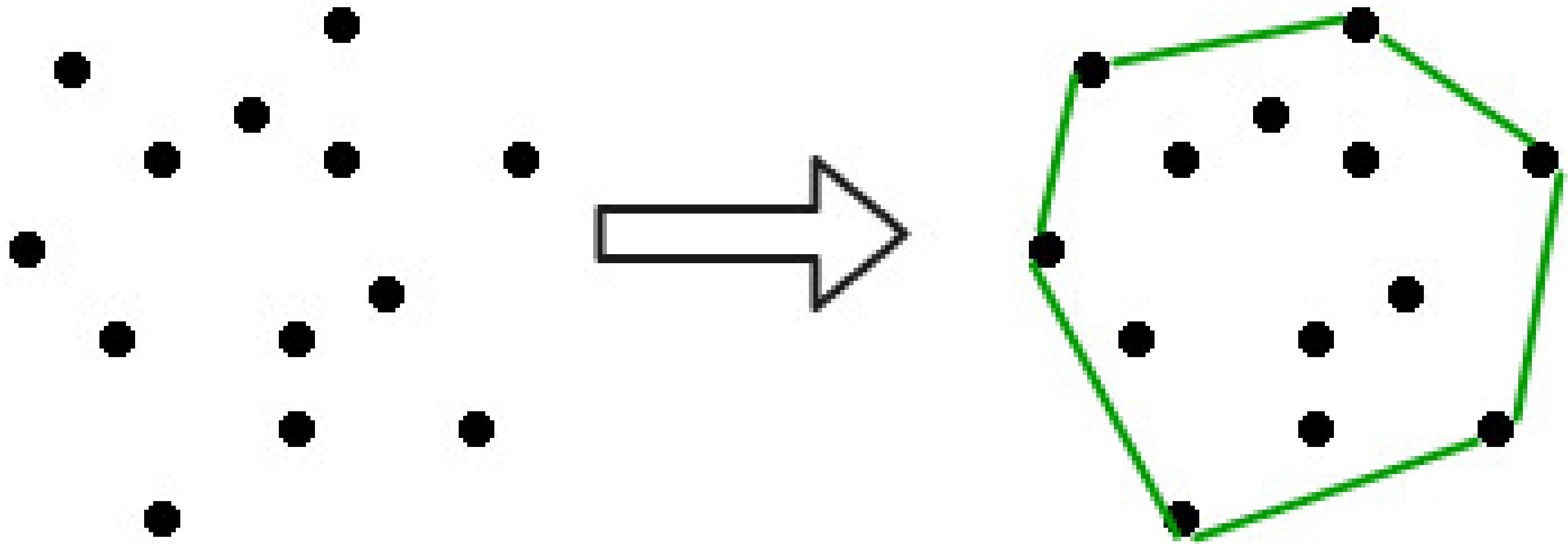
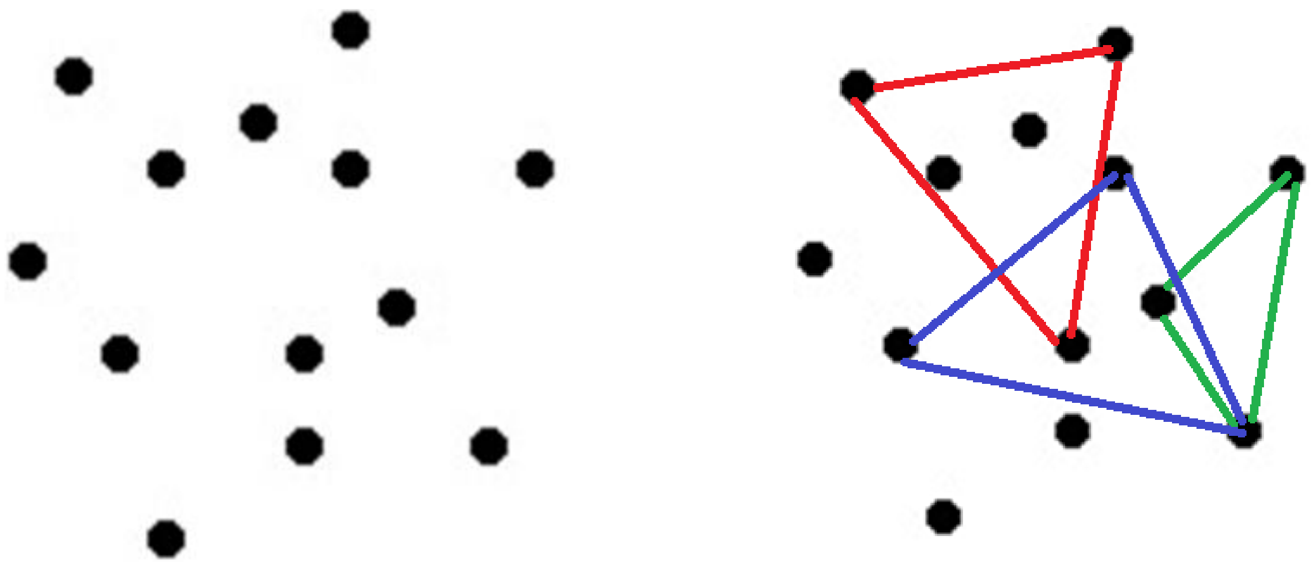
CONVEX HULL

Convex Hull (CH)



- Closed region including / enclosing all the points
- A convex hull of a set of points S in the plane is the enclosing convex polygon with:
  - Smallest area
  - Smallest perimeter

Standard algorithms for constructing a Convex Hull



Algo : Nonextreme points

Algorithm: INTERIOR POINTS

for each i do

for each $j \neq i$ do

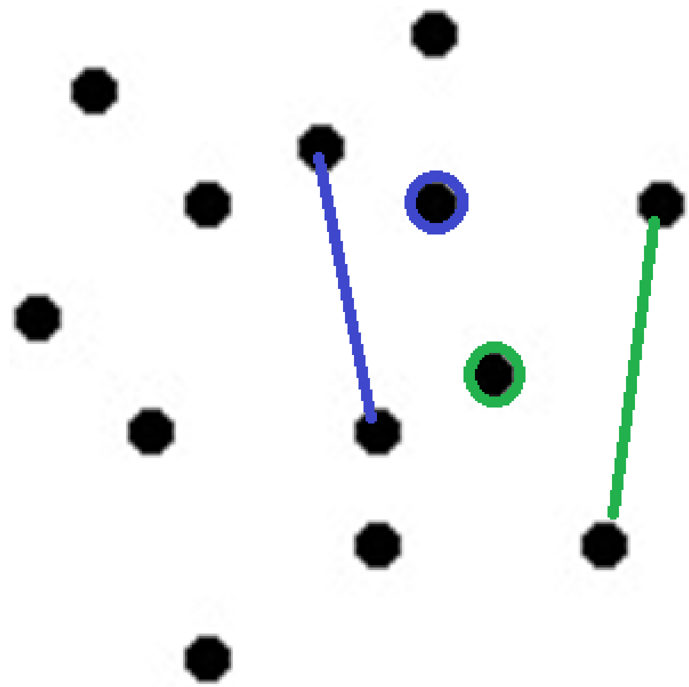
for each $k \neq i \neq j$ do

  for each $l \neq i \neq j \neq k$ do

    if $p_l \in \triangle(p_i, p_j, p_k)$

      then p_l is nonextreme

- **A directed edge is not extreme if there is some point that is not left of it or on it**

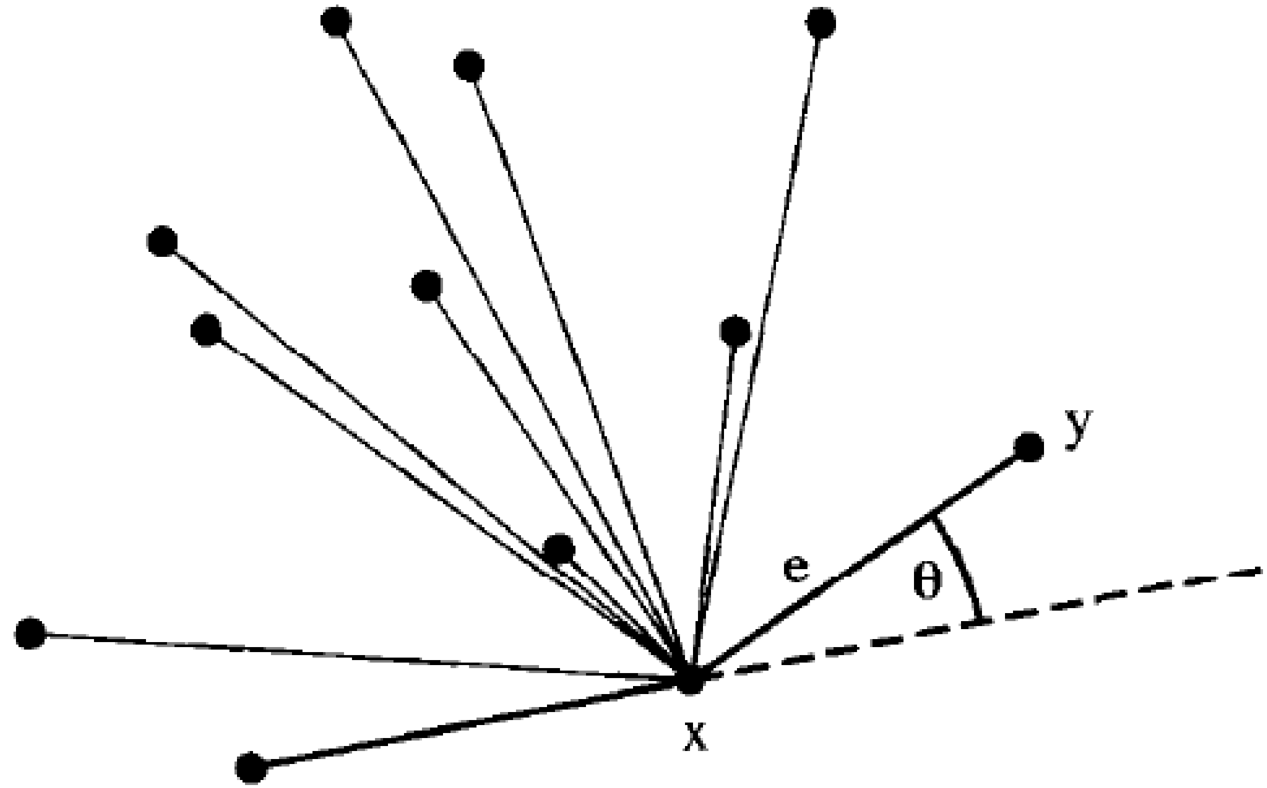


Algo

Algorithm: EXTREME EDGES

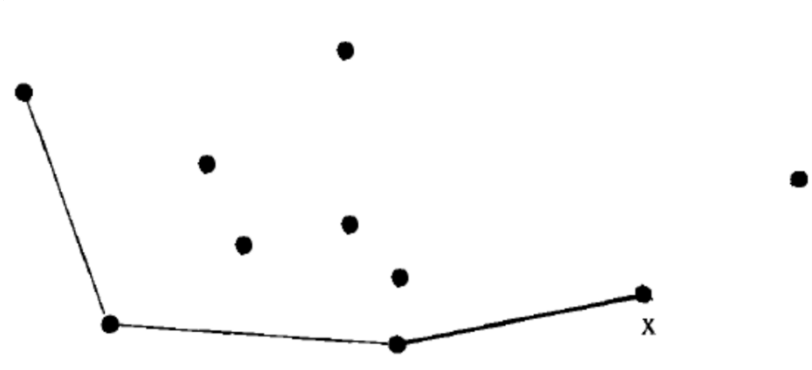
for each i do
for each $j \neq i$ do
  for each $k \neq i \neq j$ do
    if p_k is *not* left or on (p_i, p_j)
      then (p_i, p_j) is not extreme

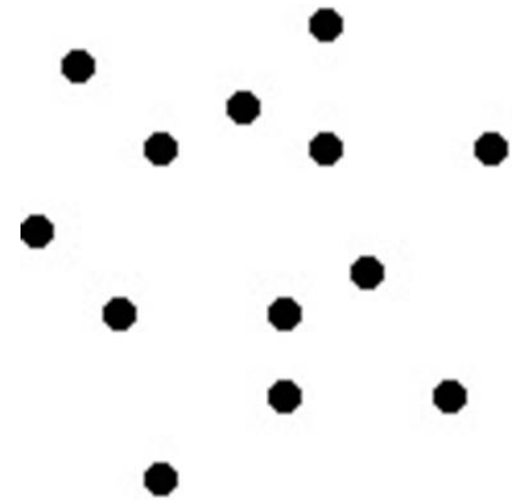
A general pic



- The point that makes the smallest counter clockwise angle Θ with respect to the previous hull edge must determine an extreme edge

How do we start Gift wrapping?





- **How do we get the first convex hull edge?**
- Find the lowest y coordinate point, i_0
- Draw a line L parallel to x axis along i_0
- Take the counter clockwise angle of the edge i_0y with respect to L for all $y \epsilon S$
- Select the point which takes the smallest counterclockwise angle

Pseudo code : Gift Wrapping

Algorithm: GIFT WRAPPING

Find the lowest point (smallest y coordinate).

Let i_0 be its index, and set $i \leftarrow i_0$.

repeat

    for each $j \neq i$ do

        Compute counterclockwise angle θ from previous hull edge.

    Let k be the index of the point with the smallest θ.

    Output (p_i, p_k) as a hull edge.

    $i \leftarrow k$

until $i = i_0$

Time Complexity

Algorithm: GIFT WRAPPING

Find the lowest point (smallest y coordinate).

Let i_0 be its index, and set $i \leftarrow i_0$.

repeat

    for each $j \neq i$ do

        Compute counterclockwise angle θ from previous hull edge.

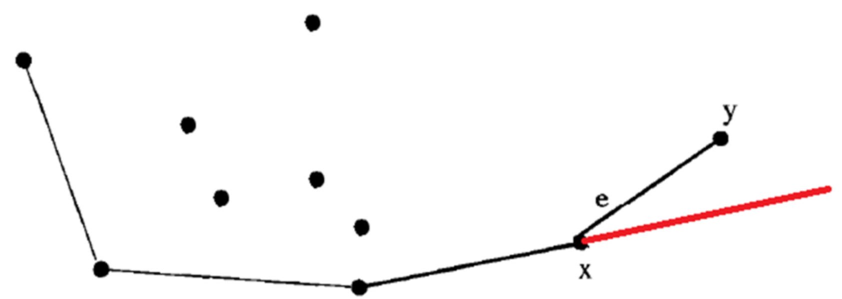
    Let k be the index of the point with the smallest θ.

    Output (p_i, p_k) as a hull edge.

    $i \leftarrow k$

until $i = i_0$

- It depends on what all parameters?
- Number of points n ?
- Number of edges of convex hull h?
- O(nh) is the time complexity
- Hence, Gift wrapping is output sensitive
- It runs faster when the hull is small
- Worst case time complexity ?
- O(n^2) : O(n) work for each hull edge

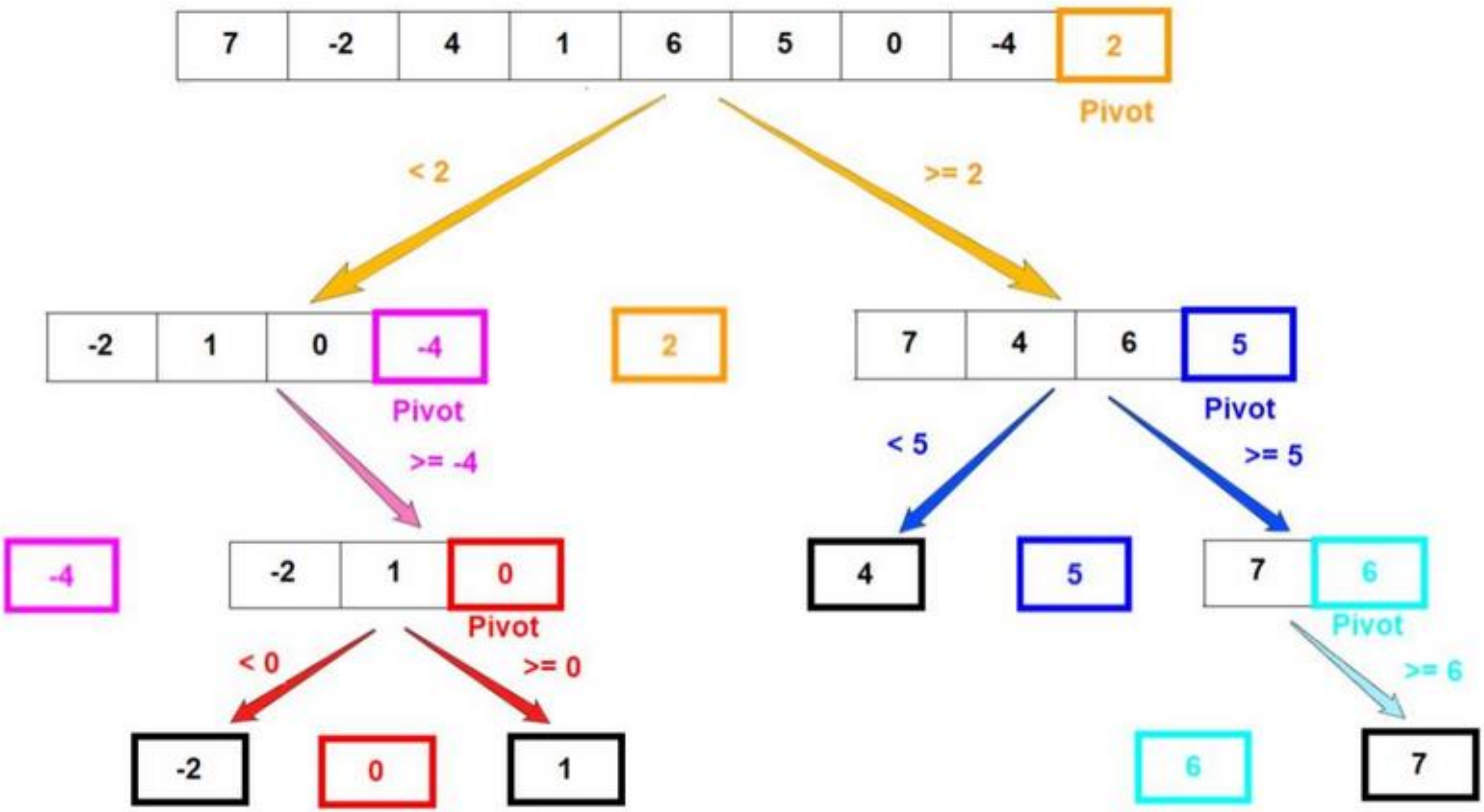


QUICK HULL

QUICK HULL

- Proposed by independent researchers in 1970’s
- Named by Preparata and Shamos [1985] due to its similarity with Quick sort algorithm
- What all we know about **Quick sort algo**?
- It is a divide and conquer algorithm
- Its average case complexity is $O(n \log n)$
- Its worst case complexity is $O(n^2)$

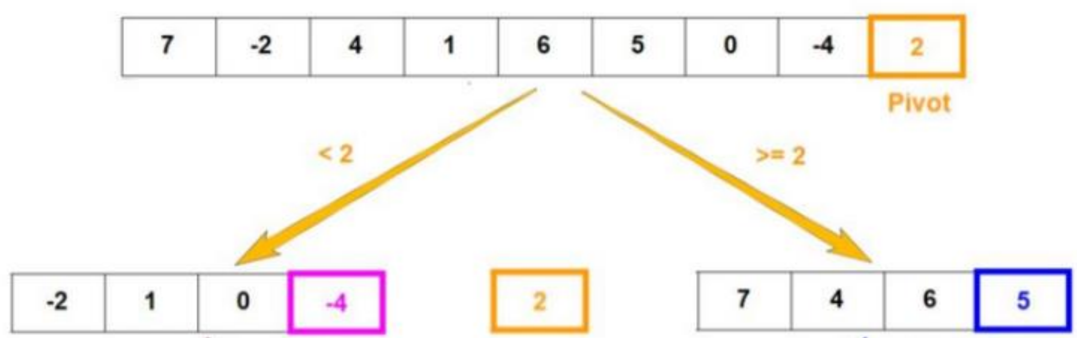
Quick Sort : Pictorial representation



Quick Sort : A Divide and Conquer strategy

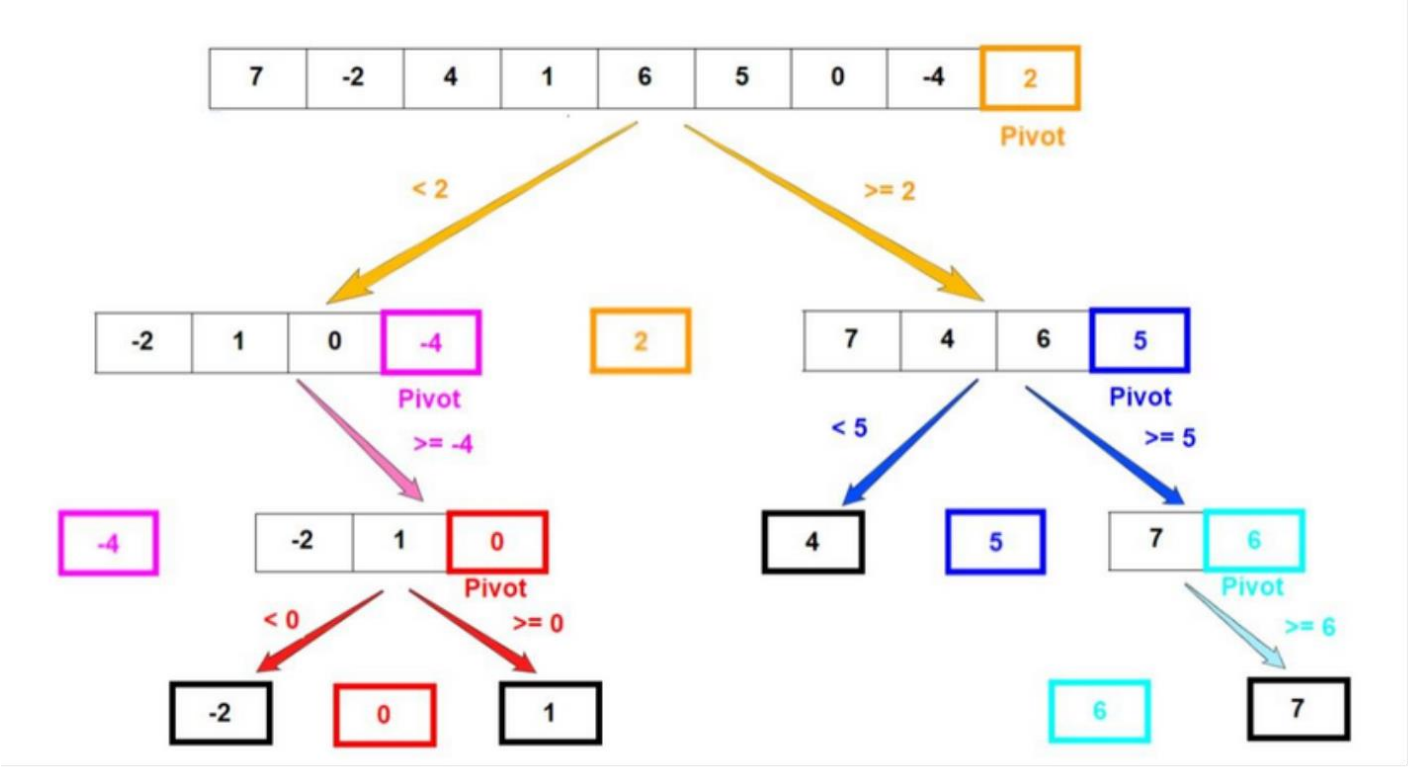
- Divide :
  - Partition the array A[p..r] to two sub arrays

    A[p..q-1] and A[q+1..r], where q is computed as part of the *PARTITION* function
  - Each element of A[p..q-1] is less than or equal to A[q]
  - Each element of A[q+1..r] is greater than A[q]
  - The sub arrays can be empty or non-empty



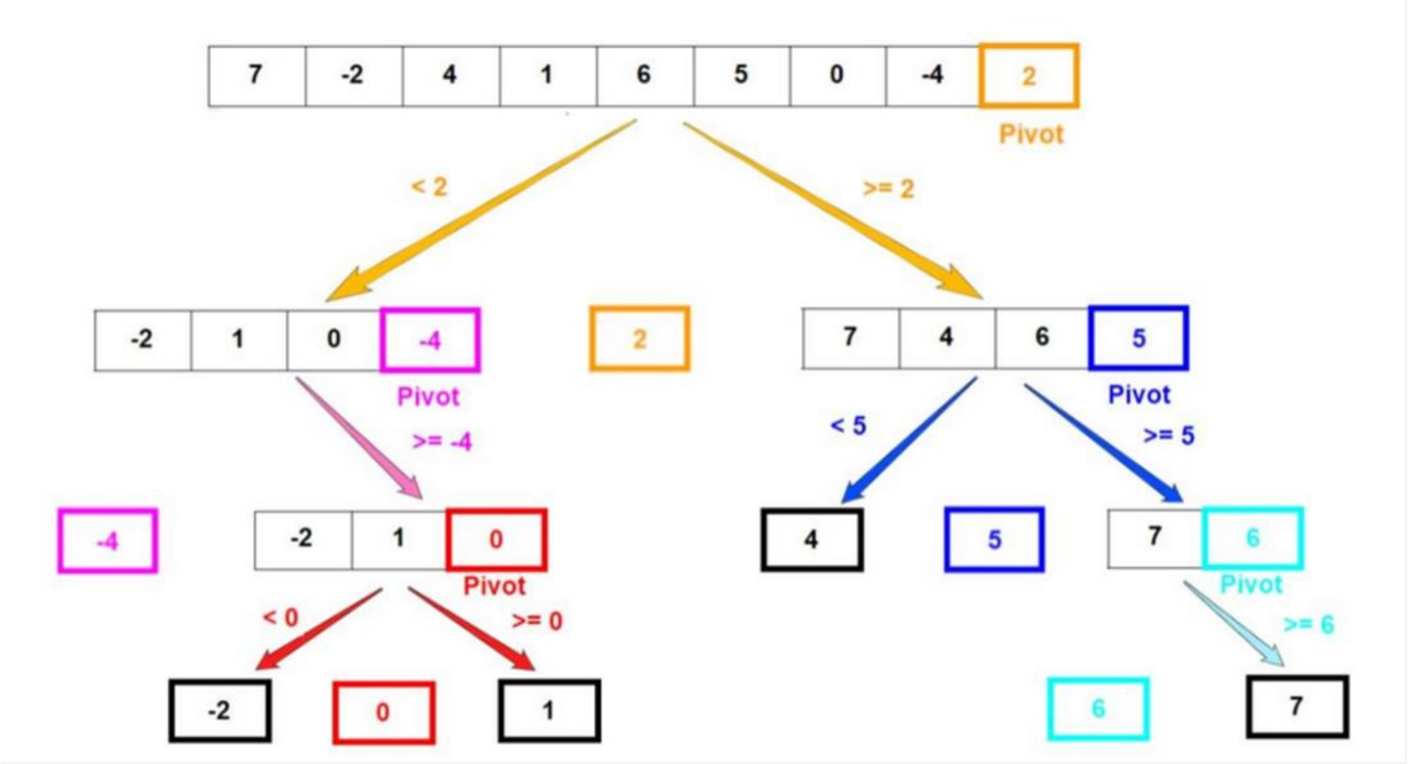
Quick Sort : A Divide and Conquer strategy

- Conquer :
  - Sort the two sub arrays A[p..q-1] and A[q+1..r]
  - By recursive calls to quick sort

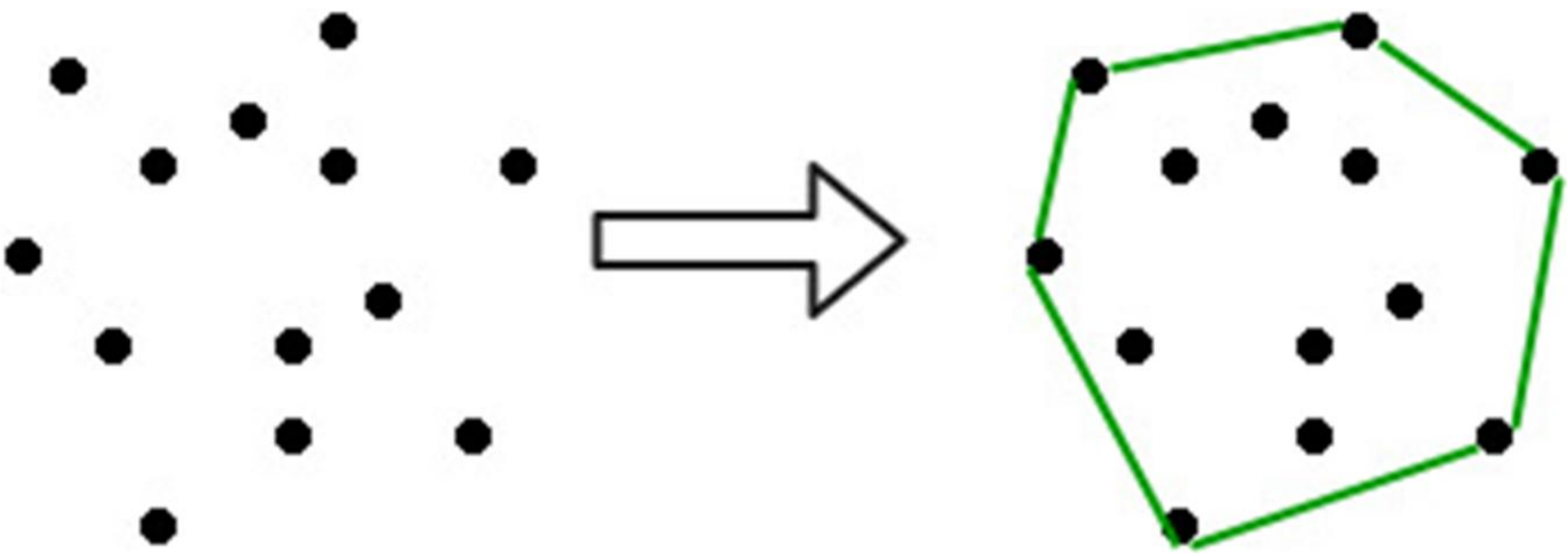


Quick Sort : A Divide and Conquer strategy

- Combine :
  - The two sub arrays are already sorted
  - The entire array A[p ..r] is already sorted
  - Nothing particular to do in *combine* step

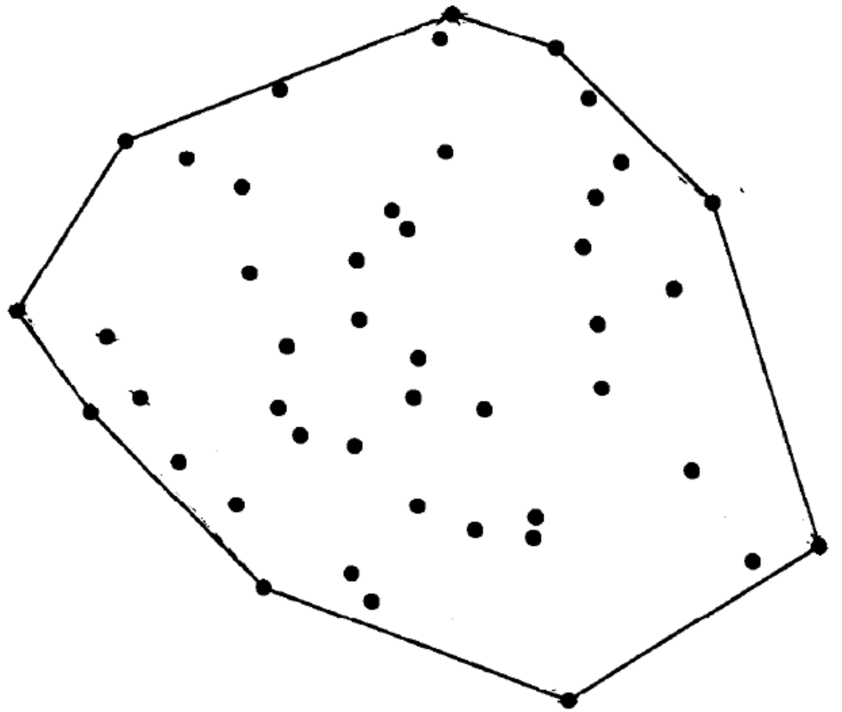


- What is the intuition of Quick Hull algo?



- Intuition of Quick Hull algo
- For most set of points, **it is easy to discard many points as definitely interior to the hull, and then concentrate on those closer to the hull boundary**
- Exercise: Draw an example input point set, where the above fact holds
- Exercise: Draw an example input point set, where the above fact does not hold

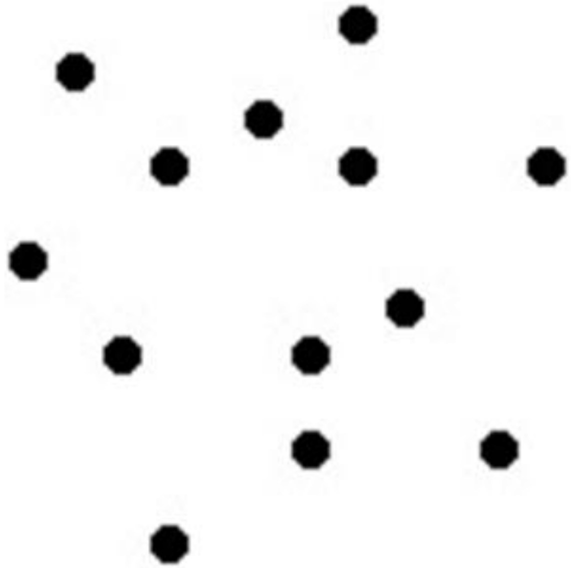
Example of a Convex hull



- Only the points on the CH /nearer to the boundary of CH have to be considered
- Some points are always part of the convex hull

Divide and Conquer Algo

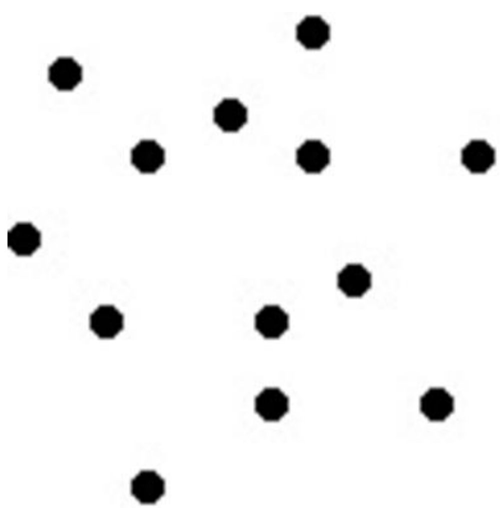
- How do we divide the input?



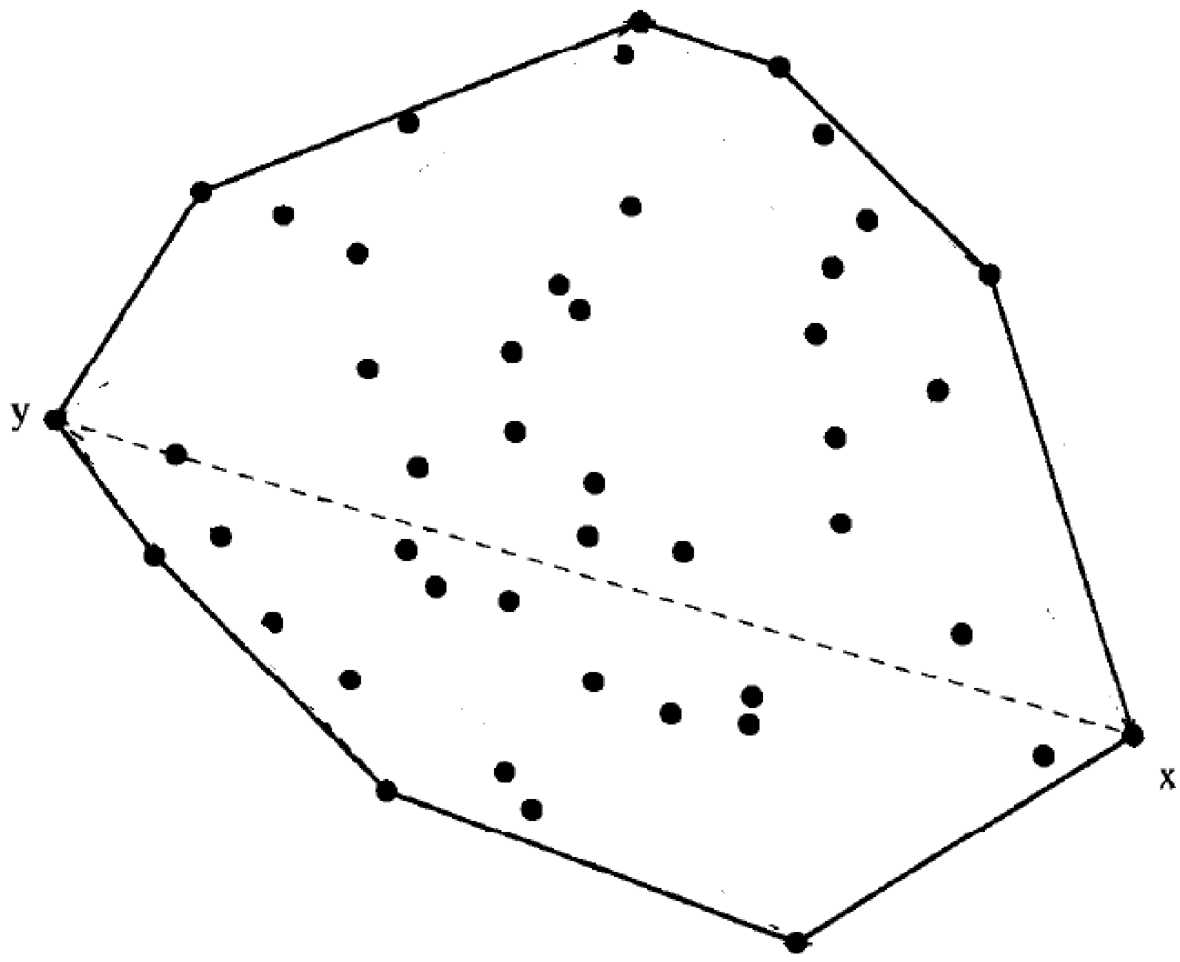
- Consider the intuition
- Recursively solve the sub problems

What are the points which are surely on the convex hull?

- Extreme points For Eg: Topmost, Lowest, Rightmost , Leftmost
- **For dividing the point set we consider the rightmost lowest and the left most highest point and draw a line L between them**

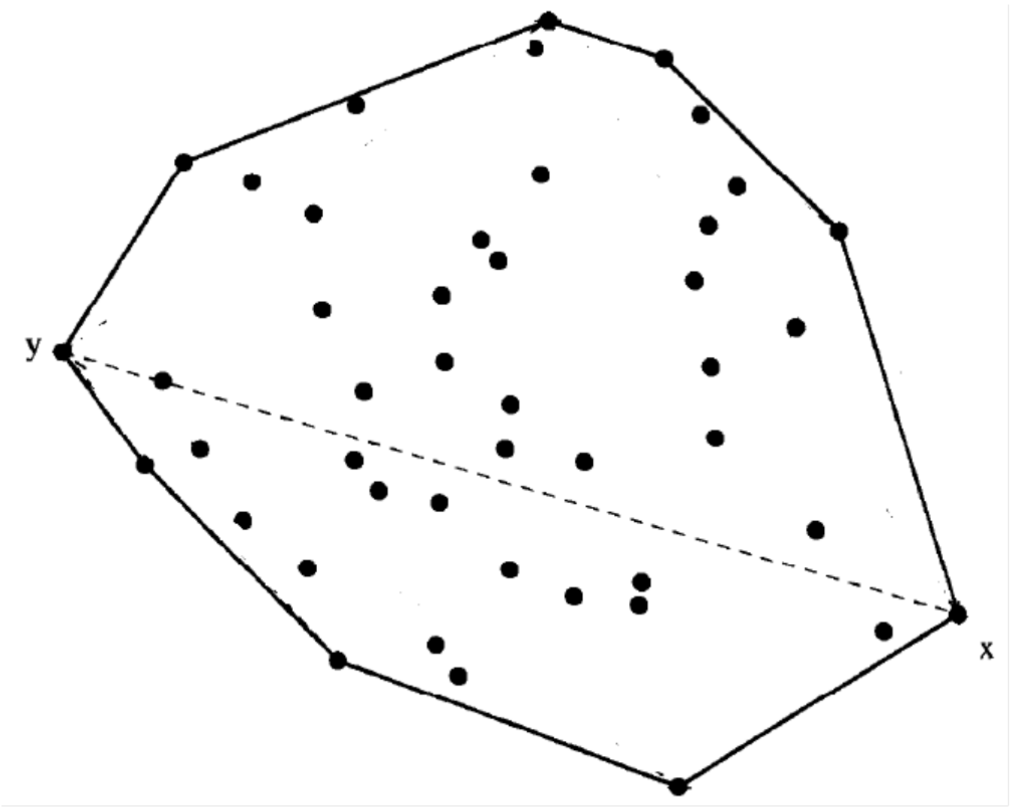


Divide the set of points in to two by Line xy



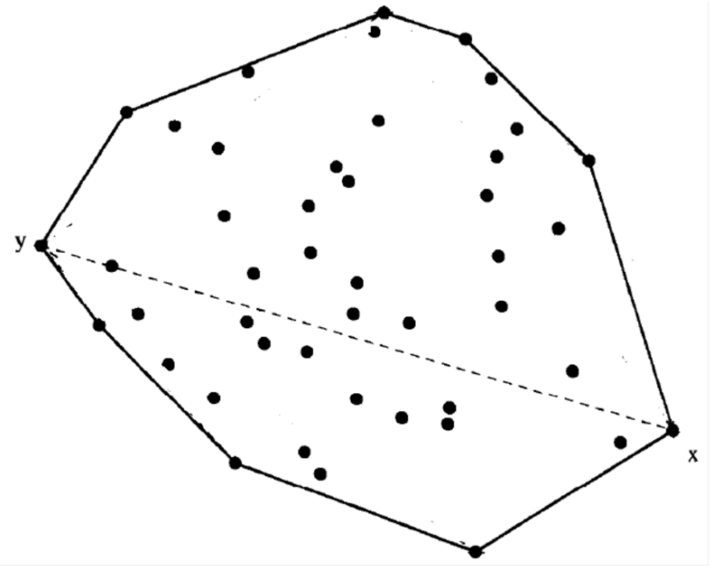
Divide and conquer

- The full hull is composed of an upper hull above the line xy and a lower hull below the line xy



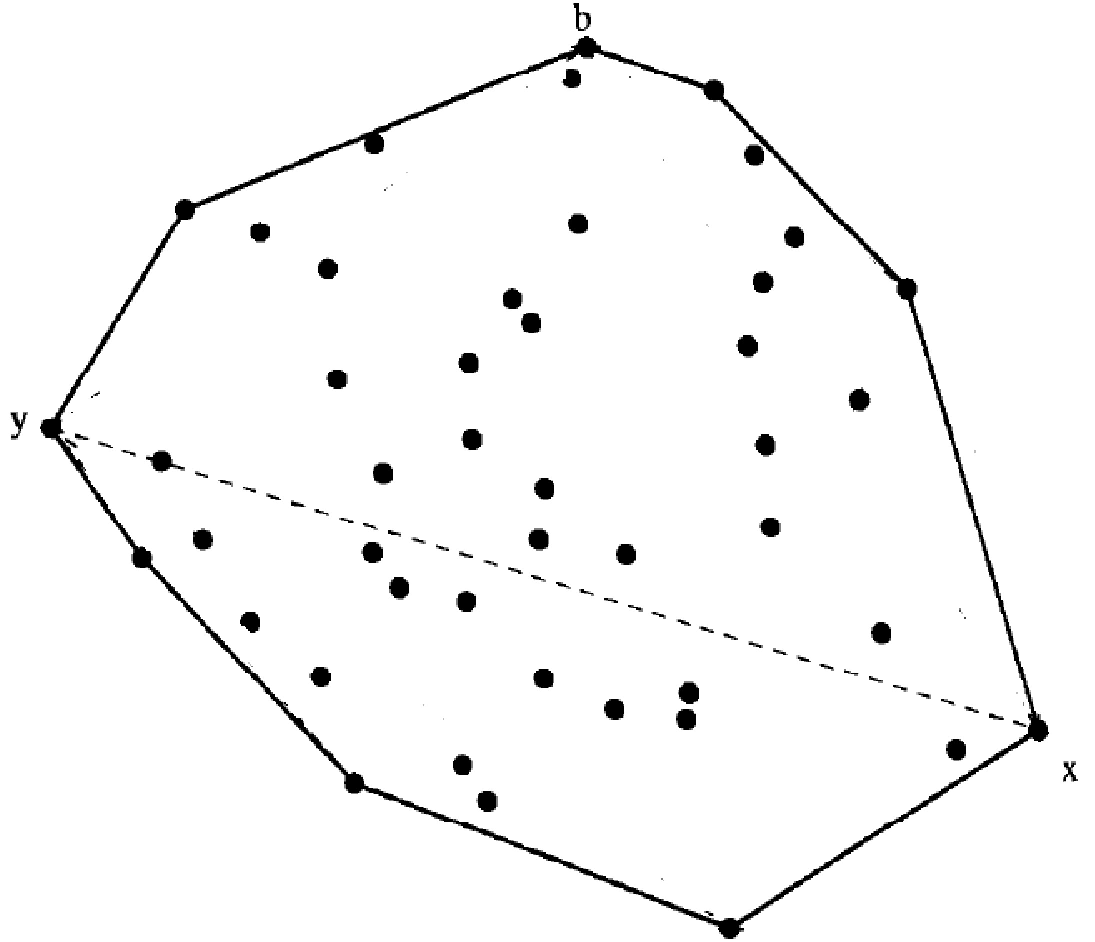
Recall Intuition

- For most set of points, it is easy to **discard many points** as definitely interior to the hull, and then concentrate on those closer to the hull boundary

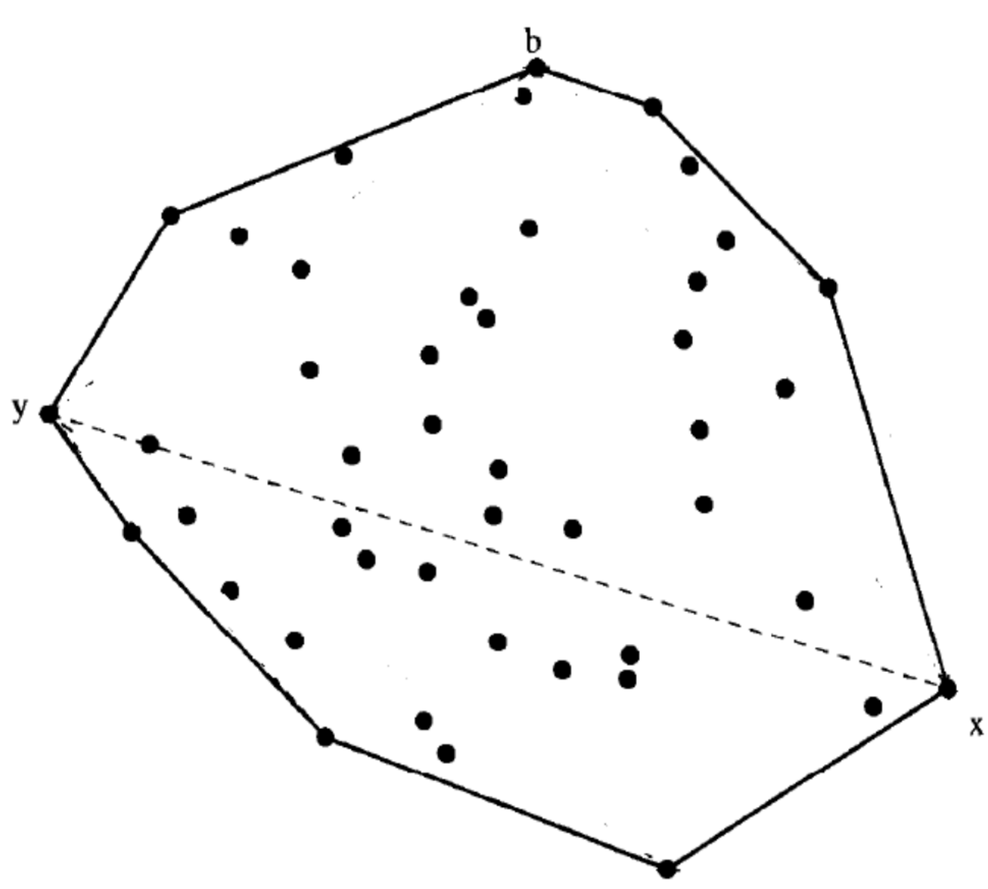


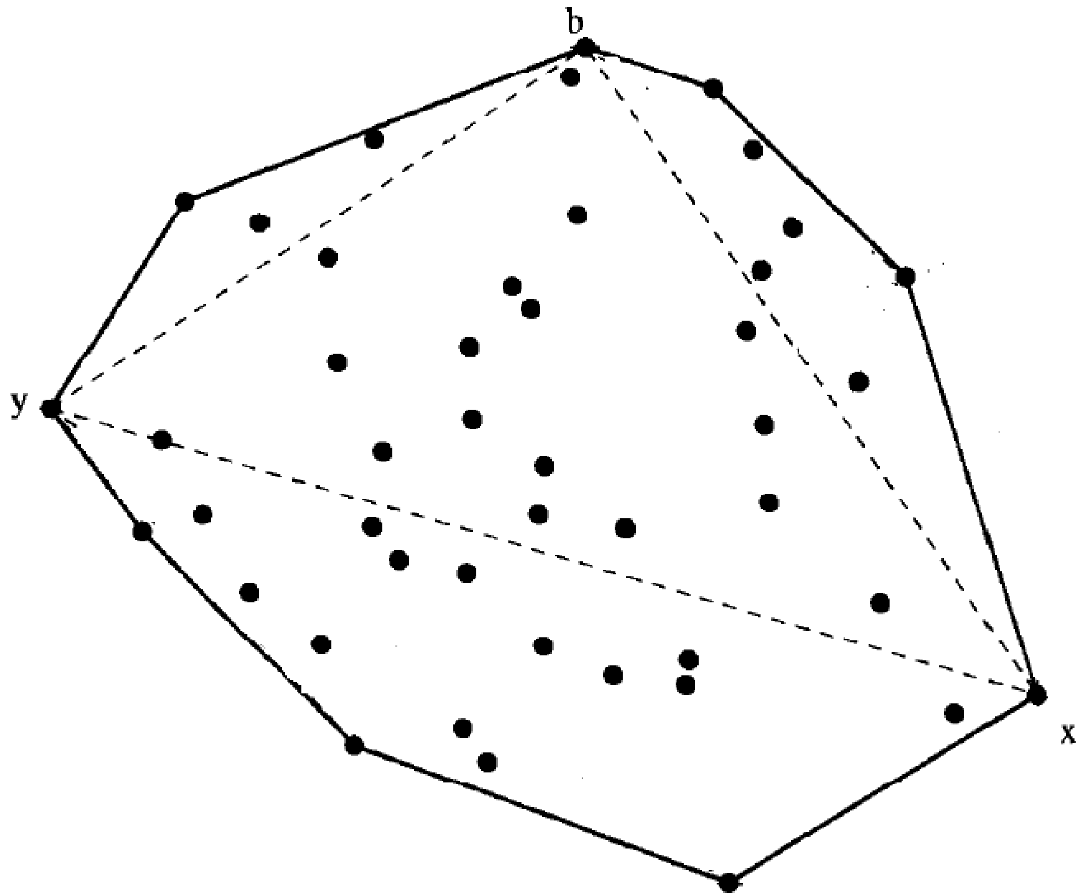
- How do we discard some interior points?

- Find a point b which is of maximum distance from the line xy



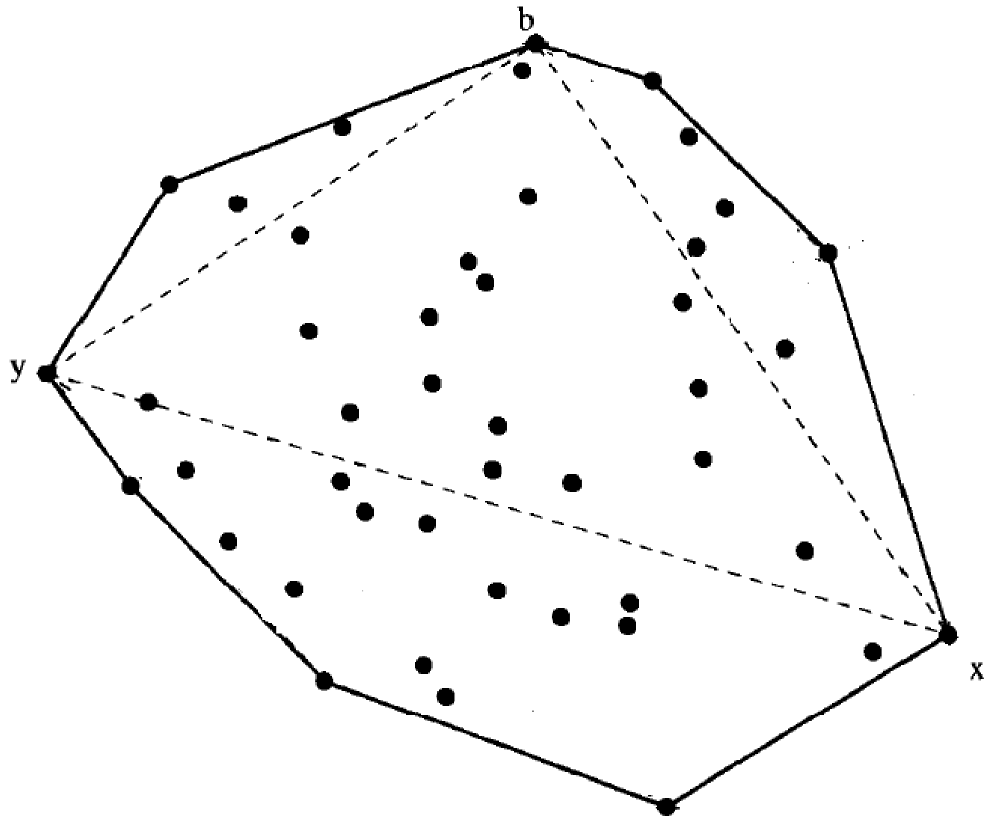
How do we discard some interior points using x, y and b?



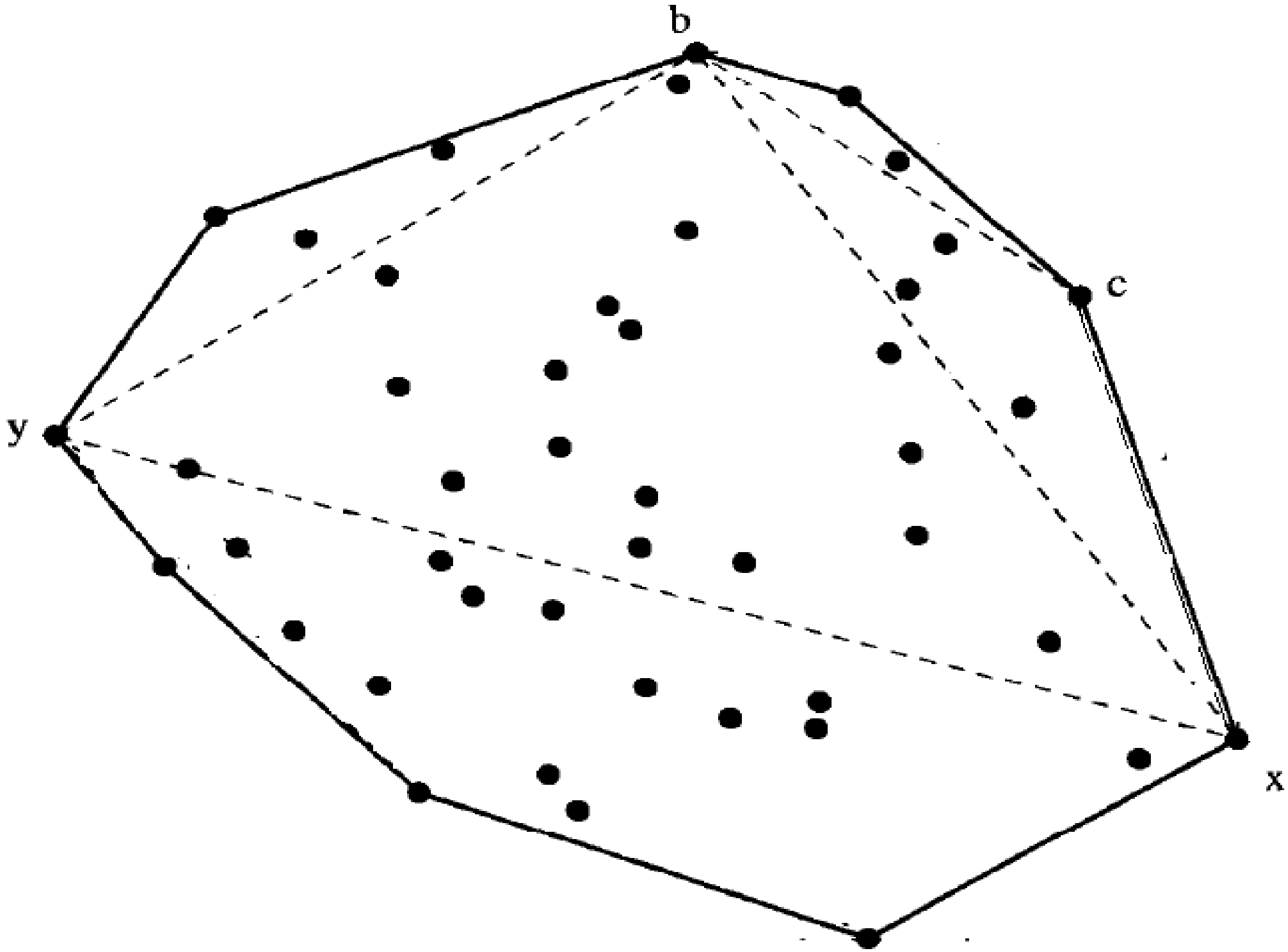


- The points interior to Δxby are not part of CH
- Recursively continue this process

Consider xb as the next line

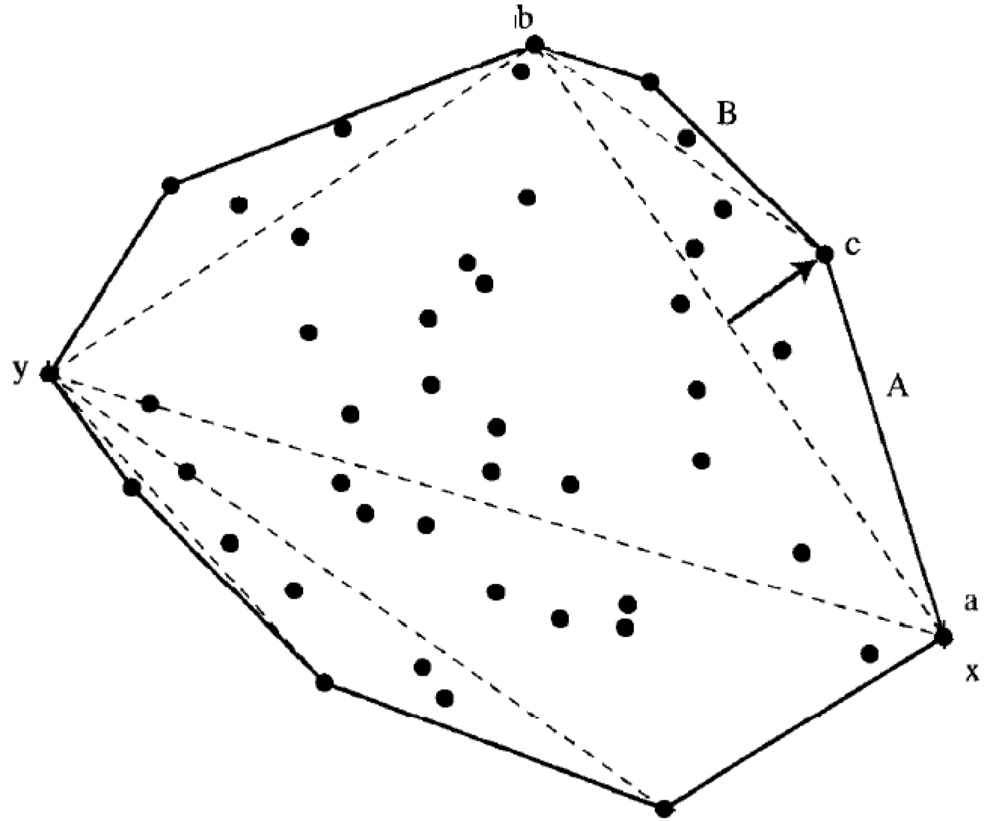


- Take the point with maximum distance from the line xb

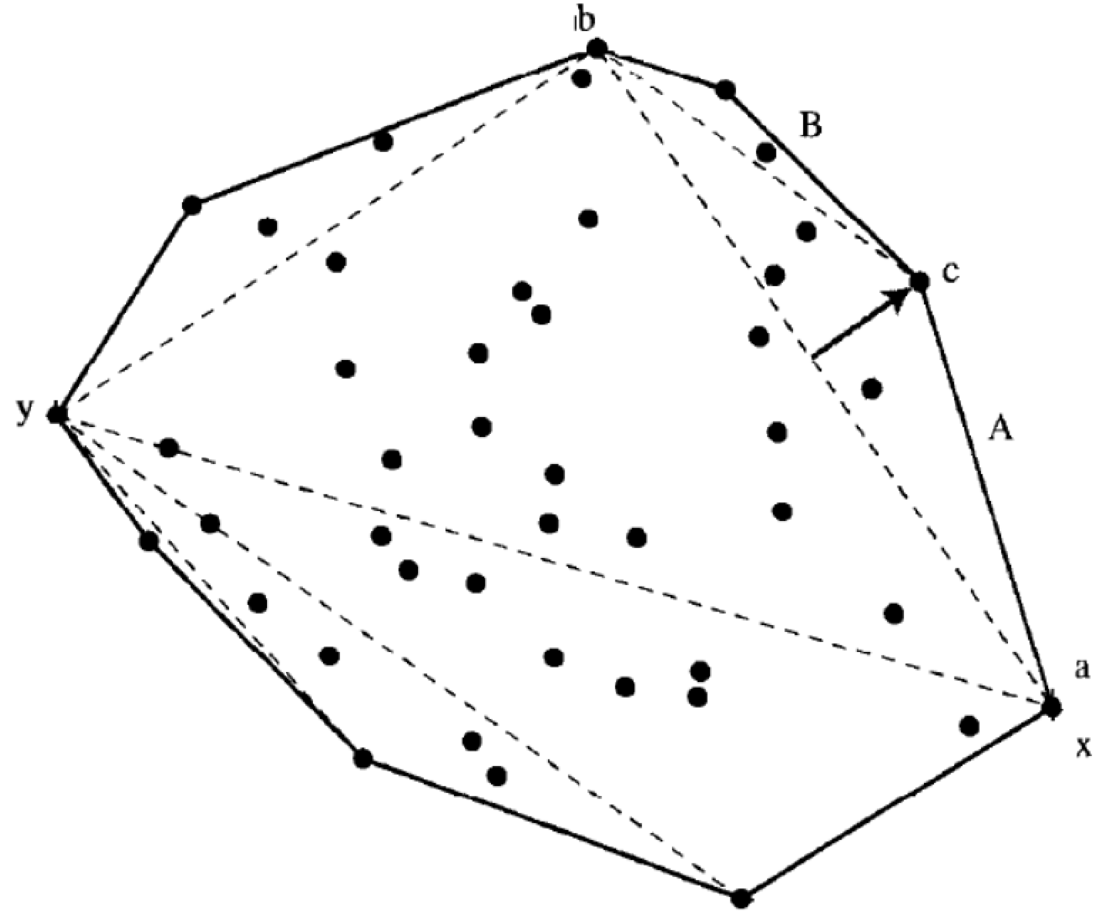


- Exclude the points from Δxcb

General pic



- Quick hull discards the points in Δabc and recurses on A and B



Algorithm: QUICKHULL

function $QuickHull(a, b, S)$

if $S = \emptyset$ then return ()

else

$c \leftarrow$ index of point with max distance from ab.

$A \leftarrow$ points strictly right of (a, c).

$B \leftarrow$ points strictly right of (c, b).

return $QuickHull(a, c, A) + (c) + QuickHull(c, b, B)$

Analysis

- Same as Quick Sort
- Best and average case : $O(n \log n)$
- Worst case : $O(n^2)$

References

- J. O Rourke, *Computational Geometry in C*, 2/e, Cambridge University Press, 1998 *)*

Thank you