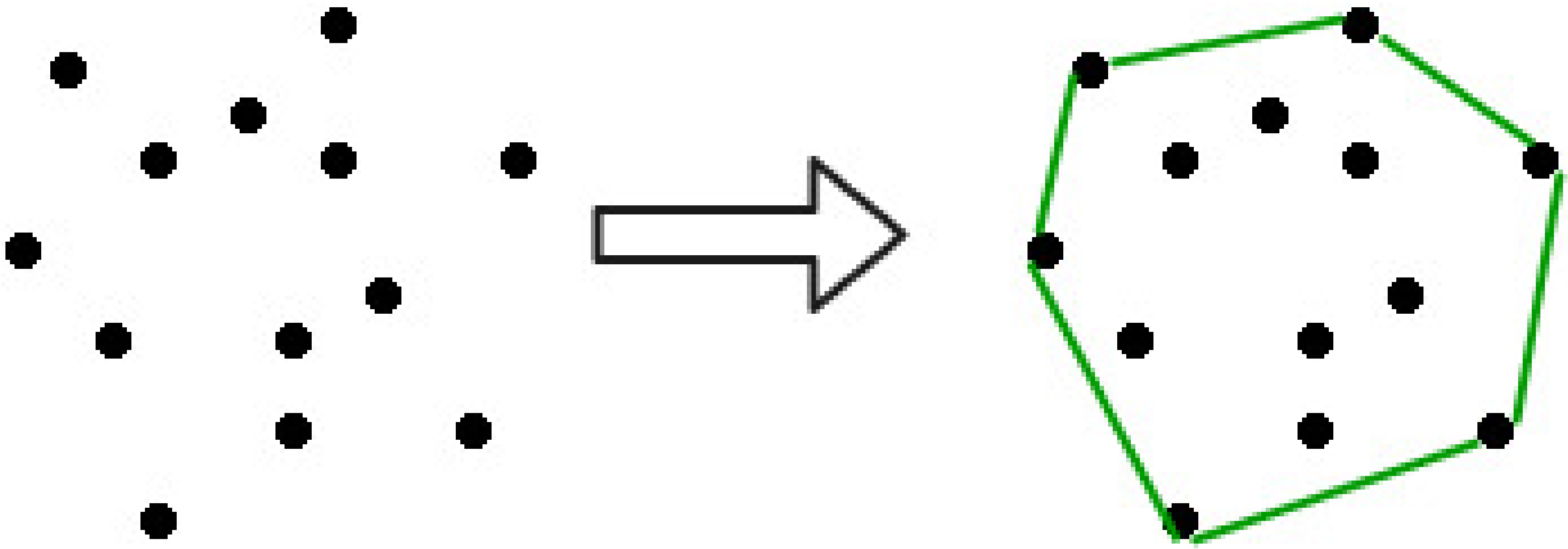
CONVEX HULL

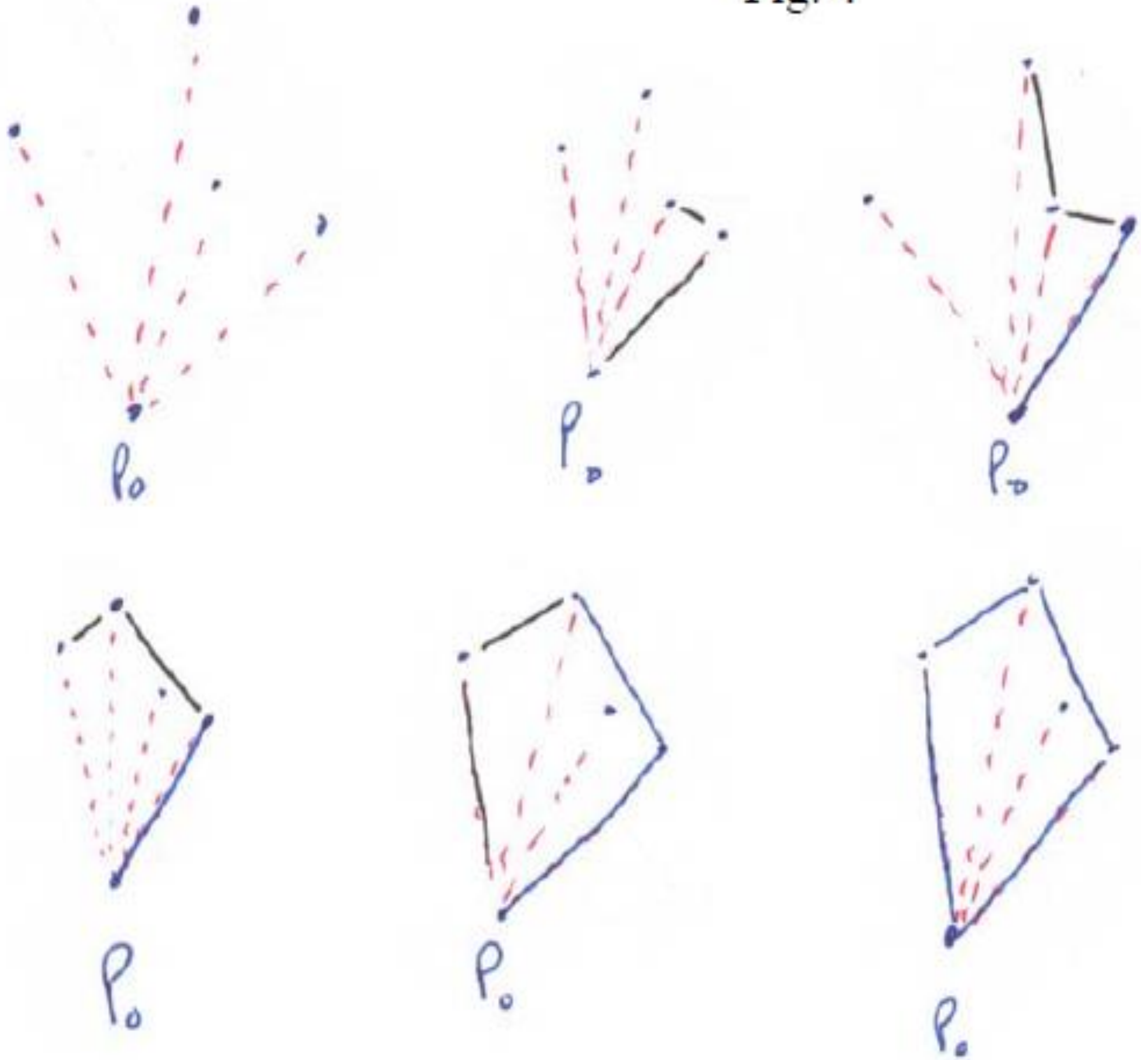
Convex Hull (CH)



Standard algorithms for constructing a Convex Hull

General pic

Fig. 4



A simple example of how the Graham Scan works, showing each step in the algorithm.

Notice that in the third step, the addition of that vertex would cause a concave hull, and therefore it is not added to the final solution.

Red Lines = Polar Lines to Points   Black Lines = Currently Considered Edges   Blue Lines = Convex Hull Edges

Pseudo code: Graham's Scan

1. Let p_0 be the first point (lowermost, left if tie)
2. Let { p_1, p_2, p_3 . . . p_n } be the rest of the points in lexicographic polar sorted order
3. Stack.push(p_0)
4. Stack.push(p_1)
5. Stack.push(p_2)
6. for(int i=3; i<=m; i++)
7.   while angle from p_i, stack.top, and stack.second is a non-left turn
8.   Stack.pop( )
9.   Stack.push(p_i)
10. return the stack

Time Complexity of Graham's Scan

- O(n log n) : For sorting the points in the counterclockwise angular order
- All other operations are of constant time
- In any cases, best, average or worst case, we have to sort the points with respect to the angles
- Lower bound on Time complexity is Ω(n log n)

Optimal algorithm for convex hull

- $\Omega(n \log n)$ algorithm is the optimal one for convex hull construction
- Why there is no algorithm for convex hull with a less complexity than $\Omega(n \log n)$?
- We already know that $\Omega(n \log n)$ is the lower bound for comparison based sorting.
- How to prove that $\Omega(n \log n)$ is the lower bound for convex hull construction too ?
- How do we prove that convex hull construction is as easy as sorting?

Comparison between different problems

- Reduction : Problem X reduces to Problem Y, if given a subroutine for Y, can solve X
- Cost of solving X = Cost of solving Y + Cost of reduction
- One of the Consequences of reduction ?
- **Establish relative difficulty between problems**
- Linear time reduction from sorting problem to convex hull construction and vice versa

Linear time reductions

- Problem X **linear reduces** to problem Y if X can be solved with:
  - Linear number of standard computational steps.
  - One call to subroutine for Y.

Sorting Problem & Convex Hull Construction

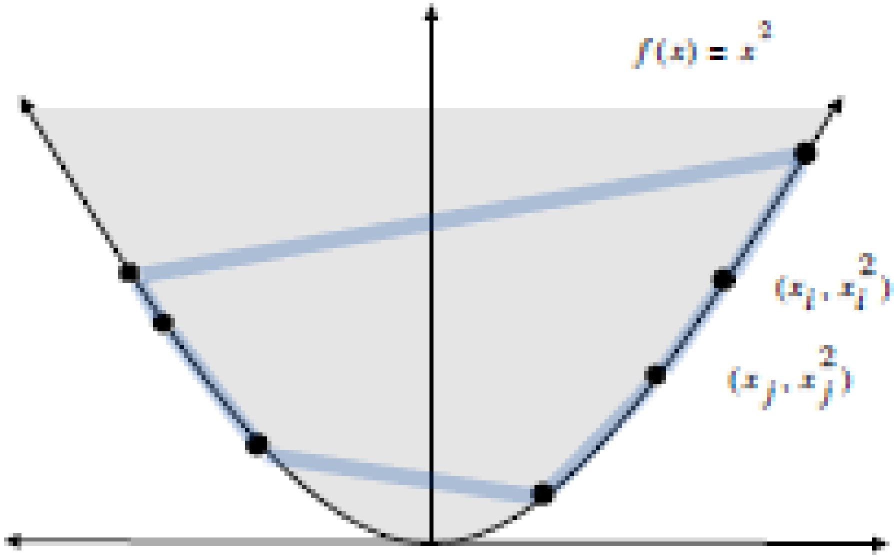
- Claim: Sorting problem linear reduces to Convex hull construction
- Sorting instance : $x_1, x_2, \ldots, x_n$
- Convex hull instance : $(x_1, x_1^2), (x_2, x_2^2), (x_3, x_3^2), (x_4, x_4^2), \ldots, (x_n, x_n^2)$
- What do we infer from the convex hull constructed ?

Exercise

- Given a Sorting instance : 4,-3,0,2,1
- Write the convex hull instance (x_1, x_1^2), (x_2, x_2^2), (x_3, x_3^2), (x_4, x_4^2),…, (x_n, x_n^2)
- Construct the convex hull
- What do we observe?

Convex function

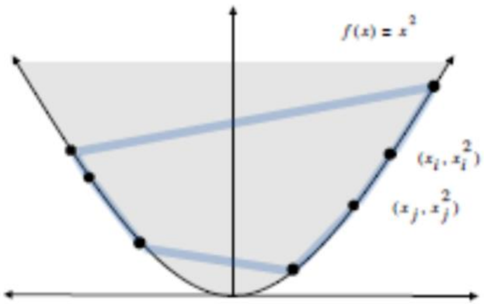
- $(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)$ can be viewed as a parabola which is a convex function



- Region $\{x: x^2 >= x\}$ is convex → all points are on hull

Observation

- Naming the convex hull in counter clockwise order, what do we observe?
- **Starting at point with most negative x, counter-clockwise order of hull points are the numbers sorted in ascending order.**
- Sorting Linear reduces to convex hull construction
- Exercise : Prove convex hull construction linear reduces to Sorting
- Hence, **lower bound of convex hull is Ω(n log n)**



More O(n log n) algorithms

- What is the motivation for yet another O(n log n) algorithm?
- When we design a computational geometric algorithm we should focus on :
- **Efficiency (Efficient in time and space)**
- **Scalability to higher dimensions**

Scalable algorithms

- Graham's scan is not directly scalable/extendable to 3D because it does angular sorting
- Angular sorting is difficult in 3D or

  it has no direct counterpart in 3D
- **Scalable algorithms** :
- Incremental algorithm : $O(n^2)$ / $O(n \log n)$
- Divide and Conquer : $O(n \log n)$

Incremental algorithm

Idea : Incremental algorithm

- First take a subset of the input small enough so that the problem is easily solved.
  - In this case, take 3 points and construct a convex hull
- Then, one by one add remaining elements (of input) while maintaining the solution at each step.

How to add points?

- **Consider points one by one:**
- next point inside current hull—ignore
- next point outside current hull—update
- **Two sub problems to solve:**
- test if point inside or outside polygon
- update hull for outside points

Assumptions

- Input (set of points) : $\{p_0, p_{1,} p_2,\ldots,p_{n-1}\}$
- Points are in general position ie. No three points are collinear

Incremental algorithm : Pseudo code

Algorithm: INCREMENTAL ALGORITHM
Let $H_2 \leftarrow \text{conv}\{p_0, p_1, p_2\}$.
for $k = 3$ to $n - 1$ do
    $H_k \leftarrow \text{conv}\{H_{k-1} \cup p_k\}$

Incremental algorithm

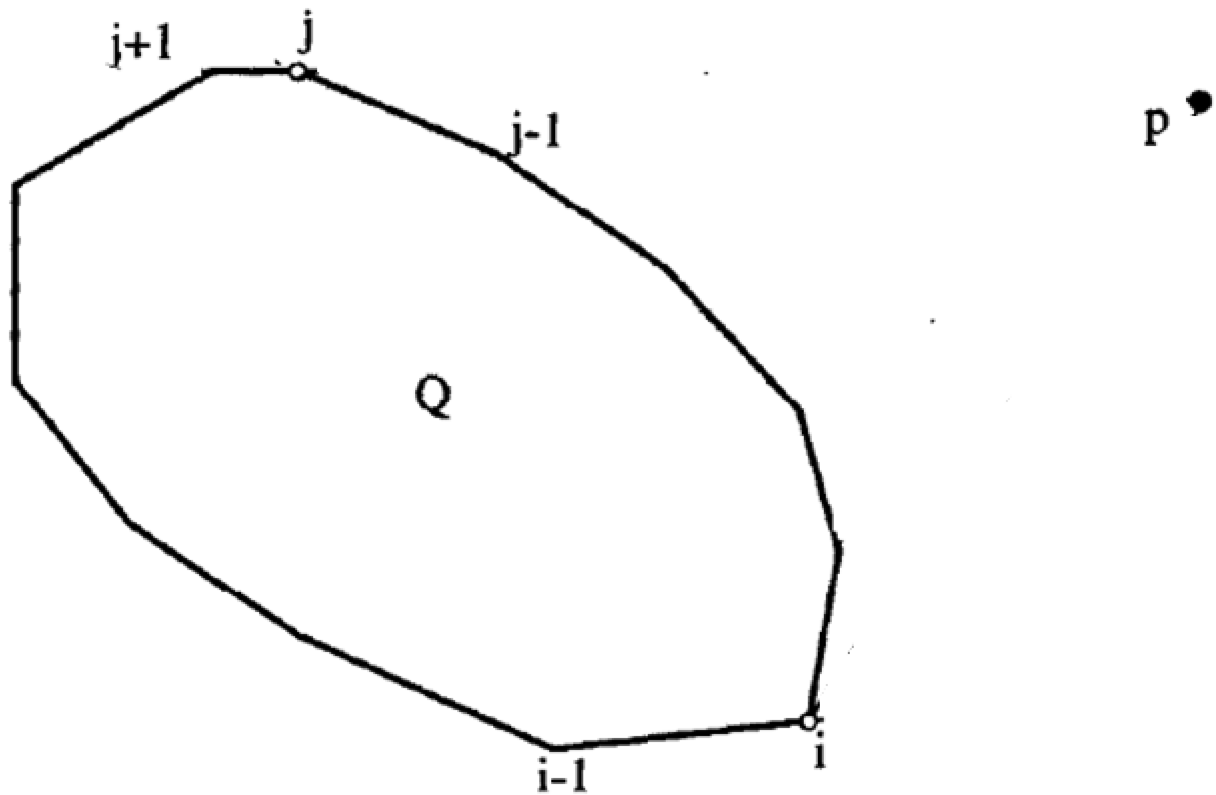
- First hull is a triangle conv$\{p_0, p_1, p_2\}$
- $Q = H_{k-1}$
- $p = p_k$
- Problem of computing convex hull $\{Q \cup p\}$ falls into two cases :
- (i)$p \in Q$
- (ii) $p \notin Q$

Point p belongs to Q

- Discard p
- Even if p is on the boundary
- How to check p belongs to Q ?
- Use Left operation
- Checks p is on to the left of all directed edges of the current polygon
- Complexity : O(n)

Point p does not belong to Q

- If any Left operation test returns False, then p does not belong to Q
- We have to compute {Q U p}
- How?

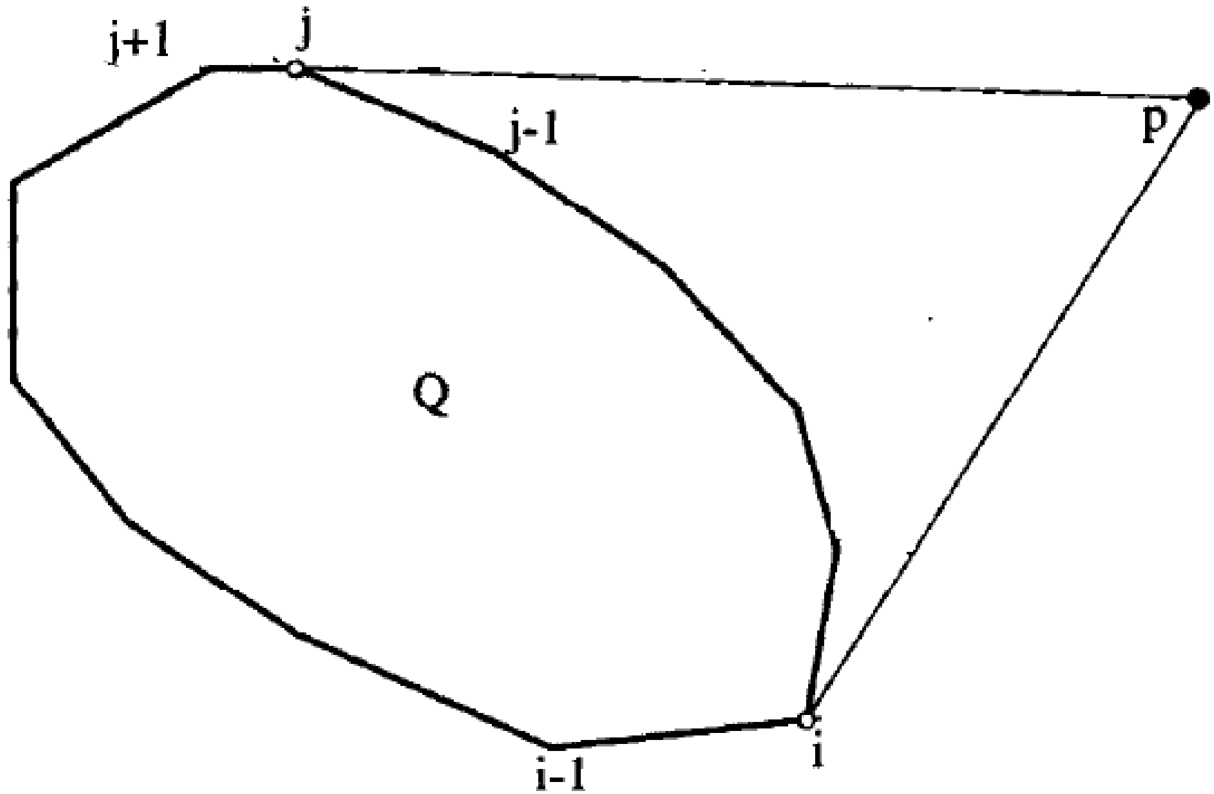


- Simpler idea is to draw line of tangency from p to Q

Line of tangency (tangent line)

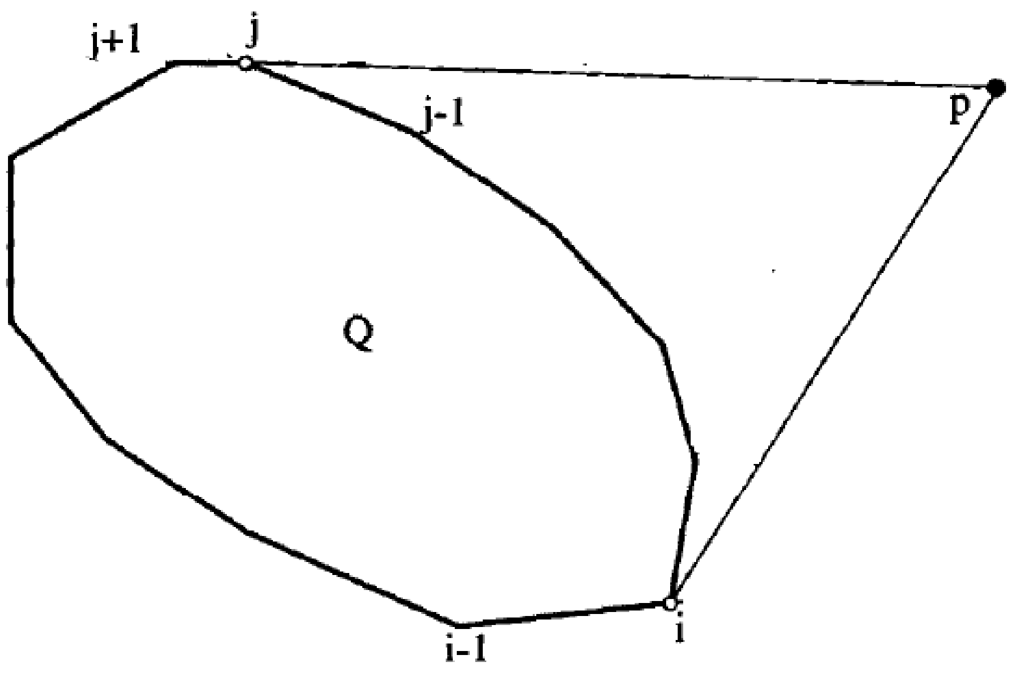
- In geometry, the **tangent line** (or simply **tangent**) to a plane curve at a given point is the straight **line** that "just touches" the curve at that point.
- German mathematician and philosopher Leibniz defined it as the **line** through a pair of infinitely close points on the curve.

Lines of tangency from p to Q

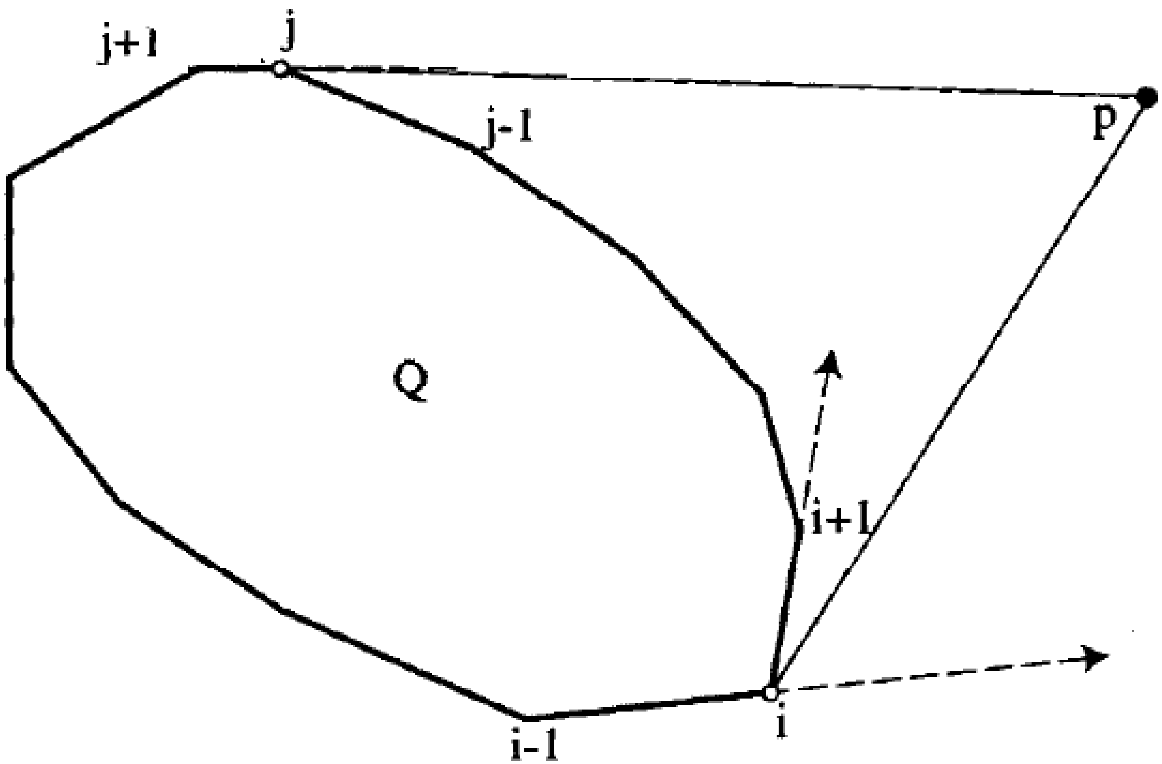


How do we find the point of tangency?

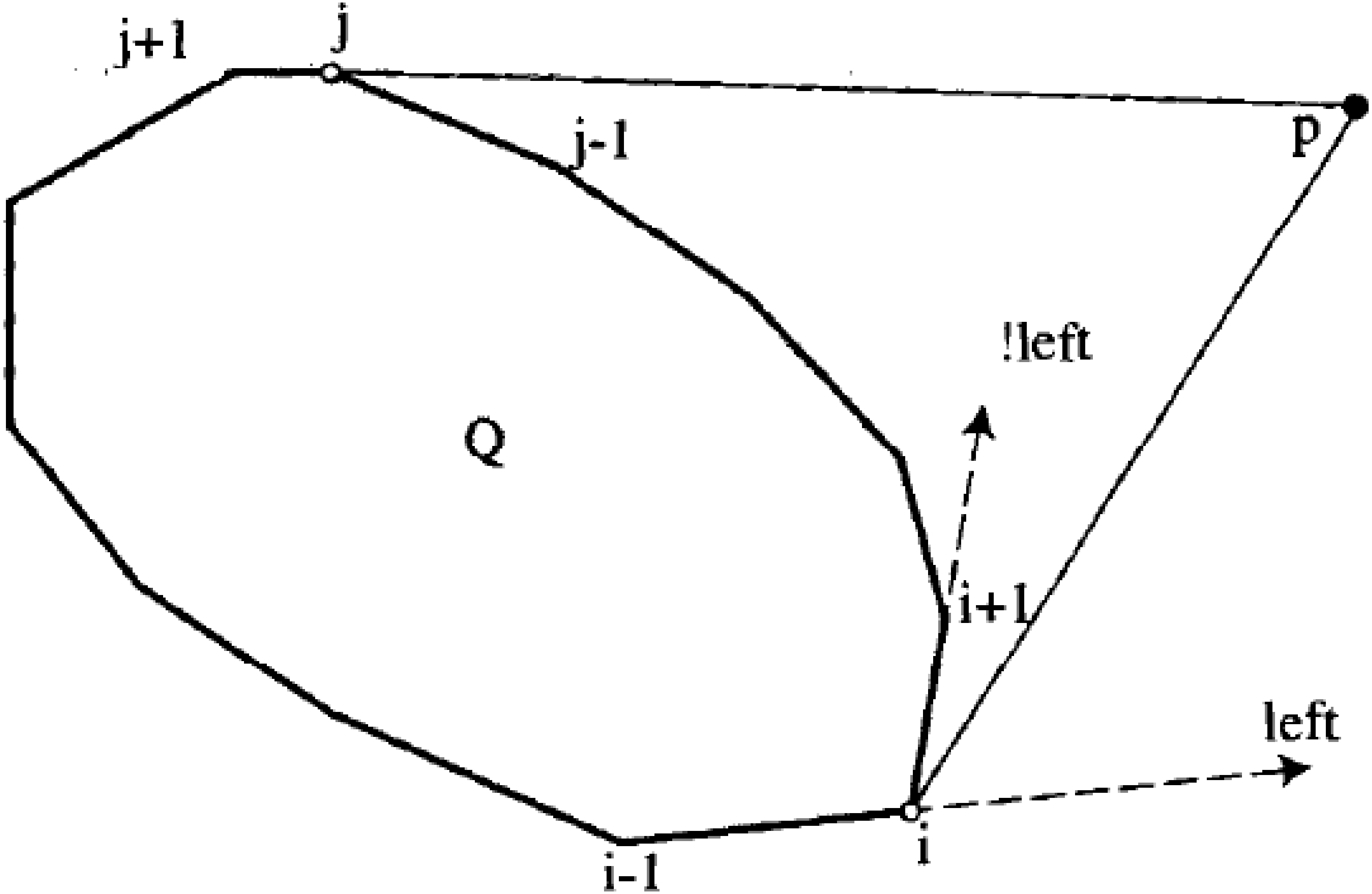
- LeftOn Test again?
- Exercise : Check how do we find p_i is the point of tangency for p



Consider p w.r.t. (p_{i-1},p_i) and (p_i,p_{i+1})

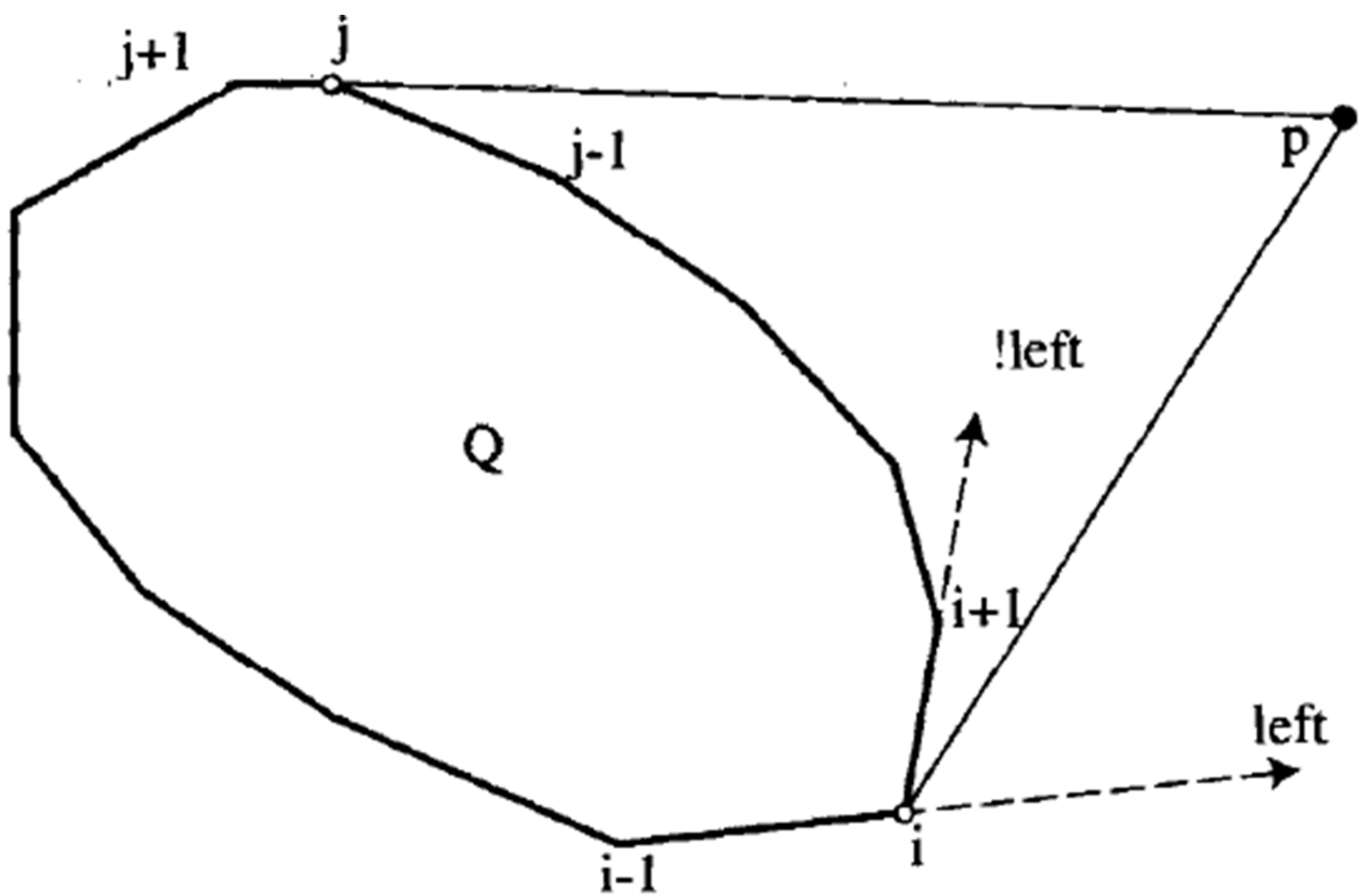


- P is to the left of (p_{i-1},p_i) and right of (p_i,p_{i+1})

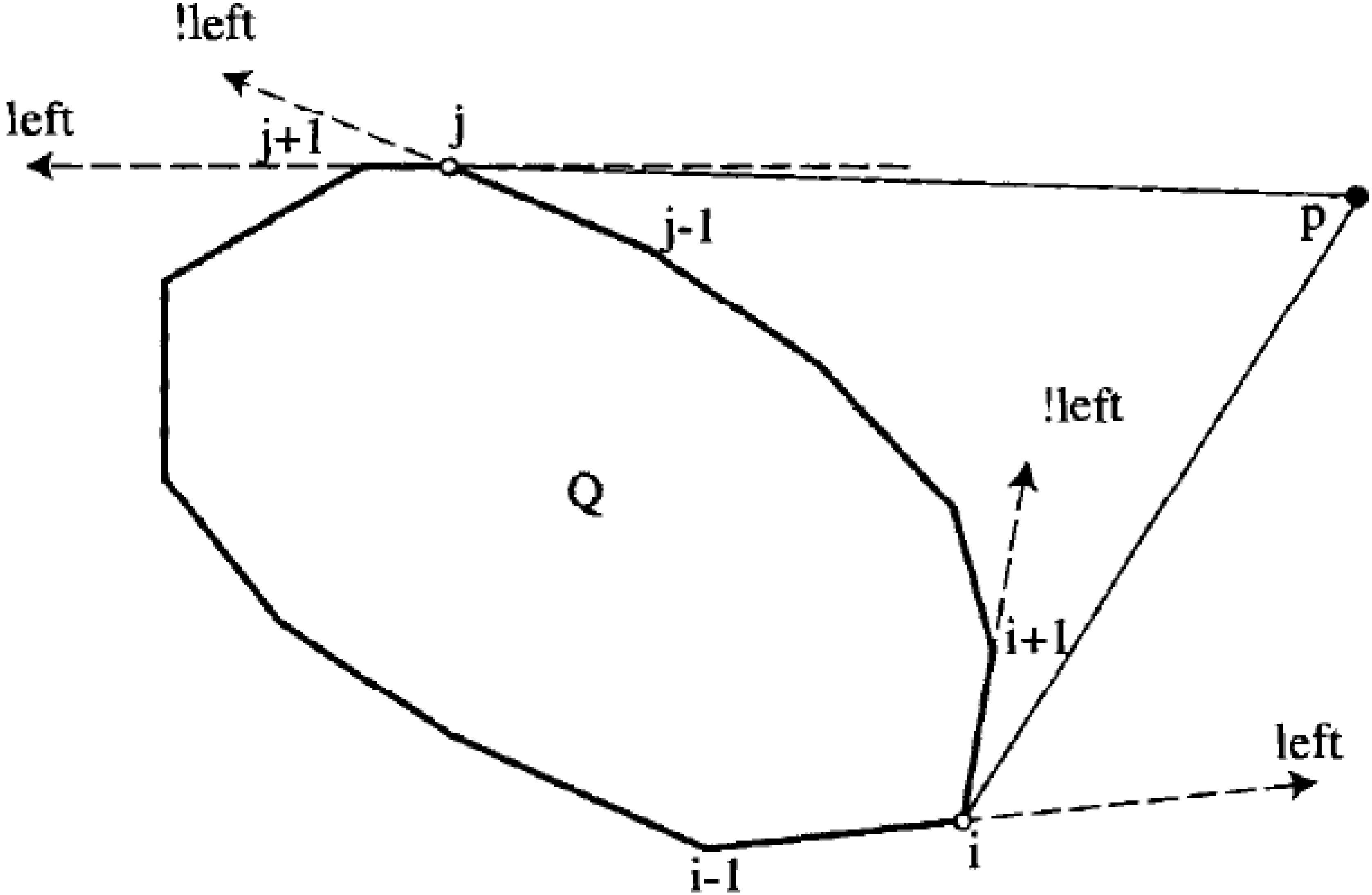

j+1
j
j-1
p
!left
Q
i+1
left
i-1
i

Exercise

- Check p w.r.t. (p_{j-1}, p_j) and (p_j, p_{j+1})

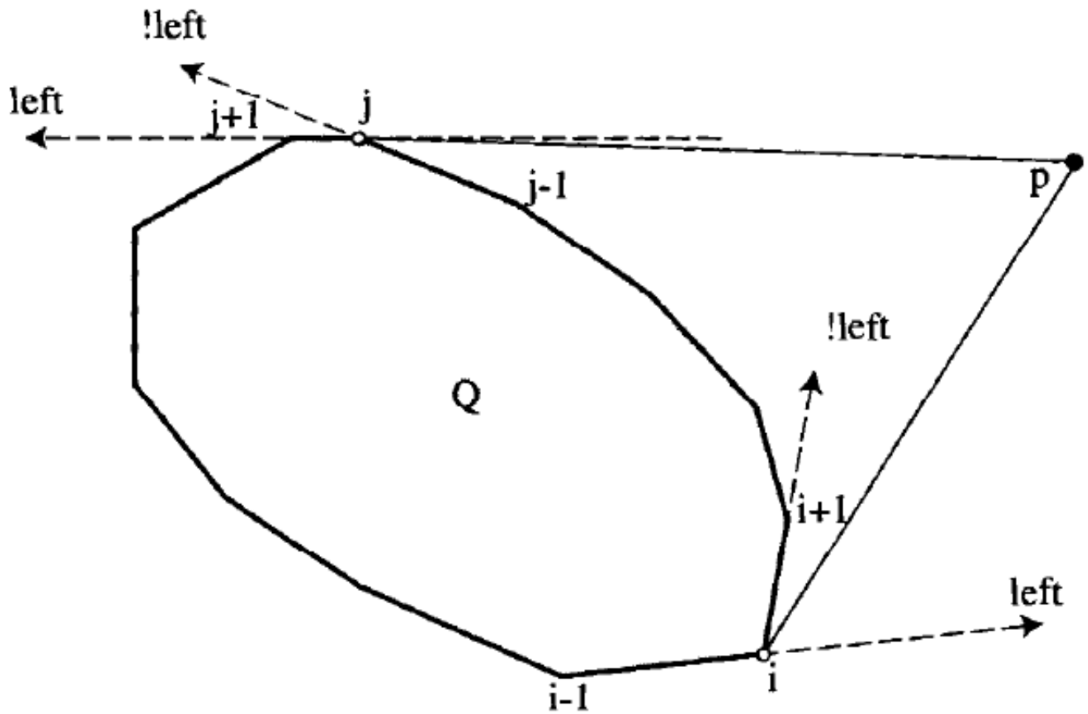


General pic



Points of tangency

- For the lower point $p_{i,}$ p is left of (p_{i-1},p_i) and right of $(p_{i,}p_{i+1})$
- For the upper point $p_{j,}$ p is right of (p_{j-1},p_j) and left of $(p_{j,}p_{j+1})$



Point of tangency

- In general : **p_i is a point of tangency, if two successive edges yield different results of Left or On**

Algorithm: TANGENT POINTS
for $i = 0$ to $n - 1$ do
  if Xor (p left or on (p_{i-1}, p_i), p left or on (p_i, p_{i+1}))
    then p_i is point of tangency

Time Complexity

- The work at each step is O(k) , where k is the number of edges of the k^{th} hull
- In the worst case, suppose p does not belong to Q, the work done is 3+4+….+n , where 3 is the number of edges of the first hull (triangle) and so on
- 3+4+….+n = ?
- $O(n^2)$
- It can be improved to O(n log n) . Think how to improve it….left as an exercise.

Reading Exercise

- DIVIDE and CONQUER

  [Preparata & Hong, 1977]

References

- J. O'Rourke: Art Gallery Theorems and Algorithms
- J. O Rourke, *Computational Geometry in C,* 2/e, Cambridge University Press, 1998 *)*
- *http://personal.denison.edu/~havill/272S04/papers/convexhulls.pdf [Prepared by Eric Eilberg, Denison University]*
- *https://www.uniweimar.de/fileadmin/user/fak/medien/professuren/Computer_Graphics/Algo19/05_MoreGeomAlgs.pdf*

Thank you