OI-OPPA. European Social Fund
Prague & EU: We invest in your future.

Diameter of a point set

Martin Krošlák

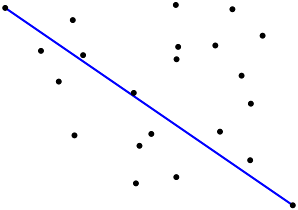
Computer Graphics
Open Informatics
CTU in Prague

October 24, 2012

Contents

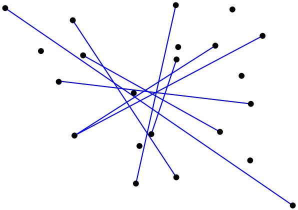
Theory

- diameter of a set of points
  - maximum distance between any 2 points in the set
  - in 2D it is diameter of a bounding circle (n-sphere in general) enclosing all points of the set



Theory

- Problem: Which points to use?
  - brute force solution - $\Theta(n^2)$ where n is number of points

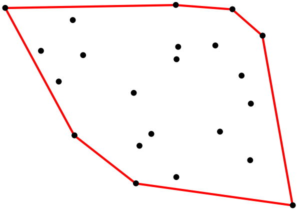


Theory

Fact

The diameter of a set is equal to the diameter of its convex hull.

- convex hull typically consists of much fewer points
- in worst case, all points are on convex hull (eg. circular distribution)

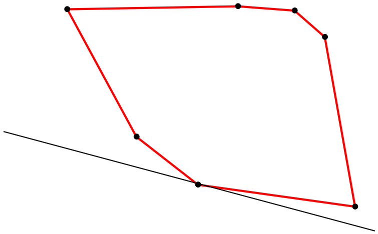


Theory

Definition

Given a convex polygons P, a line of support l is a line intersecting P and the interior of P lies to one side of l.

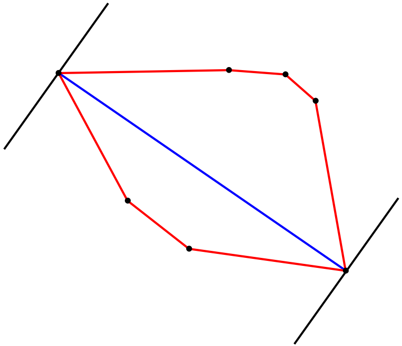
- "tangent" of a convex polygon



Theory

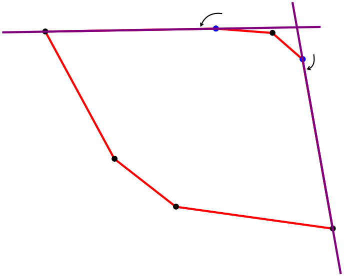
Fact

The diameter of a convex polygon is the greatest distance between its parallel lines of support.



Theory

- If support lines passing two points on convex hull cannot be parallel, these points cannot form diameter!!!



Theory

Definition

A pair of points that allows parallel supporting lines is called *antipodal*.

- it has been shown, that the number of *antipodal* pairs is linearly dependent on number of points of convex hull
  - specifically, it is at most $3n/2$

Contents

Algorithm

Algorithm for finding diameter of a point set:

- construct convex hull of given set of points
  - complexity $O(n \log n)$ where n is number of points in the set
- find antipodal pairs
  - complexity in 2D is $O(h)$ where h is number of points on the convex hull
- find the diametral pair among antipodal pairs and determine it's length
  - complexity $O(p)$ where p is number of antipodal pairs

Algorithm

Find antipodal pairs

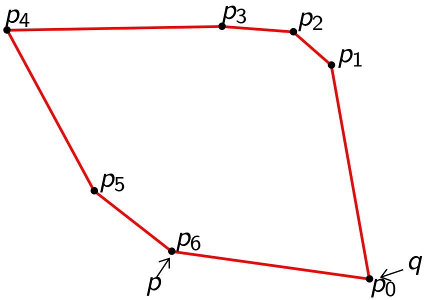
- uses two pointers, p and q, which iterate over points of convex hull in counter-clockwise order
- repeatedly calculates area of triangles formed by p, q and points immediately following p and q
- wraps point indexing; point p_0 is following after p_n

Algorithm

Find antipodal pairs

1. start with $p = p_n$ and $q = p_0$
2. repeatedly move q forward until first antipodal pair is found
3. set q_0 to current position of q
4. in main loop, each time q or p is incremented, or when when we find two parallel lines, (p, q) pair is added to antipodal pairs
5. main loop terminates when whole convex hull has been traversed by q (when $q = p_0$)

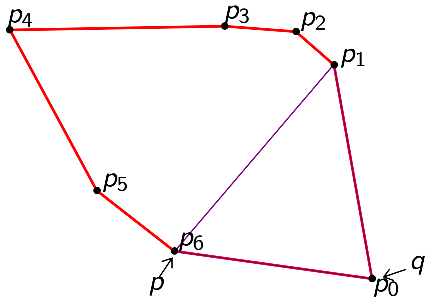
Algorithm



$p = p_n$;

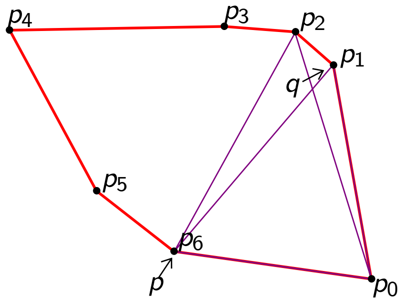
$q = p$.next;

Algorithm



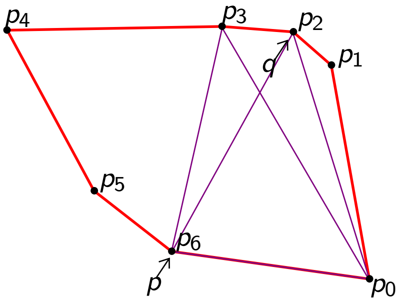
while $area(p,\ p.next,\ q.next) > area(p,\ p.next,\ q)$ **do**
$\quad q = q$.next;

Algorithm



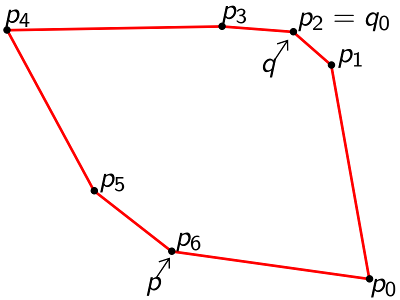
while $area(p,\ p.next,\ q.next) > area(p,\ p.next,\ q)$ **do**
  $q = q.\text{next};$

Algorithm



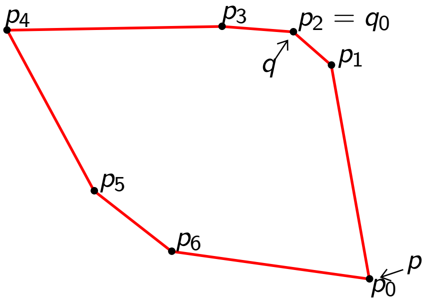
while *area*$(p, p.next, q.next) >$ *area*$(p, p.next, q)$ **do**
$\quad q = q$.next;

Algorithm



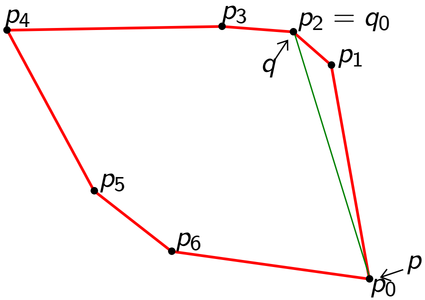
$q_0 = q$;
while $q \neq p_0$ **do** main loop

Algorithm



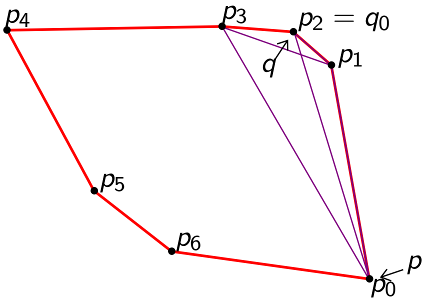
$p = p.\text{next};$

Algorithm



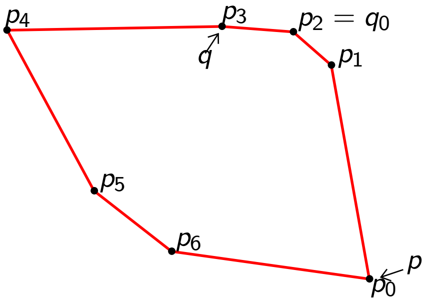
pairs.add(p,q);

Algorithm



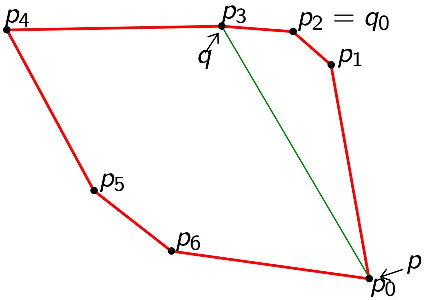
while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**
    $q = q$.next;
    if $(p, q) \neq (q_0, p_0)$ **then**
        pairs.add(p,q);

Algorithm



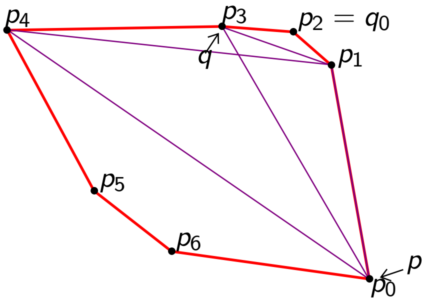
while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**
  $q = q$.next;
  if $(p, q) \neq (q_0, p_0)$ **then**
    pairs.add(p,q);

Algorithm



while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**
  $q = q$.next;
  if $(p, q) \neq (q_0, p_0)$ **then**
    pairs.add(p,q);

Algorithm



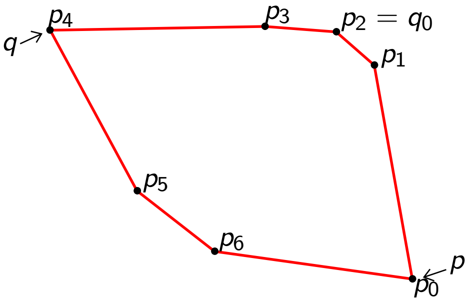
while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**

  $q = q$.next;

  if $(p, q) \neq (q_0, p_0)$ **then**

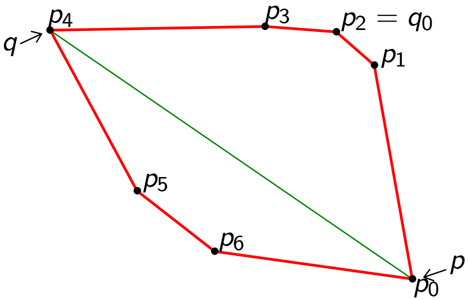
    pairs.add(p,q);

Algorithm



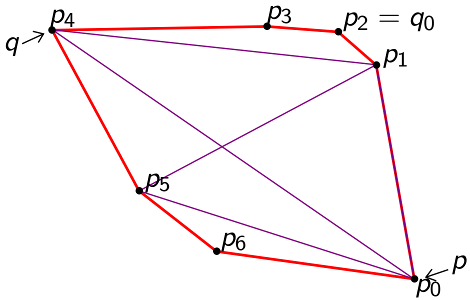
while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**
$q = q$.next;
if $(p, q) \neq (q_0, p_0)$ **then**
pairs.add(p,q);

Algorithm



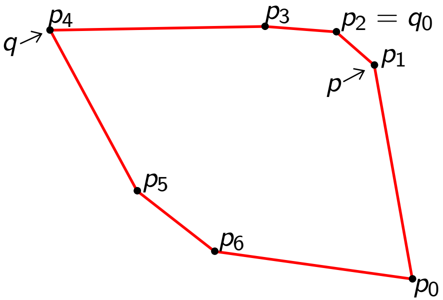
while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**
    $q = q$.next;
    if $(p, q) \neq (q_0, p_0)$ **then**
        pairs.add(p,q);

Algorithm



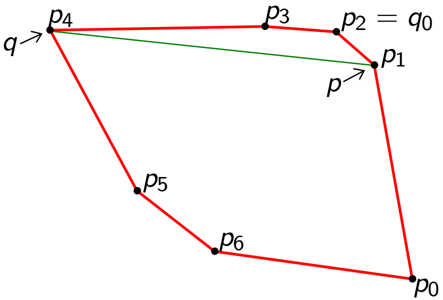
while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**
$q = q$.next;
if $(p, q) \neq (q_0, p_0)$ **then**
pairs.add(p,q);

Algorithm



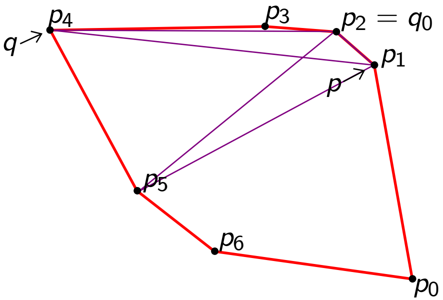
$p = p$.next;

Algorithm



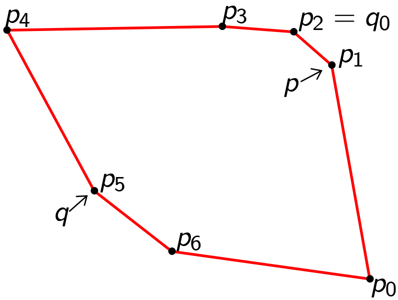
pairs.add(p,q);

Algorithm



while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**
    $q = q$.next;
    if $(p, q) \neq (q_0, p_0)$ **then**
        pairs.add(p,q);

Algorithm

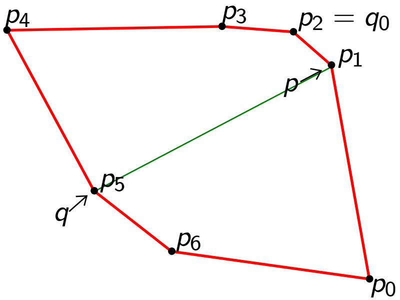


while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**

  $q = q$.next;

  if $(p, q) \neq (q_0, p_0)$ **then**

    pairs.add(p,q);



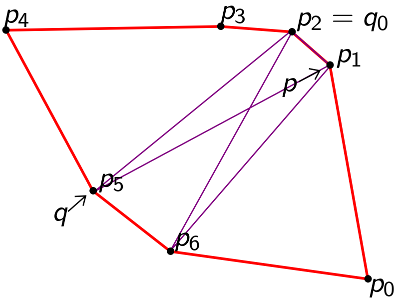
while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**

  $q = q$.next;

  if $(p, q) \neq (q_0, p_0)$ **then**

    pairs.add(p,q);

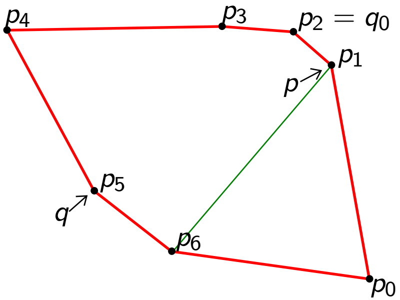
Algorithm



while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**
  // ...

if *area(p, p.next, q.next)* $=$ *area(p, p.next, q)* & $(p, q) \neq (q_0, p_n)$ **then**
  pairs.add(p,q.next);

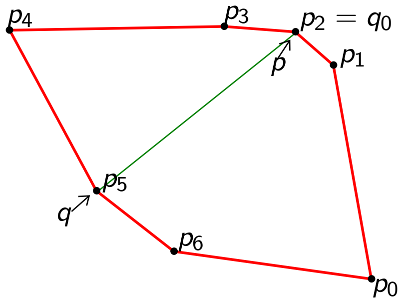
Algorithm



if $area(p,\ p.next,\ q.next) = area(p,\ p.next,\ q)\ \&\ (p, q) \neq (q_0, p_n)$ **then**

 $pairs$.add(p,q.next);

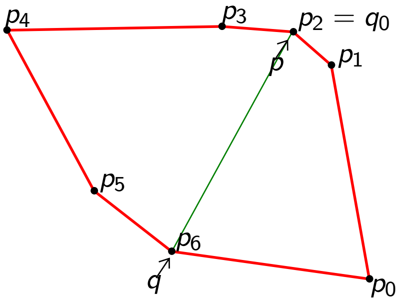
Algorithm



$p = p$.next;

$pairs$.add(p,q);

Algorithm

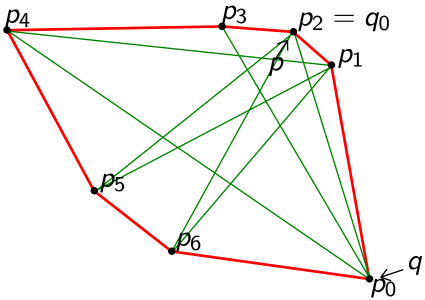


while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**

  $q = q$.next;

  if $(p, q) \neq (q_0, p_0)$ **then**

    pairs.add(p,q);



while $q \neq p_0$ **do** main loop
  // ...

Algorithm

$p = p_n$;

$q = p$.next;

while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**

  $q = q$.next;

$q_0 = q$;

while $q \neq p_0$ **do**

  $p = p$.next;

  pairs.add(p,q);

  while *area(p, p.next, q.next)* $>$ *area(p, p.next, q)* **do**

    $q = q$.next;

    if $(p, q) \neq (q_0, p_0)$ **then**

      pairs.add(p,q);

  if *area(p, p.next, q.next)* $=$ *area(p, p.next, q)* & $(p, q) \neq (q_0, p_n)$ **then**

    pairs.add(p,q.next);

Contents

Final notes and summary

- We can find diameter of a set of points in $O(n \log n)$.
  - using convex hull and filtering points that do not form antipodal pairs

Final notes and summary

- We can find diameter of a set of points in $O(n \log n)$.
  - using convex hull and filtering points that do not form antipodal pairs
- The algorithm was shown in 2D only.
- What about more dimensions?
  - we can find convex hull in 3D
  - antipodal points can be defined in 3D as well
  - but ...

Final notes and summary

- We can find diameter of a set of points in $O(n \log n)$.
  - using convex hull and filtering points that do not form antipodal pairs
- The algorithm was shown in 2D only.
- What about more dimensions?
  - we can find convex hull in 3D
  - antipodal points can be defined in 3D as well
  - but ...
  - number of antipodal pairs in 3D is $O(N^2)$
  - there is a lot of computation involved
  - brute force will most likely be faster here ☹

References

- Franco P. Preparata, Michael Ian Shamos, *Computational Geometry: An Introduction*. Springer-Verlag, New York, 2nd Edition, 1988
- Grégoire Malandain, Jean-Daniel Boissonnat, *Computing the Diameter of a Point Set*. INRIA - Institut Natianal de Recherche en Informatique et en Automatique, July 27, 2001

Questions?

Questions?
Thank you for your attention.

OI-OPPA. European Social Fund

Prague & EU: We invest in your future.