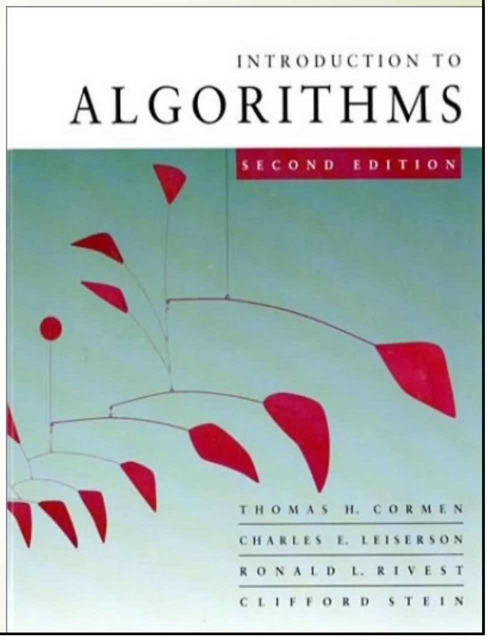
Convex Hull



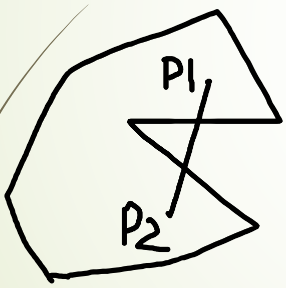
SUBJECT-CODE : KCS-503

Dr. Ragini Karwayun

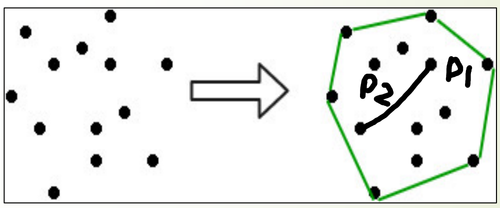
1

Convex Hull

- The convex hull of a set of S points in the plane is defined to be the smallest convex polygon containing all the points of S.
- A polygon is said to be convex if for any two points P1 & P2 inside the polygon, the directed segment from P1 to P2 <P1,P2> is fully contained in the polygon.



Not convex



convex

2

Convex Hull

- The vertices of the convex hull of a set S of points form a (not necessarily proper) subset of S. Given a finite set of points {P1,P2,…,Pn} the convex hull of P is the smallest convex set C such that $P \subset C$.
- There are two variants of the convex hull problem :
  - Obtain the vertices of the convex hull (also referred to as extreme points)
  - Obtain the vertices of the convex hull in some order.
- Convex Hull of Q is denoted by CH(Q).
- We shall discuss two algorithms that compute the convex hull of a set of n points.
  - GRAHAM-SCAN
  - DIVIDE-AND-CONQUER METHOD
- Both algorithms runs in O(n lg n) time.

Graham's Scan

- ***Graham's scan*** solves the convex-hull problem by maintaining a stack S of candidate points.
- Each point of the input set Q is pushed once onto the stack, and the points that are not vertices of CH(Q) are eventually popped from the stack.
- When the algorithm terminates, stack S contains exactly the vertices of CH(Q), in counter-clockwise order of their appearance on the boundary.
- The procedure GRAHAM-SCAN takes as input a set Q of points, where $|Q| \geq 3$.
- It calls the functions TOP(S), which returns the point on top of stack S without changing S, and NEXT-TO-TOP(S), which returns the point one entry below the top of stack S without changing S.
- The stack S returned by GRAHAM-SCAN contains, from bottom to top, exactly the vertices of CH(Q) in counter-clockwise order.

Graham's Scan

GRAHAM-SCAN(*Q*)

1. let P0 be the point in Q with the minimum y-coordinate, or the leftmost such point in case of a tie
2. let ⟨P1, P2,..., Pm⟩ be the remaining points in Q, sorted by polar angle in counter-clockwise order around P0 (if more than one point has the same angle, remove all but the one that is farthest from P0)
3. PUSH(P0,S)
4. PUSH(P1,S)
5. PUSH(P2,S)
6. for i←3 to m
7. do while the angle formed by points NEXT-TO-TOP(S), TOP(S), and Pi makes a non-left turn
8. do POP(S)
   PUSH(Pi , S)
9. return S

$$\text{If } (x1-x0)(y2-y0) - (x2-x0)(y1-y0) \begin{cases} < 0 & \text{Left turn} \\ = 0 & \text{Straight} \\ > 0 & \text{Right turn} \end{cases}$$

IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

5

Graham's Scan

S=P0,P1,P2

P1P2P3 – right turn

i=3
POP P2
PUSH P3
S=P0,P1,P3

P1P3P4 – left turn

i=4
PUSH P4
S=P0,P1,P3,P4

P3P4P5 – right turn

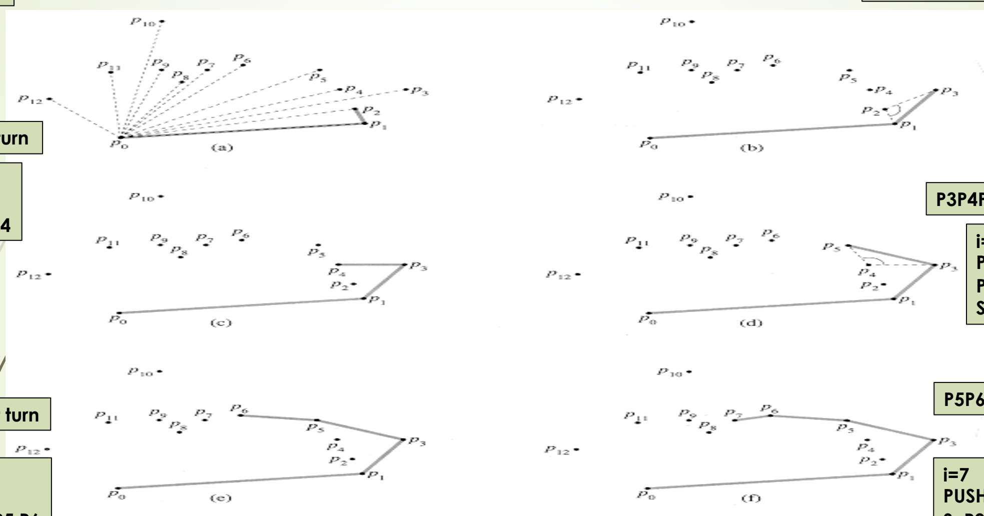
i=5
POP P4
PUSH P5
S=P0,P1,P3,P5

P3P5P6 – left turn

i=6
PUSH P6
S=P0,P1,P3,P5,P6

P5P6P7 – left turn

i=7
PUSH P7
S=P0,P1,P3,P5,P6,P7



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

6

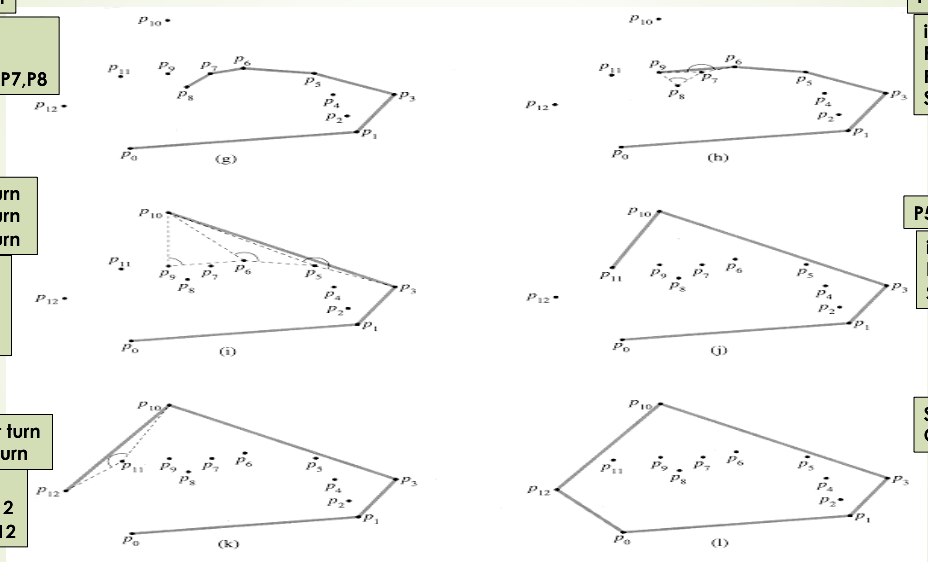
Graham's Scan

P6P7P8 – left turn

i=8
PUSH P8
S=P0,P1,P3,P5,P6,P7,P8

P7P8P9 – right turn
P6P7P9 – right turn

i=9
POP P8 ,POP P7
PUSH P9
S=P0,P1,P3,P5,P6,P9



P6P9P10 – right turn
P5P6P10 – right turn
P3P5P10 – right turn

i=10
POP P9 ,POP P6
POP P5,PUSH P10
S=P0,P1,P3,P10

P5P10P11 – left turn

i=11
PUSH P11
S=P0,P1,P3,P10,P11

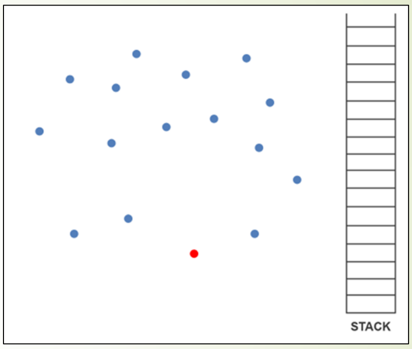
P10P11P12 – right turn
P3P11P12 – left turn

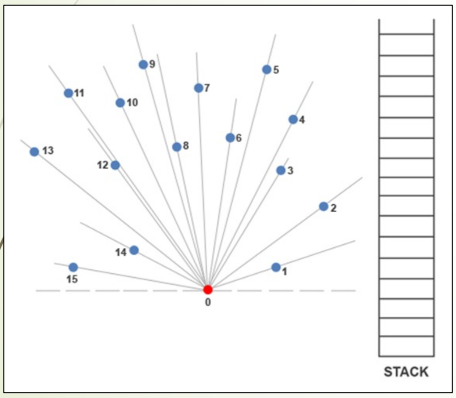
i=12
POP P11, PUSH P12
S=P0,P1,P3,P10,P12

S=P0,P1,P3,P10,P12
Convex Hull

Graham's Scan

We have a set of points. Select the point that has the lowest y-coordinate value.

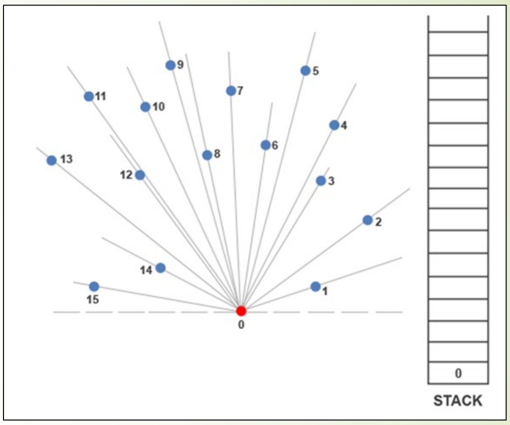


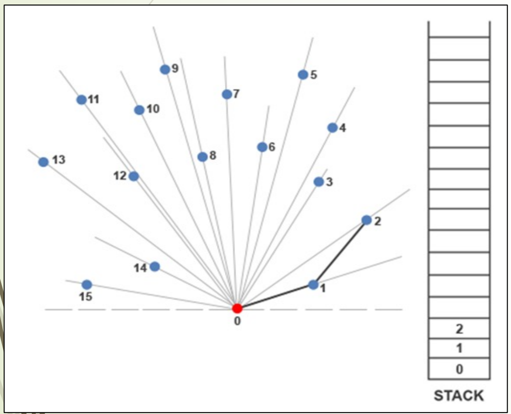


The points are ordered in increasing angle with respect to point 0

Graham's Scan

Now we can follow Graham's scan to find out which points create the convex hull. Point 0 is pushed onto the stack.





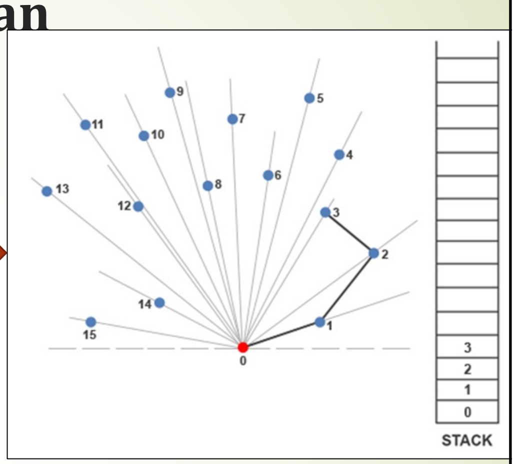
Point 1 and 2 are pushed onto the stack immediately after. A line is made from point 1 to point 2.

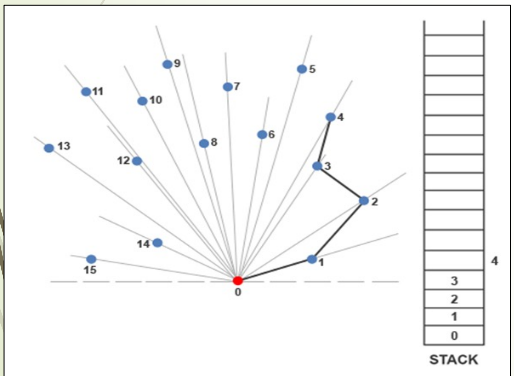
IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

9

Graham's Scan

Whenever a left turn is made, the point is presumed to be part of the convex hull. We can clearly see a left turn being made to reach 2 from point 1. To get to point 3, another left turn is made. Currently, point 3 is part of the convex hull. A line segment is drawn from point 2 to 3 and 3 is pushed onto the stack.





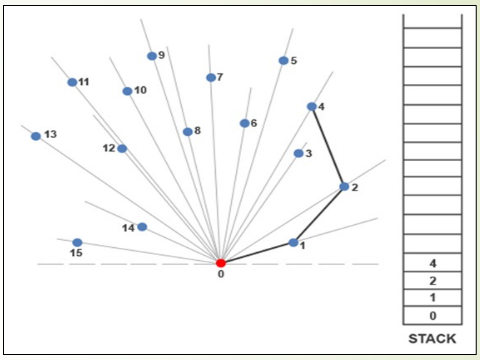
We make a right turn going to point 4. We'll draw the line to point 4 but will not push it onto the stack.

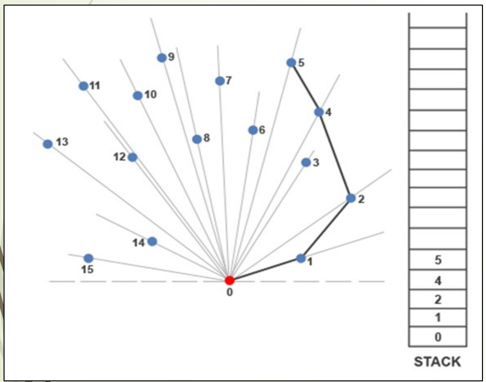
IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

10

Graham's Scan

Whenever a right turn is made, Graham's scan algorithm pops the previous value from the stack and compares the new value with the top of the stack again. In this case, we'll pop 3 from the top of the stack and we'll see if going from point 2 to point 4 creates a left bend. In this case it does, so we'll draw a line segment from 2 to 4 and push 4 onto the stack.



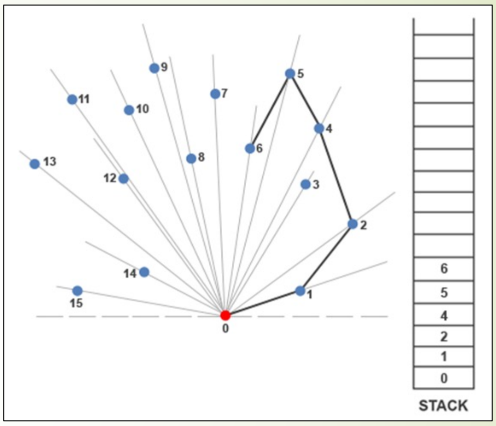


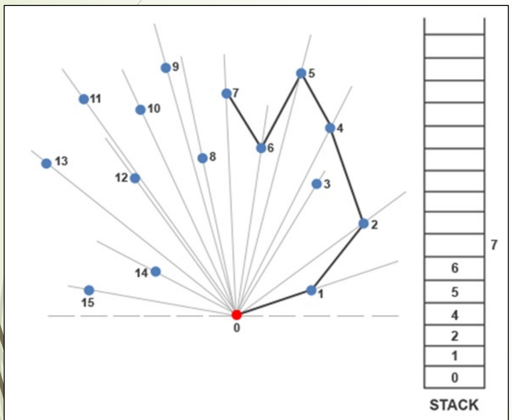
Since going from 4 to 5 creates a left turn, we'll push 5 onto the stack. Point 5 is currently part of the convex hull.

11

Graham's Scan

Moving from point 5 to 6 creates a left-hand turn, so we'll push 6 onto the stack. Point 6 is currently part of the convex hull.



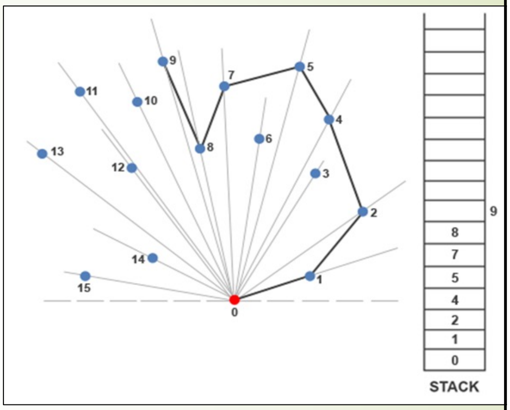


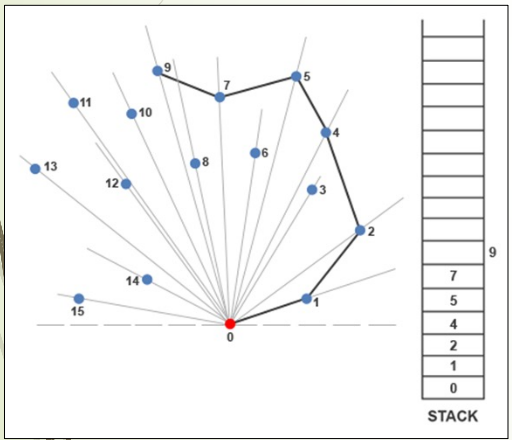
To get to point 7, we must make a right-hand turn at 6

12

Graham's Scan

Going to point 9 requires a right-hand turn at point 8.



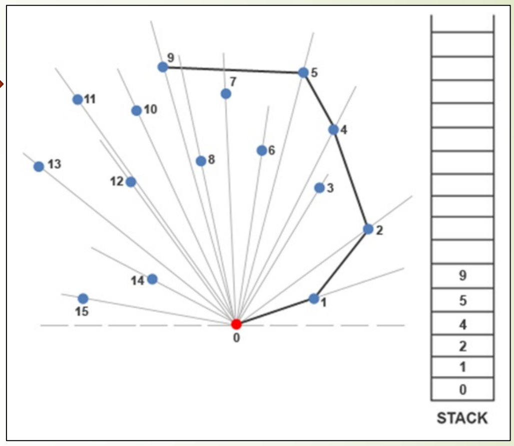


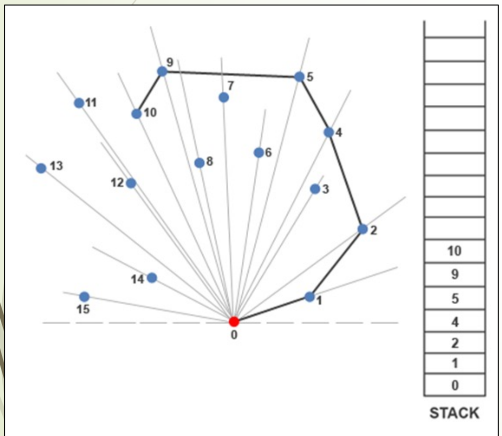
Since there's a right-hand turn, point 8 is popped from the stack and point 9 is compared with point 7.

13

Graham's Scan

To get to point 9 from point 7 requires another right-turn, so we pop point 7 from the stack too and compare point 9 to point 5. We make a left-hand turn at point 5 to get to point 9, so 9 is pushed onto the stack.



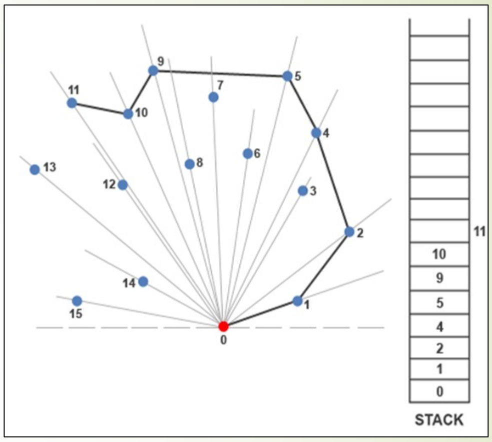


Next, we make a left turn to get to point 10. Point 10 is currently part of the convex hull.

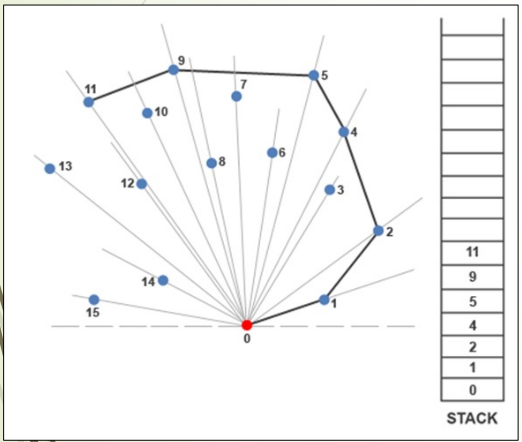
14

Graham's Scan

A right turn is required to get to point 11 from point 10

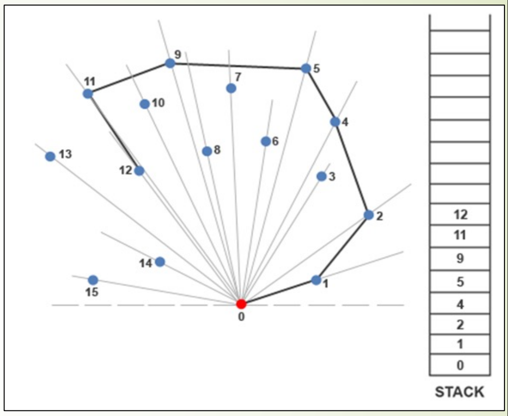


Since a right turn is taken at point 10, point 10 is popped from the stack and the path to point 10 from point 9 is examined. Since a left turn is made at point 9 to get to point 11, point 11 is pushed onto the stack.

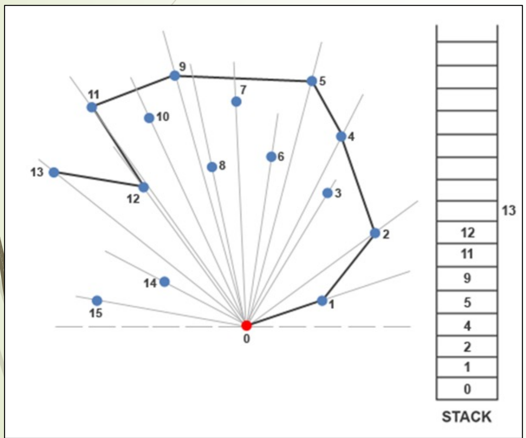


Graham's Scan

A left turn is made at point 11 to get to point 12. Point 12 is therefore pushed onto the stack and is currently considered part of the convex hull

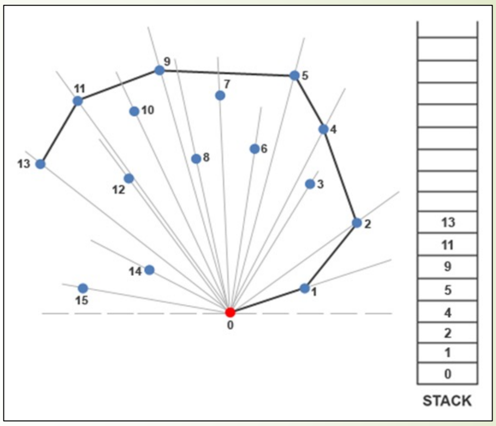


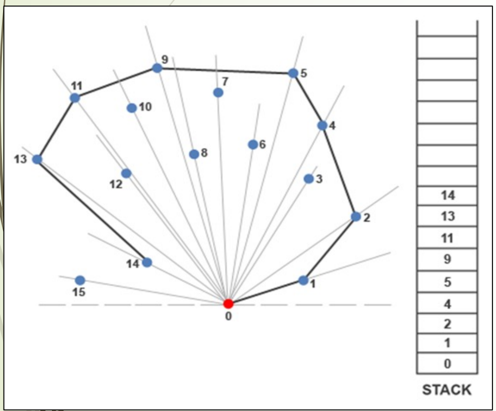
A right turn is required to go to point 13 from point 12



Graham's Scan

Point 12 is popped from the stack and the path to point 13 from point 11 is examined. Since a left turn is made at point 11, point 13 is pushed onto the stack.



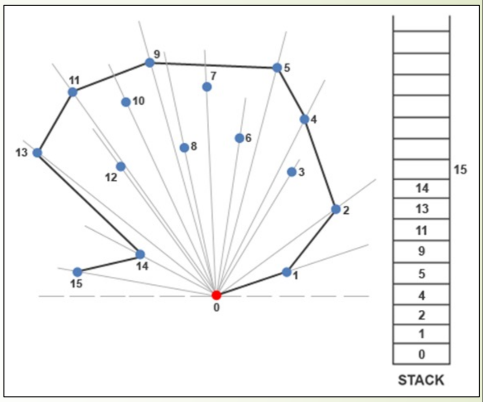


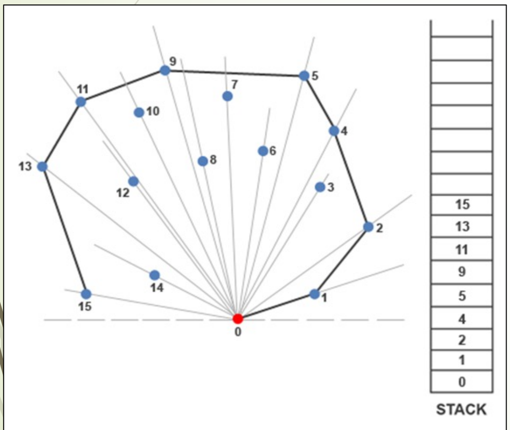
A left turn is made at point 13 to get to point 14, so point 14 is pushed onto the stack.

17

Graham's Scan

A right turn is required to go from point 14 to point 15.





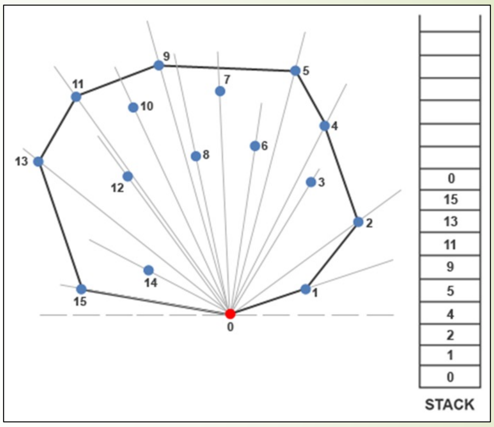
Since a right turn was made at point 14, point 14 is popped from the stack. The path to point 15 from point 13 is examined next. A left turn is made at point 13 to get to point 15, so point 15 is pushed onto the stack.

IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

18

Graham's Scan

Going from point 15 to the starting point 0 requires a left turn. Since the initial point was the point that we needed to reach to complete the convex hull, the algorithm ends.



The points that are needed to create the convex hull are

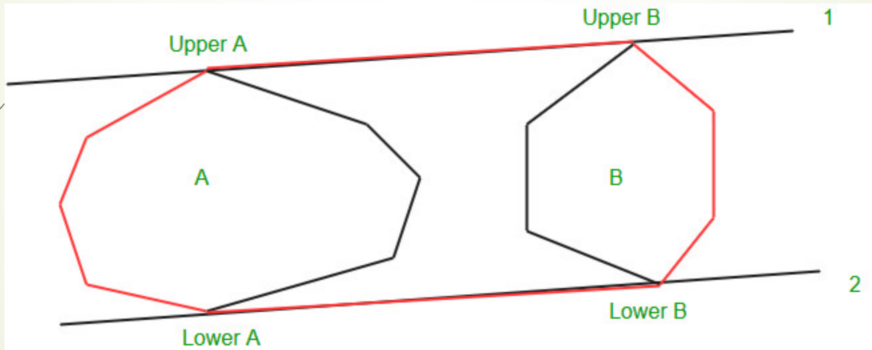
0–1–2–4–5–9–11–13–15.

Divide and Conquer Algorithm

- Each recursive invocation of the algorithm takes as input a subset $P \subseteq Q$ and arrays X and Y, each of which contains all the points of the input subset P.
- The points in array X are sorted so that their x-coordinates are monotonically increasing.
- Similarly, array Y is sorted by monotonically increasing y-coordinate.
- Suppose we know the convex hull of the left half points and the right half points, then the problem now is to merge these two convex hulls and determine the convex hull for the complete set.
- This can be done by finding the upper and lower tangent to the right and left convex hulls.

Divide and Conquer Algorithm

Let the left convex hull be A and the right convex hull be B. Then the lower and upper tangents are named as 1 and 2 respectively, as shown in the figure. Then the red outline shows the final convex hull.



Divide and Conquer Algorithm

- **Divide:**

- It finds a vertical line l that bisects the point set P into two sets P_L and P_R such that $|P_L| = \lceil |P|/2 \rceil$, $|P_R| = \lfloor |P|/2 \rfloor$, all points in P_L are on or to the left of line l, and all points in P_R are on or to the right of l.
- The array X is divided into arrays X_L and X_R, which contain the points of P_L and P_R respectively, sorted by monotonically increasing x-coordinate.
- Similarly, the array Y is divided into arrays Y_L and Y_R, which contain the points of P_L and P_R respectively, sorted by monotonically increasing y-coordinate.
- The division terminates if |P| <= 3.

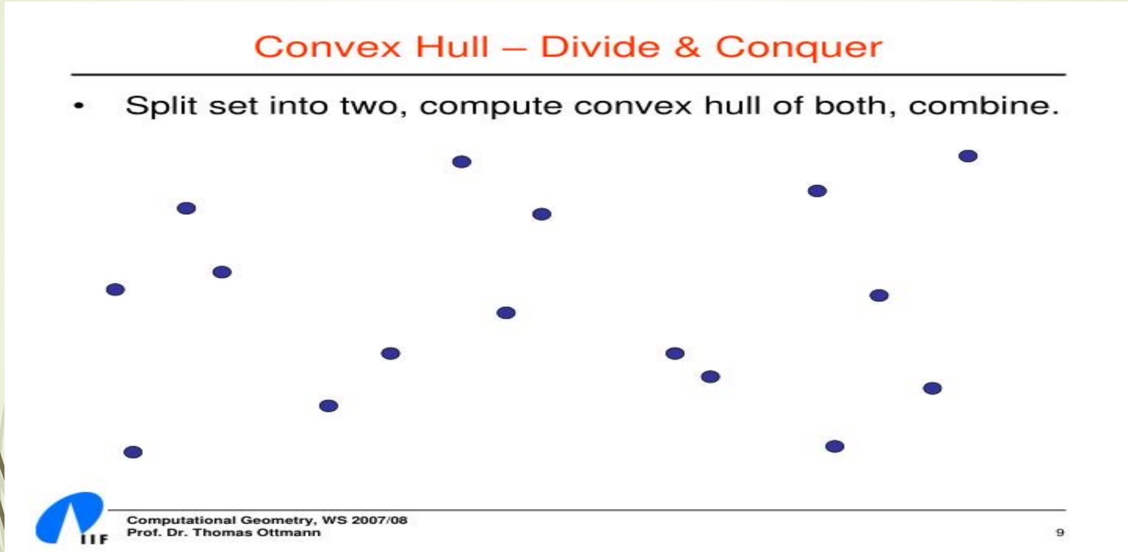
Divide and Conquer Algorithm

- **Conquer:**
- Having divided P into P_L and P_R , it makes two recursive calls, one to find the closest pair of points in P_L and the other to find the closest pair of points in P_R.
- The inputs to the first call are the subset P_L and arrays X_L and Y_L; the second call receives the inputs P_R, X_R, and Y_R.
- Let the closest-pair distances returned for P_L and P_R be δ_L and δ_R , respectively, and let $\delta = \min(\delta_L , \delta_R)$.
- Closest" refers to the usual euclidean distance: the distance between points $p_1=(x_1,y_1)$ and $p_2 = (x_2,y_2)$ is $[(x_1-x_2)^2+(y_1-y_2)^2]^{1/2}$
- Two points in set Q may be coincident, in which case the distance between them is zero.

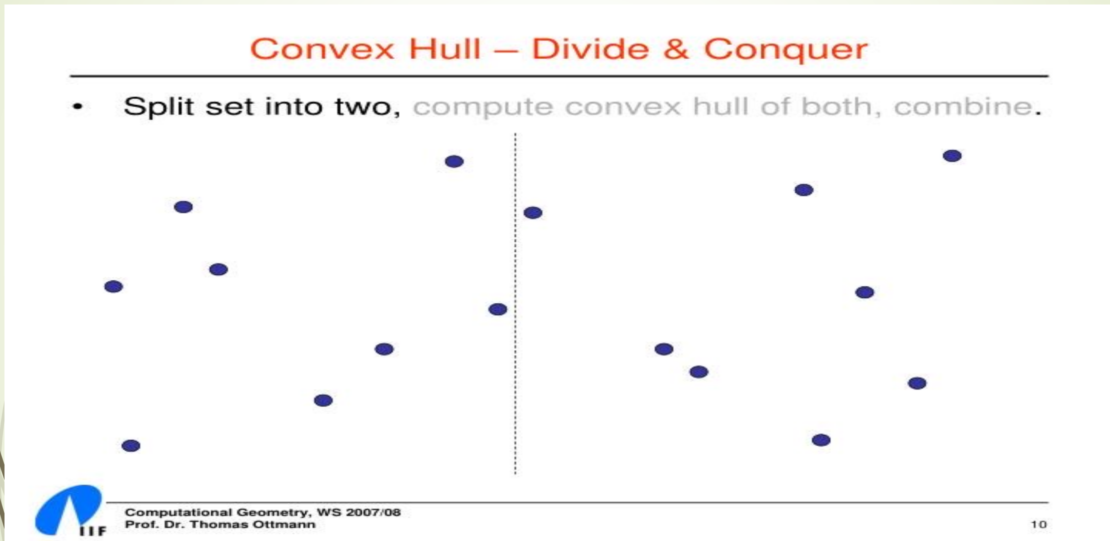
Divide and Conquer Algorithm

- **Combine:** The closest pair is either the pair with distance δ found by one of the recursive calls, or it is a pair of points with one point in *PL* and the other in *PR* .
- To find such a pair, if one exists, the algorithm does the following :
  - It creates an array Y', which is the array Y with all points not in the 2δ-wide vertical strip removed. The array Y' is sorted by y-coordinate, just as Y is.
  - For each point p in the array Y', the algorithm tries to find points in Y' that are within δ units of p. The algorithm computes the distance from p to each of these points and keeps track of the closest-pair distance δ' found over all pairs of points in Y .
  - If $\delta' < \delta$, then the vertical strip does indeed contain a closer pair than was found by the recursive calls. This pair and its distance δ' are returned. Other- wise, the closest pair and its distance δ found by the recursive calls are returned.

Divide and Conquer Algorithm



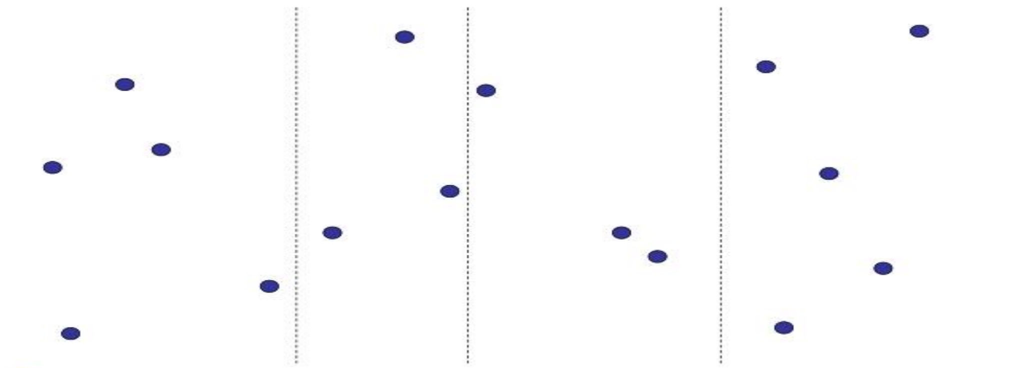
Divide and Conquer Algorithm



Divide and Conquer Algorithm

Convex Hull – Divide & Conquer

- Split set into two, compute convex hull of both, combine.



Computational Geometry, WS 2007/08
Prof. Dr. Thomas Ottmann
11

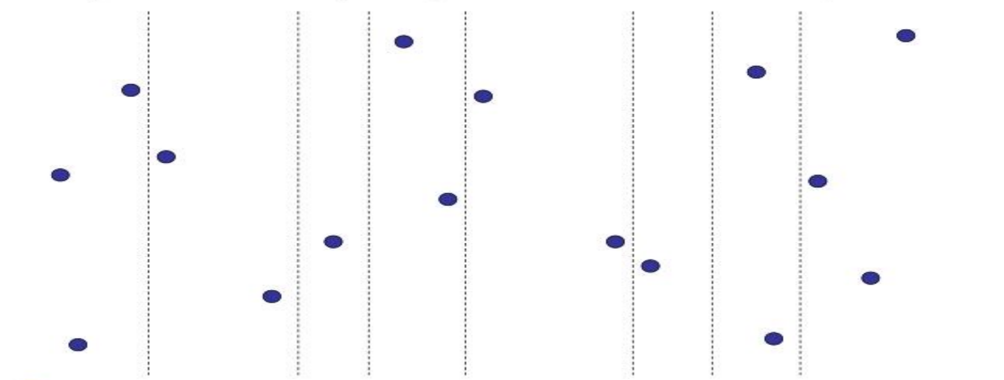
IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

27

Divide and Conquer Algorithm

Convex Hull – Divide & Conquer

- Split set into two, compute convex hull of both, combine.

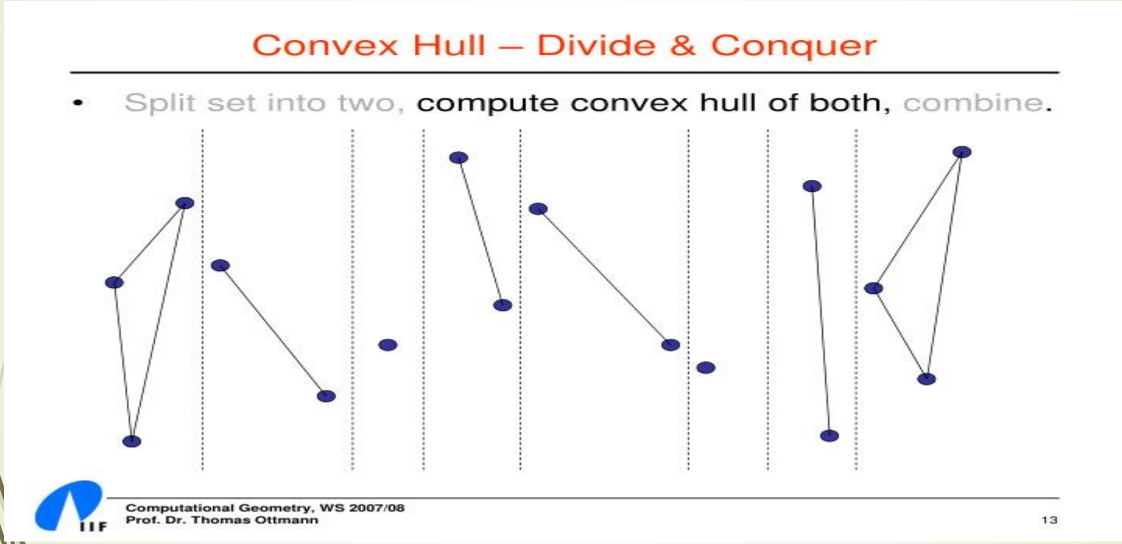


Computational Geometry, WS 2007/08
Prof. Dr. Thomas Ottmann
12

IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

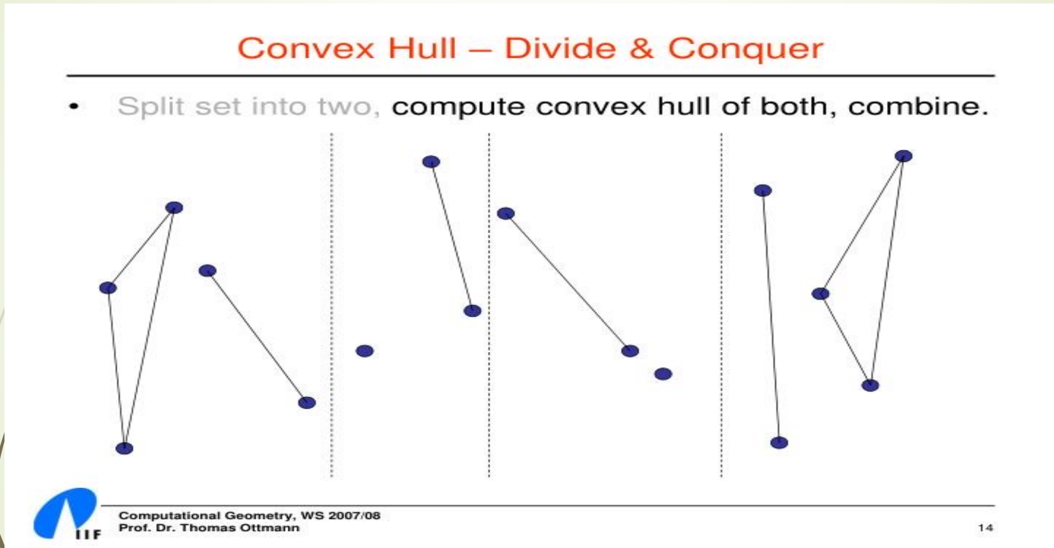
28

Divide and Conquer Algorithm



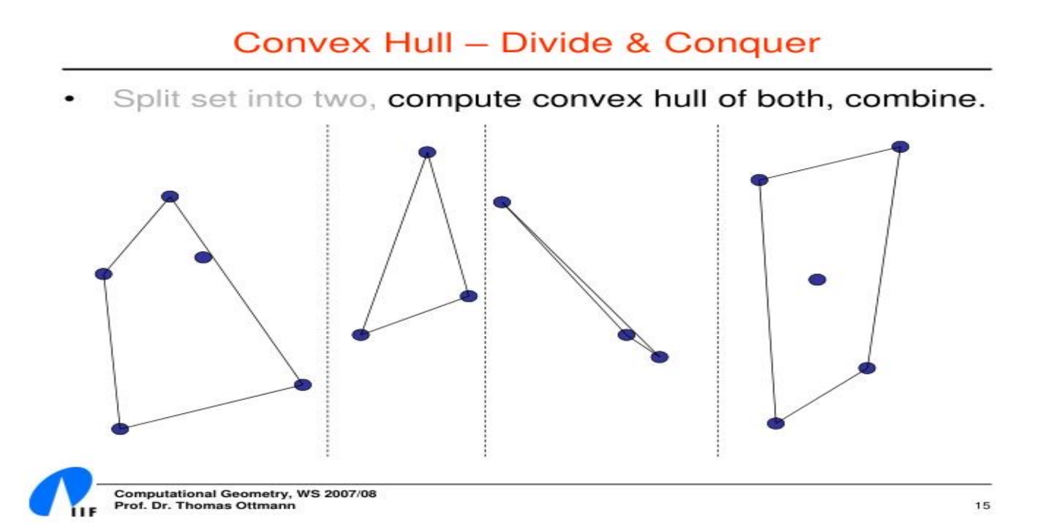
29

Divide and Conquer Algorithm



30

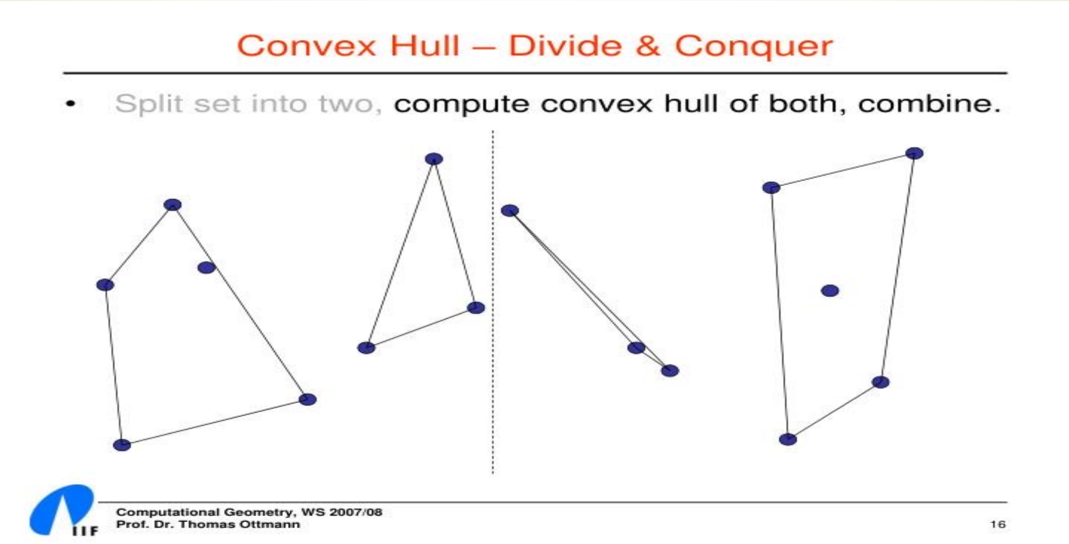
Divide and Conquer Algorithm



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

31

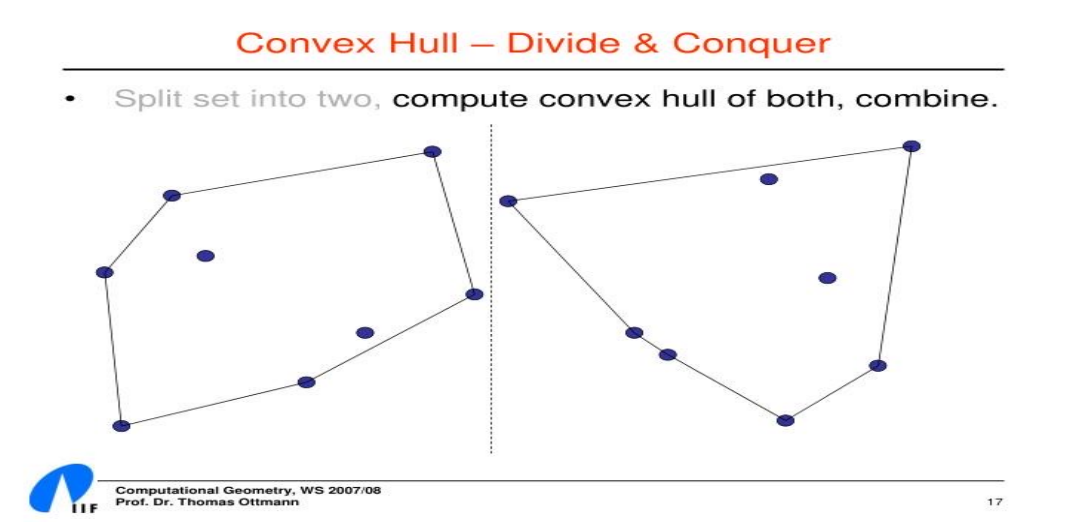
Divide and Conquer Algorithm



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

32

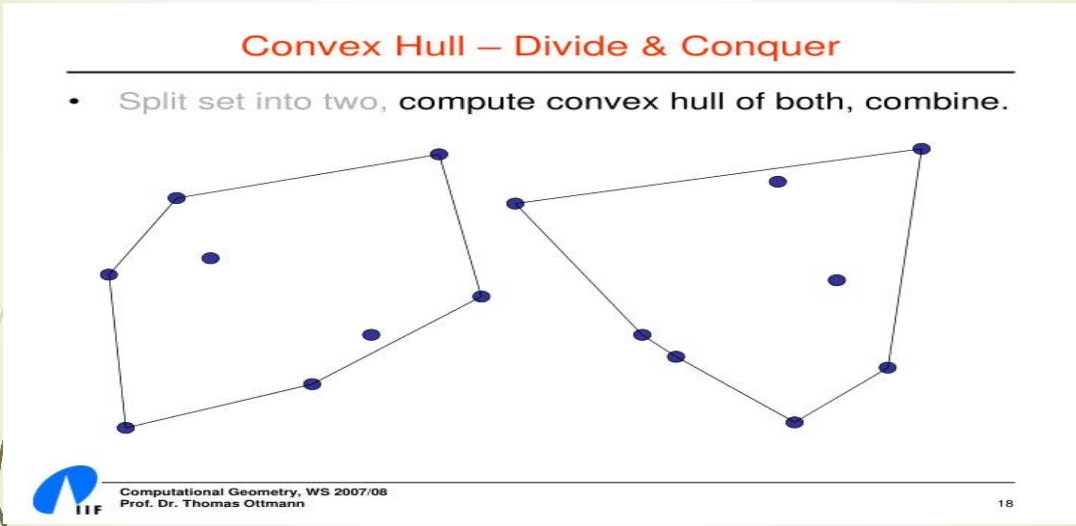
Divide and Conquer Algorithm



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

33

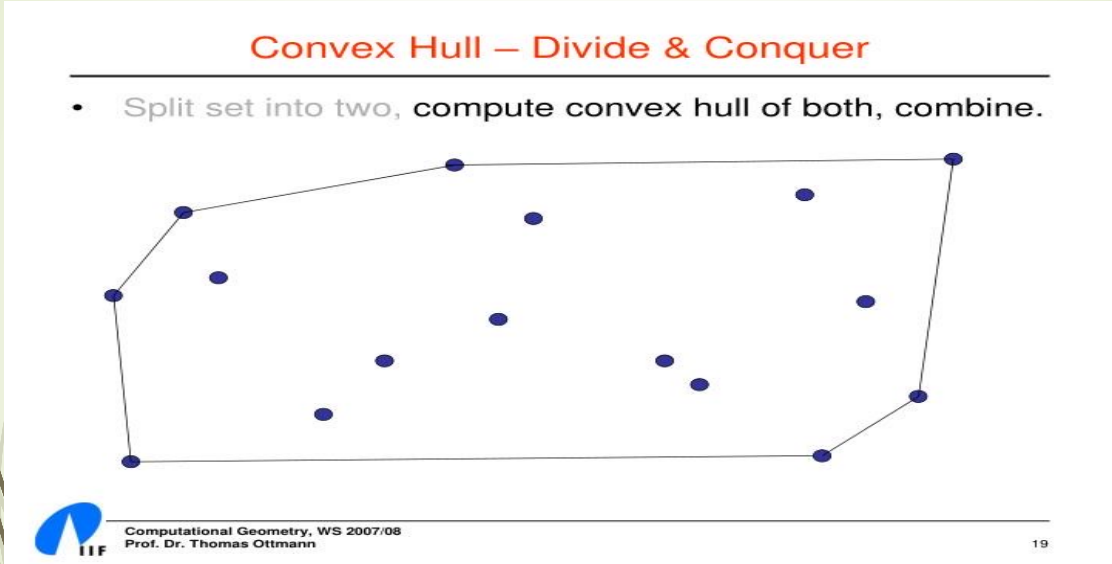
Divide and Conquer Algorithm



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

34

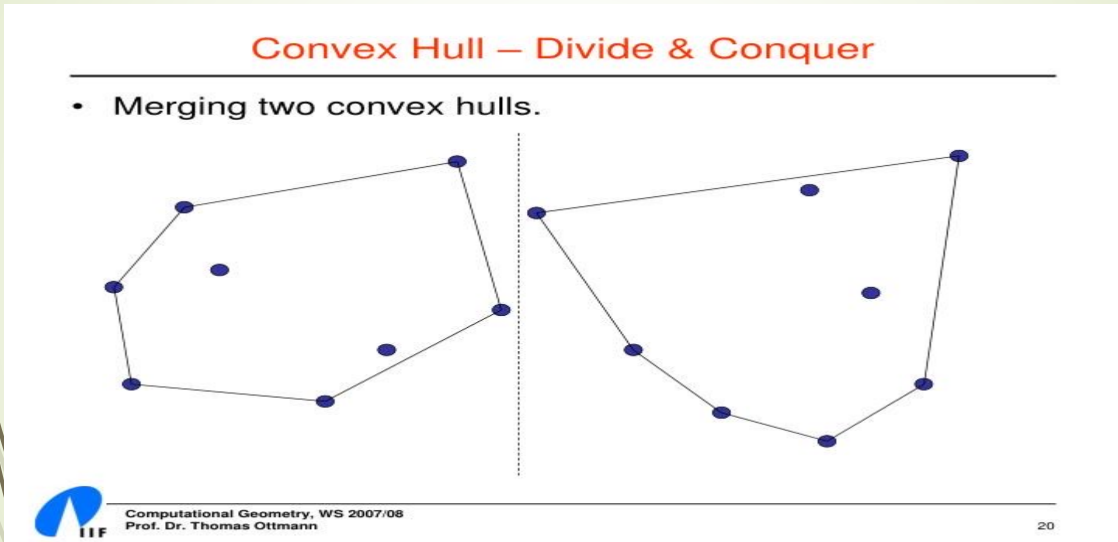
Divide and Conquer Algorithm



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

35

Divide and Conquer Algorithm



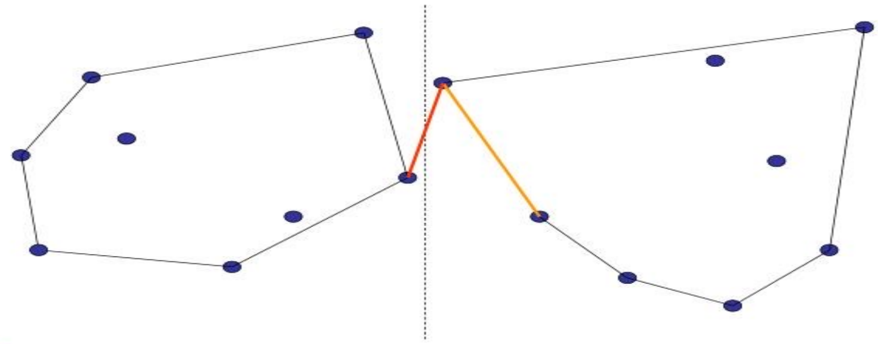
IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

36

Divide and Conquer Algorithm

Convex Hull – Divide & Conquer

- Merging two convex hulls: (i) Find the lower tangent.



Computational Geometry, WS 2007/08
Prof. Dr. Thomas Ottmann
21

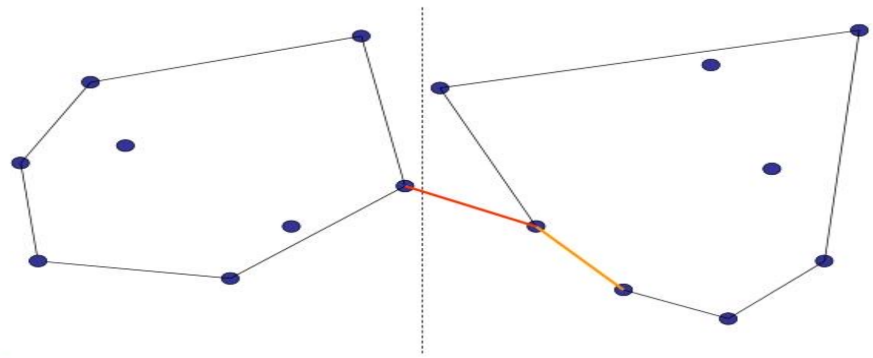
IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

37

Divide and Conquer Algorithm

Convex Hull – Divide & Conquer

- Merging two convex hulls: (i) Find the lower tangent.



Computational Geometry, WS 2007/08
Prof. Dr. Thomas Ottmann
22

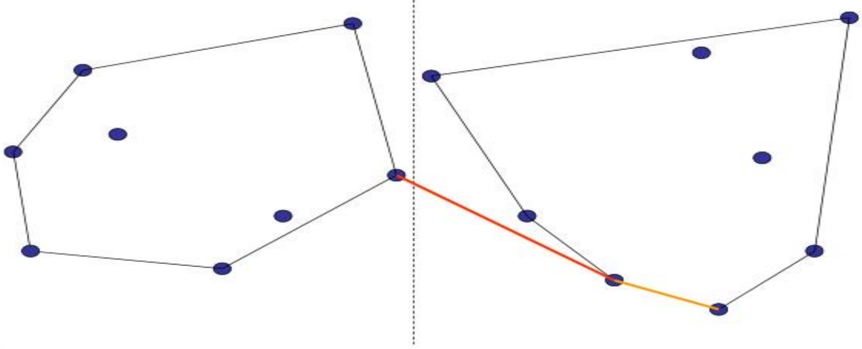
IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

38

Divide and Conquer Algorithm

Convex Hull – Divide & Conquer

- Merging two convex hulls: (i) Find the lower tangent.

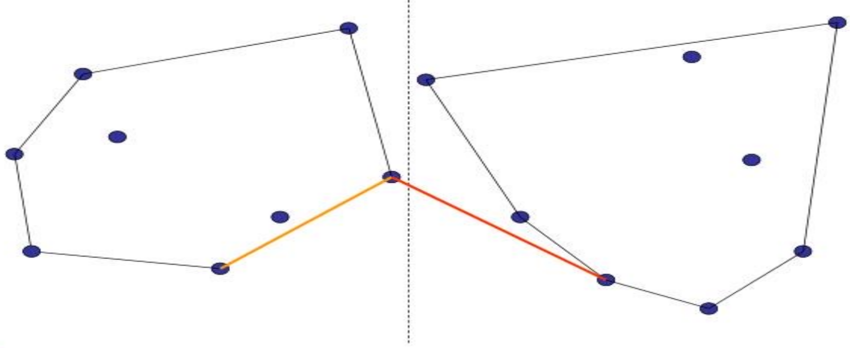


Computational Geometry, WS 2007/08
Prof. Dr. Thomas Ottmann 23

Divide and Conquer Algorithm

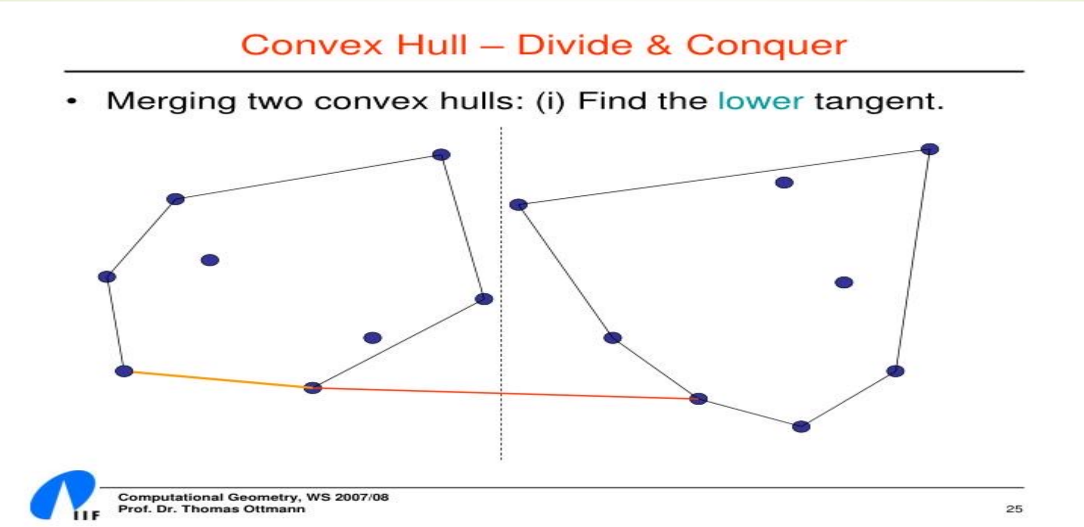
Convex Hull – Divide & Conquer

- Merging two convex hulls: (i) Find the lower tangent.



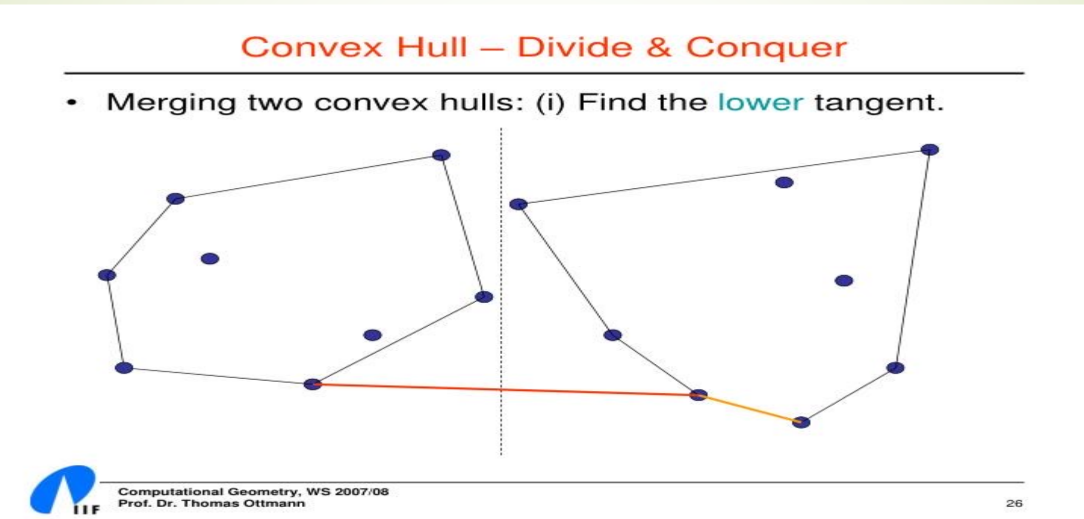
Computational Geometry, WS 2007/08
Prof. Dr. Thomas Ottmann 24

Divide and Conquer Algorithm



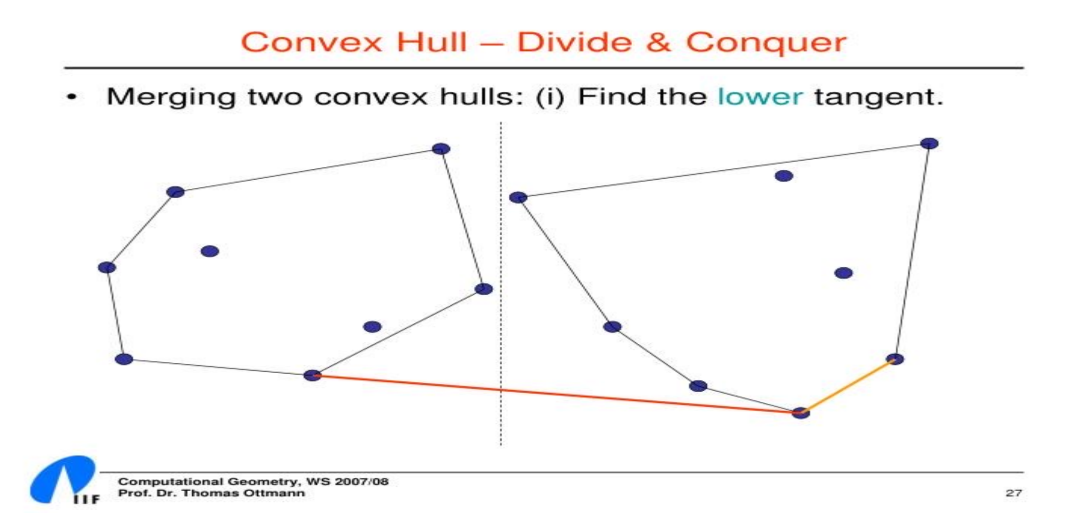
41

Divide and Conquer Algorithm



42

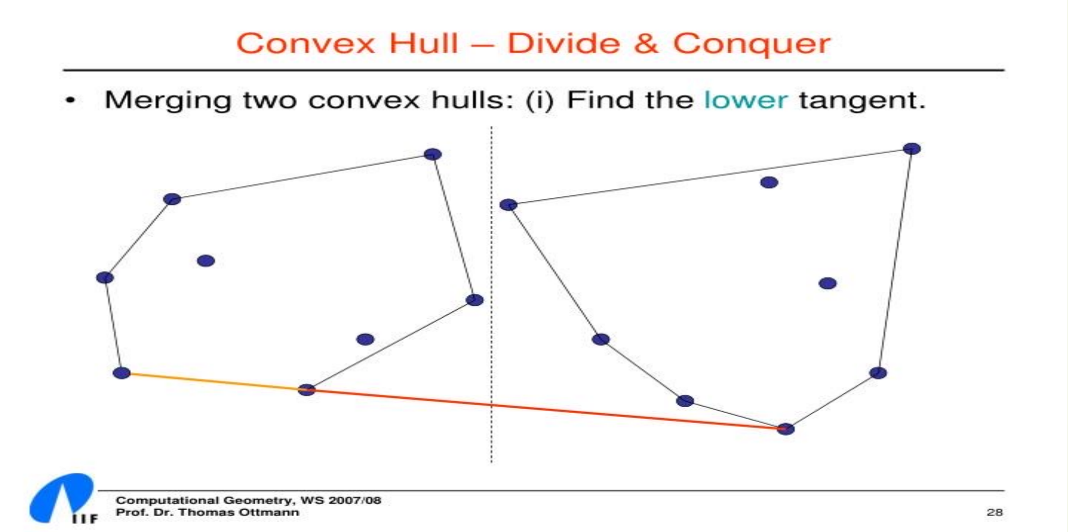
Divide and Conquer Algorithm



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

43

Divide and Conquer Algorithm



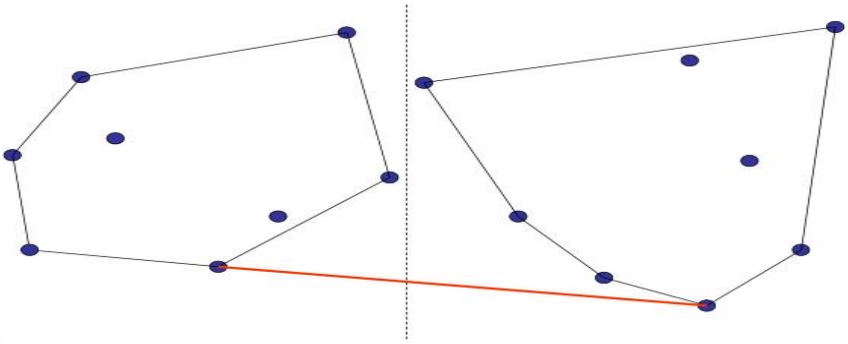
IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

44

Divide and Conquer Algorithm

Convex Hull – Divide & Conquer

- Merging two convex hulls: (i) Find the lower tangent.



Computational Geometry, WS 2007/08
Prof. Dr. Thomas Ottmann
29

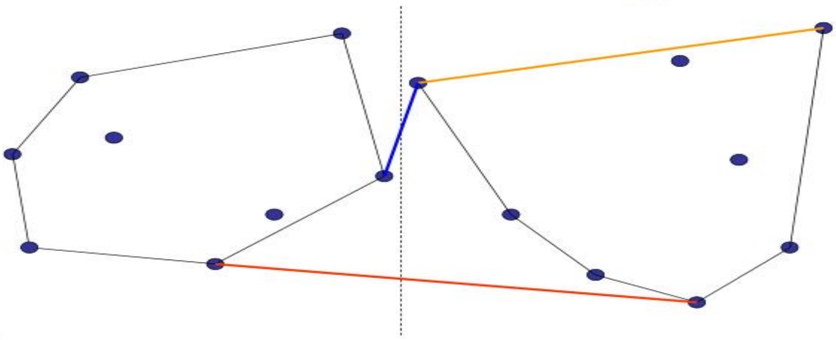
IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

45

Divide and Conquer Algorithm

Convex Hull – Divide & Conquer

- Merging two convex hulls: (ii) Find the upper tangent.

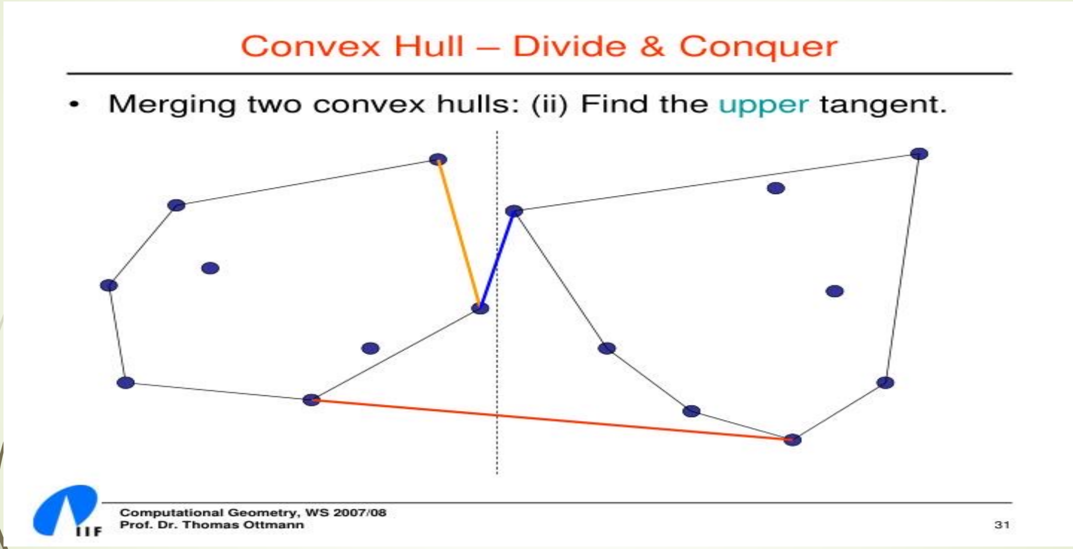


Computational Geometry, WS 2007/08
Prof. Dr. Thomas Ottmann
30

IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

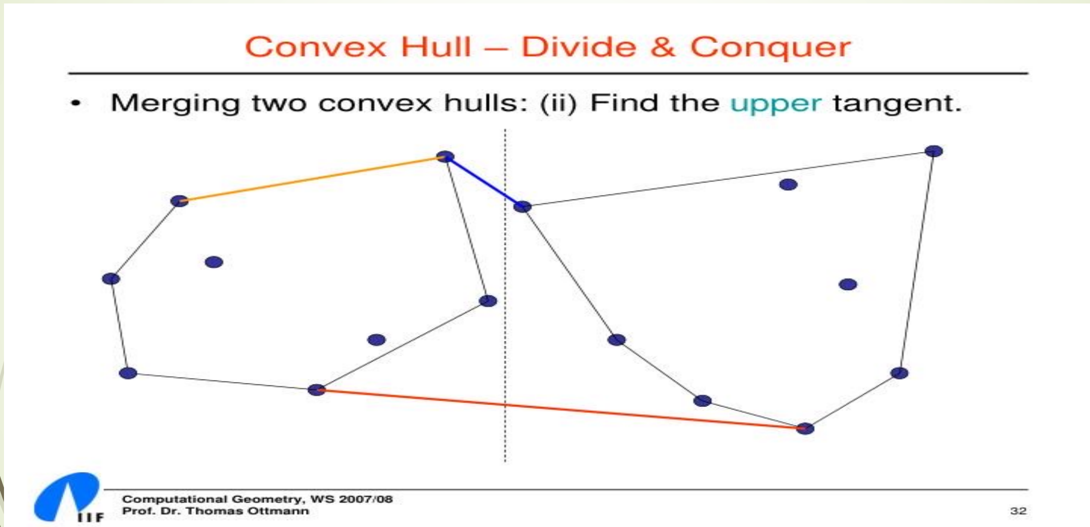
46

Divide and Conquer Algorithm



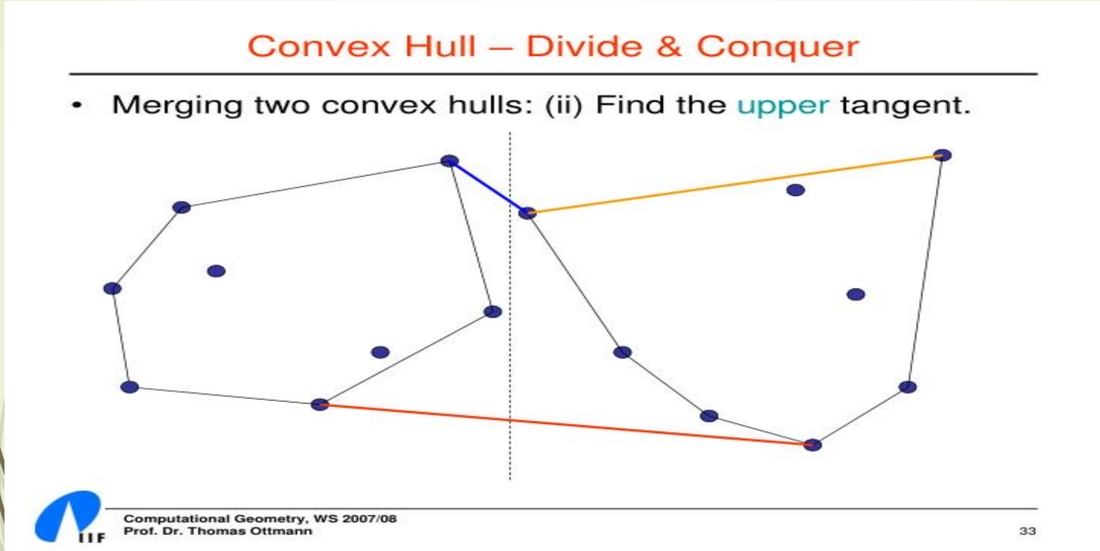
47

Divide and Conquer Algorithm



48

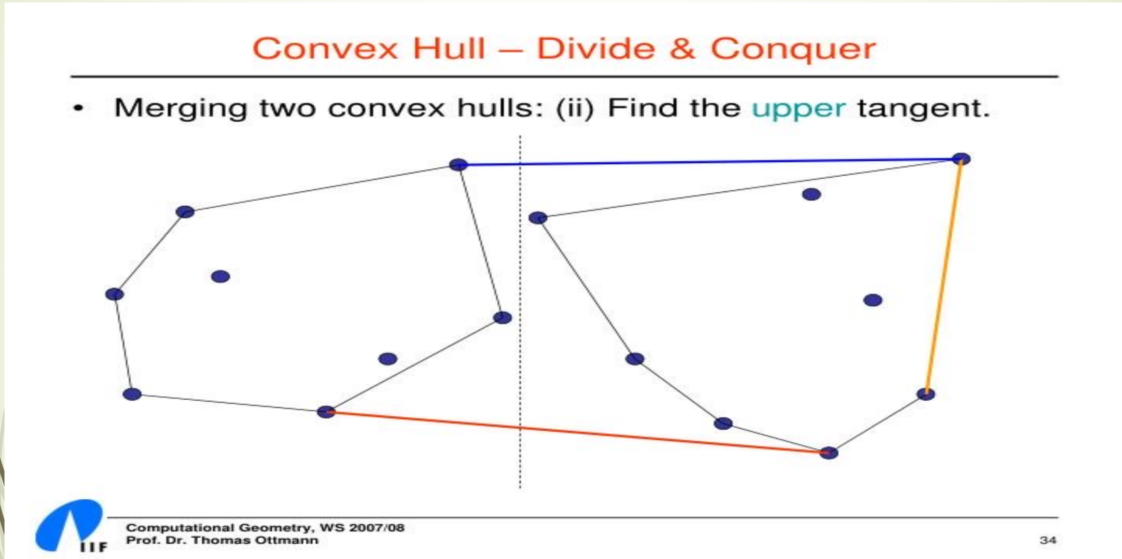
Divide and Conquer Algorithm



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

49

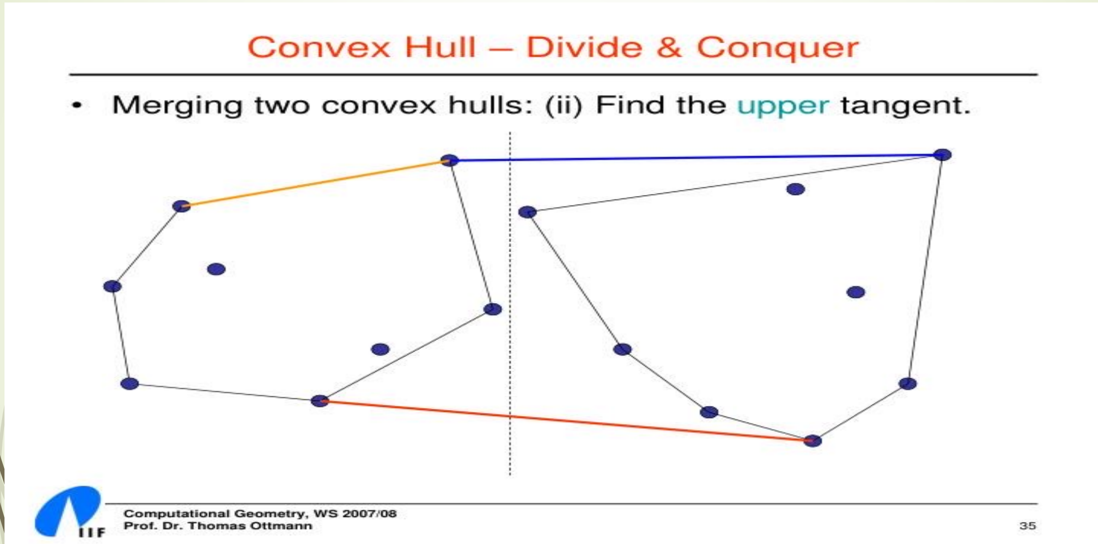
Divide and Conquer Algorithm



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

50

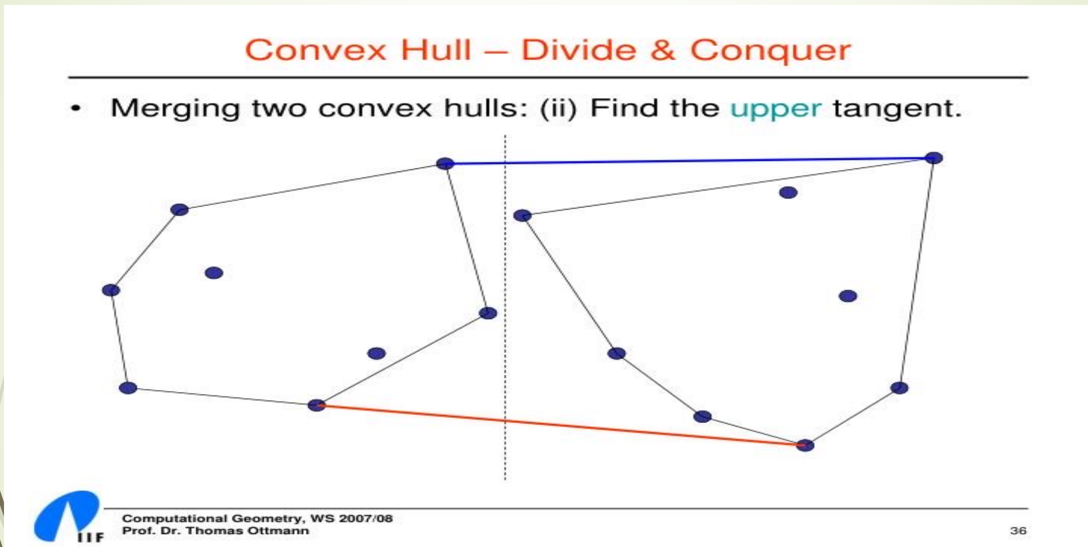
Divide and Conquer Algorithm



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

51

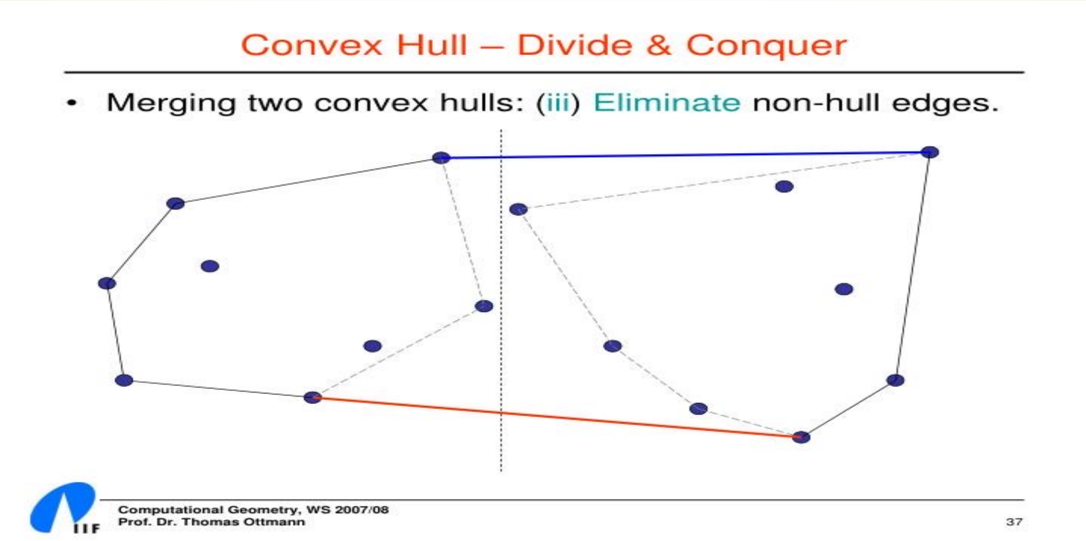
Divide and Conquer Algorithm



IPEC KCS-503. Design and Analysis of Algorithms Dr. Ragini Karwayun

52

Divide and Conquer Algorithm



Divide and Conquer Algorithm

- Analysis :
- In $\theta(n)$time the set of n points is divided into two subsets, one containing the leftmost (n/2)points and one containing the right most (n/2) points, the convex hulls of the subsets are computed recursively, and then it takes O(n) time to combine the hulls.
- ∴ T(n) = 2T(n/2) + O(n) = O(n lg n)