Computational Geometry

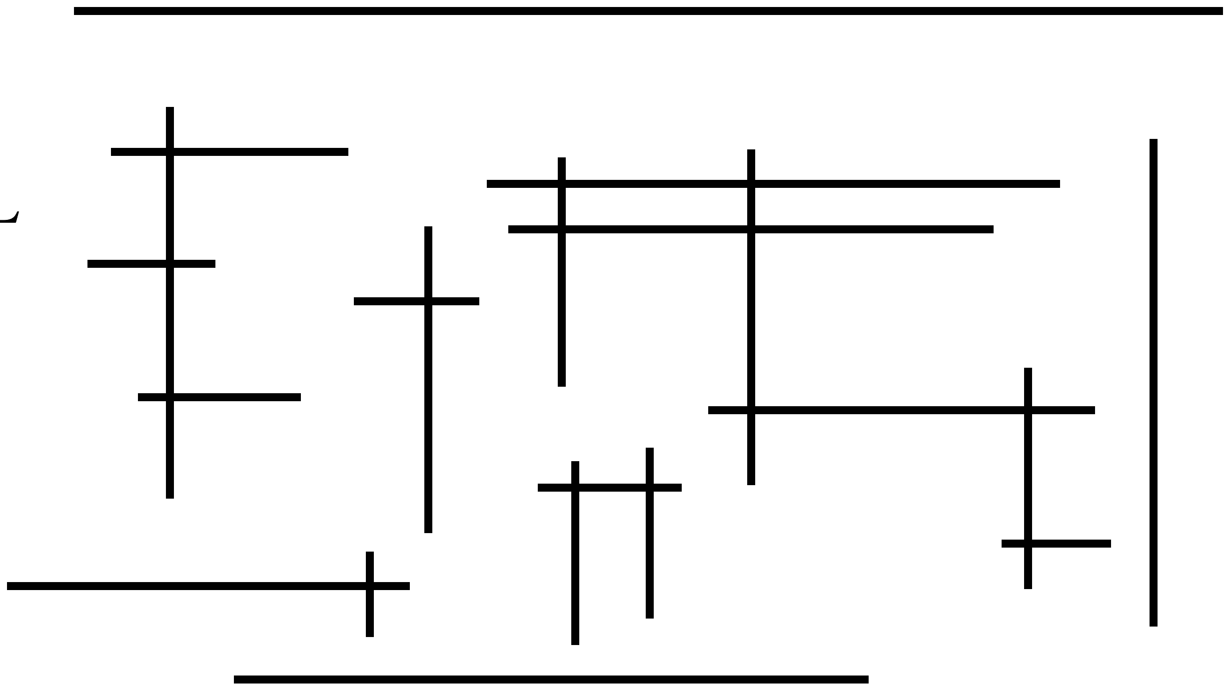
Dr. Gur Saran Adhar

Reference clrs, Chapter 33, Page 933-

Intersection of Horizontal and Vertical Line Segments

The Problem: Given a set of n horizontal and m vertical line segments in the plane, find all the intersections among them

L



Algorithm Intersection$((v_1, v_2, \ldots, v_m), (h_1, h_2, \ldots, h_n))$
Input: $(v_1, v_2, \ldots, v_m)$ (a set of vertical line segments and
$(h_1, h_2, \ldots, h_n)$ (a set of horizontal line segments)
Output: The set of all pairs of intersecting segments.
$\{y_B(v_i), y_T(v_i)\}$ denote the bottom and top of
the vertical segment v_i

`Reference Udi Manber page 286`

Algorithm Intersection(S, K)

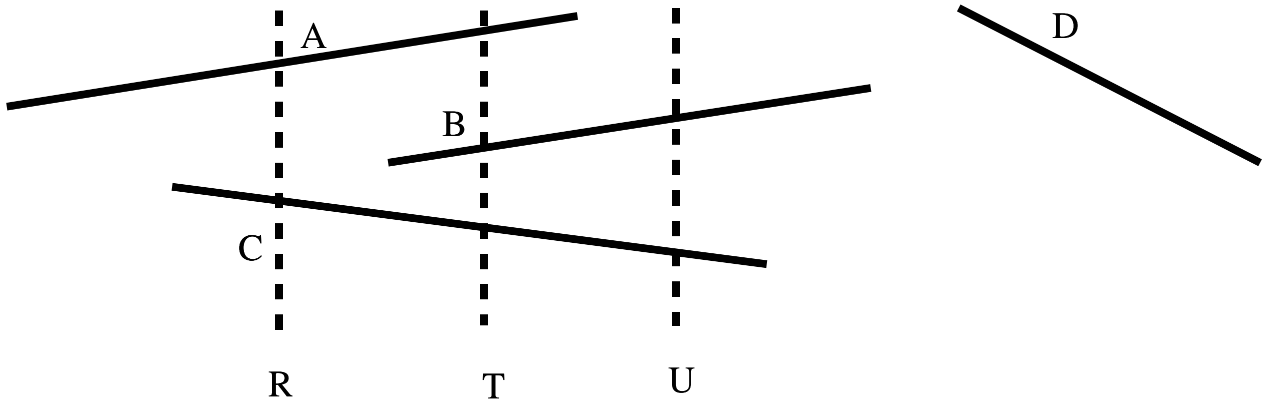
begin
  sort all x coordinates in increasing order
  and place them in Q
  $V := \emptyset$
  while Q is not empty do
    remove the first end point p from Q
    if p is the right endpoint of h_k **then**
      remove h_k from V.
      else if p is the left endpoint of h_k **then**
        insert h_k from V.
          else if p is the x coordinate of a
            vertical line v_i **then**
            perform a one-dimensional range
            query for the range $y_B(v_i)$ to $y_T(v_i)$
end.

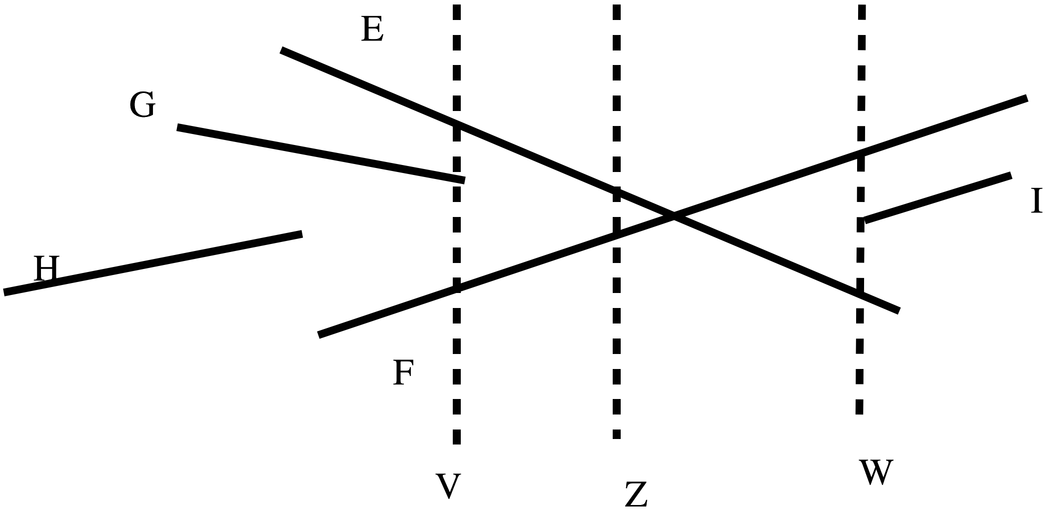
`Reference Udi Manber page 286`

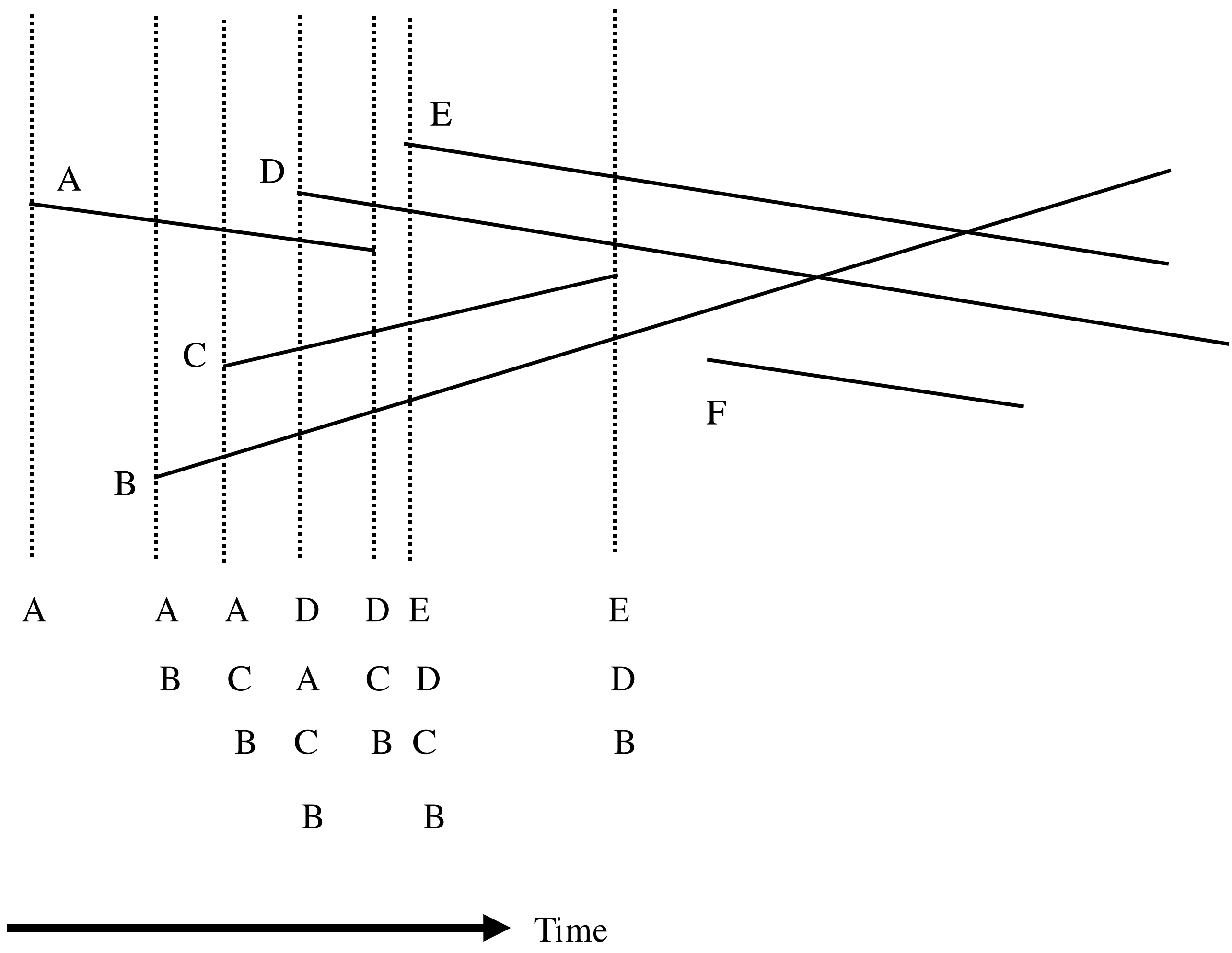
Intersection of Line Segments

The Problem: Given a set of n line segments in the plane, find all the intersections among them.

Assumption:
No input segment is vertical
No three input segments intersect at a single point


A
D
B
C
R
T
U


E
G
I
H
F
V
Z
W


E
D
A
C
F
B
A
A B
A C B
D A C B
D C B
E D C B
E D B
Time

Algorithm Any-Segments-Intersect(S)

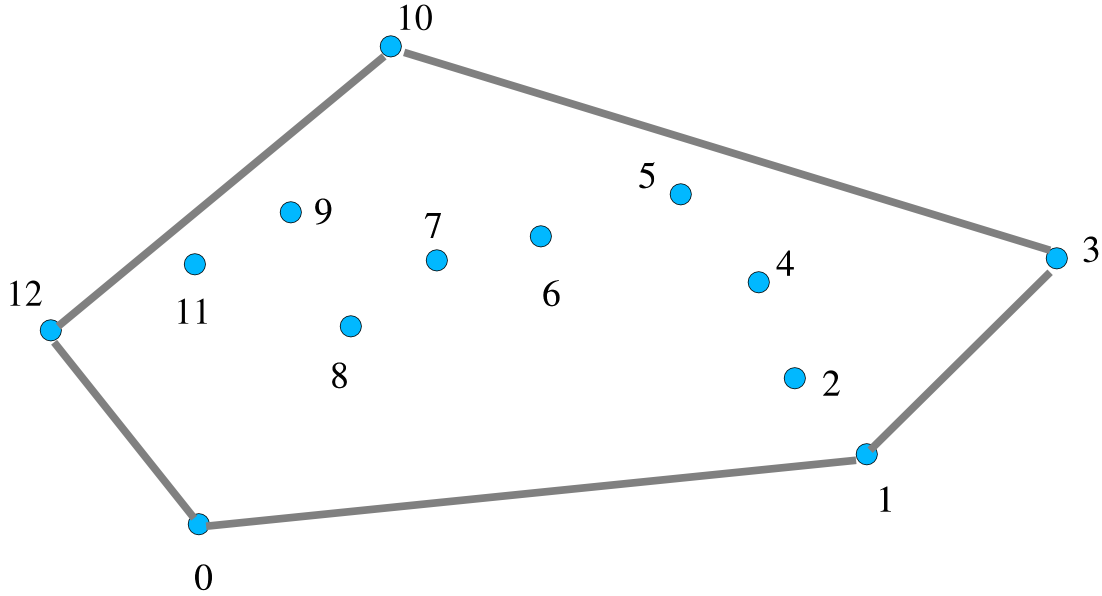
$T := \emptyset$
sort the endpoints of the segments in S from left to right breaking ties by putting points with lower y-coordinates first
for each point p in the sorted list of endpoints
  do if p is the left endpoint of a segment s
    then $INSERT(T, s)$
      if ($ABOVE(T, s)$ exists and intersects s)
      then return TRUE
  if p is the right endpoint of a segment s
    then if both $ABOVE(T, s)$ and $BELOW(T, s)$ exists
      and ($ABOVE(T, s)$ intersects $BELOW(T, s)$)
      then return TRUE
      $DELETE(T, s)$
return FALSE

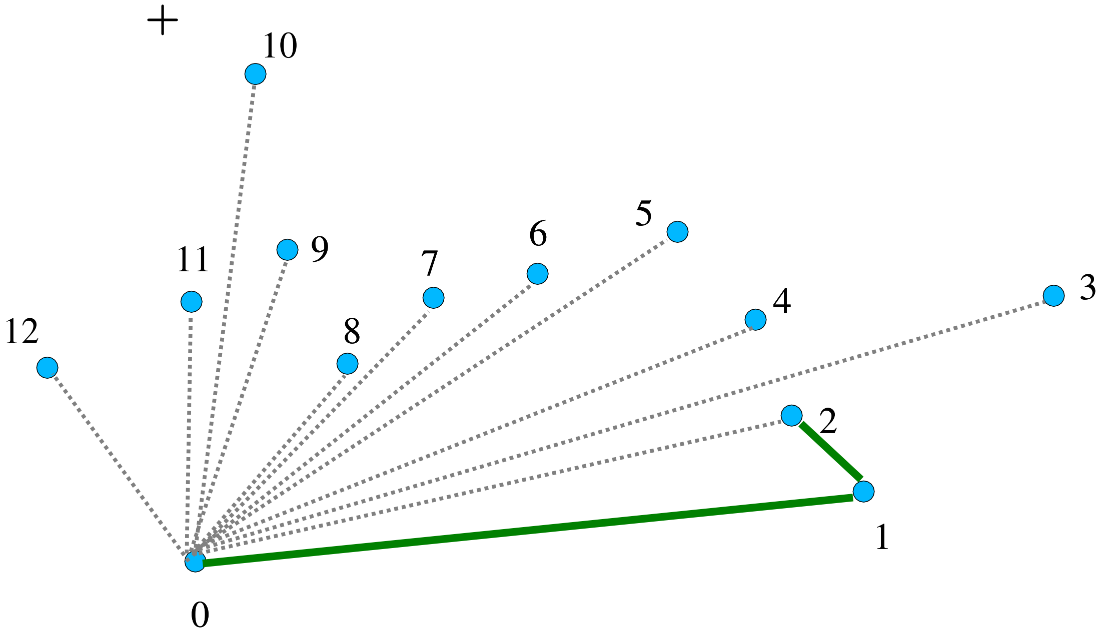
`Reference clrs2e page 943`

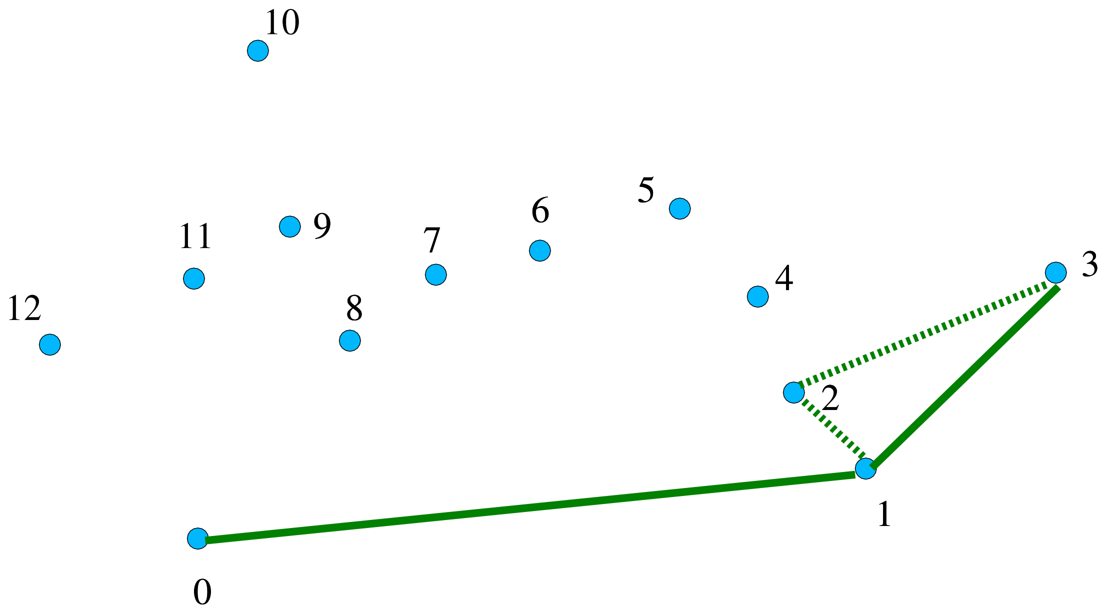
Convex Hull of Points

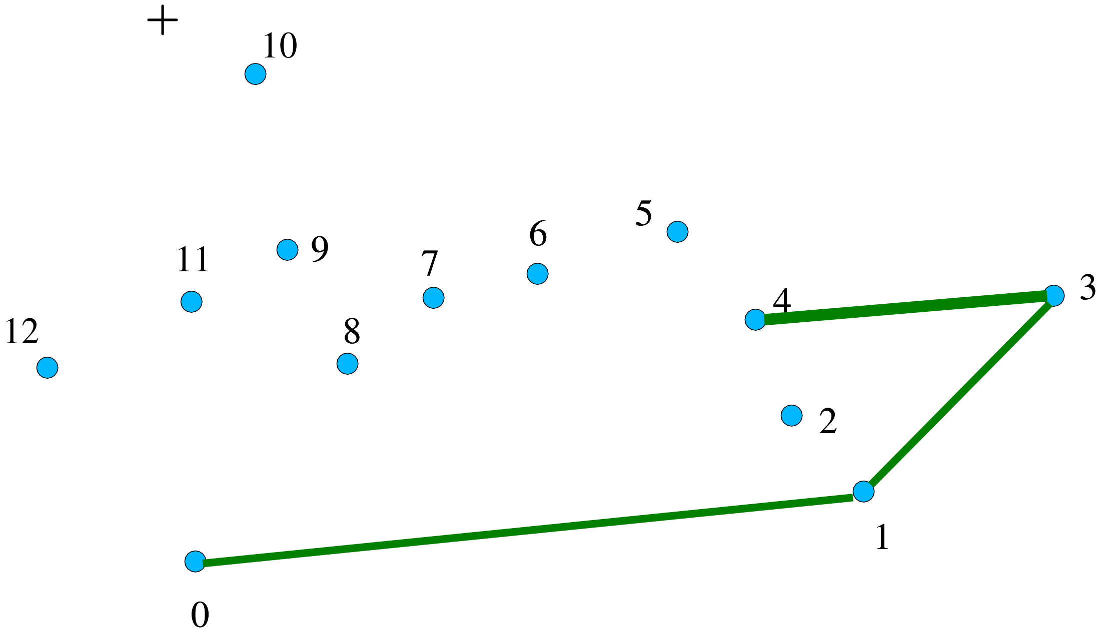
Problem: Given a set of n points in the plane, find a smallest convex polygon for which either each point is on the boundary or inside the polygon.

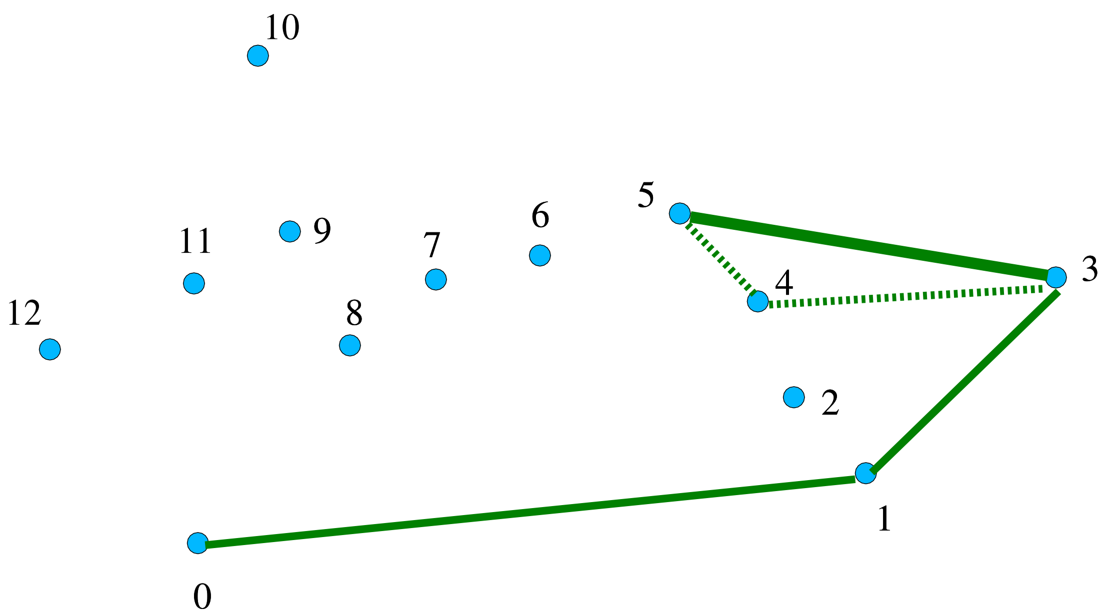
`Reference clrs page947-`

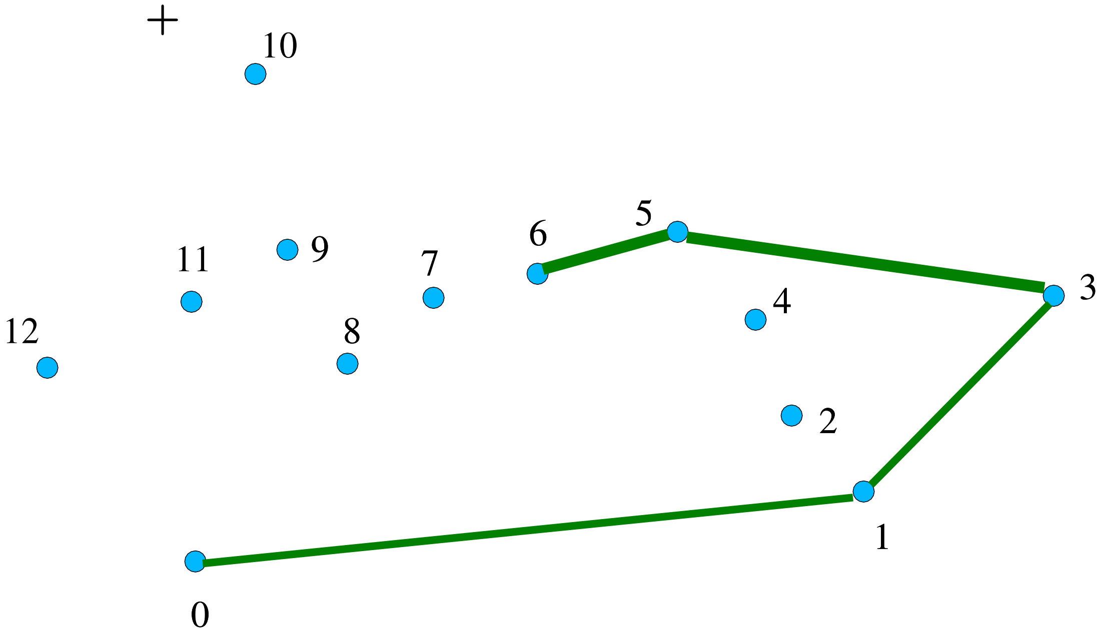

10
5
9
7
3
4
12
6
11
2
8
1
0


10
11
9
7
6
5
4
3
12
8
2
1
0

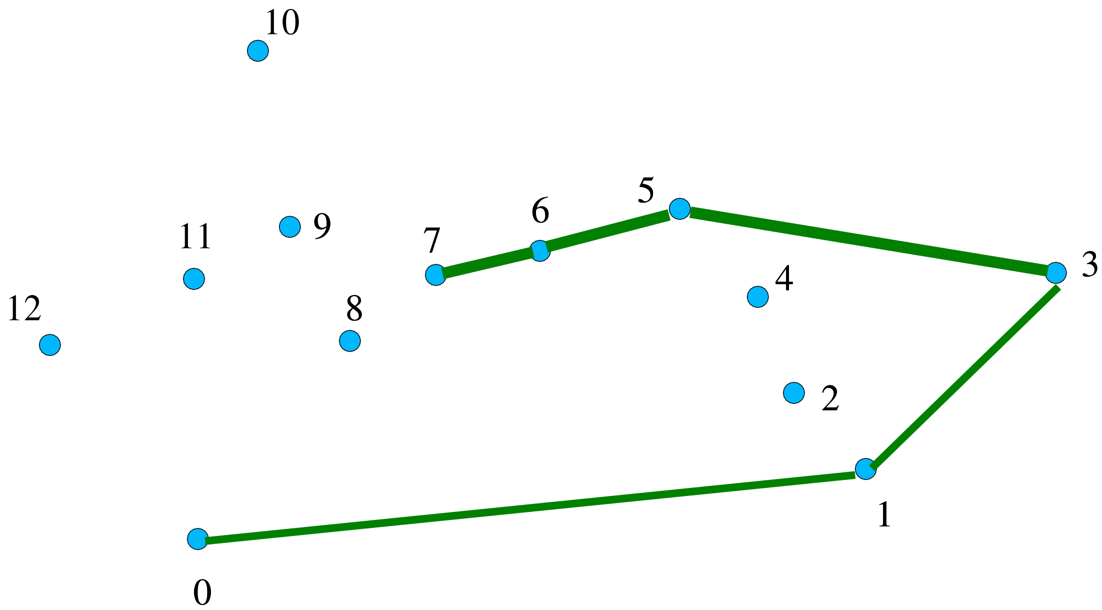

10
5
6
11
9
7
3
4
12
8
2
1
0


0
1
2
3
4
5
6
7
8
9
10
11
12

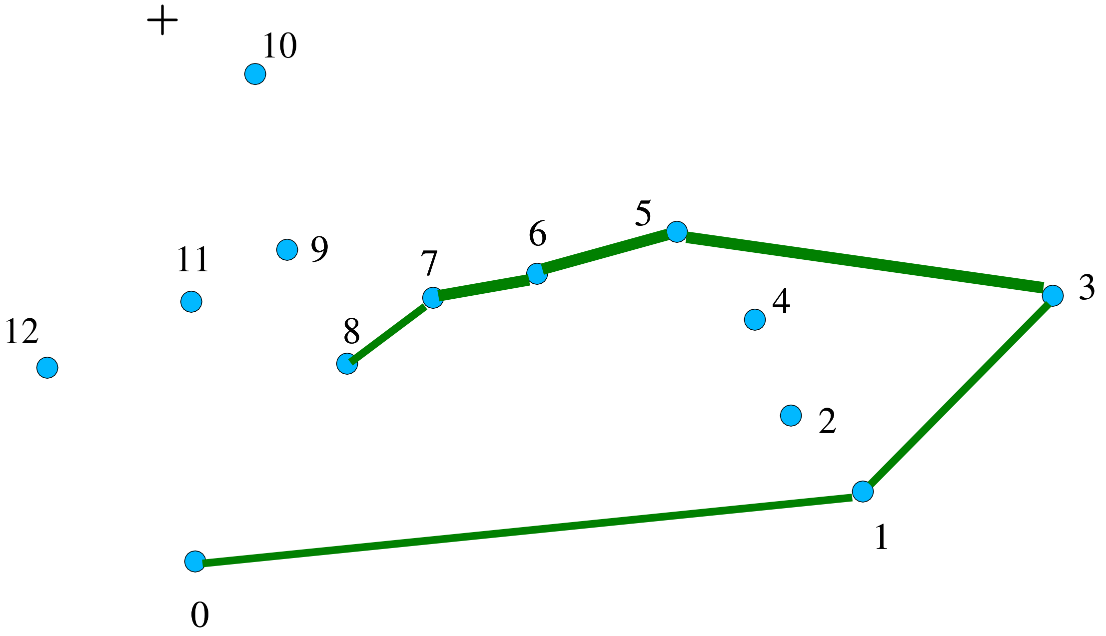

0
1
2
3
4
5
6
7
8
9
10
11
12

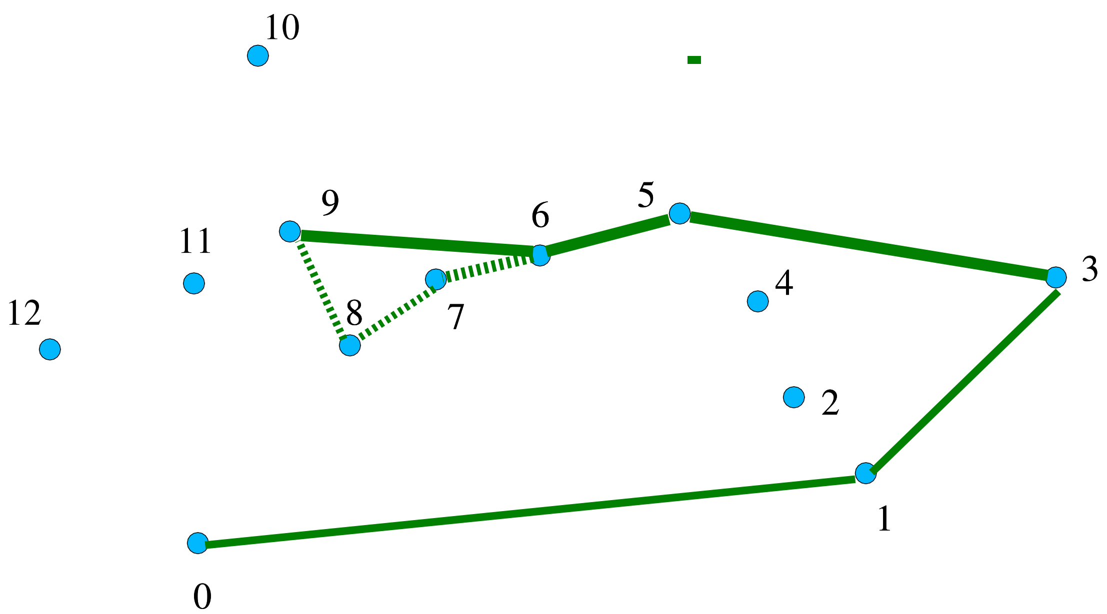


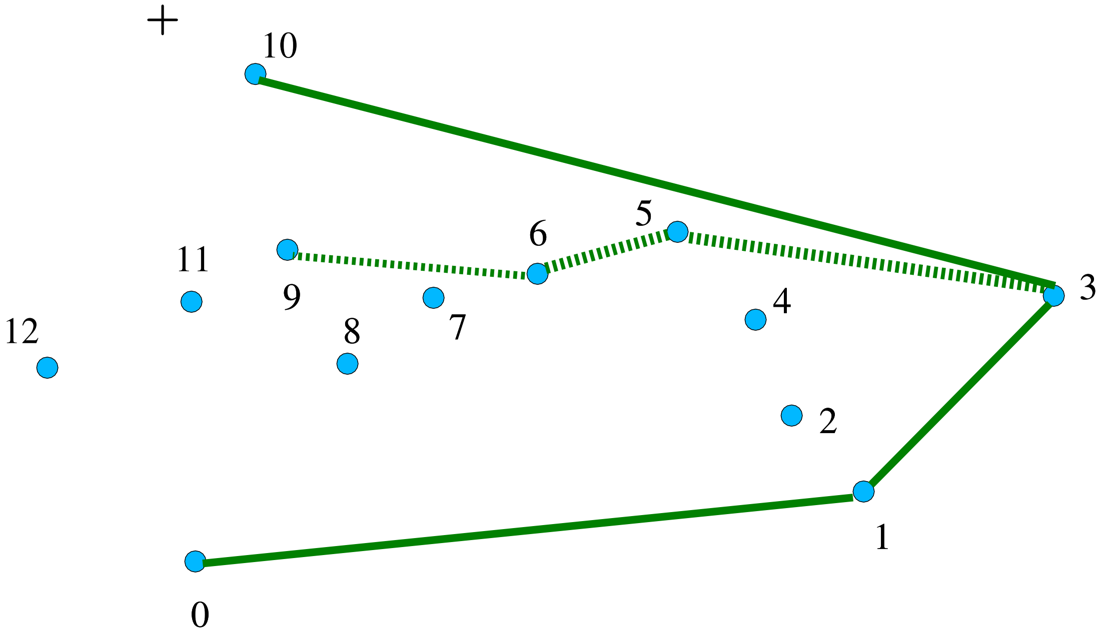
10
11
9
7
6
5
3
12
8
4
2
1
0

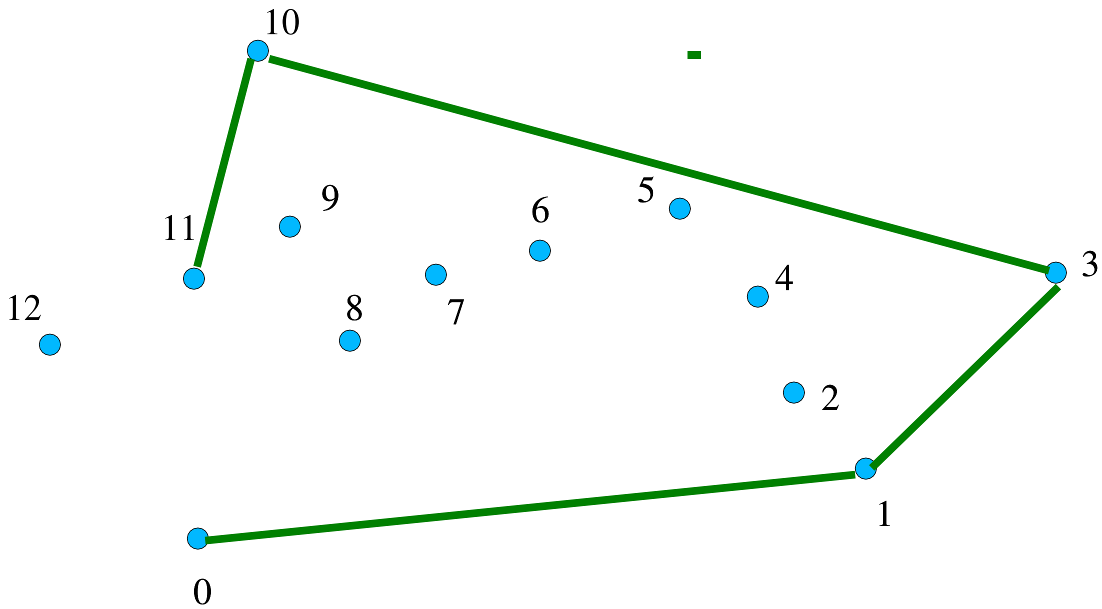


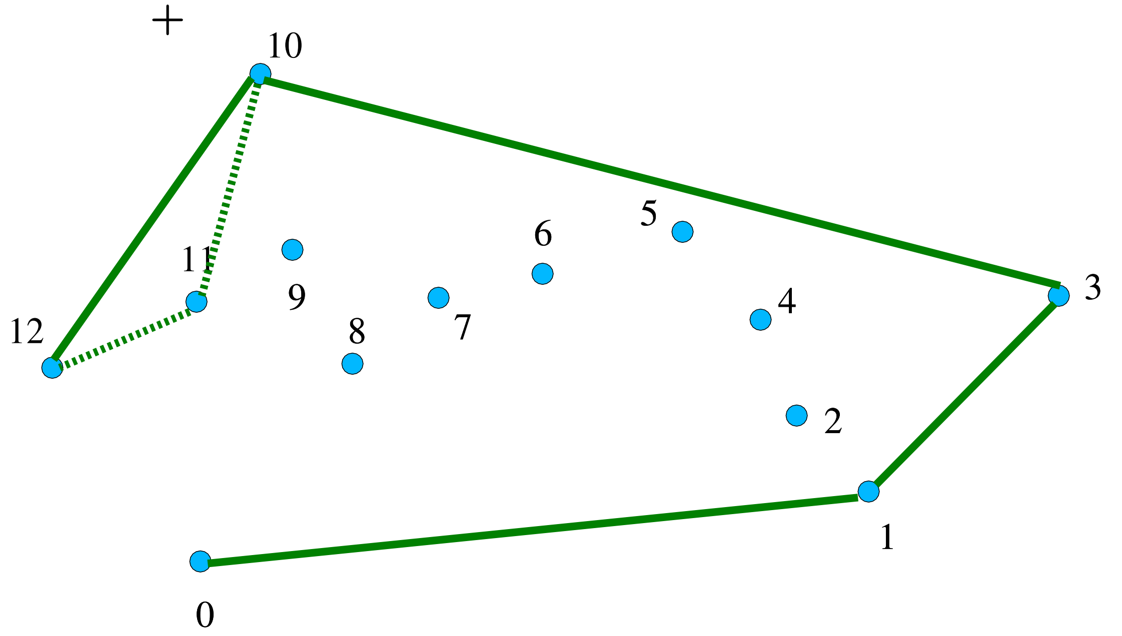
10
11
9
7
6
5
3
12
8
4
2
1
0

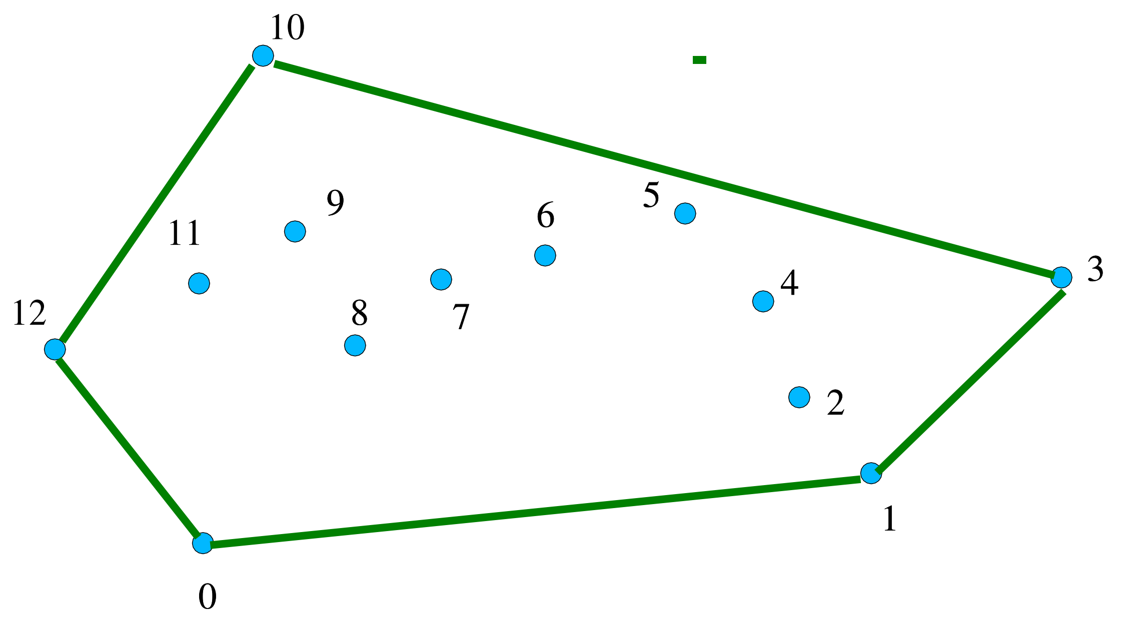

10
5
6
7
11
9
3
12
8
4
2
1
0


10
9
6
5
11
3
4
12
8
7
2
1
0


10
11
9
6
5
12
8
7
4
3
2
1
0


10
11
9
6
5
12
8
7
4
3
2
1
0


10
11
9
6
5
7
8
4
3
12
2
1
0


10
9
11
6
5
8
7
4
3
12
2
1
0

Algorithm Graham-Scan(Q)

Let p_0 be the point with minimum y-coordinate

or the leftmost such point in case of a tie

Let $< p_1, p_2, \ldots, p_m >$ be the remaining points in Q

sorted by polar angles in counterclockwise order

around p_0

$PUSH(p_0, S)$

$PUSH(p_1, S)$

$PUSH(p_2, S)$

for $i \leftarrow 3$ to m

  do while the angle formed by points

  NEXT-TO-TOP(S), TOP(S), and p_i

  makes a non-left (right turn)

    do $POP(S)$

      then return TRUE

  $PUSH(p_i, S)$

return S

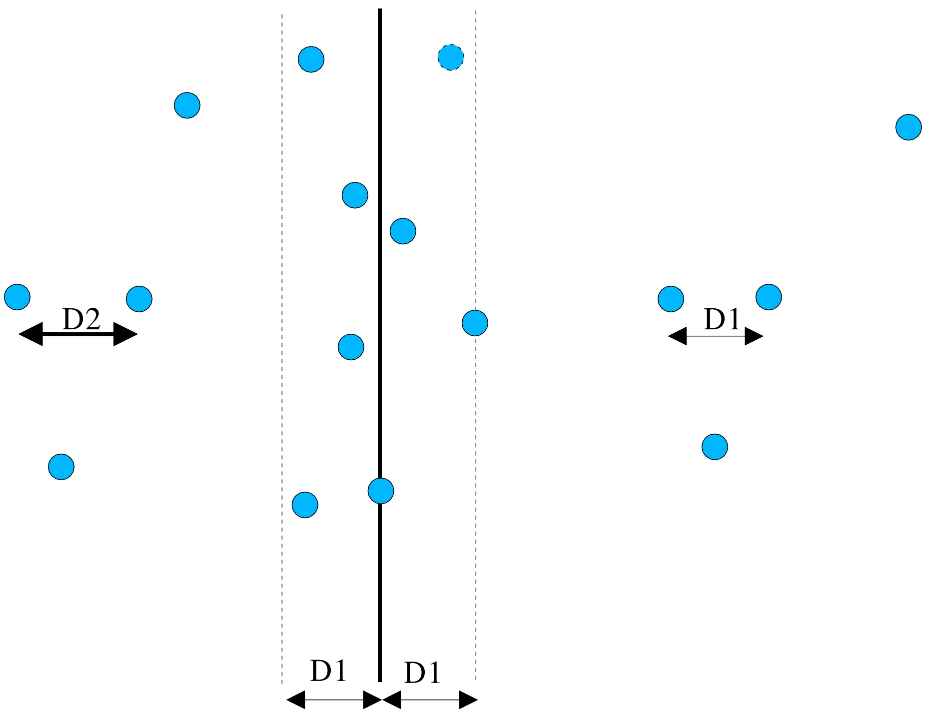
`Reference clrs2e page 943`

Approach: Divide and Conquer- The Closest Pair of Points

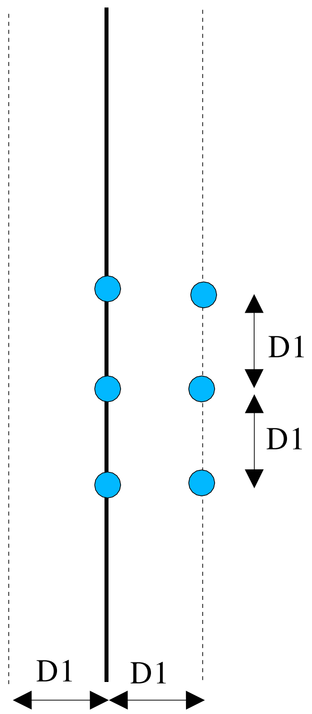
Problem: Given a set of n points in the plane, find a pair of closest points

`Reference clrs page957-,`

`Udi Manber page 279`



Closest Pair Problem



The worst case of six points d1 apart

Algorithm Closest_Pair$(p_1, p_2, \ldots, p_n)$
Input: $p_1, p_2, \ldots, p_n$ a set of n points in the plane
Output: d (the distance between the two closest points)

begin
    Sortpoints according to their x-coordinates;
    {comment-this sorting is done only once }
    divide the set into two equal-sized parts;
    Recursively, compute the minimal distance
        in each part;
    Let d be the minimal of the two minimal distances;
    Eliminate points that lie farther than d apart
        from the separation line
    Sort the remaining points according to
        their y coordinates;
    Scan the remaining points in the y order and find
        the distance of each point to its five neighbors;
    if any of these distances is less than d
        then update d
end.

Reference Udi Manber page 280