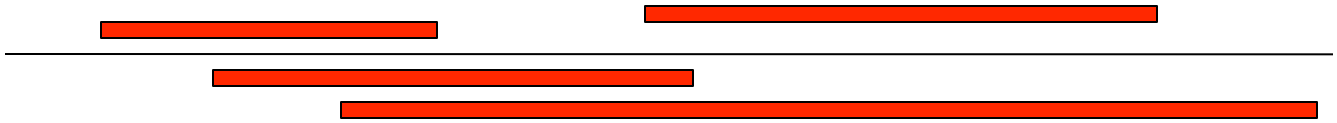
1(d) dynamic dictionary

Assignment 0



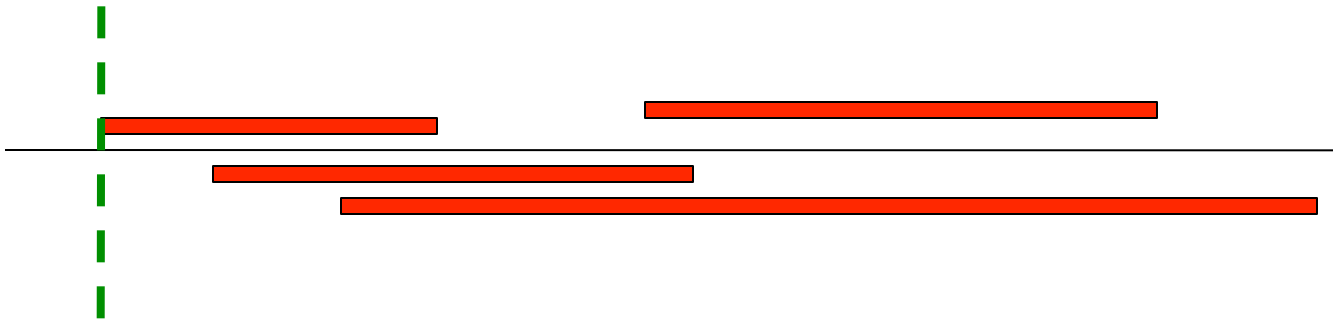
1(e) count number of *distinct* elements

Assignment 0



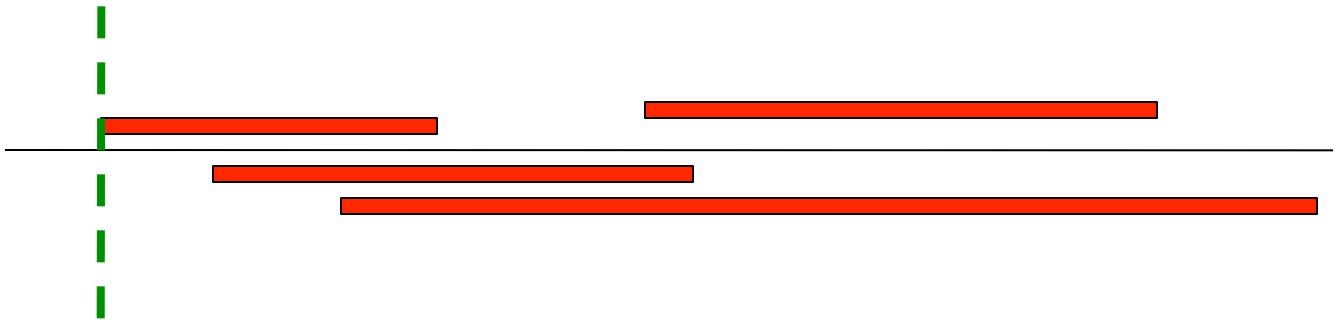
1(f) report intersecting intervals

Assignment 0



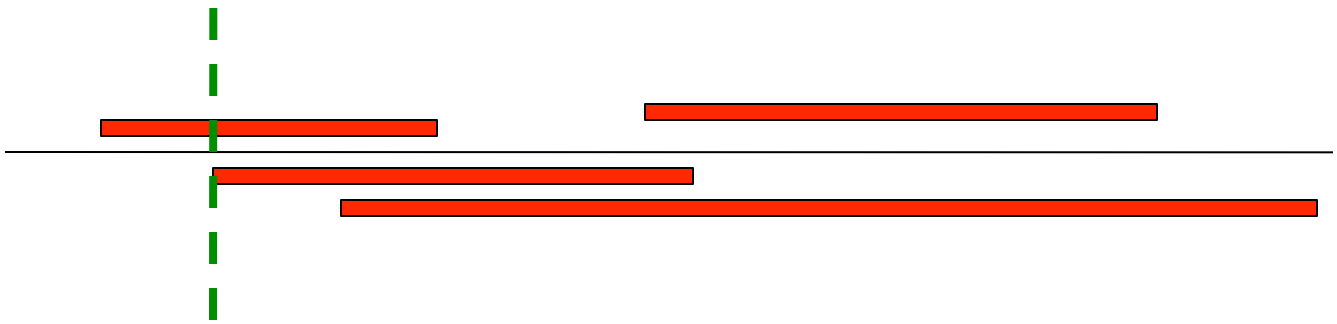
1(f) report intersecting intervals

Assignment 0



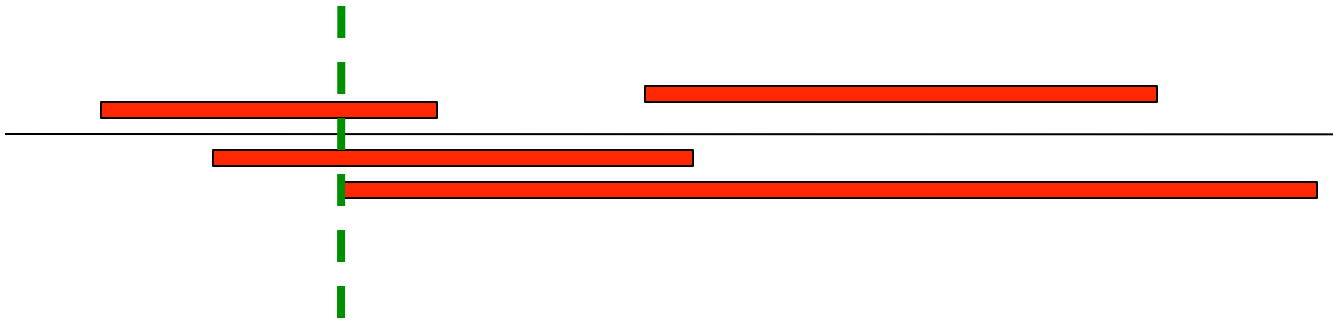
1(f) report intersecting intervals

Assignment 0



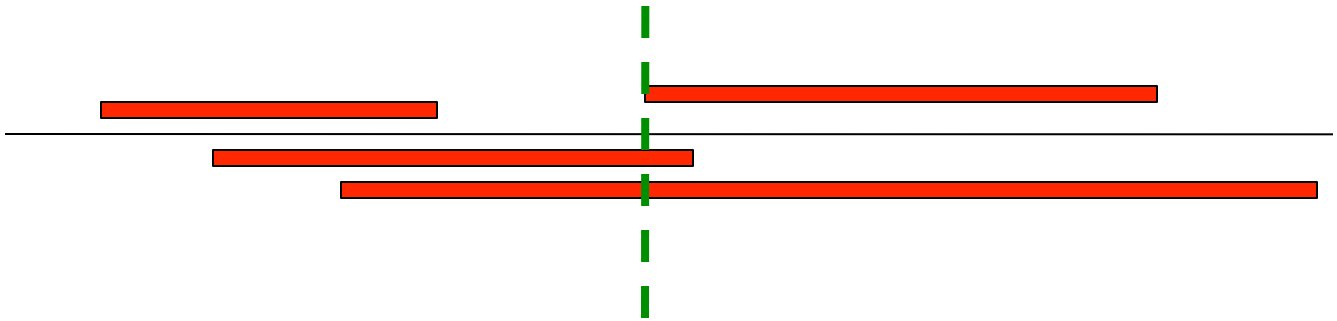
1(f) report intersecting intervals

Assignment 0



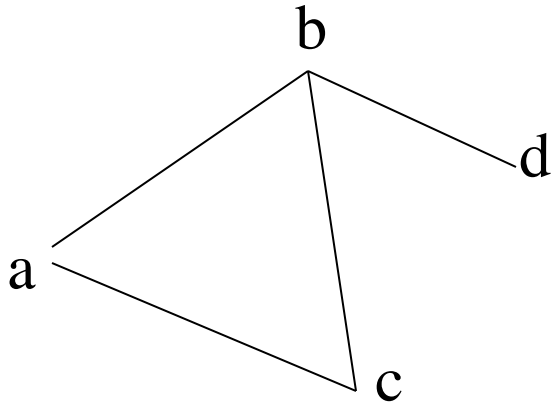
1(f) report intersecting intervals

Assignment 0



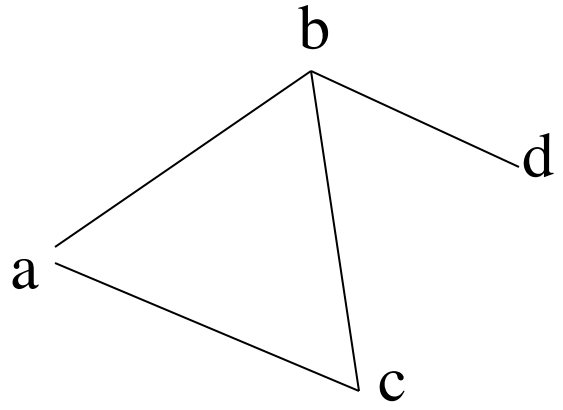
1(f) report intersecting intervals

Assignment 0



1(g)

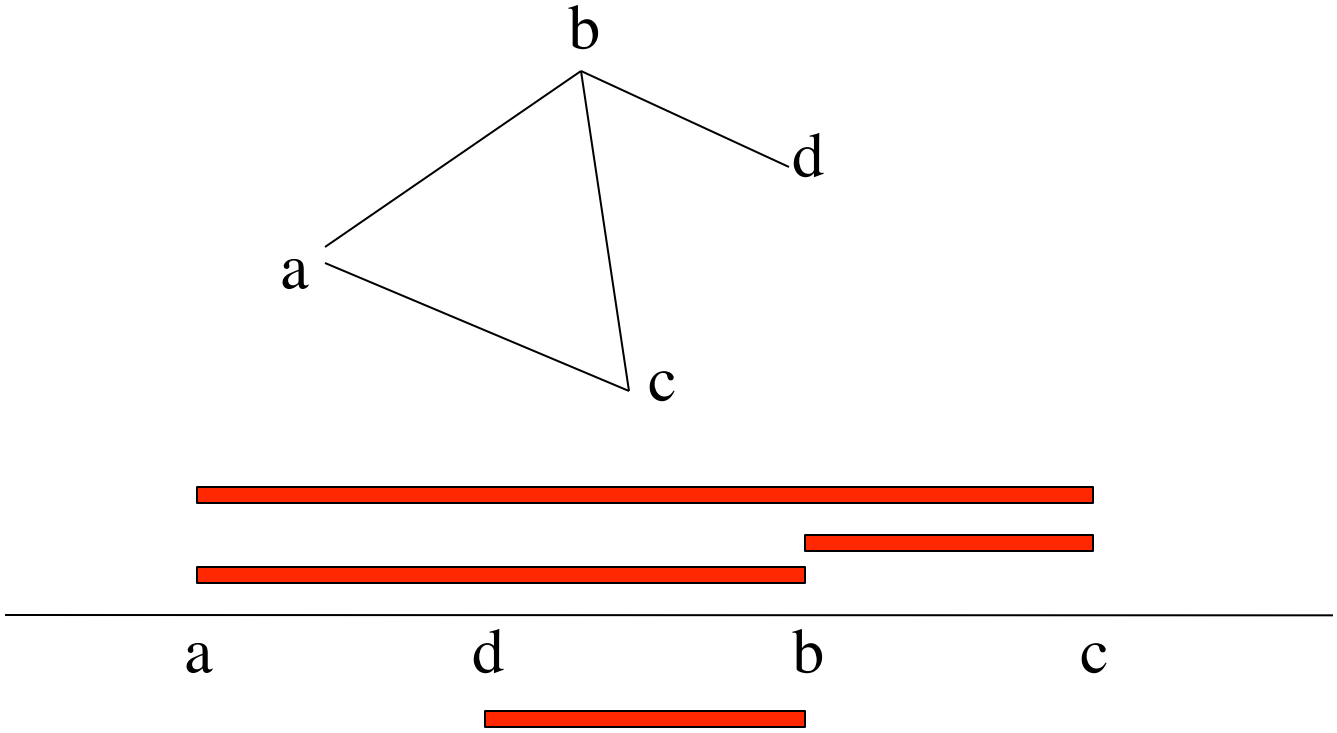
Assignment 0



a d b c

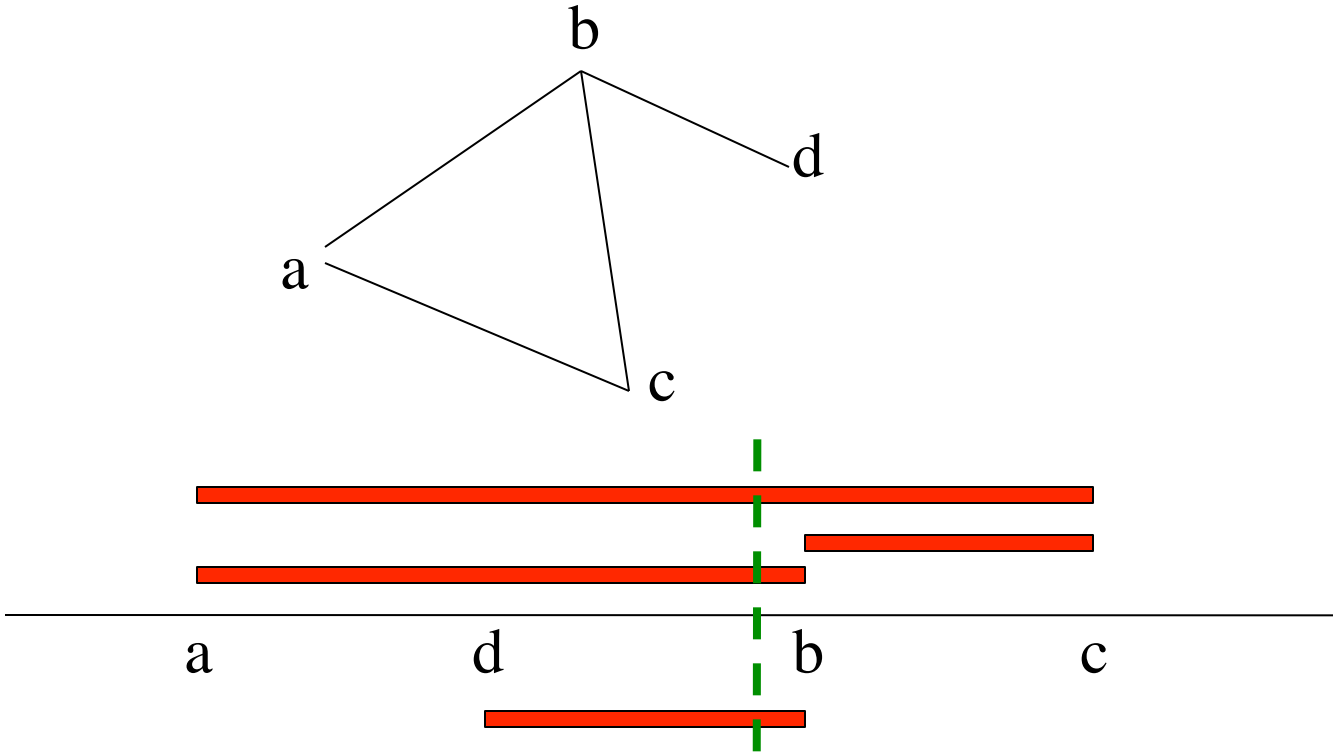
1(g)

Assignment 0



1(g)

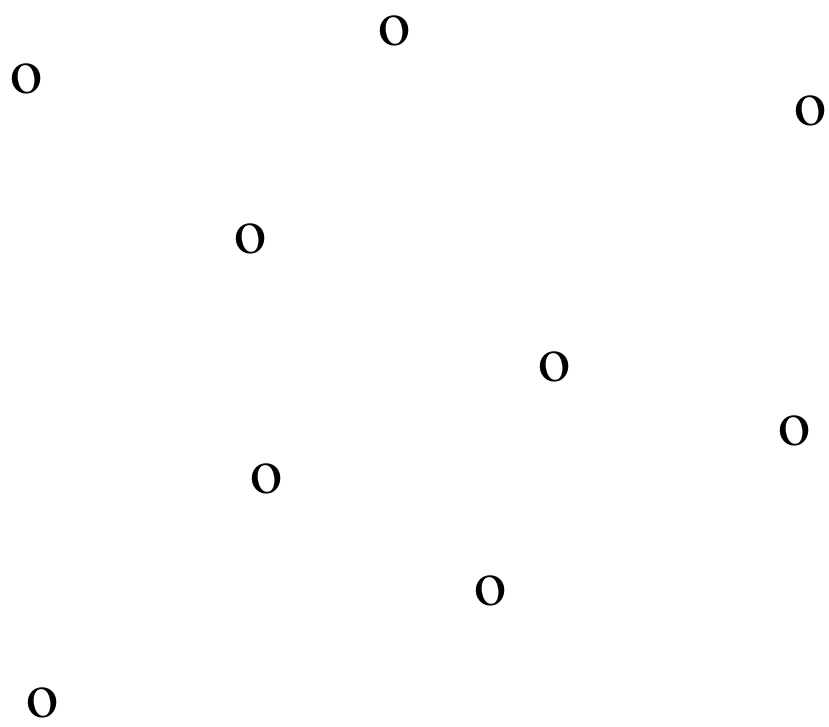
Assignment 0



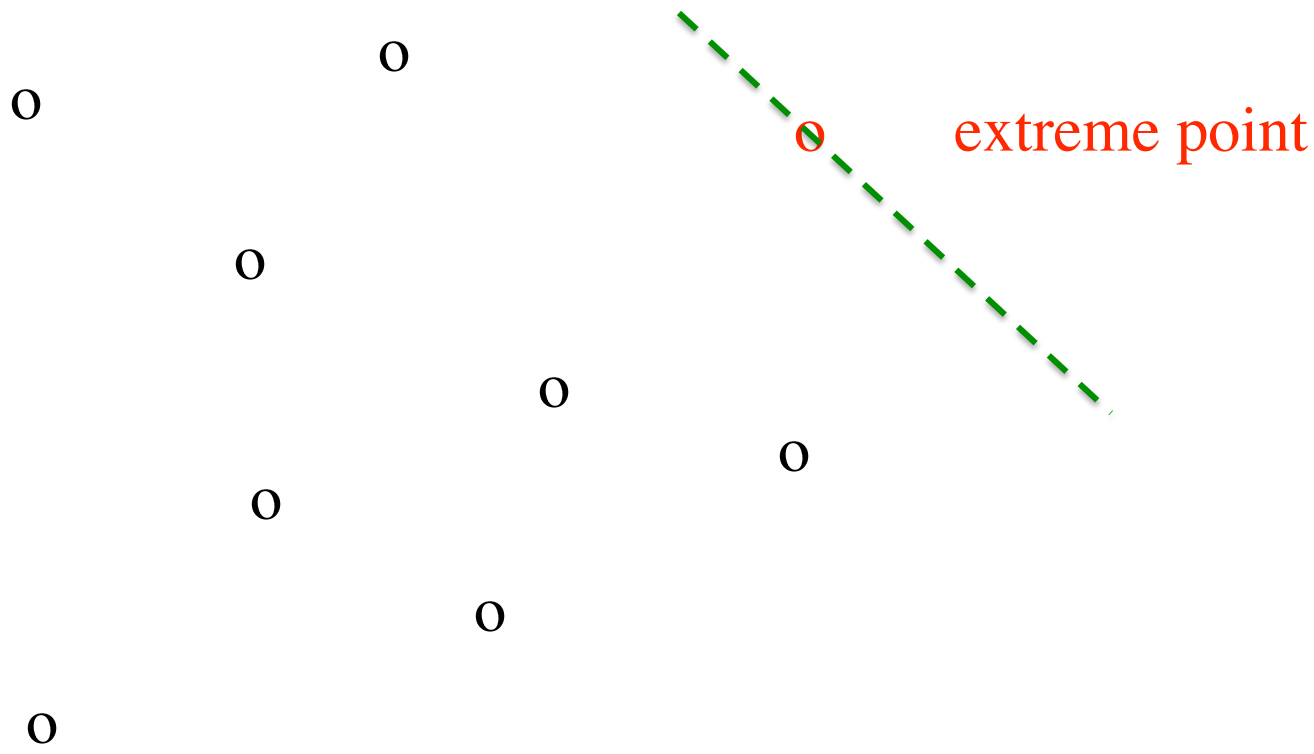
1(g) min cut linear arrangement

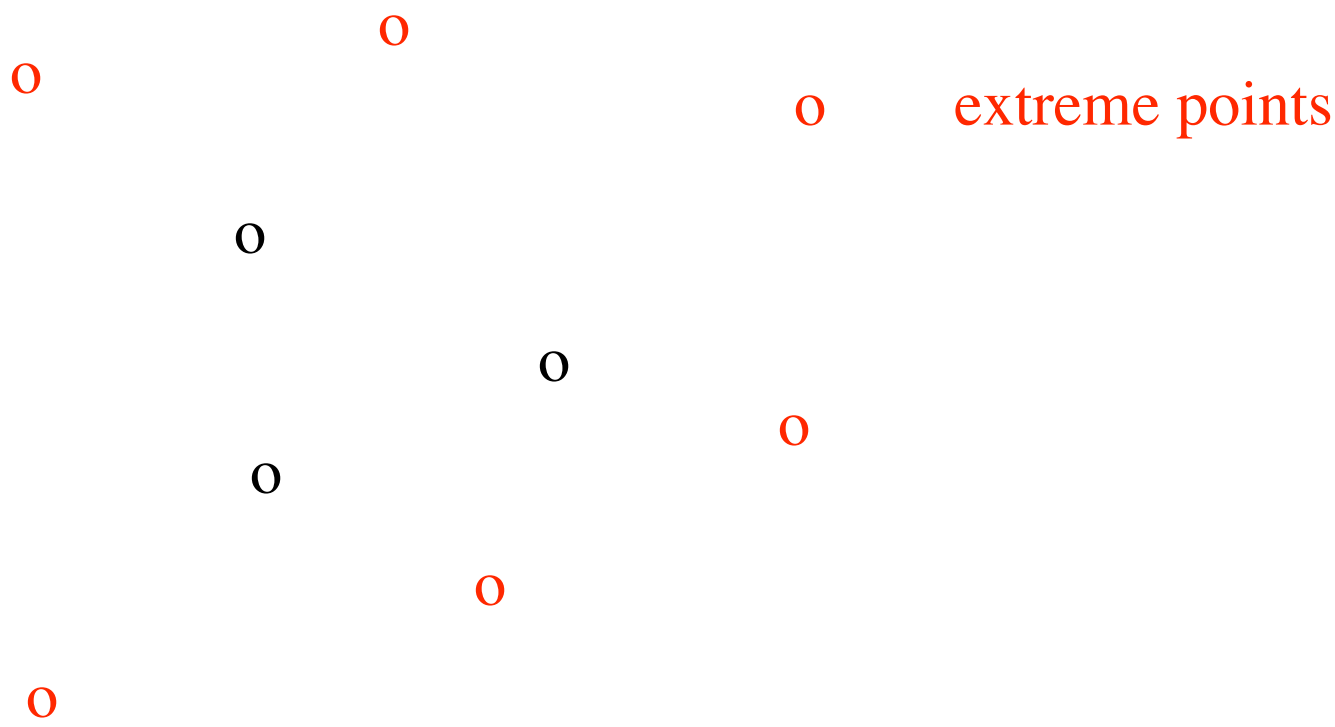
Today…

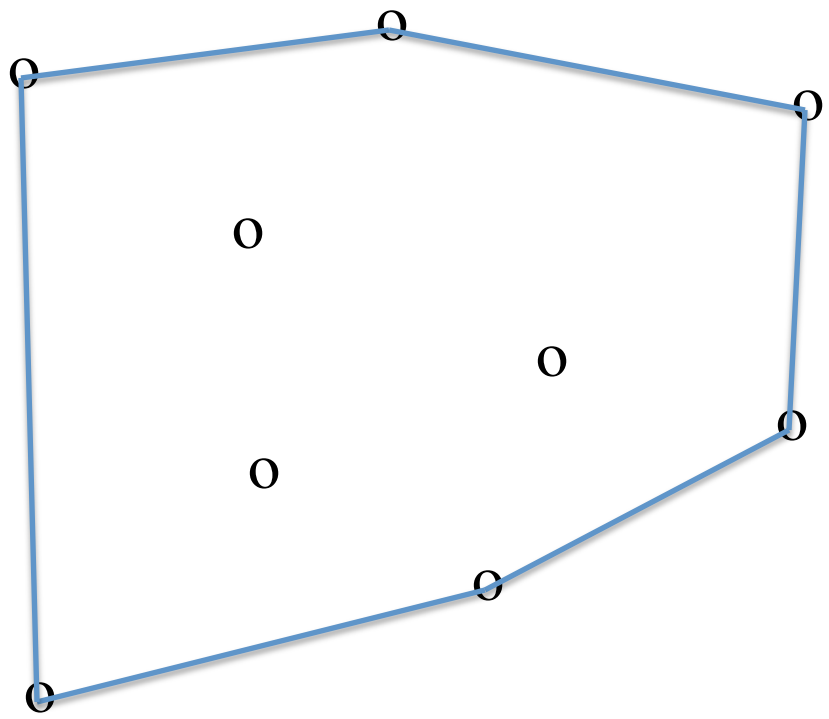
- Brief review of Asst0
- Convex hulls, halfspace intersections and duality
- Two equivalences concerning convex hulls:
  - Structural equivalence (via duality) with (origin-containing) half-space intersection
  - Algorithmic equivalence (via reducibility argument) with sorting



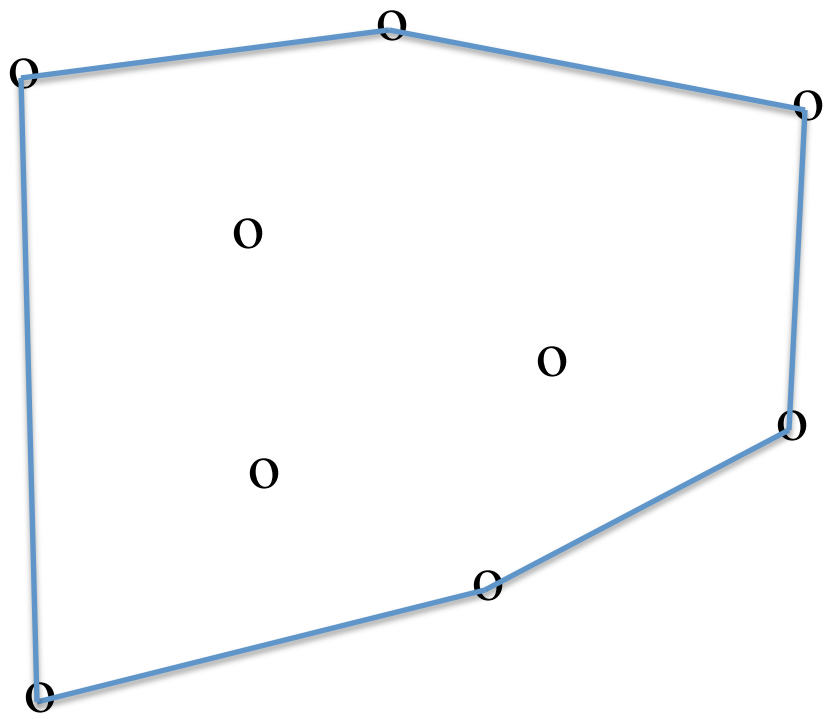
point set S


extreme point

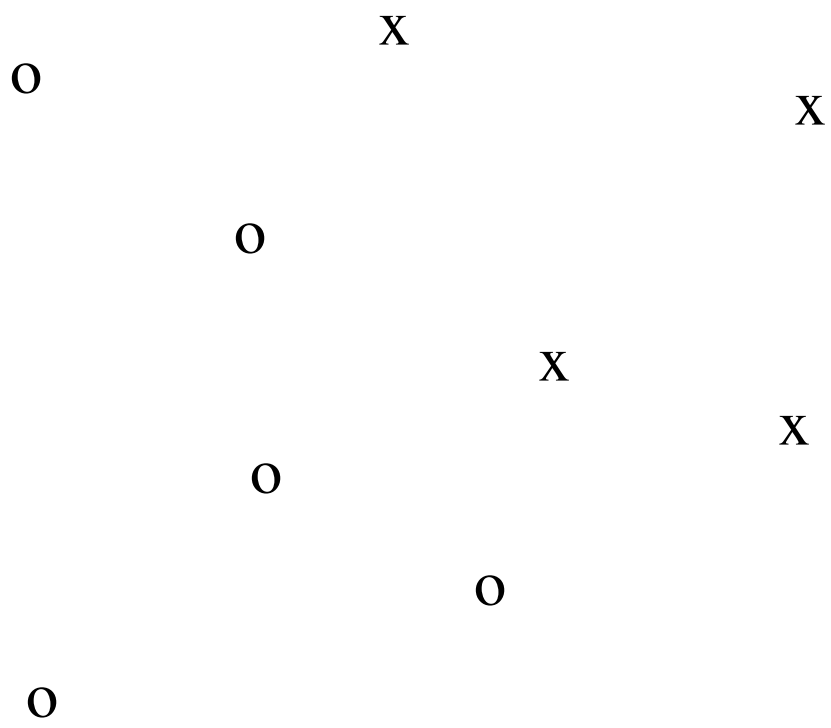

extreme points



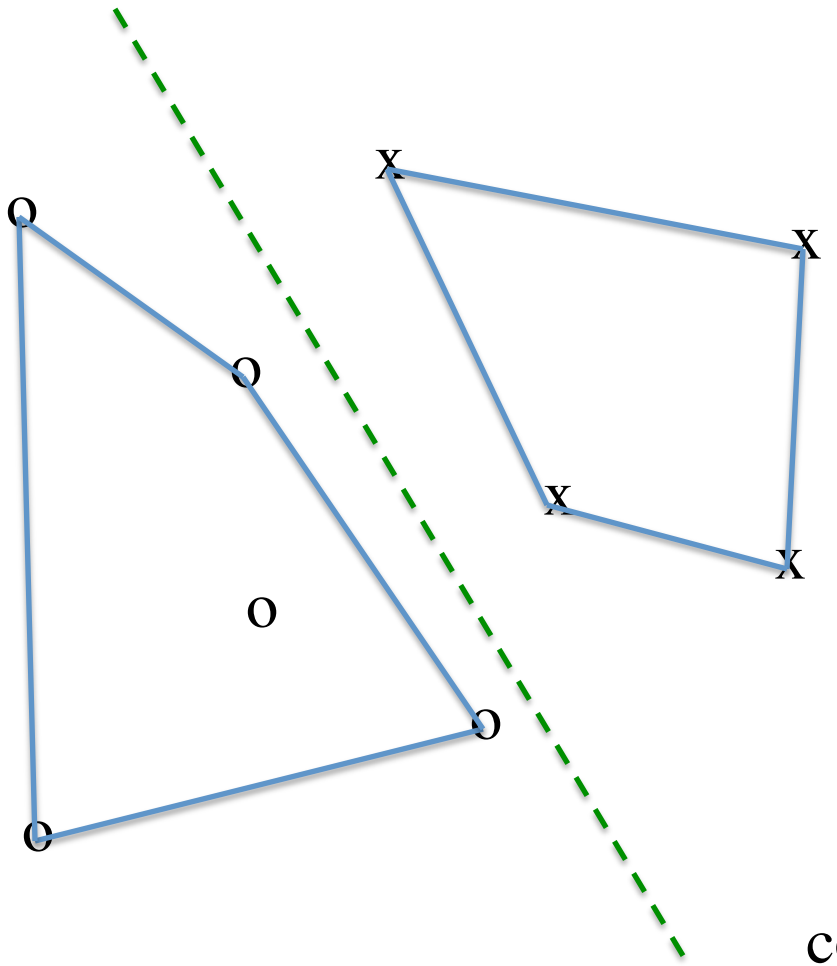
convex-hull =
polygon whose vertices
are extreme points



convex-hull :
shape approx



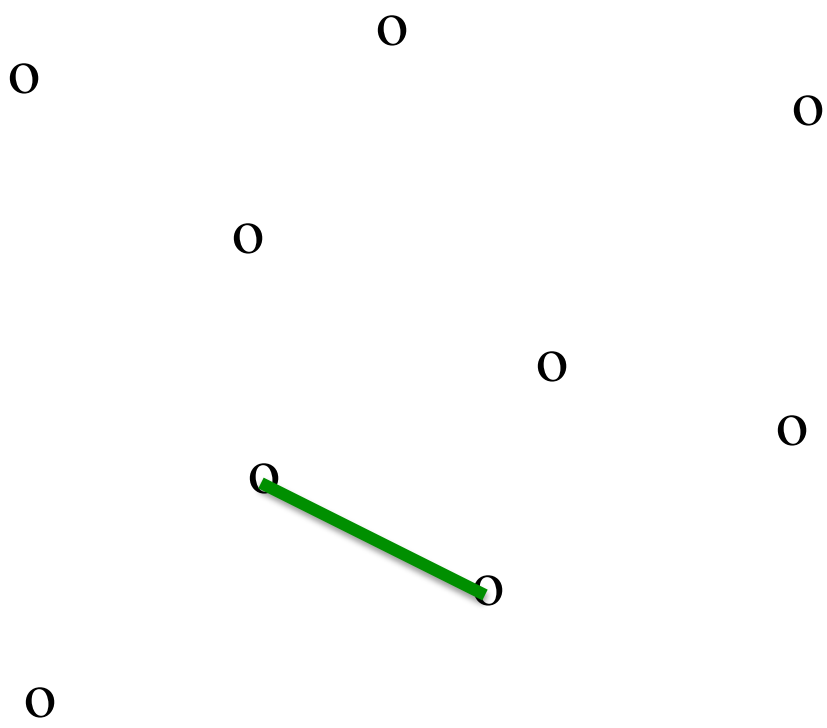
convex-hull :
shape approx
linear separability



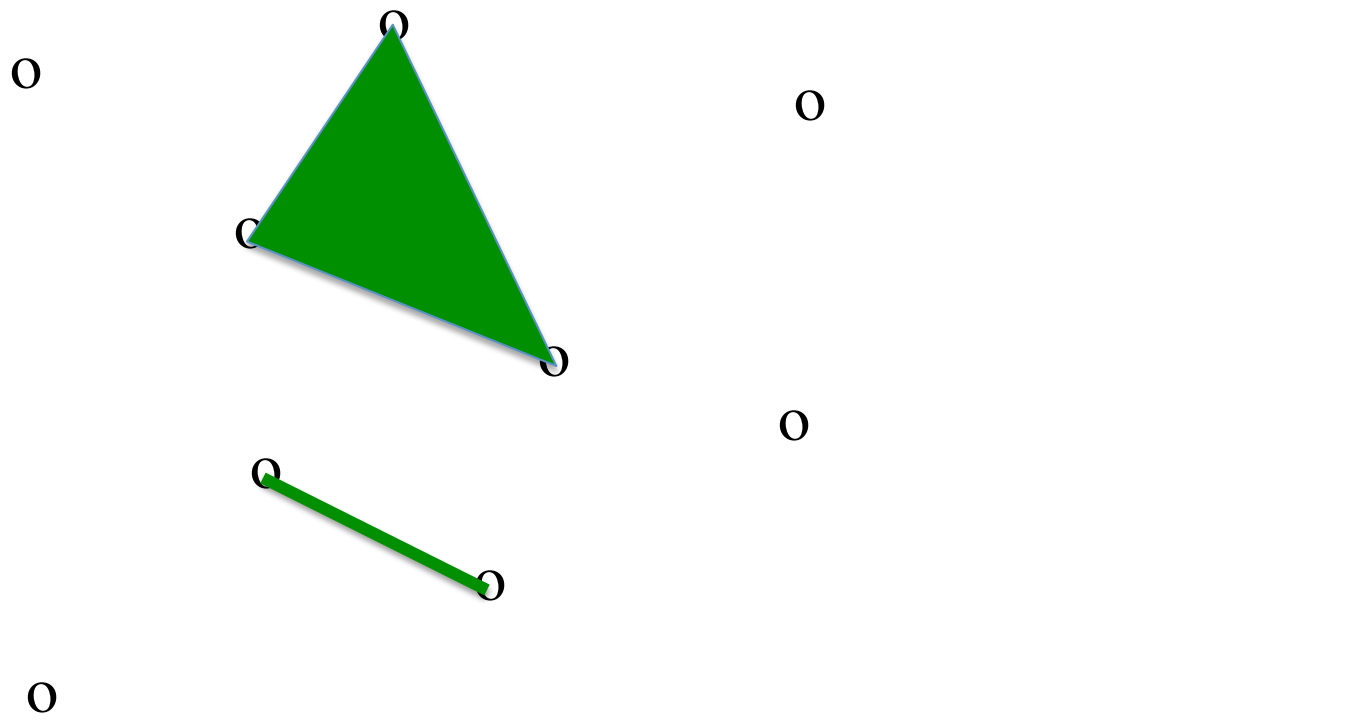
convex-hull :

shape approx

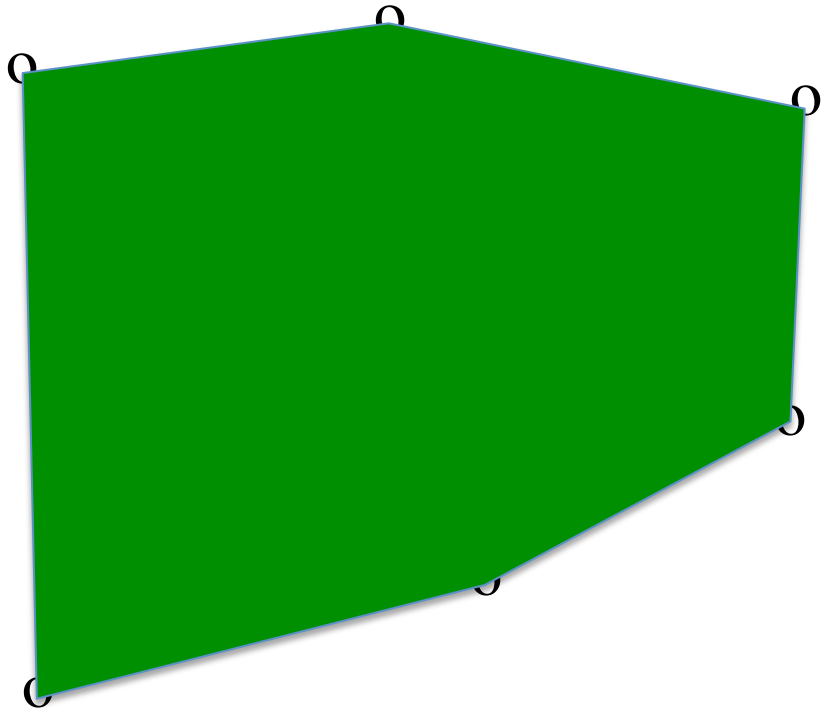
linear separability



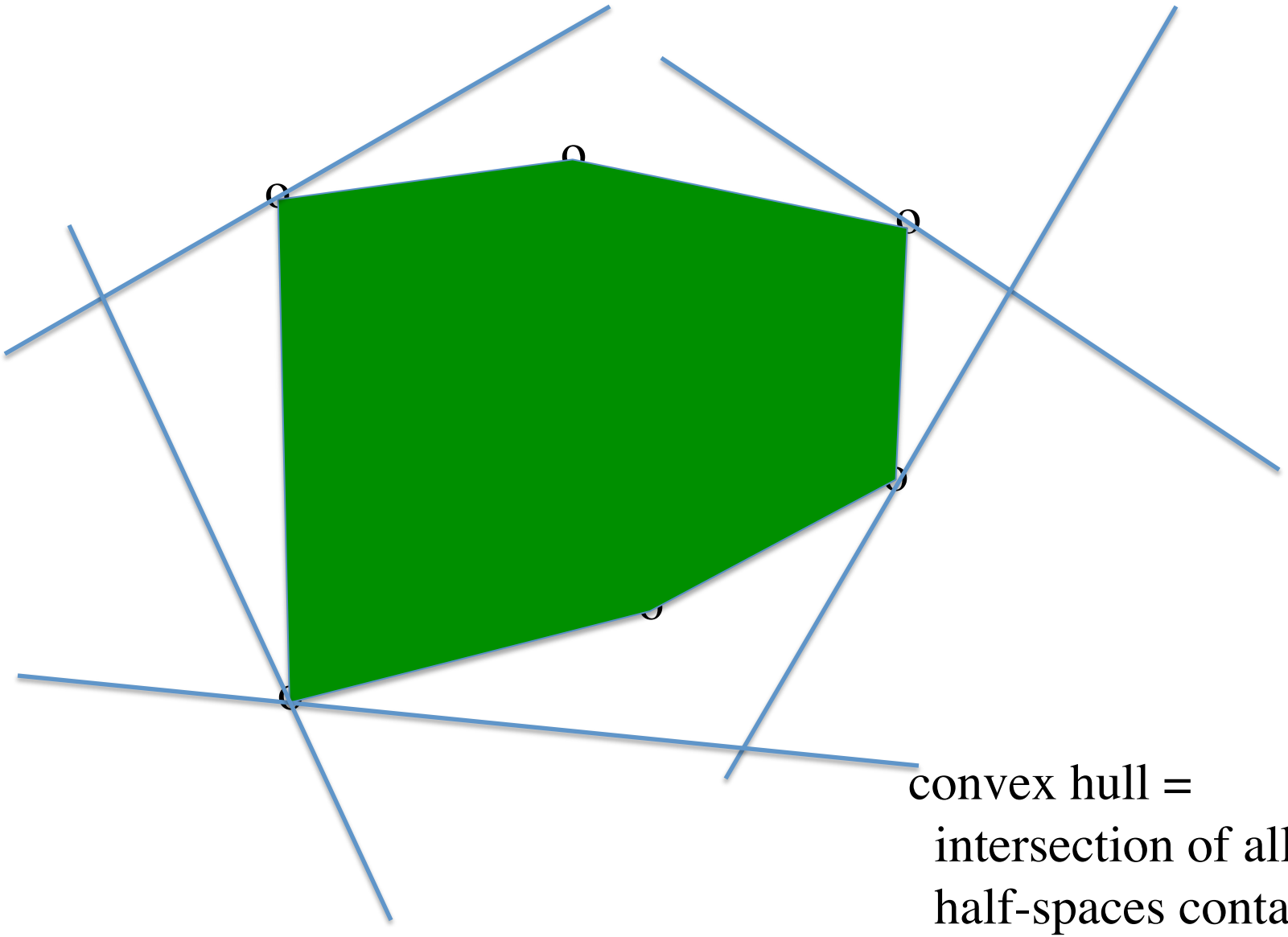
convex combination



convex combinations



convex hull =
union of all
convex combinations



convex hull =
intersection of all
half-spaces containing S

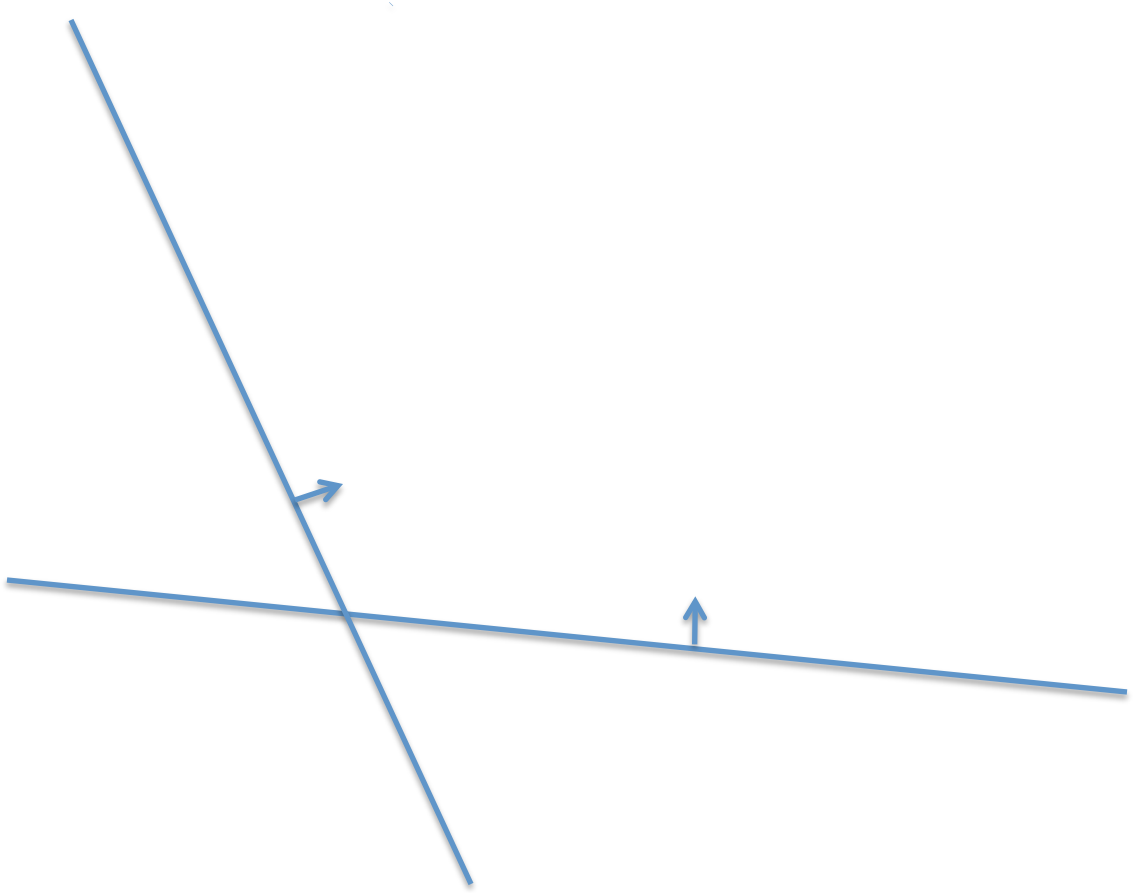
Convex hull CH(*S*)

- CH(*S*) is *smallest convex set* containing *S*.
- In $\mathbf{R}^2$, CH(*S*) is smallest area (or perimeter) convex polygon containing *S*.
- In $\mathbf{R}^2$, CH(*S*) is union of all triangles formed by triples of points in *S*.

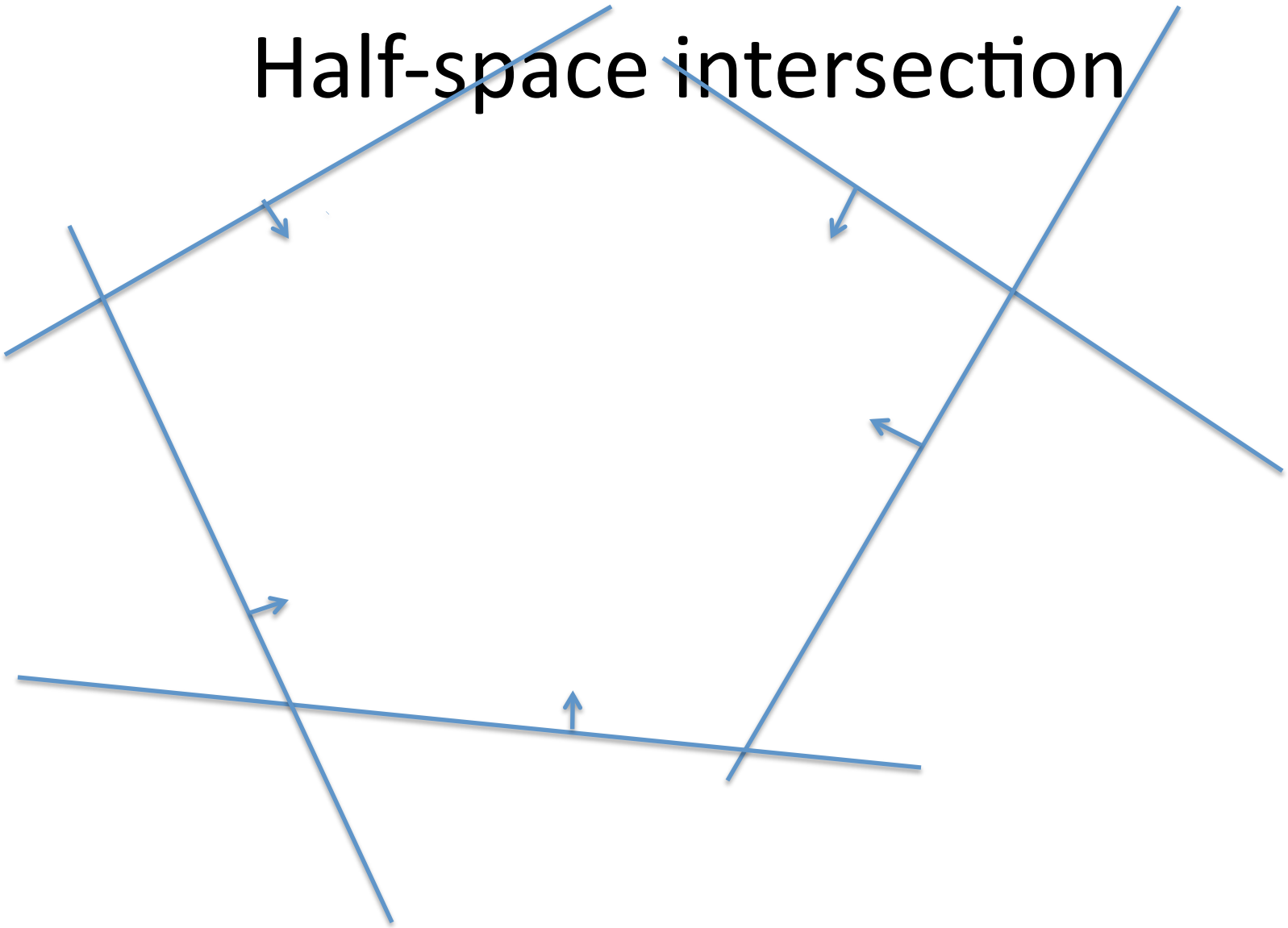
Convex hull CH(*S*)

- CH(*S*) is *smallest convex set* containing *S*.
- In $\mathbf{R}^2$, CH(*S*) is smallest area (or perimeter) convex polygon containing *S*.
- In $\mathbf{R}^2$, CH(*S*) is union of all triangles formed by triples of points in *S*.
- None of these descriptions/properties yield efficient algorithms; at best $O(|S|^3)$.

Half-space intersection

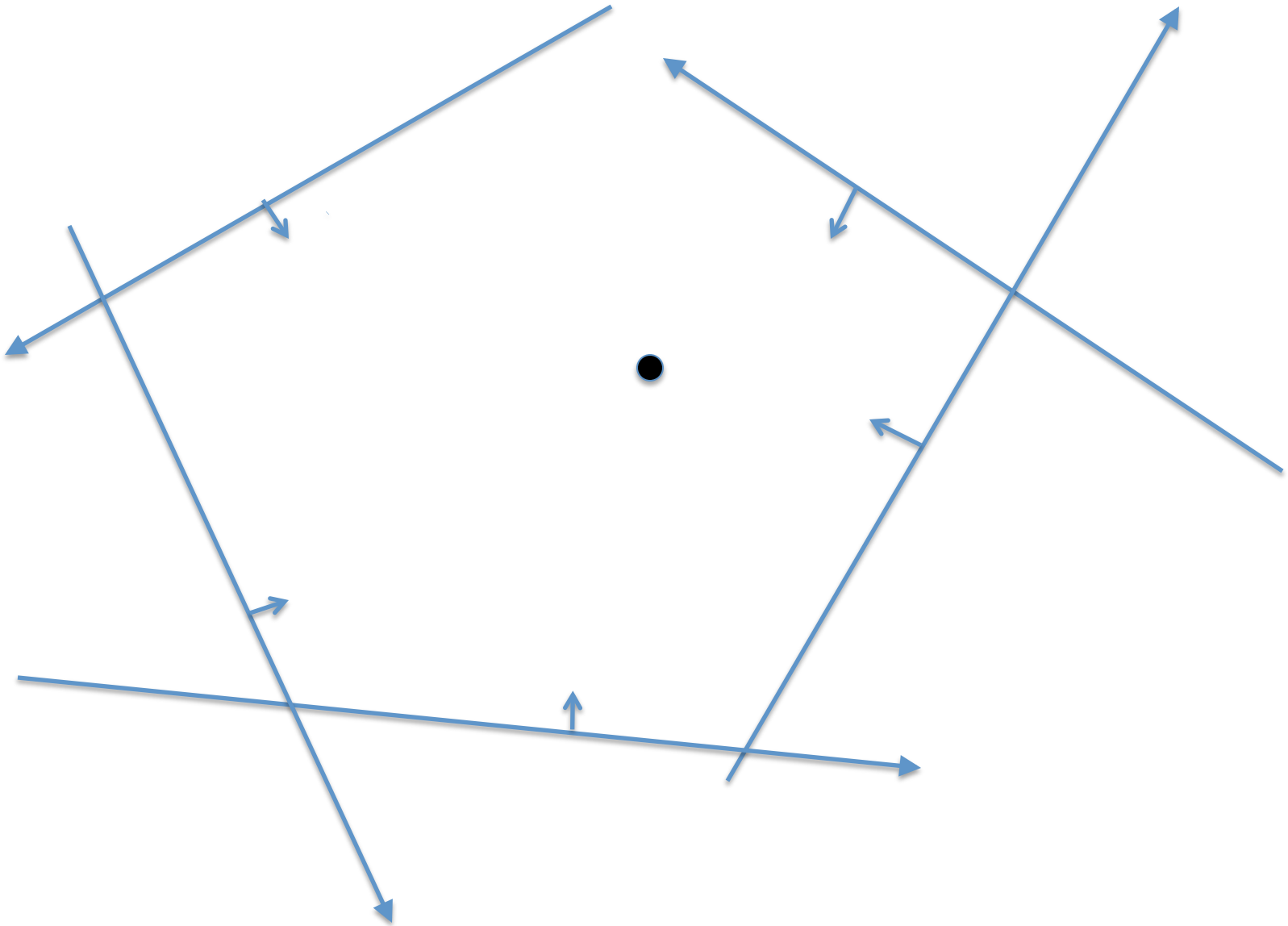


Half-space intersection



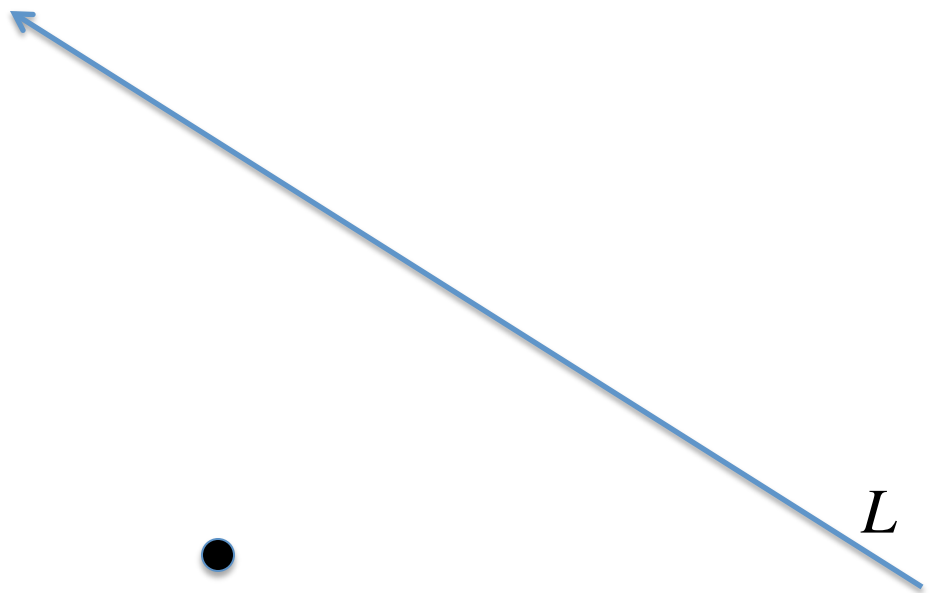
Half-space intersection

- suppose we have a *witness* to the non-emptiness of the intersection—may as well be the origin
- such half-spaces are defined by *oriented* lines (directed so that origin lies to the left)

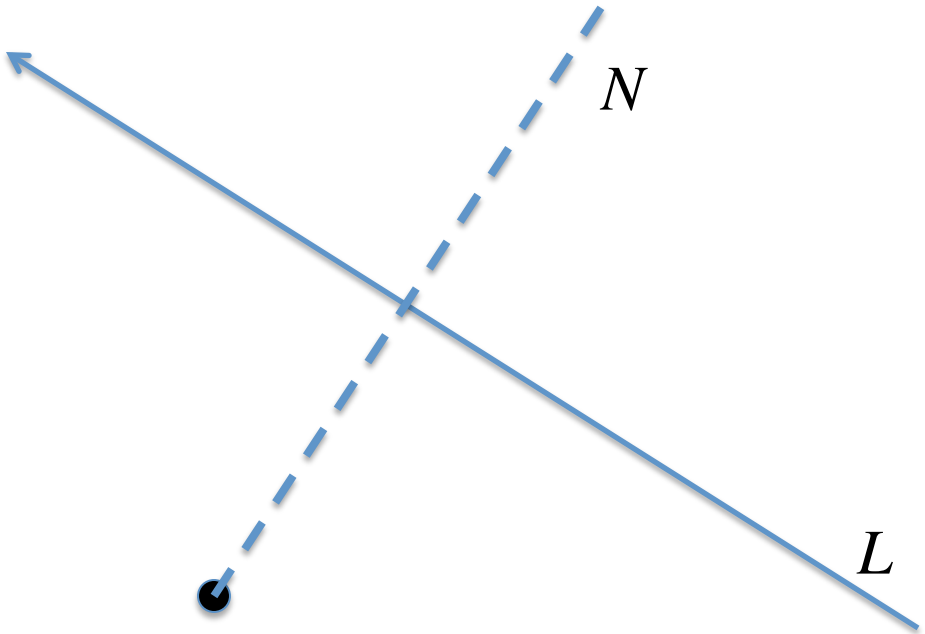


Polarity transform

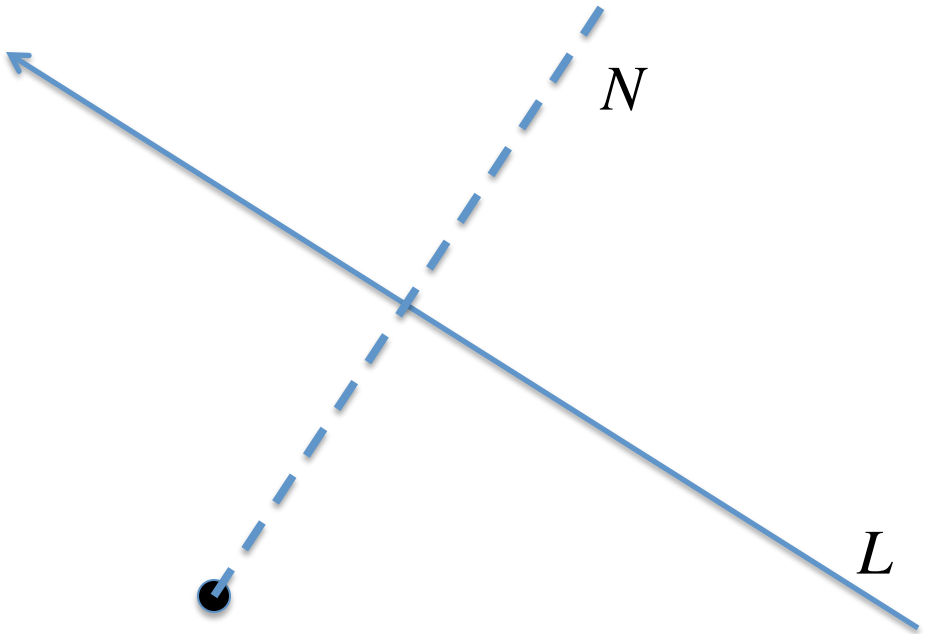
- an arbitrary line *L* that *avoids* the origin has the an equation of the form: $ax + by - 1 = 0$
- view as directed line where points to *left* (respectively *right*) make $ax + by - 1$ *negative* (respectively, *positive*)



$L: \; ax+by-1 = 0$

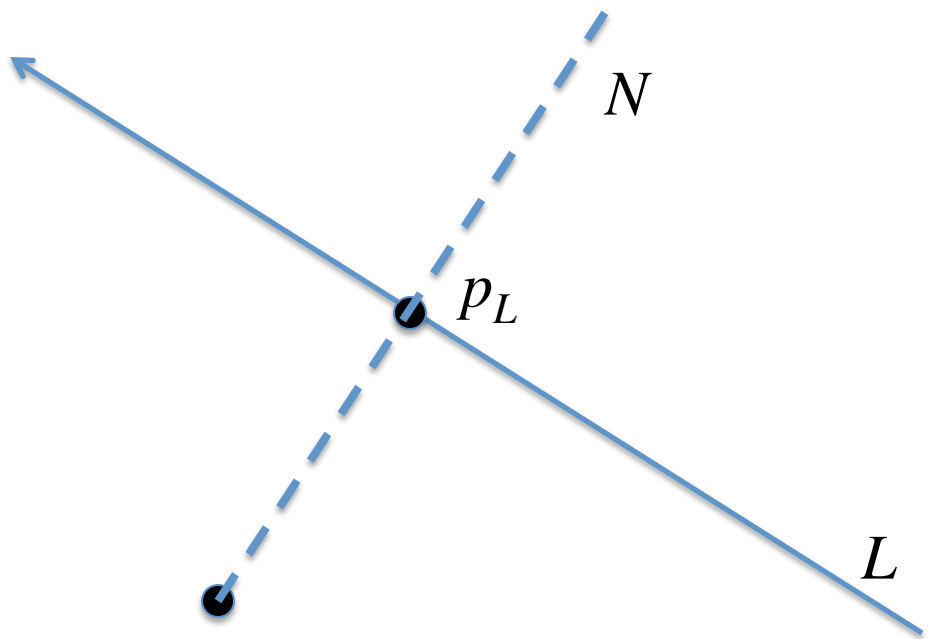


$L: \; ax+by-1 = 0$



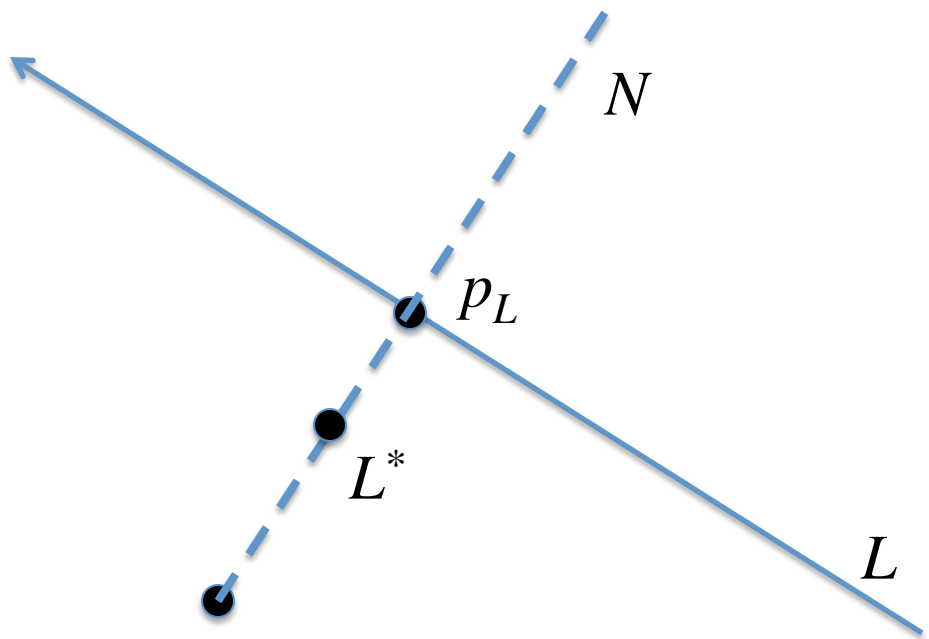
$L: \; ax+by-1 = 0$

$N: bx-ay = 0$



L: $ax+by-1 = 0$

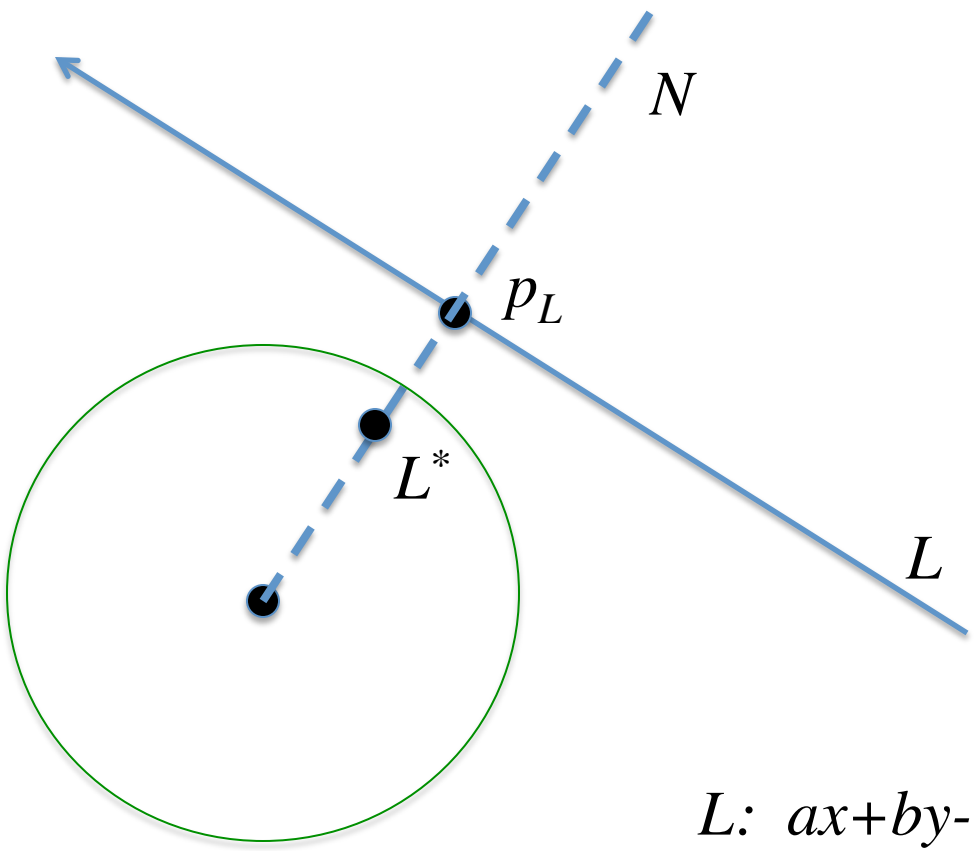
N: $bx-ay = 0$



L: $ax+by-1 = 0$

N: $bx-ay = 0$

L^*: (a,b)

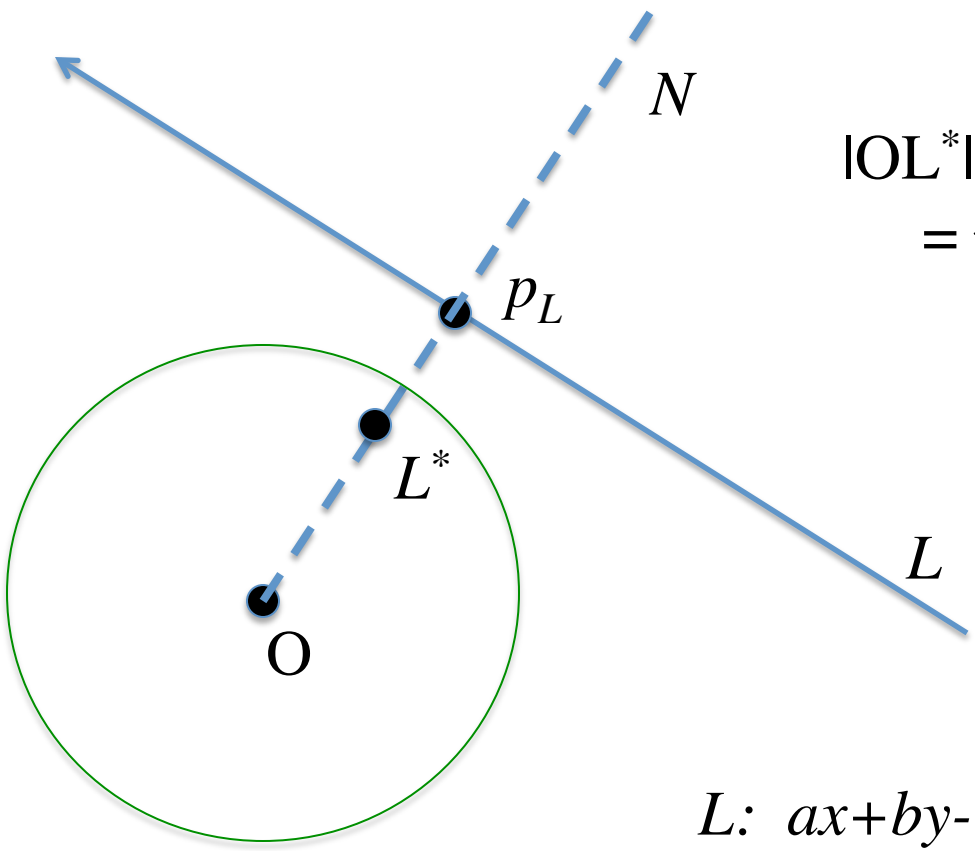


unit circle

L: $ax+by-1 = 0$

N: $bx-ay = 0$

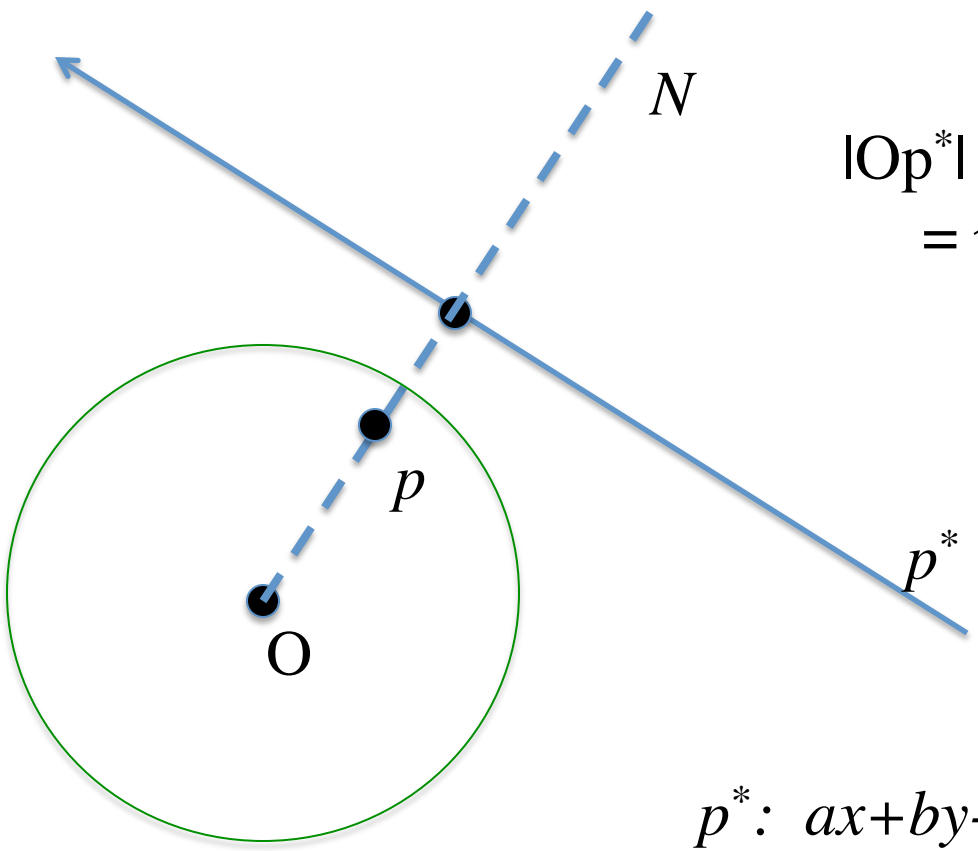
L^*: (a,b)



$|OL^*| = 1 / |OL|$
$= \sqrt{(a^2 + b^2)}$

unit circle

L: ax+by-1 = 0
N: bx-ay = 0
L^*: (a,b)



$|Op^*| = 1 / |Op|$
$= \sqrt{(a^2 + b^2)}$

unit circle

p^*: $ax+by-1 = 0$
N: $bx-ay = 0$
p: (a,b)

Properties

- [self inverse] $(p^*)^* = p$

Properties

- [self inverse] $(p^*)^* = p$ (and $(L^*)^* = L$)

Properties

- [self inverse] $(p^*)^* = p$
- [incidence preserving] if p belongs to L, then L^* belongs to p^*

Properties

- [self inverse] $(p^*)^* = p$
- [incidence preserving] if p belongs to L, then L^* belongs to p^*
- [sidedness reversing] if p lies to the left (right) of L, then L^* lies to the left (right) of p^*

Properties

- [self inverse] $(p^*)^* = p$
- [incidence preserving] if p belongs to L, then L^* belongs to p^*
- [sidedness reversing] if p lies to the left (right) of L, then L^* lies to the left (right) of p^*
- the line joining points p_1 and p_2 is the dual of the point formed by the intersection of the lines p_1^* and p_2^*

Equivalent problems

- [half-space intersection] finding all points that lie to the left of the (primal) lines defining the half-spaces
- [convex hull] finding all lines that lie to the right of all of the (dual) points
- in both cases a succinct description is a polygon (polytope); the boundary order is preserved under duality.

Remark

- Mount (lecture 8) presents a *different point-line duality transform* (based on slope & *y*-intercept):

  point *p:(a,b)* maps to line *p*: y=ax-b*

  (and vice-versa)

  this takes points to non-vertical lines

Same properties

- [self inverse] $(p^*)^* = p$
- [incidence preserving] if p belongs to L, then L^* belongs to p^*
- [sidedness reversing] if p lies to the left (right) of L, then L^* lies to the left (right) of p^*
- the line joining points p_1 and p_2 is the dual of the point formed by the intersection of the lines p_1^* and p_2^*

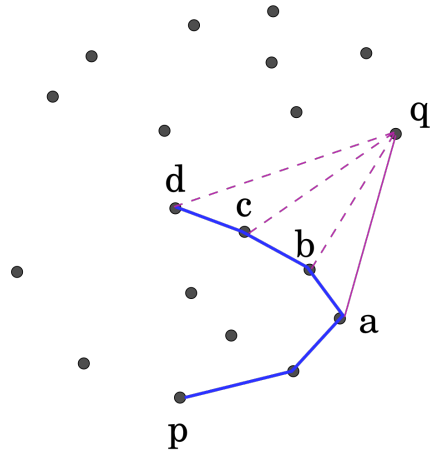
2-d convex hulls and sorting

- let $T_{sort}(n)$ and $T_{CH}(n)$ denote the worst case complexities of the sorting and convex hull problems (for input instances of size n)

2-d convex hulls and sorting

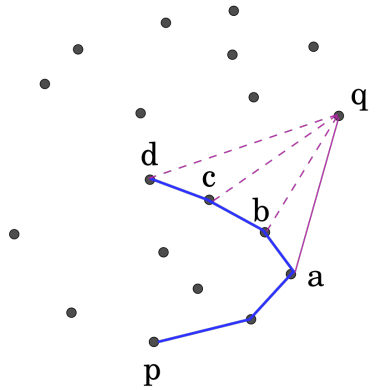
- let $T_{sort}(n)$ and $T_{CH}(n)$ denote the worst case complexities of the sorting and convex hull problems (for input instances of size n)
- we will show :
  - $T_{sort}(n) \leq T_{CH}(n) + \Theta(n)$
  - $T_{CH}(n) \leq T_{sort}(n) + \Theta(n)$

Graham Scan



1. **Sort by Y-order;** $p_1, p_2, \ldots, p_n$.
2. **Initialize.** **push** $(p_i, stack)$, $i = 1, 2$.
3. **for** $i = 3$ **to** n **do**
   while $\angle$ **next, top,** $p_i \neq$ **Left-Turn**
   pop $(stack)$
   push $(p_i, stack)$.
4. **return** *stack*.
5. **Invented by R. Graham '73.** (**Left and Right convex hull chains separately.**)

Analysis of Graham Scan



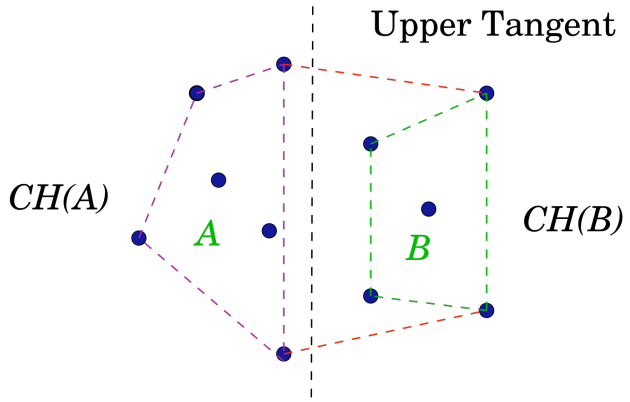
1. **Invariant: $\langle p_1, \ldots, top(stack) \rangle$ is convex. On termination, points in *stack* are on CH.**

2. **Orientation Test:** $D = \begin{Vmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{Vmatrix}$

   $\angle p, q, r$ is LEFT if $D > 0$, RIGHT if $D < 0$, and straight if $D = 0$.

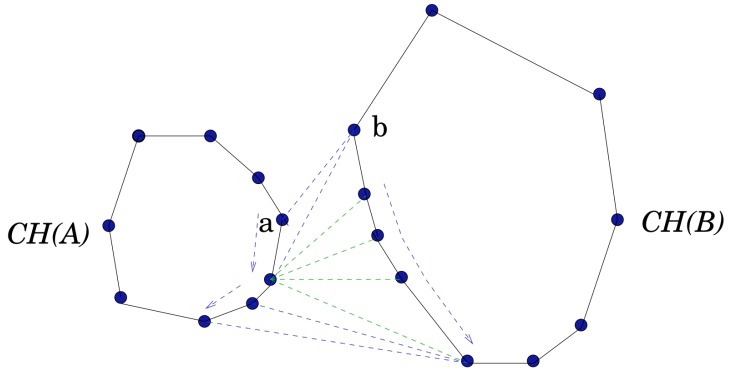
3. **After sorting, the scan takes $O(n)$ time: in each step, either a point is deleted, or added to stack.**

Divide and Conquer



- **Sort points by X-coordinates.**
- **Let A be the set of $n/2$ leftmost points, and B the set of $n/2$ rightmost points.**
- **Recursively compute $CH(A)$ and $CH(B)$.**
- **Merge $CH(A)$ and $CH(B)$ to obtain $CH(S)$.**

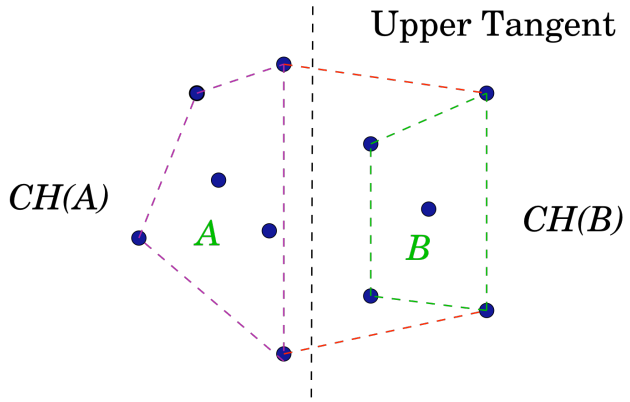
Merging Convex Hulls



Lower Tangent

- $a =$ **rightmost point of** $CH(A)$**.**
- $b =$ **leftmost point of** $CH(B)$**.**
- **while** ab **not lower tangent of** $CH(A)$ **and** $CH(B)$ **do**
  1. **while** ab **not lower tangent to** $CH(A)$
     set $a = a - 1$ **(move** a **CW);**
  2. **while** ab **not lower tangent to** $CH(B)$
     set $b = b + 1$ **(move** b **CCW);**
- **Return** ab

Analysis of D&C



- **Initial sorting takes $O(N \log N)$ time.**
- **Recurrence for divide and conquer**
  $T(N) = 2T(N/2) + O(N)$
- **$O(N)$ for merging (computing tangents).**
- **Recurrence solves to $T(N) = O(N \log N)$.**

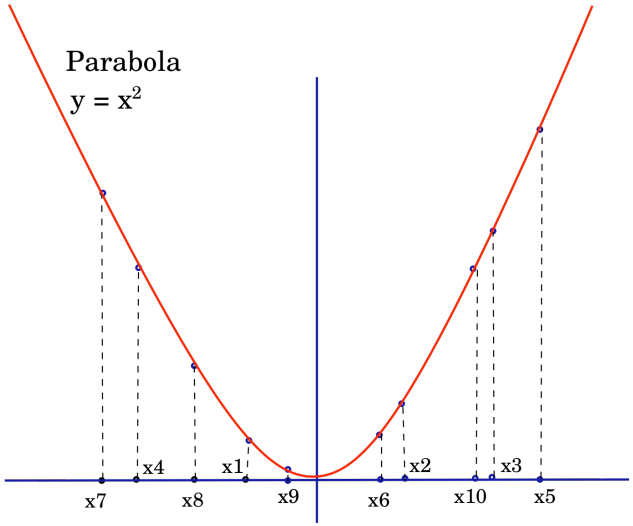
Can hulls be merged more efficiently?

What if hulls are not linearly separated?

Other sorting-inspired algorithms

Can hulls be constructed more efficiently?

Lower Bounds



- **Reduce sorting to convex hull.**
- **List of numbers to sort $\{x_1, x_2, \ldots, x_n\}$.**
- **Create point $p_i = (x_i, x_i^2)$, for each i.**
- **Convex hull of $\{p_1, p_2, \ldots, p_n\}$ has points in sorted x-order. $\Rightarrow$ CH of n points must take $\Omega(n \log n)$ in worst-case time.**

Other approaches…

- Convex hull algorithms that avoid sorting:
  - gift-wrapping (Jarvis) O(n h)
  - discard/filter interior points (QuickHull)