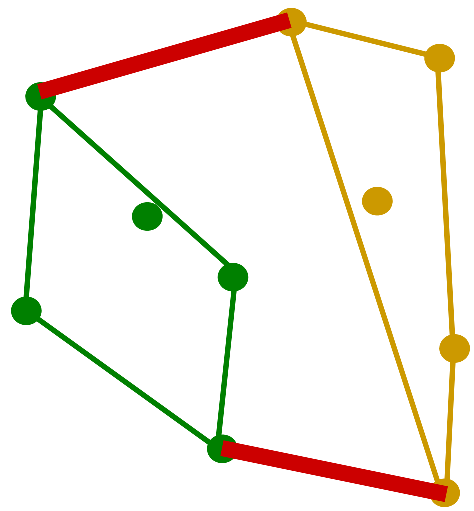
CMPS 6640/4040: Computational Geometry
Spring 2016



Convex Hulls

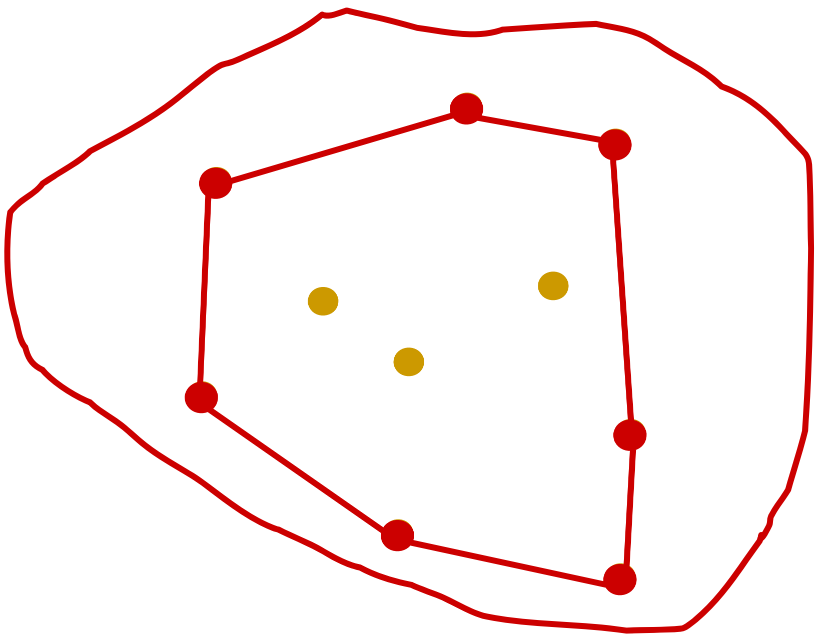
Carola Wenk

Convex Hull Problem

- Given a set of pins on a pinboard

  and a rubber band around them.

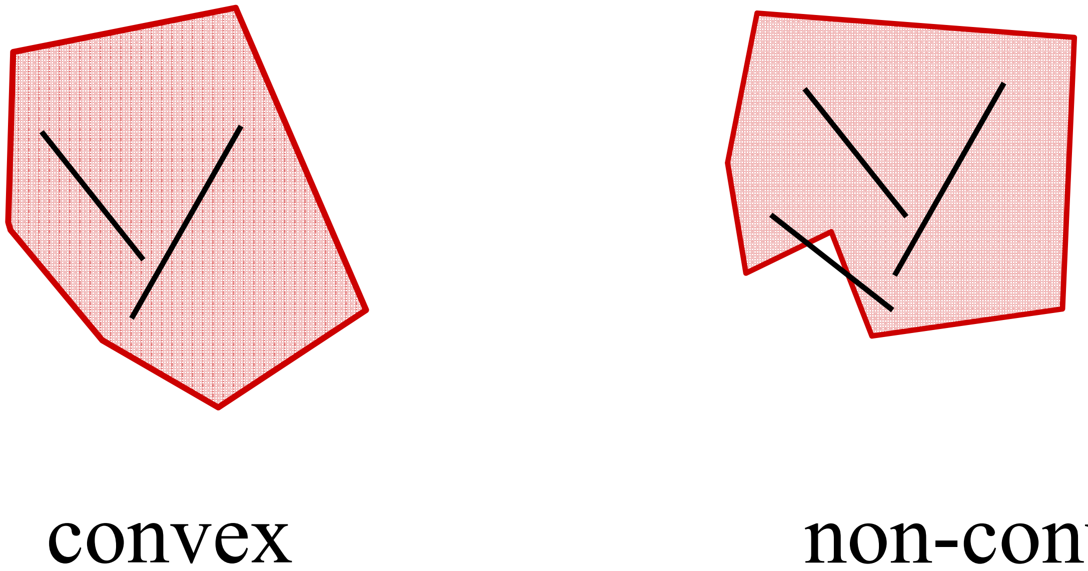
  How does the rubber band look when it snaps tight?



- The convex hull of a point set is one of the simplest shape approximations for a set of points.

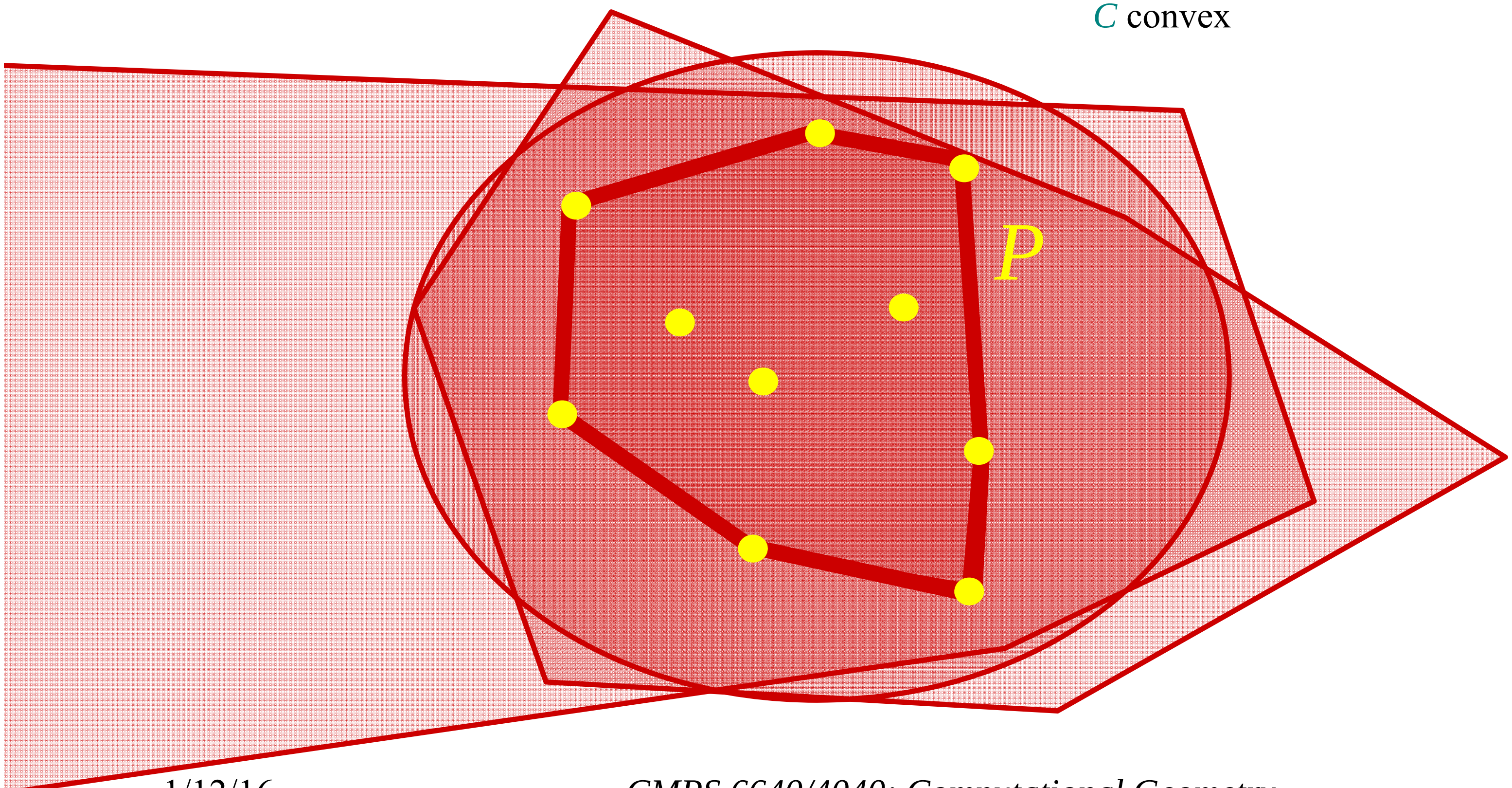
Convexity

- A set $C \subseteq \mathbf{R}^2$ is *convex* if for every two points $p,q \in C$ the line segment $\overline{pq}$ is fully contained in C.



Convex Hull

- The convex hull $CH(P)$ of a point set $P \subseteq \mathbf{R}^2$ is the smallest convex set $C \supseteq P$. In other words $CH(P) = \bigcap_{\substack{C \supseteq P \\ C \text{ convex}}} C$.

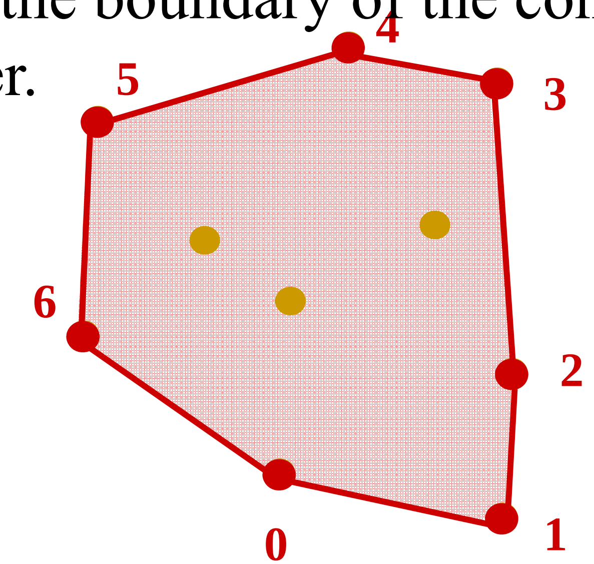


Convex Hull

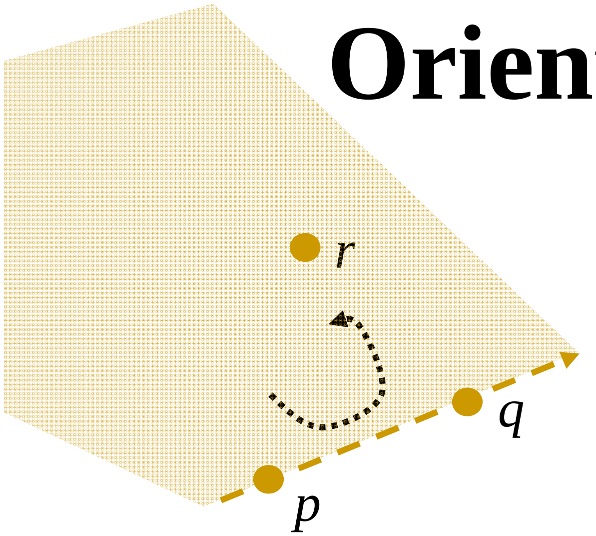
- **Observation:** CH(P) is the unique convex polygon whose vertices are points of P and which contains all points of P.
- **Goal:** Compute CH(P).

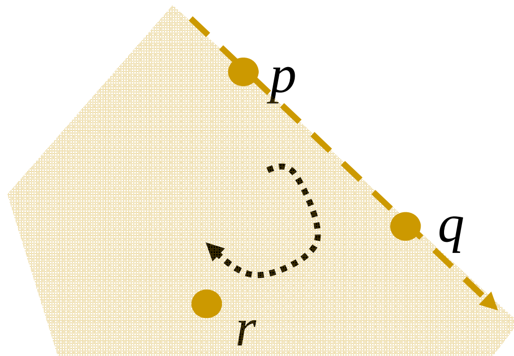
What does that mean? How do we represent/store CH(P)?

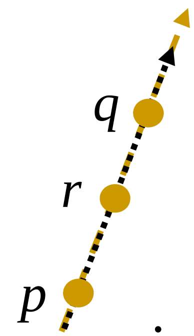
⇒ Represent the convex hull as the sequence of points on the convex hull polygon (the boundary of the convex hull), in counter-clockwise order.



Orientation Test / Halfplane Test







- positive orientation (counter-clockwise)
- *r* lies to the left of *pq*

- negative orientation (clockwise)
- $\overline{r}$ lies to the right of $\overrightarrow{pq}$

- zero orientation
- r lies on the line $\overrightarrow{pq}$

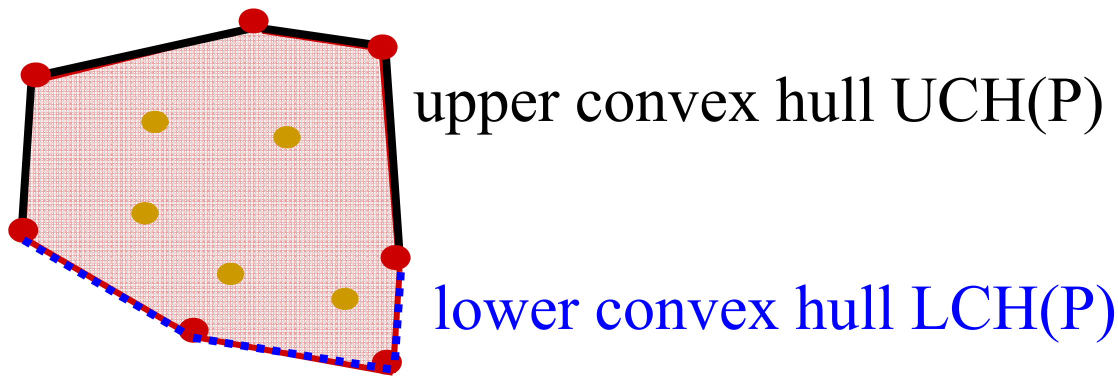
- Orient(p,q,r) = sign det $\begin{pmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{pmatrix}$,where $p = (p_x,p_y)$
- Can be computed in constant time

Graham’s Scan

Another incremental algorithm

- Compute solution by incrementally adding points
- Add points in which order?
  - Sorted by *x*-coordinate
  - But convex hulls are cyclically ordered

  → Split convex hull into **upper** and **lower** part



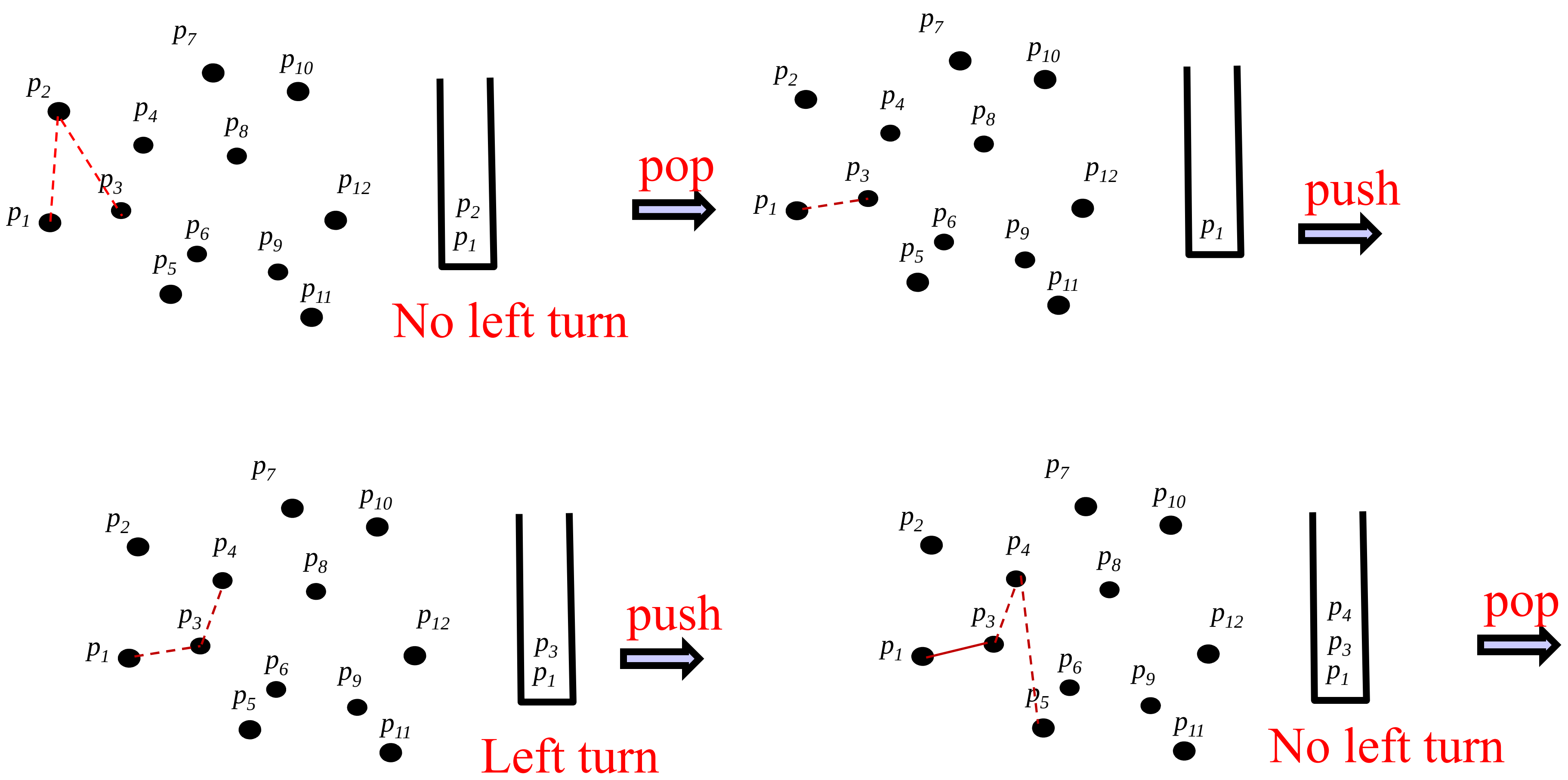
Graham's LCH

O(n log n) — Sort step
O(n) — for-loop

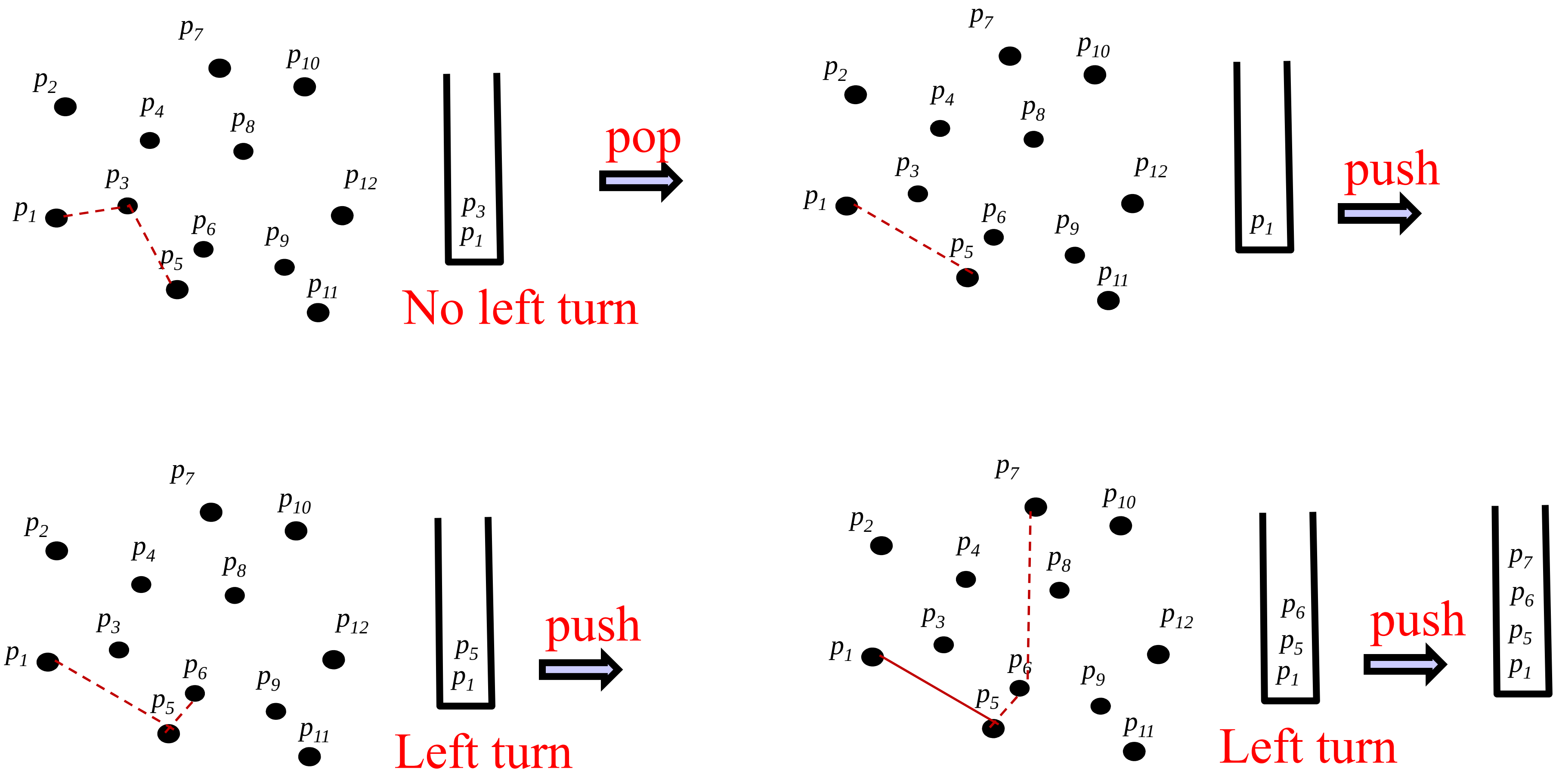
```
Algorithm Grahams_LCH(P):
// Incrementally compute the lower convex hull of P
Input: Point set P ⊆ R^2
Output: A stack S of vertices describing LCH(P) in counter-clockwise order

Sort P in increasing order by x-coordinate → P = {p_1,…,p_n}        O(n log n)
S.push(p_1)
S.push(p_2)
for i=3 to n                                                         O(n)
    while |S|>=2 and orientation(S.second(), S.top(), p_i,) <= 0 // no left turn
        S.pop()
    S.push(p_i)
```

- Each element is appended only once, and hence only deleted at most once $\Rightarrow$ the for-loop takes O(n) time
- O(n log n) time total

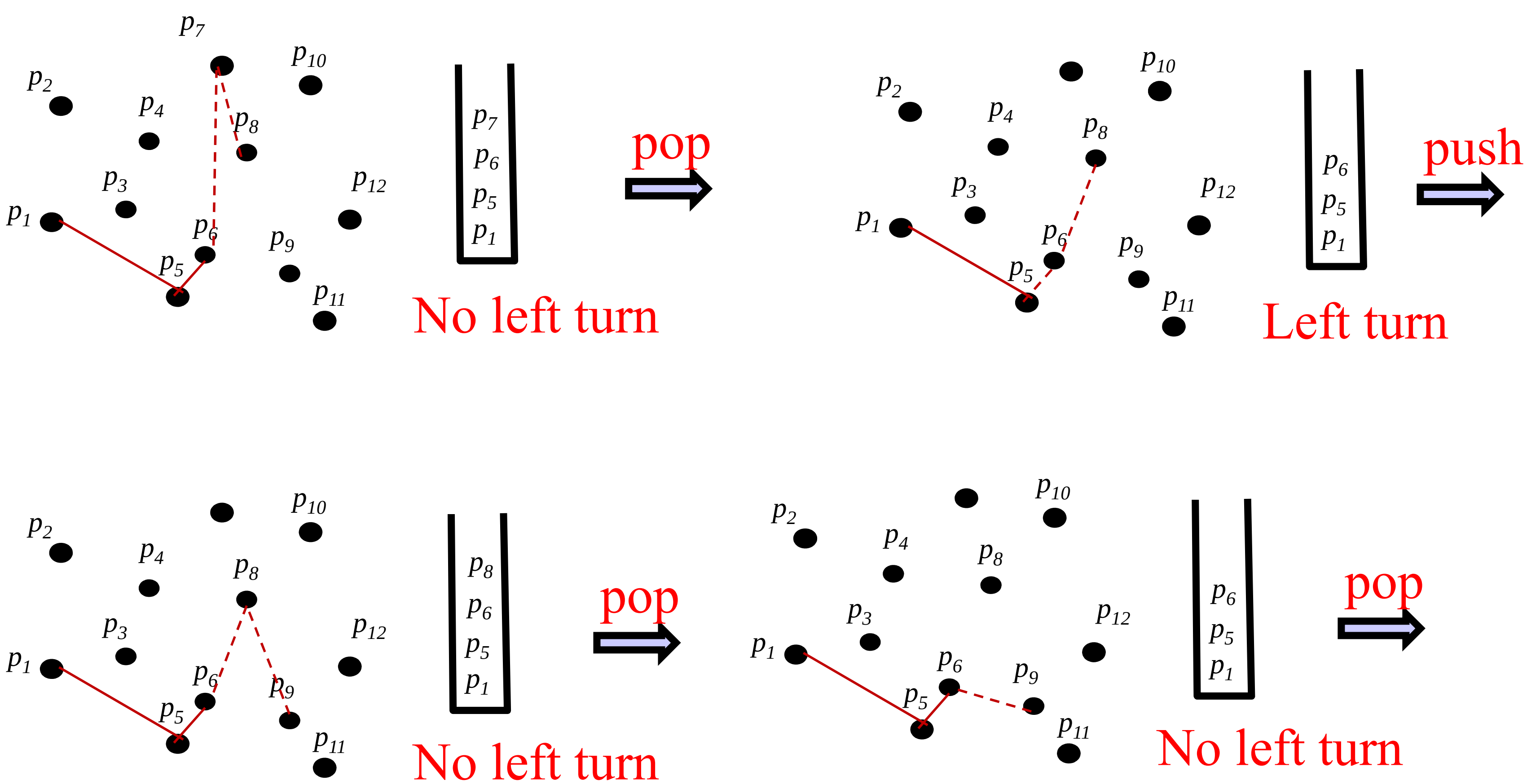
Graham's Scan



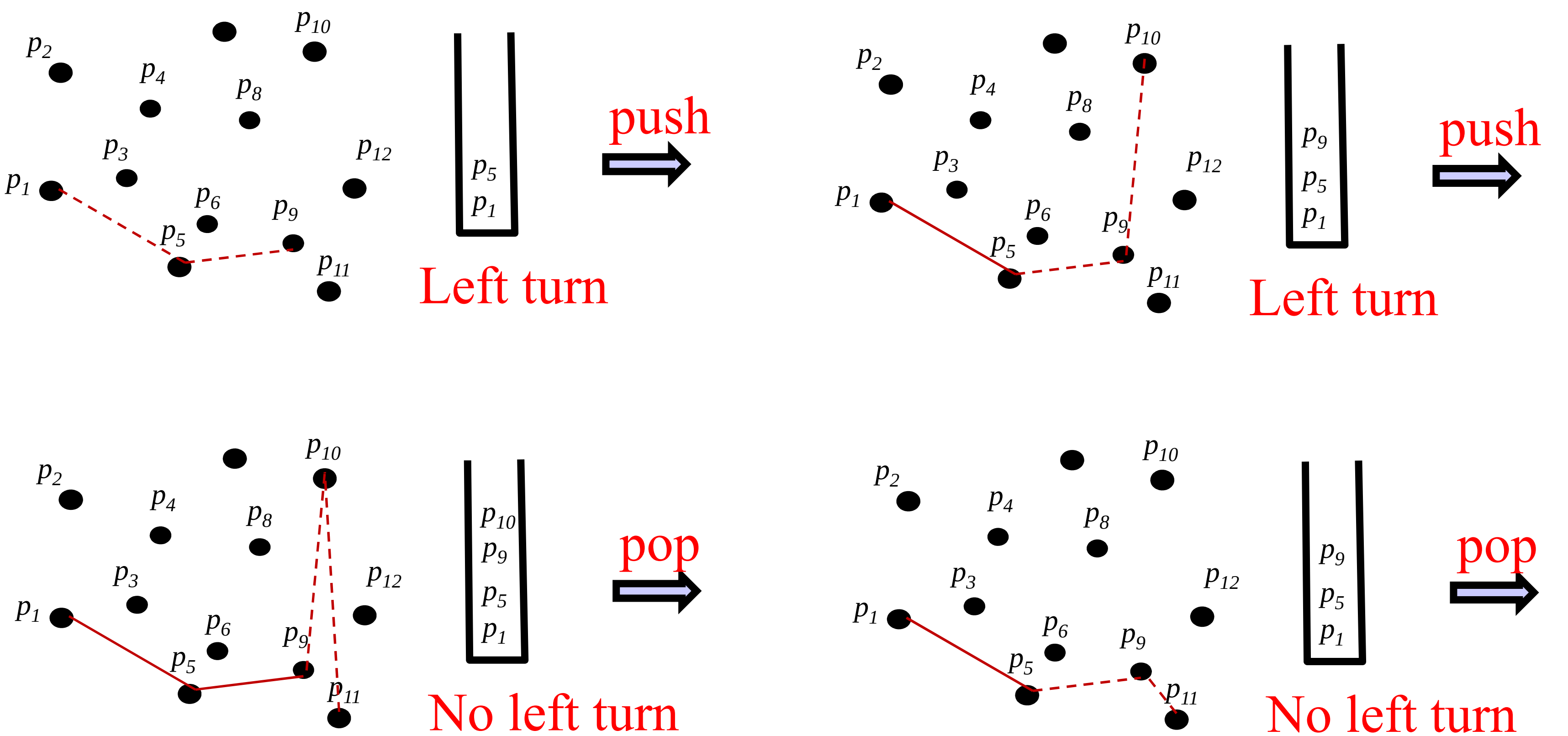
Graham's Scan



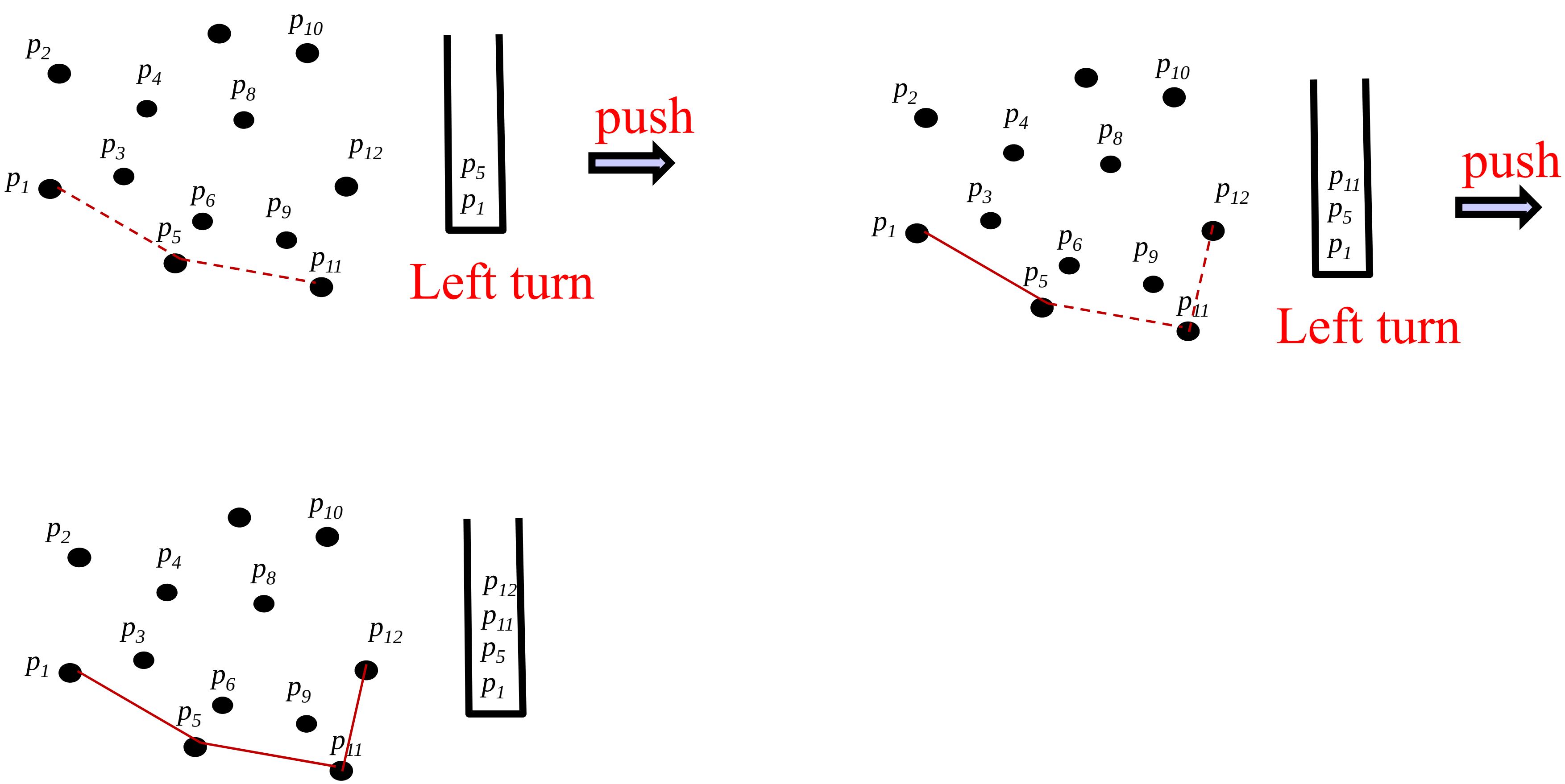
Graham's Scan



Graham's Scan

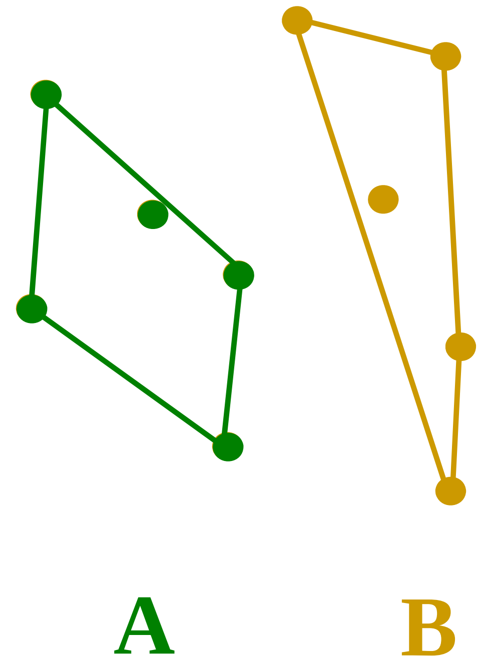


Graham's Scan



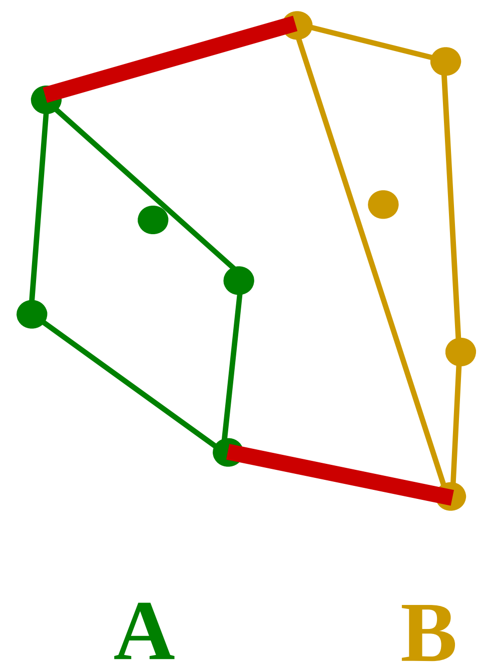
Convex Hull: Divide & Conquer

- Preprocessing: sort the points by x-coordinate
- Divide the set of points into two sets **A** and **B**:
    - **A** contains the left $\lfloor n/2 \rfloor$ points,
    - **B** contains the right $\lceil n/2 \rceil$ points
- Recursively compute the convex hull of **A**
- Recursively compute the convex hull of **B**
- Merge the two convex hulls



Merging

- **Find upper and lower tangent**
- With those tangents the convex hull of A∪B can be computed from the convex hulls of A and the convex hull of B in O(n) linear time



Finding the lower tangent

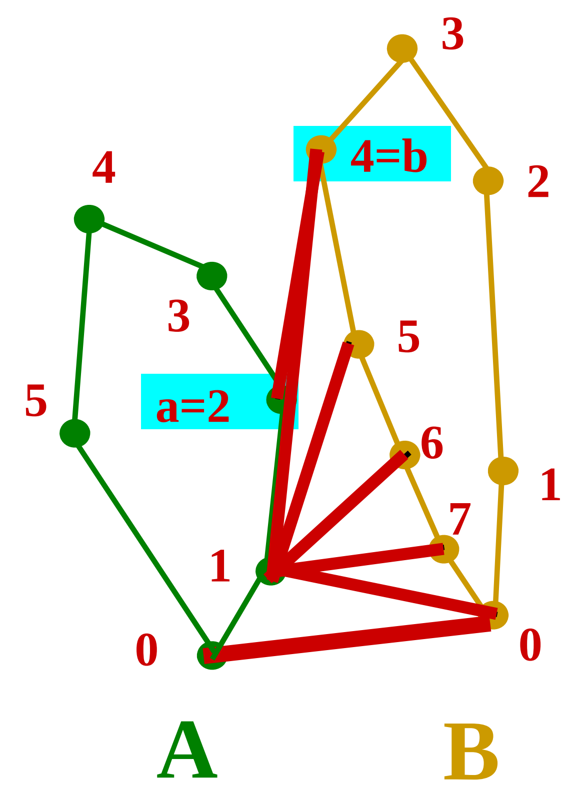
a = rightmost point of A

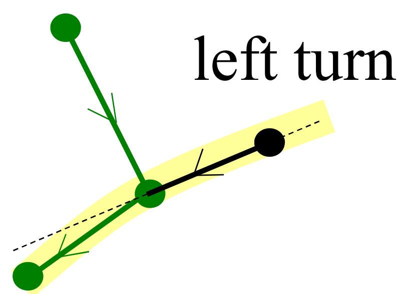
b = leftmost point of B

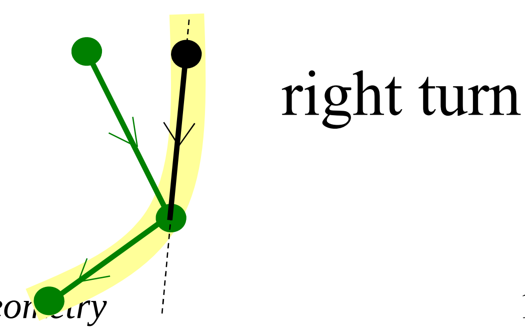
while T=ab not lower tangent to both
    convex hulls of A and B do{

    while T not lower tangent to
     convex hull of A do{
        a=a-1
    }
    while T not lower tangent to
     convex hull of B do{
        b=b+1
    }
}

check with orientation test







Convex Hull: Runtime

- Preprocessing: sort the points by x-coordinate — $O(n \log n)$ just once
- Divide the set of points into two sets **A** and **B**: — $O(1)$
  - **A** contains the left $\lfloor n/2 \rfloor$ points,
  - **B** contains the right $\lceil n/2 \rceil$ points
- Recursively compute the convex hull of **A** — $T(n/2)$
- Recursively compute the convex hull of **B** — $T(n/2)$
- Merge the two convex hulls — $O(n)$

Convex Hull: Runtime

- Runtime Recurrence:

  $T(n) = 2\,T(n/2) + cn$

- Solves to $T(n) = \Theta(n \log n)$

Master theorem

$$T(n) = a\,T(n/b) + f(n)\ ,$$

where $a \geq 1$, $b > 1$, and f is asymptotically positive.

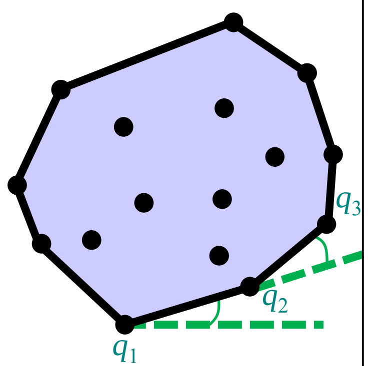
CASE 1: $f(n) = O(n^{\log_b a - \varepsilon})$
$\Rightarrow T(n) = \Theta(n^{\log_b a})$.

CASE 2: $f(n) = \Theta(n^{\log_b a} \log^k n)$
$\Rightarrow T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.

CASE 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$ and $a\,f(n/b) \leq c\,f(n)$
$\Rightarrow T(n) = \Theta(f(n))$.

Convex hull: $a = 2$, $b = 2$ $\Rightarrow$ $n^{\log_b a} = n$
$\Rightarrow$ CASE 2 $(k = 0)$ $\Rightarrow$ $T(n) = \Theta(n \log n)$.

Jarvis' March (Gift Wrapping)



Algorithm Giftwrapping_CH(P):
// Compute CH(P) by incrementally inserting points from left to right
Input: Point set $P \subseteq \mathbf{R}^2$
Output: List $q_1, q_2, \ldots$ of vertices in counter-clockwise order around CH(P)
q_1 = point in P with smallest y (if ties, with smallest x)
q_2 = point in P with smallest angle to horizontal line through q_1
$i = 2$
do {
  i++
  q_i = point with smallest angle to line through q_{i-2} and q_{i-1}
} while $q_i \neq q_1$

- Runtime: O(hn) , where $n = |P|$ and h = #points on CH(P)
- Output-sensitive algorithm

Chan's Algorithm

- Runtime goal: $O(n \log h)$, where $n = |P|$ and h = #points on CH(P)
- Output-sensitive algorithm

Lower Bound

- Comparison-based sorting of n elements takes $\Omega(n \log n)$ time.
- How can we use this lower bound to show a lower bound for the computation of the convex hull of n points in $\mathbf{R}^2$?

Lower Bound

- Comparison-based sorting of n elements takes $\Omega(n \log n)$ time.
- How can we use this lower bound to show a lower bound for the computation of the convex hull of n points in $\mathbf{R}^2$?
- Devise a sorting algorithm which uses the convex hull and otherwise only linear-time operations
  - $\Rightarrow$ Since this is a comparison-based sorting algorithm, the lower bound $\Omega(n \log n)$ applies
  - $\Rightarrow$ Since all other operations need linear time, the convex hull algorithm has to take $\Omega(n \log n)$ time

CH_Sort

Algorithm CH_Sort(S):

/* Sorts a set of numbers using a convex hull algorithm.
Converts numbers to points, runs CH, converts back to sorted sequence. */

Input: Set of numbers $S \subseteq \mathbf{R}$

Output: A list L of of numbers in S sorted in increasing order

$P = \emptyset$

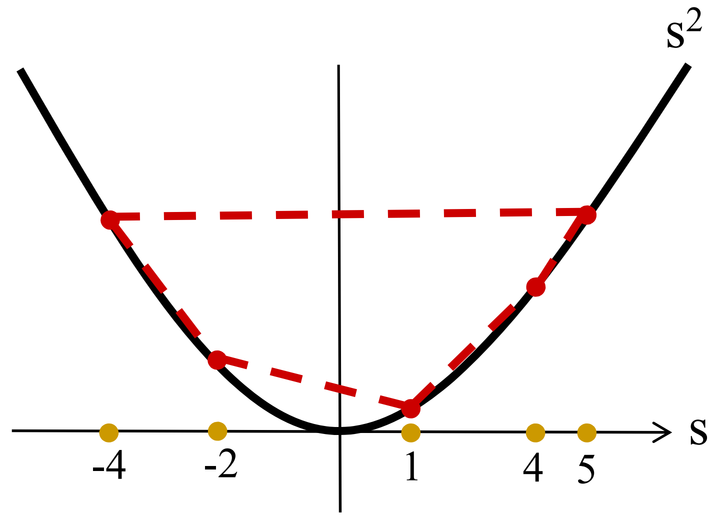
for each $s \in S$ insert (s, s^2) into P

$L' = \text{CH}(P)$ // compute convex hull

Find point $p' \in P$ with minimum x-coordinate

for each $p = (p_x, p_y) \in L'$, starting with p',
add p_x into L

return L



Convex Hull Summary

- Graham's scan: $O(n \log n)$
- Divide-and-conquer: $O(n \log n)$
- Jarvis' march (gift wrapping): $O(nh)$
- Chan's algorithm: $O(n \log h)$

- Lower bound: $\Omega(n \log n)$