Introduction to Algorithms

Meng-Tsung Tsai

10/03/2019

Course Materials

Textbook

Introduction to Algorithms (I2A) 3rd ed. by Cormen, Leiserson, Rivest, and Stein.

Reference Book

Algorithms (JfA) 1st ed. by Erickson. An e-copy can be downloaded from author's website: http://jeffe.cs.illinois.edu/teaching/algorithms/

Websites

http://e3new.nctu.edu.tw for slides, written assignments, and solutions.

http://oj.nctu.me for programming assignments.

Office Hours

Lecturer's

On Wednesdays 16:30 - 17:20 at EC 336 (工程三館).

TA. Erh-Hsuan Lu (呂爾軒) and Tsung-Ta Wu (吳宗達)

On Mondays 10:10 - 11:00 at ES 724 (電資大樓).

TA. Yung-Ping Wang (王詠平) and Chien-An Yu (俞建安)

On Thursdays 11:10 - 12:00 at ES 724 (電資大樓).

Announcements

Programming Assignment 1 is due by Oct 9, 23:59. at https://oj.nctu.me

Written Assignment 1 is due by Oct 15, 10:20. You need hand in your writeup in class. No late submissions because the solution will be announced right after the deadline. at https://e3new.nctu.edu.tw

We will not normalize the points that you receive from assignments. 100 points is a perfect score, and extra points are considered as a bonus.

Caution: it is very difficult to solve all problems in an assignment.

A list ...

-A 多段code幾乎完全相同
A 結構和coding sytle非常類似 看起來僅改了一些變數名和賦值順序
幾乎相同 僅加了空行
幾乎相同 僅加了空行
code多處類似
code多處類似 甚至變數名相同
C 幾乎相同 僅改matrix大小
D又在加空行....
-E 幾乎相同

Last Reminder

You will definitely fail this course if you plagiarize someone else's work or *lend* your solution to someone else or cheat in a quiz/exam.

If you do plagiarize on programming assignment #1, write an e-mail to TA to withdraw your (maybe other's) programs by next Wednesday. Next time, you will fail this course immediately ...

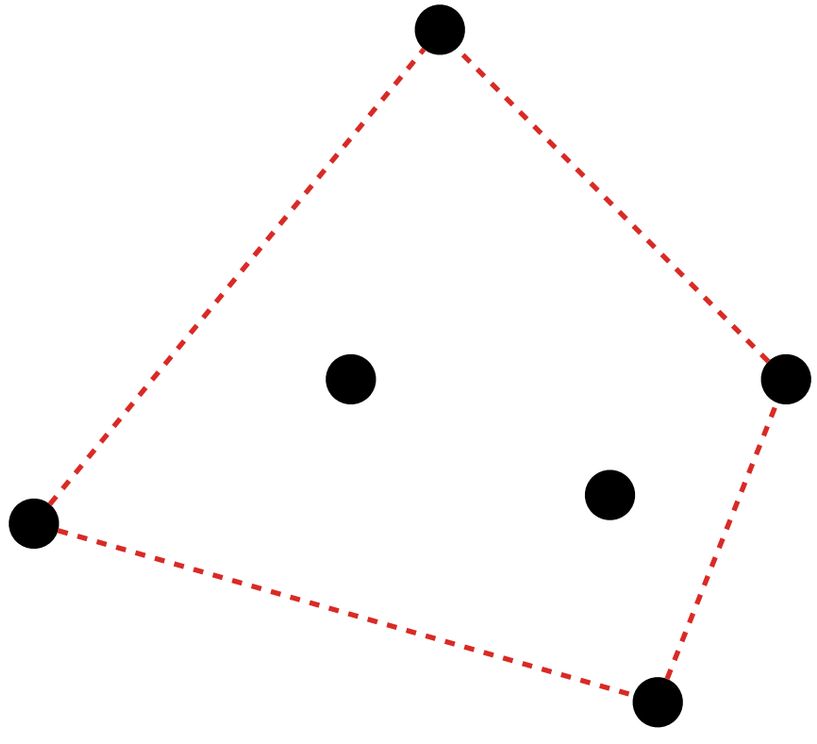
You are encouraged to discuss, but the program/writeup shall be your own. That is, you can solve that problem again without other's help. It is very easy to verify this. Please do not challenge me at this point. Thanks.

Convex Hulls

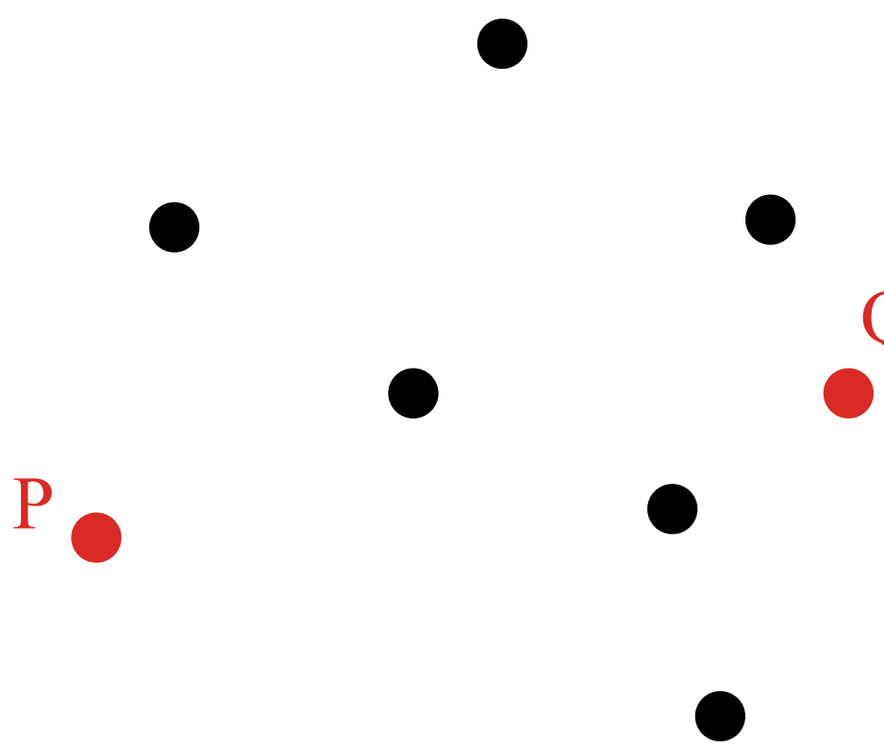
Convex Hulls

Input: a set S of n points in the plane, where $S = \{p_1, p_2, \ldots, p_n\}$.

Output: the *convex hull* of S, i.e. the smallest convex polygon that contains the given point set.

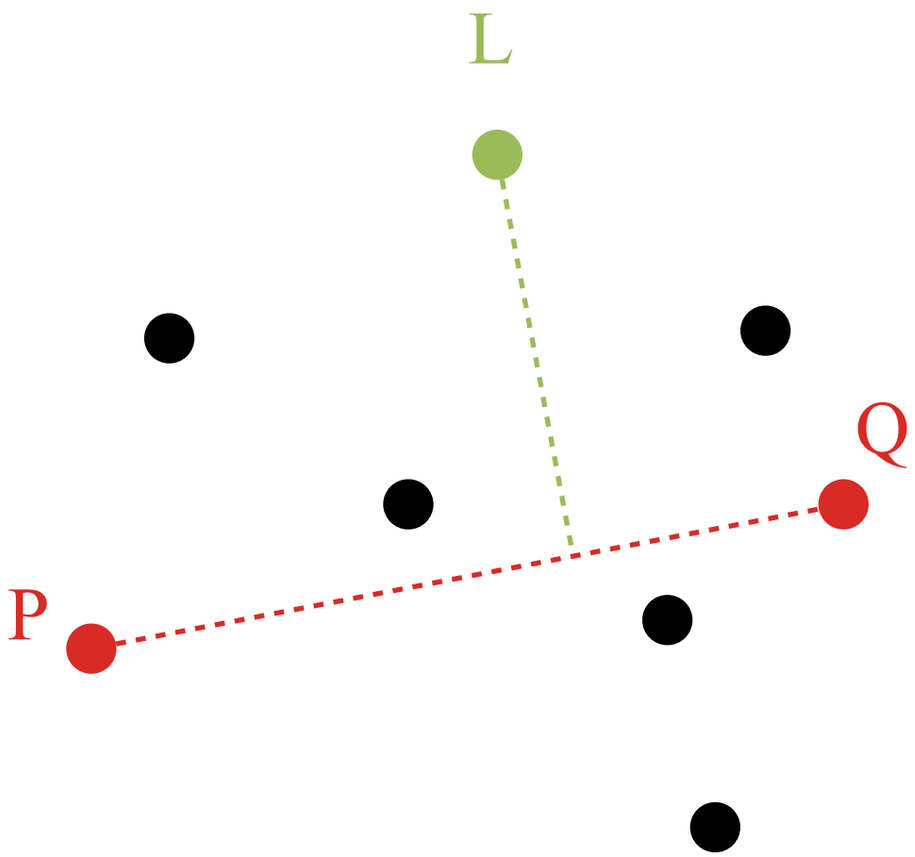


Quick Hulls



Step 1. The point with the largest and the smallest x coordinate must be on the convex hull, say P and Q -- tie breaking by y coordinates.

Quick Hulls

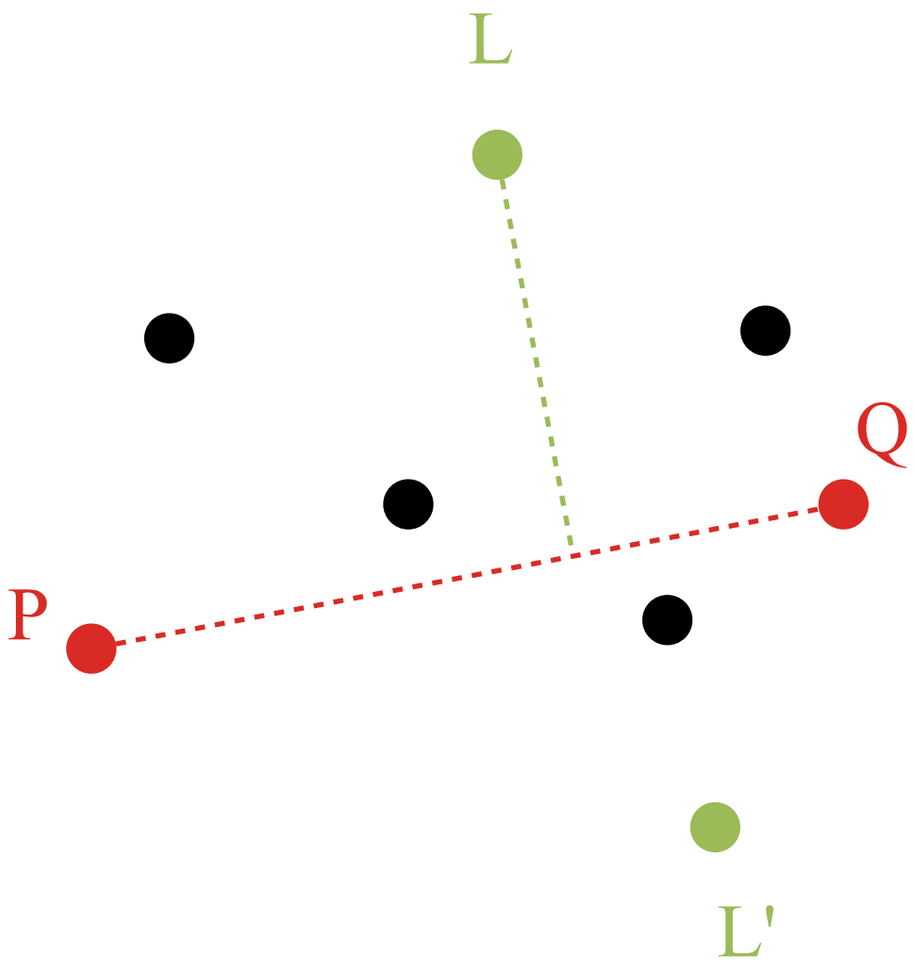


Step 1. The point with the largest and the smallest x coordinate must be on the convex hull, say P and Q -- tie breaking by y coordinates.

Define ℓ(P, Q) to be the leftmost extreme point to vector $\overrightarrow{PQ}$.

Clearly, L = ℓ(P, Q) and L must be on the convex hull.

Quick Hulls



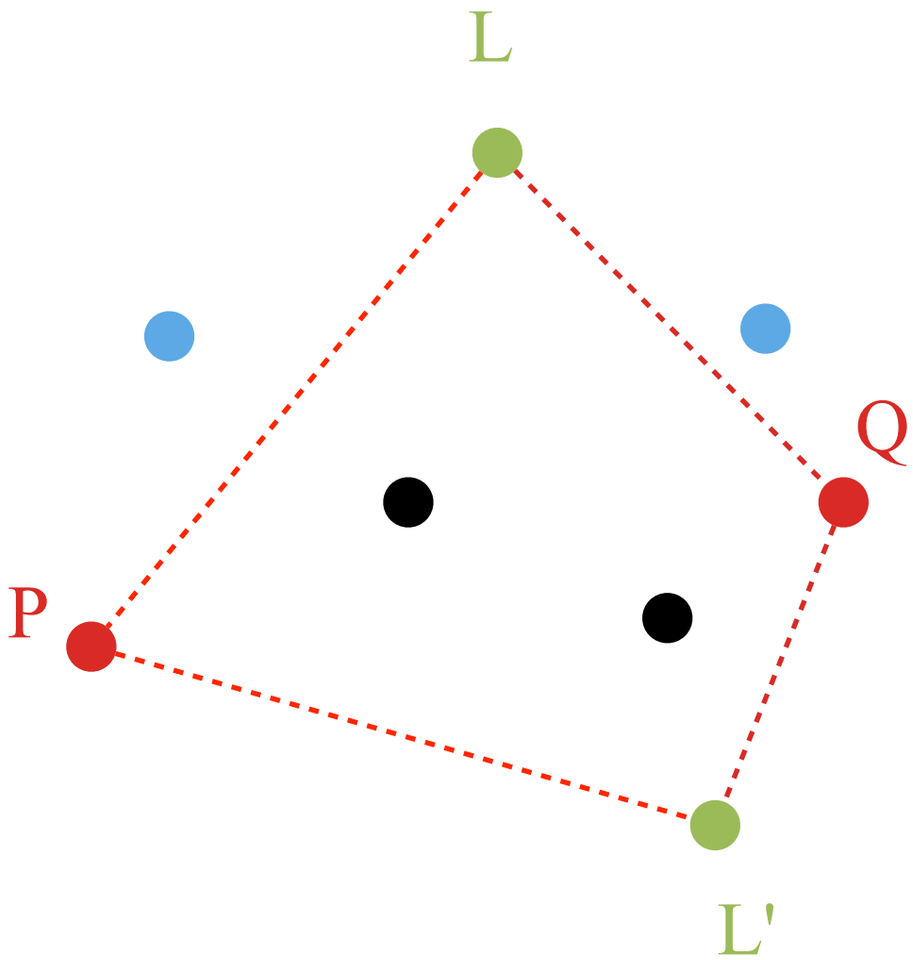
Step 1. The point with the largest and the smallest x coordinate must be on the convex hull, say P and Q -- tie breaking by y coordinates.

Define ℓ(P, Q) to be the leftmost extreme point to vector $\overrightarrow{PQ}$.

Clearly, L = ℓ(P, Q) and L must be on the convex hull.

Step 2. Find ℓ(P, Q) and ℓ(Q, P).

Quick Hulls



Step 1. The point with the largest and the smallest x coordinate must be on the convex hull, say P and Q -- tie breaking by y coordinates.

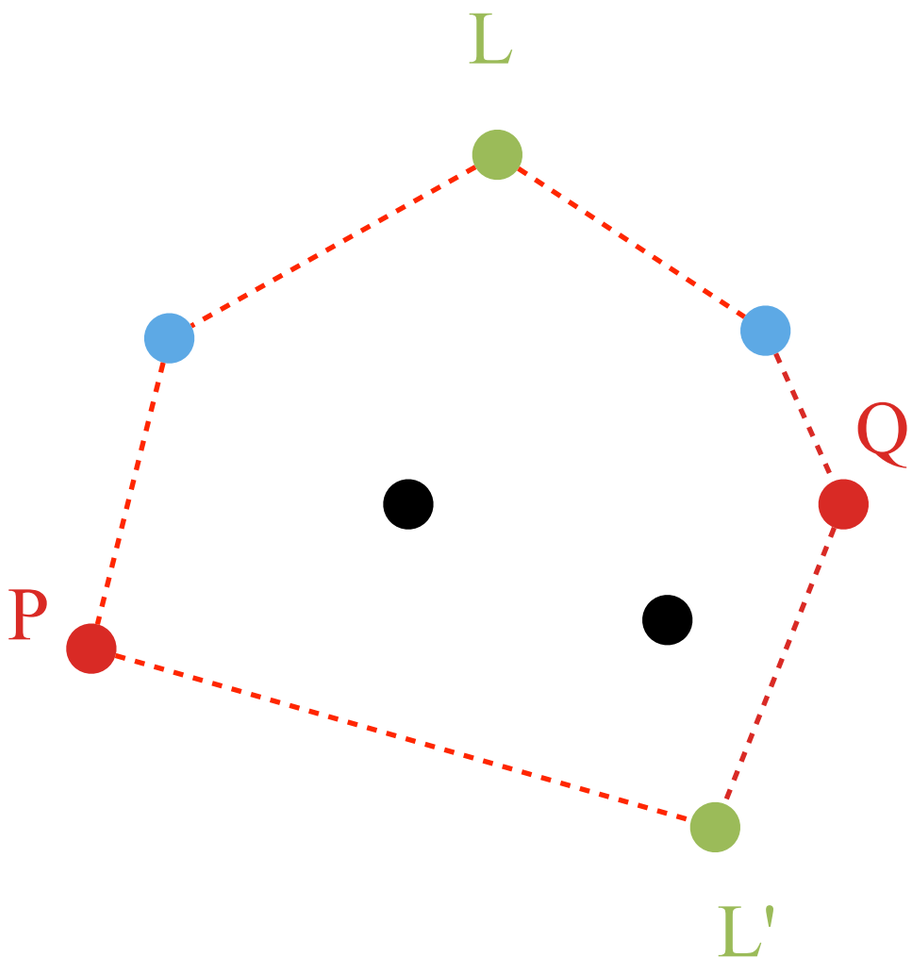
Define ℓ(P, Q) to be the leftmost extreme point to vector $\overrightarrow{PQ}$.

Clearly, L = ℓ(P, Q) and L must be on the convex hull.

Step 2. Find ℓ(P, Q) and ℓ(Q, P).

Step 3. Find ℓ(P, L), ℓ(L, Q), ℓ(Q, L'), ℓ(L', P).

Quick Hulls



Step 1. The point with the largest and the smallest x coordinate must be on the convex hull, say P and Q -- tie breaking by y coordinates.

Define ℓ(P, Q) to be the leftmost extreme point to vector $\overrightarrow{PQ}$.

Clearly, L = ℓ(P, Q) and L must be on the convex hull.

Step 2. Find ℓ(P, Q) and ℓ(Q, P).

Step 3. Find ℓ(P, L), ℓ(L, Q), ℓ(Q, L'), ℓ(L', P) -- iterate this procedure until no new point can be found.

Running Time

The running time is O(nh) where h is the number of vertices on the convex hull. (Why?)

There are many algorithms that can compute convex hulls in O(n log n) time, for example Graham scan. We will not cover the details in the course. Please refer to the textbook.

--- more discussions ---

Some people claim that Quick Hulls runs as fast as O(n log n) in practice. It is not easy to construct a point set with bounded-precision coordinates that makes Quick Hulls runs slowly.

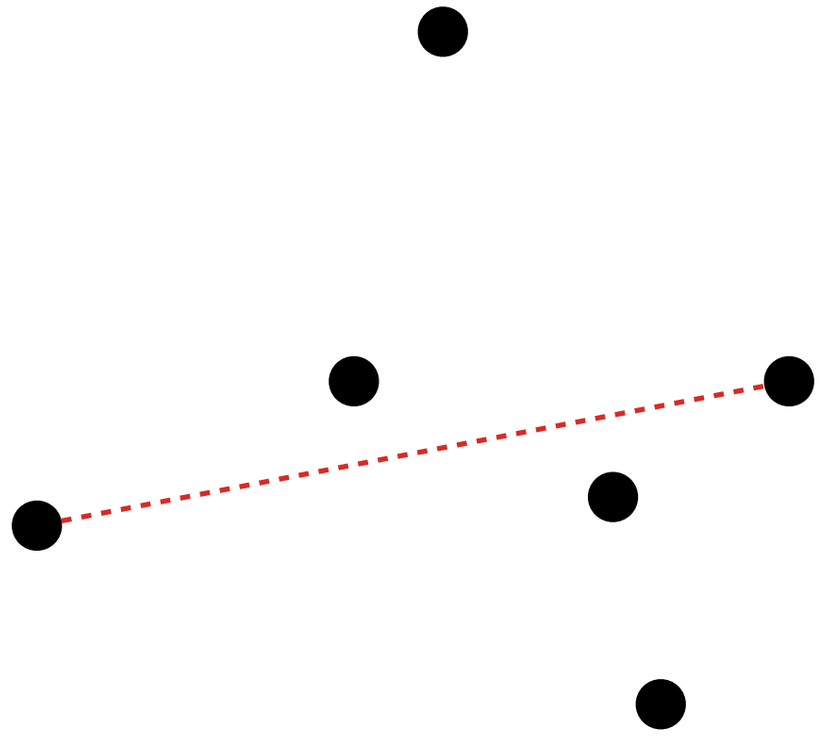
Farthest Pair

The Farthest Pair of Points

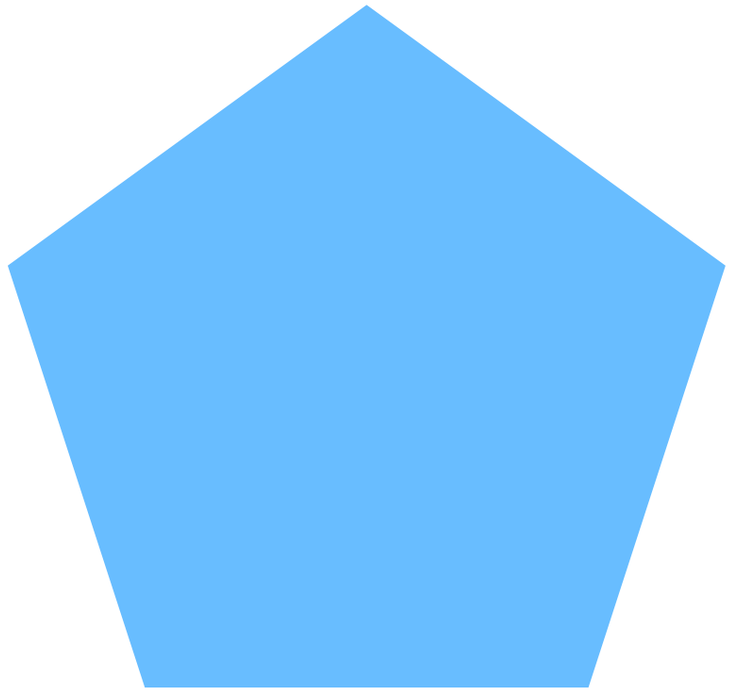
Input: a set S of n points in the plane, where S = $\{p_1, p_2, \ldots, p_n\}$.

Output: a pair (p_i, p_j) whose Euclidean distance is maximum among all point pairs.

Example.

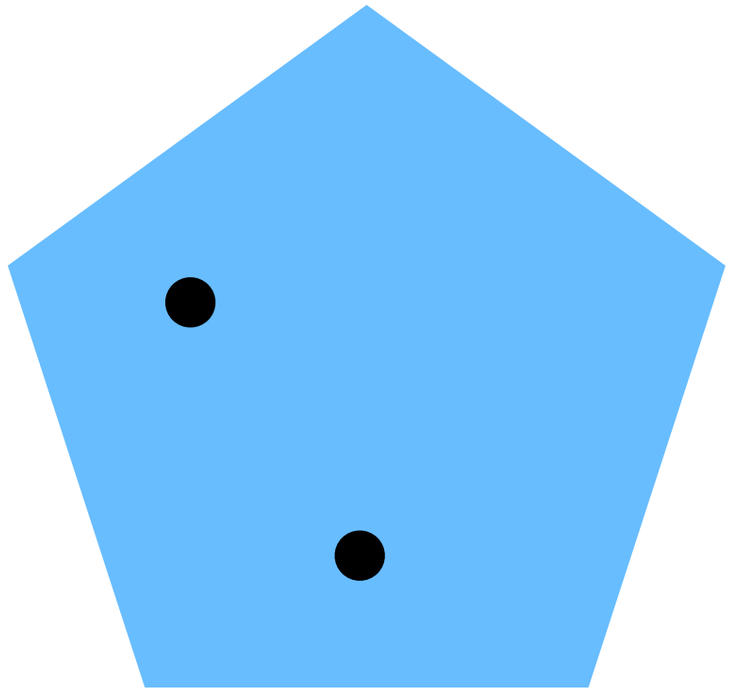


The Farthest Pair of Points



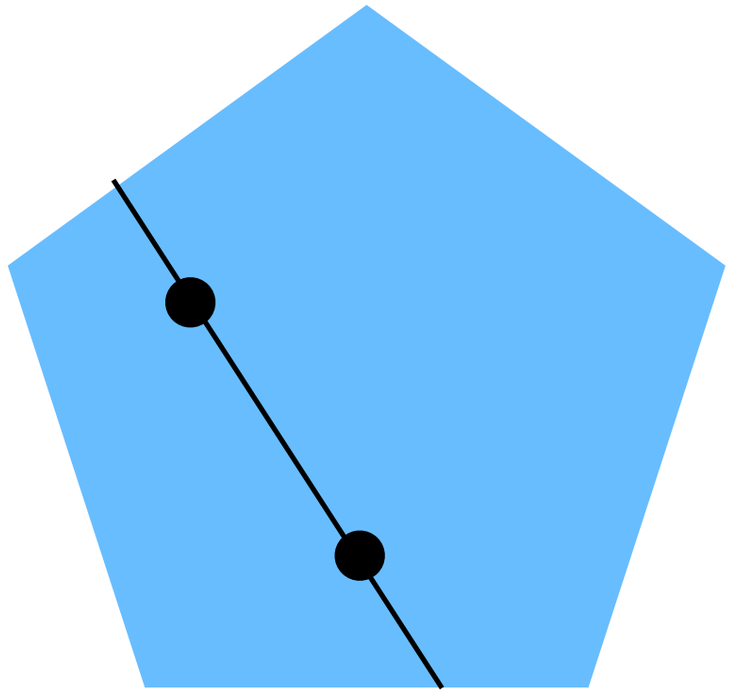
What is the farthest pair of points in a convex polygon?

The Farthest Pair of Points



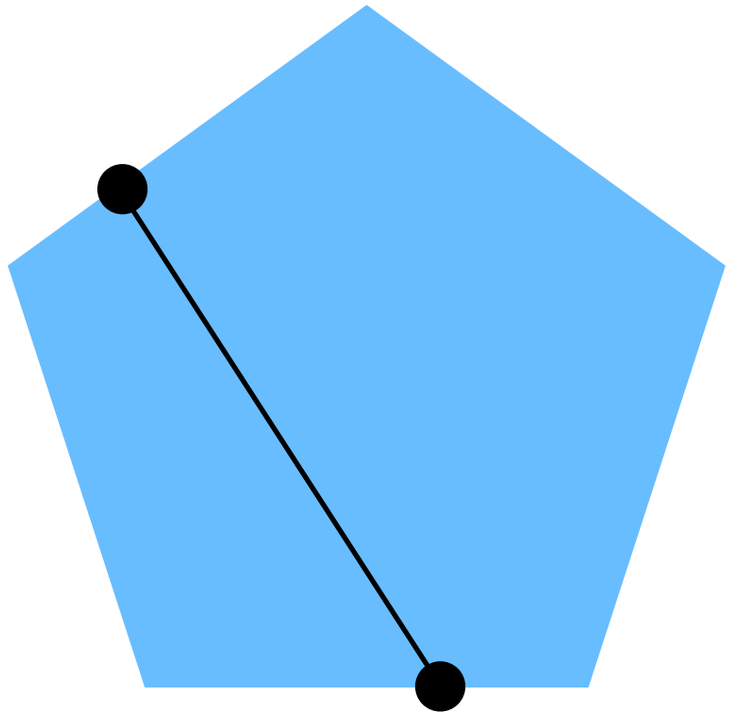
Can the farthest pair comprise some interior points?

The Farthest Pair of Points



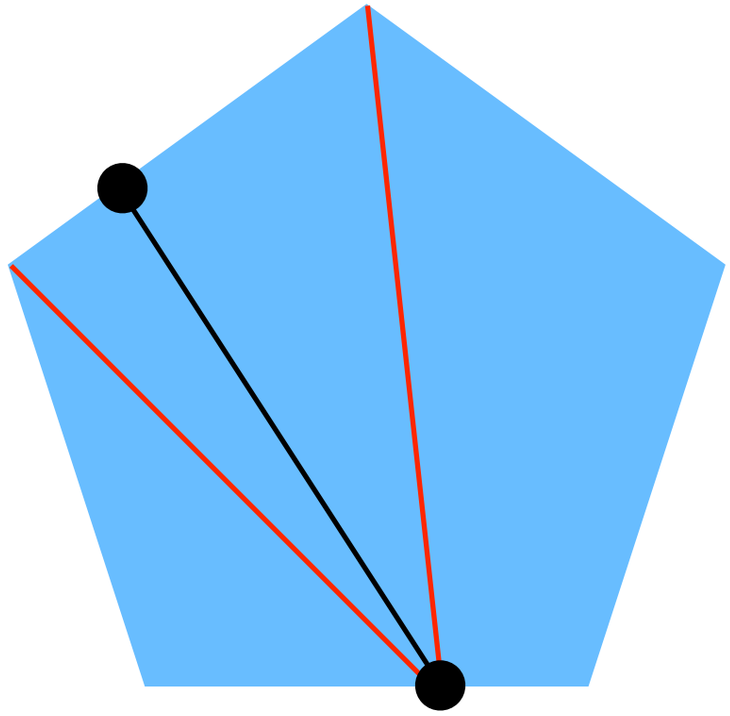
One can draw a line passing through the farthest pair and intersecting the boundary at two new points. The distance between the boundary points is larger.

The Farthest Pair of Points



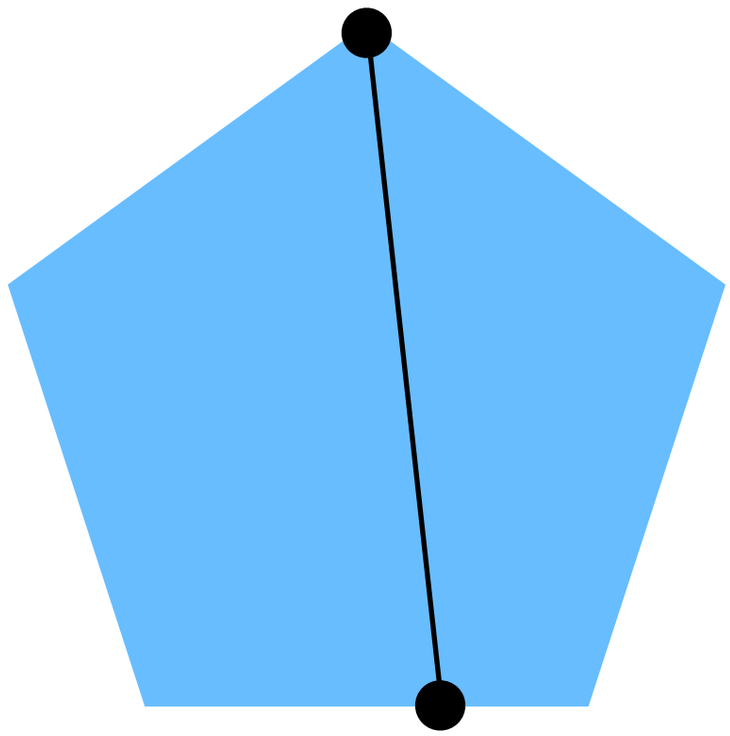
Can the farthest pair comprise points
on the edge of boundary?

The Farthest Pair of Points



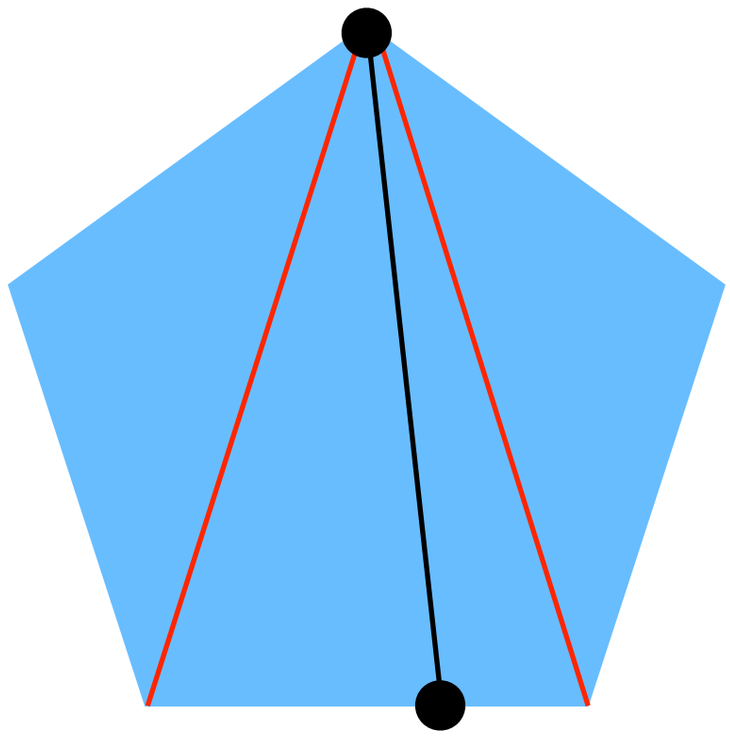
One of the red segments is longer.

The Farthest Pair of Points



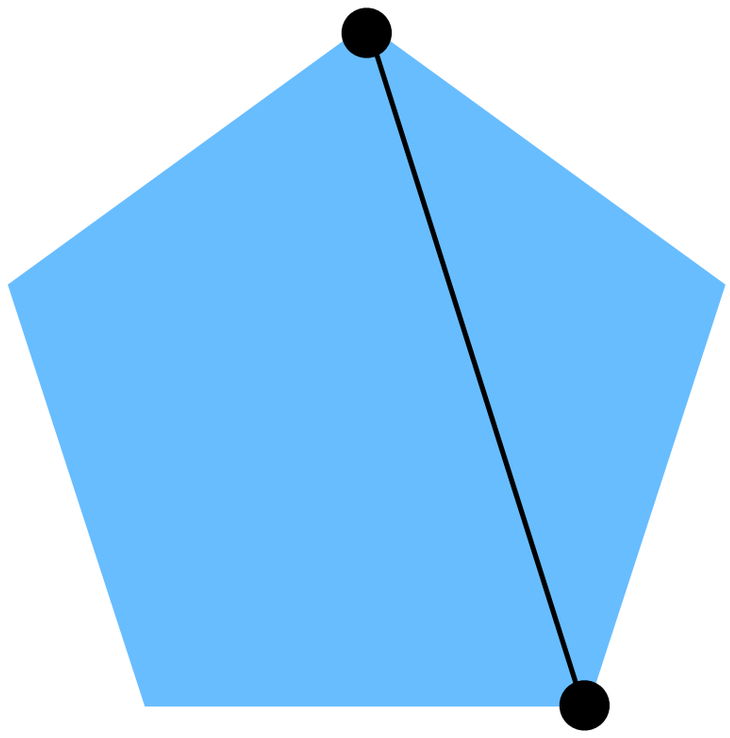
Can the farthest pair comprise points
on the edge of boundary?

The Farthest Pair of Points



One of the red segments is longer.

The Farthest Pair of Points



The farthest pair comprise two points at vertices.

Algorithm to find a farthest pair

Step 1. Compute the convex hull of the n given points.

Step 2. Find a farthest pair among vertices of the convex hulls.

This algorithm has running time $O(n \log n) + O(h^2)$ and can be as worse as $O(n^2)$.

Can we do better?

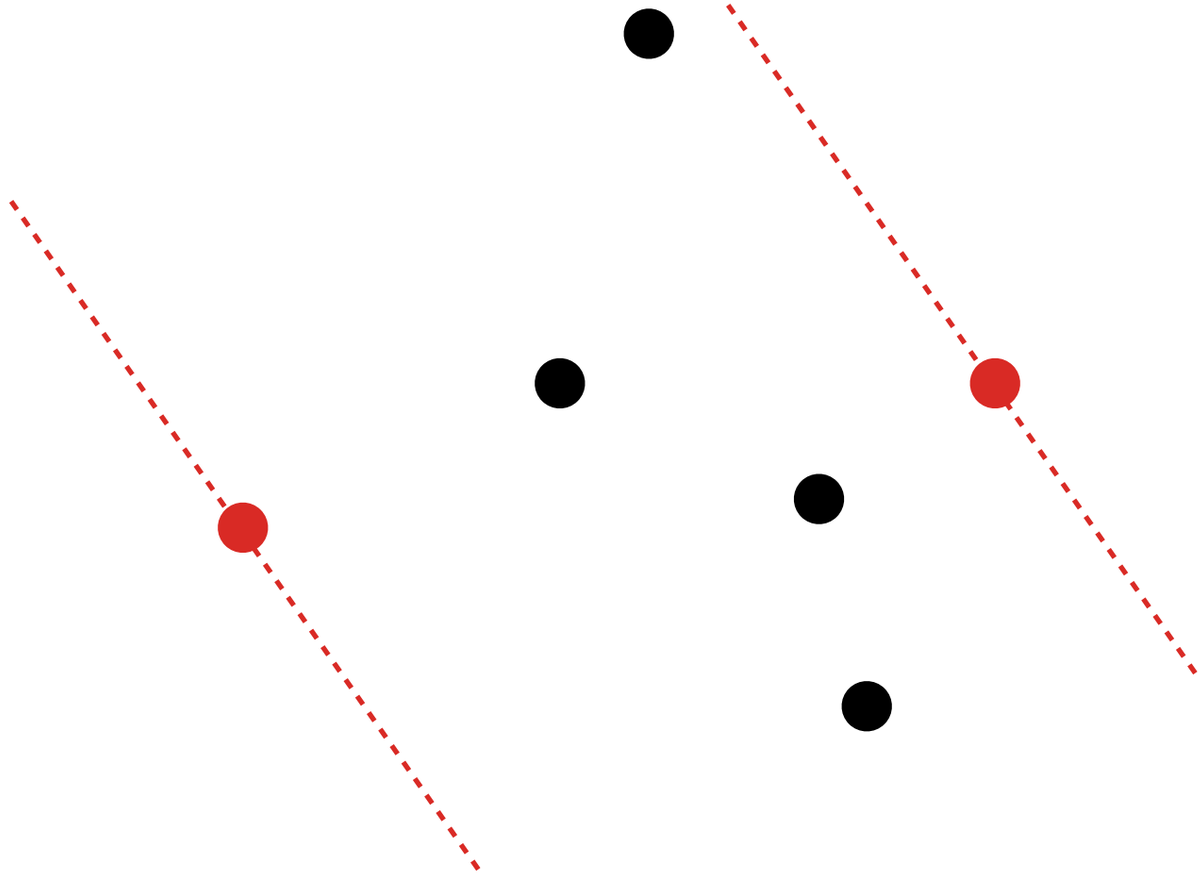
Antipodal Pair

Antipodal Pair

Input: a set S of n points in the plane, where S = $\{p_1, p_2, \ldots, p_n\}$.

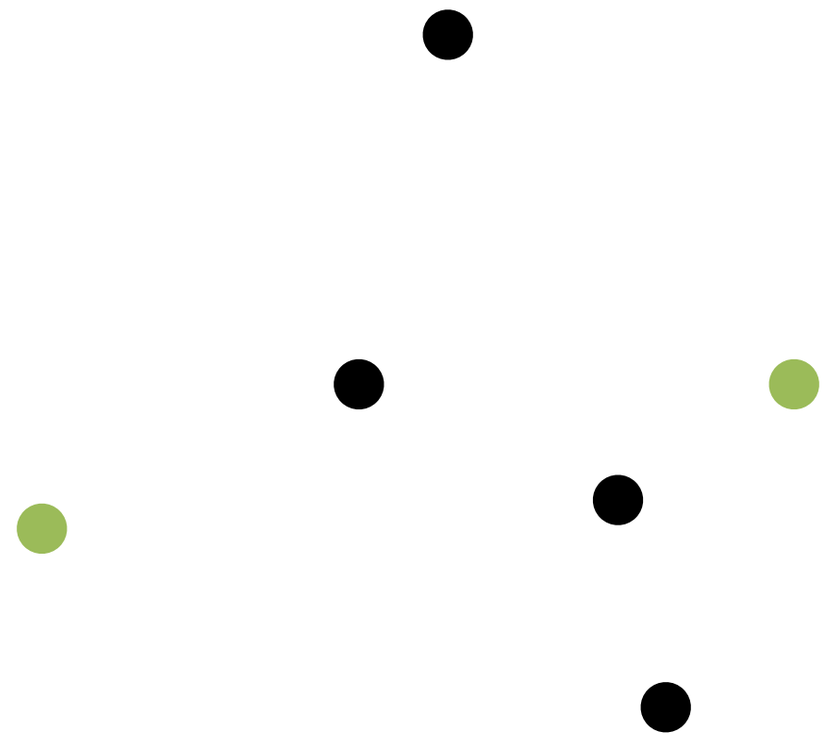
Output: an *antipodal pair* (p_i, p_j), i.e. you can draw two parallel lines passing through the pair so that all points are in-between.

Example.



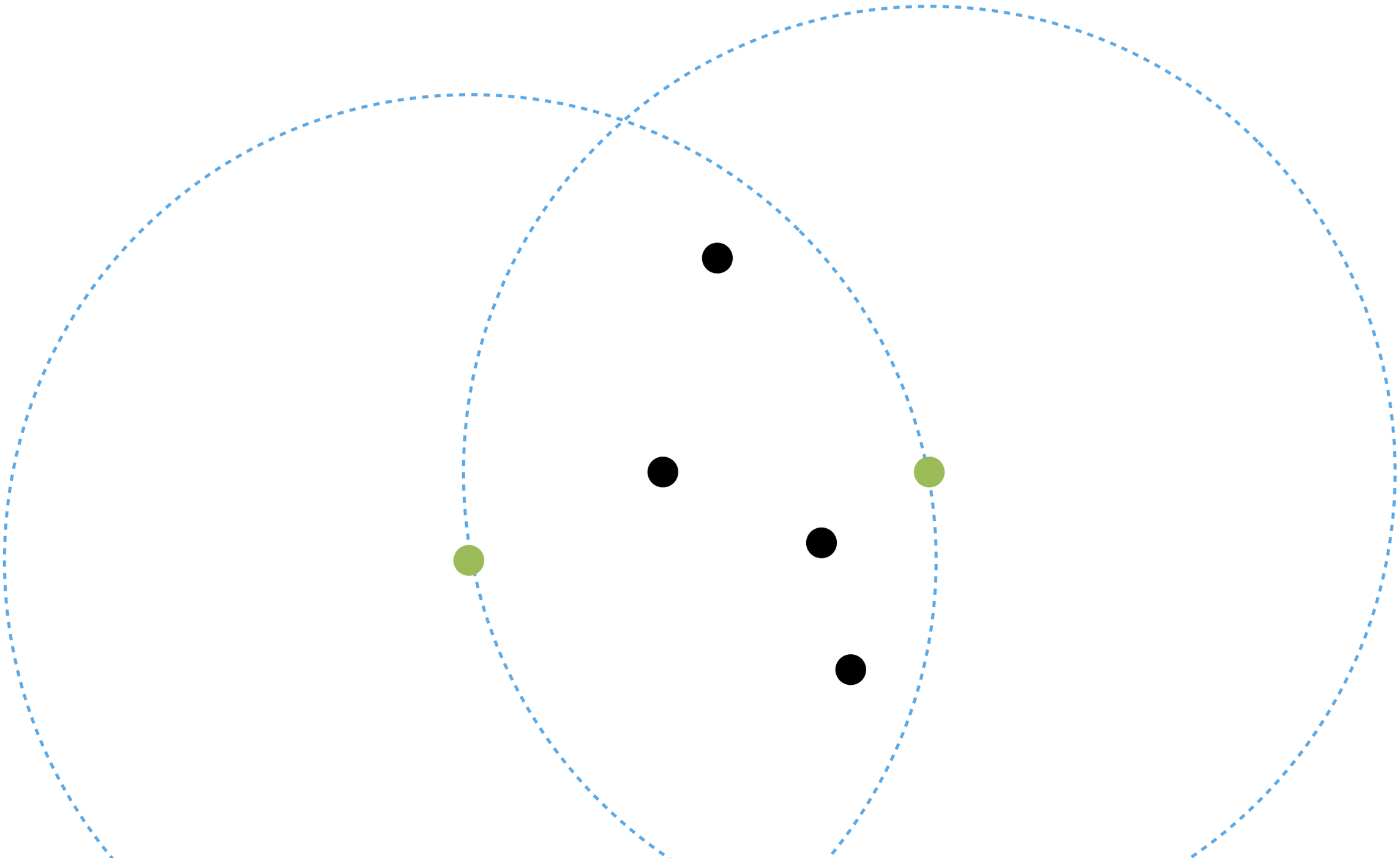
Claim 1

Any farthest pair must be an antipodal pair.



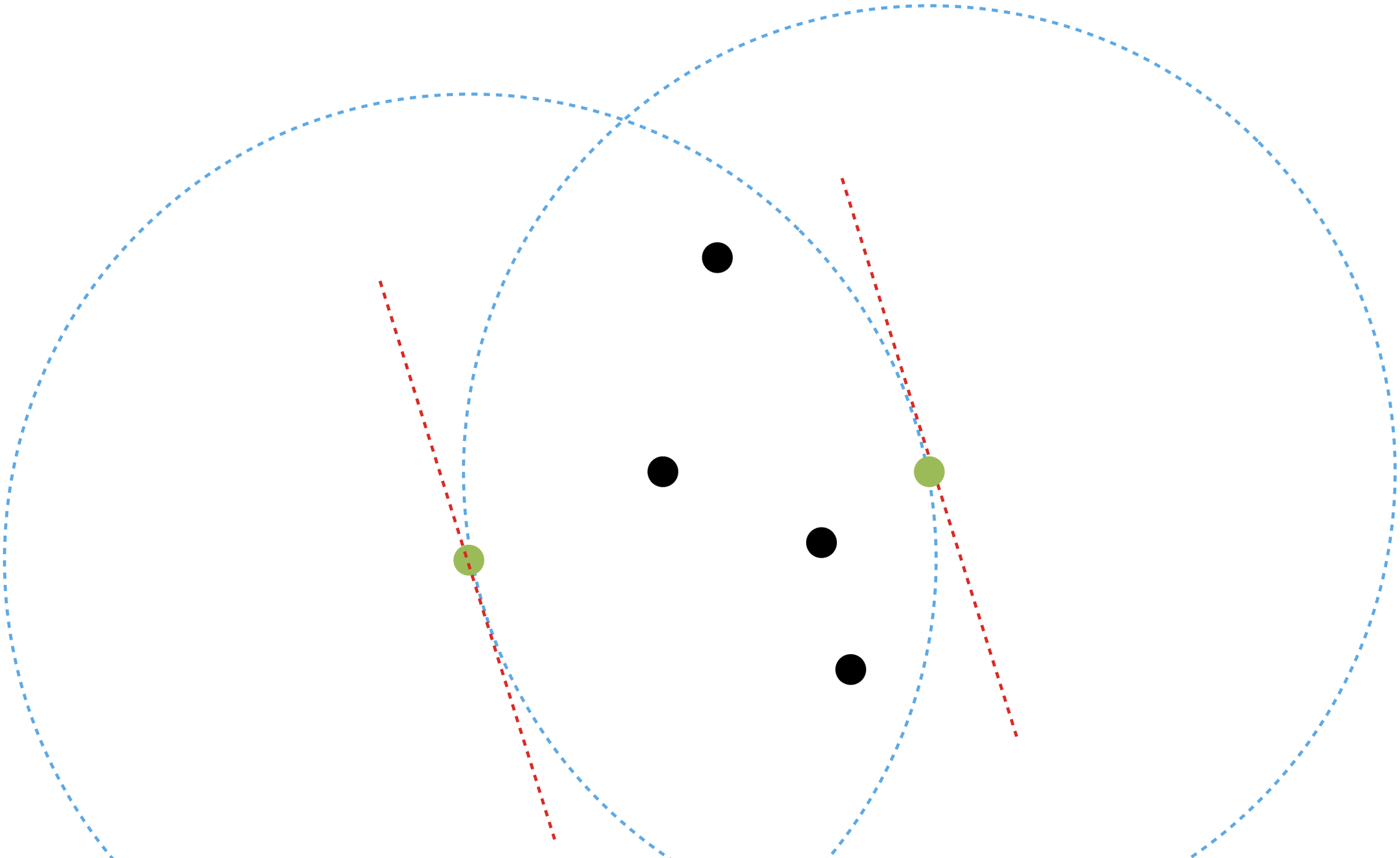
Proof of Claim 1

Any farthest pair must be an antipodal pair.



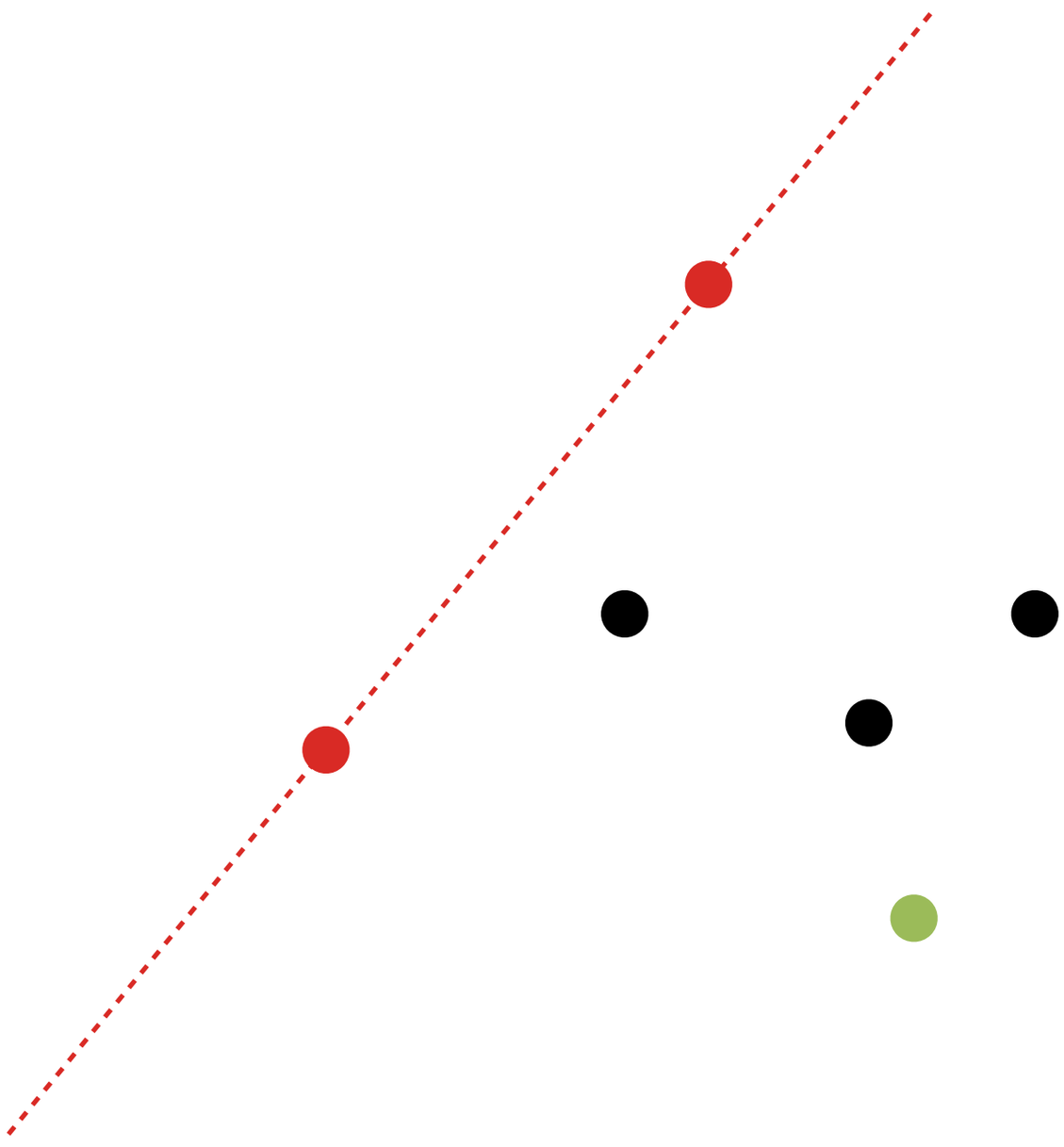
Proof of Claim 1

Any farthest pair must be an antipodal pair.



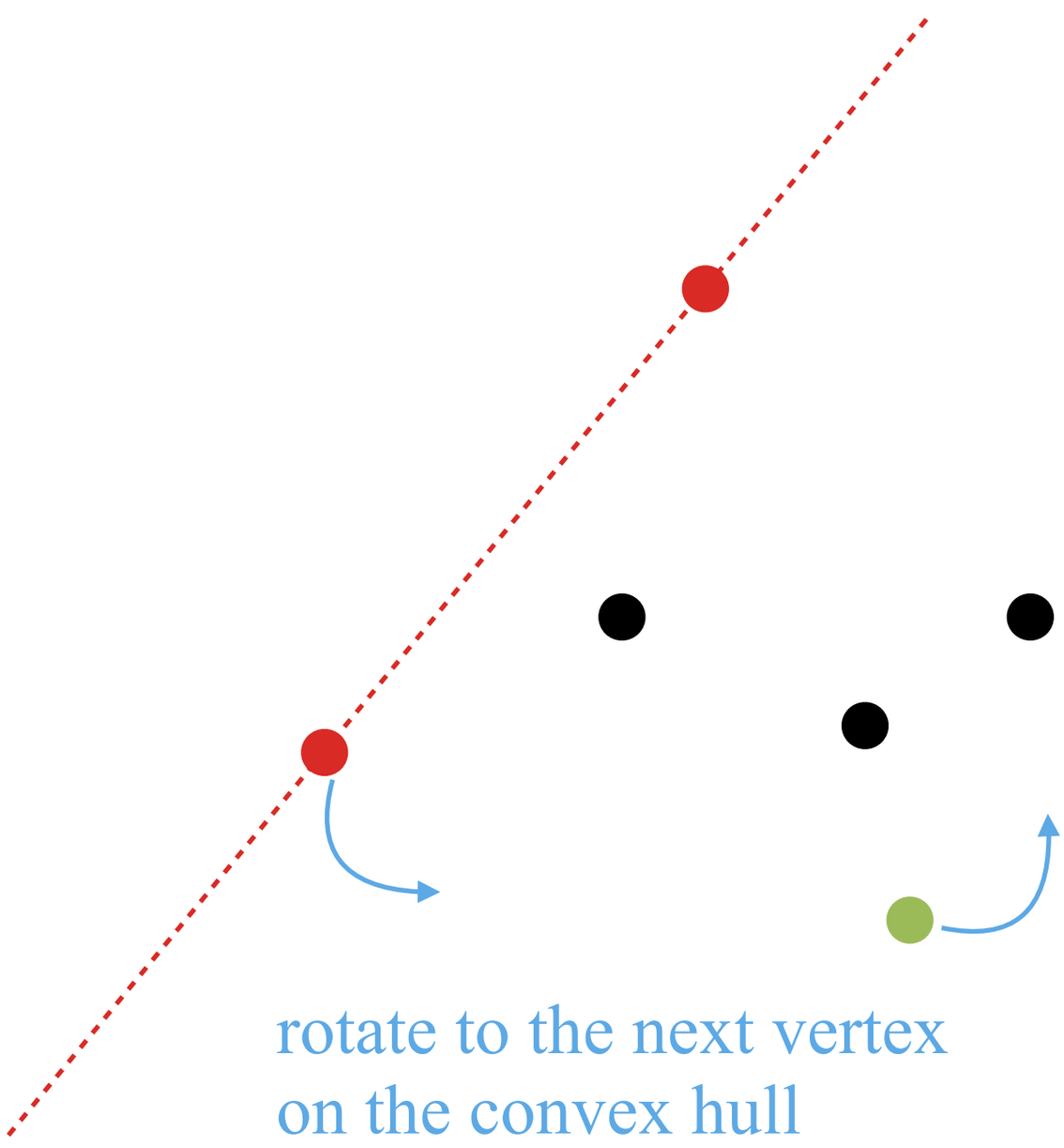
Claim 2

There are O(n) antipodal pairs for a point set without 3 points colinear, and they can be enumerated in O(n) time.



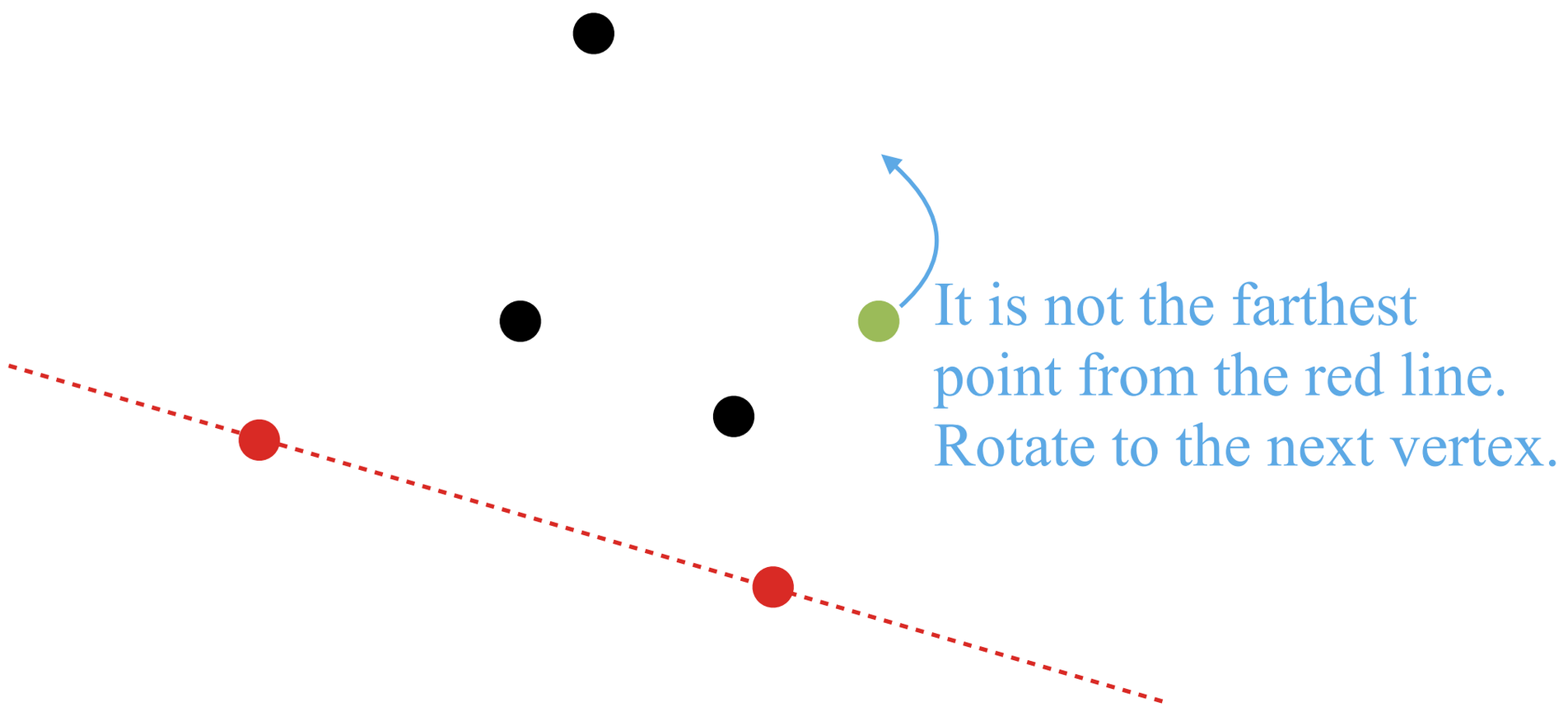
Proof of Claim 2 -- Rotating Calipers

There are O(n) antipodal pairs for a point set without 3 points colinear, and they can be enumerated in O(n) time.



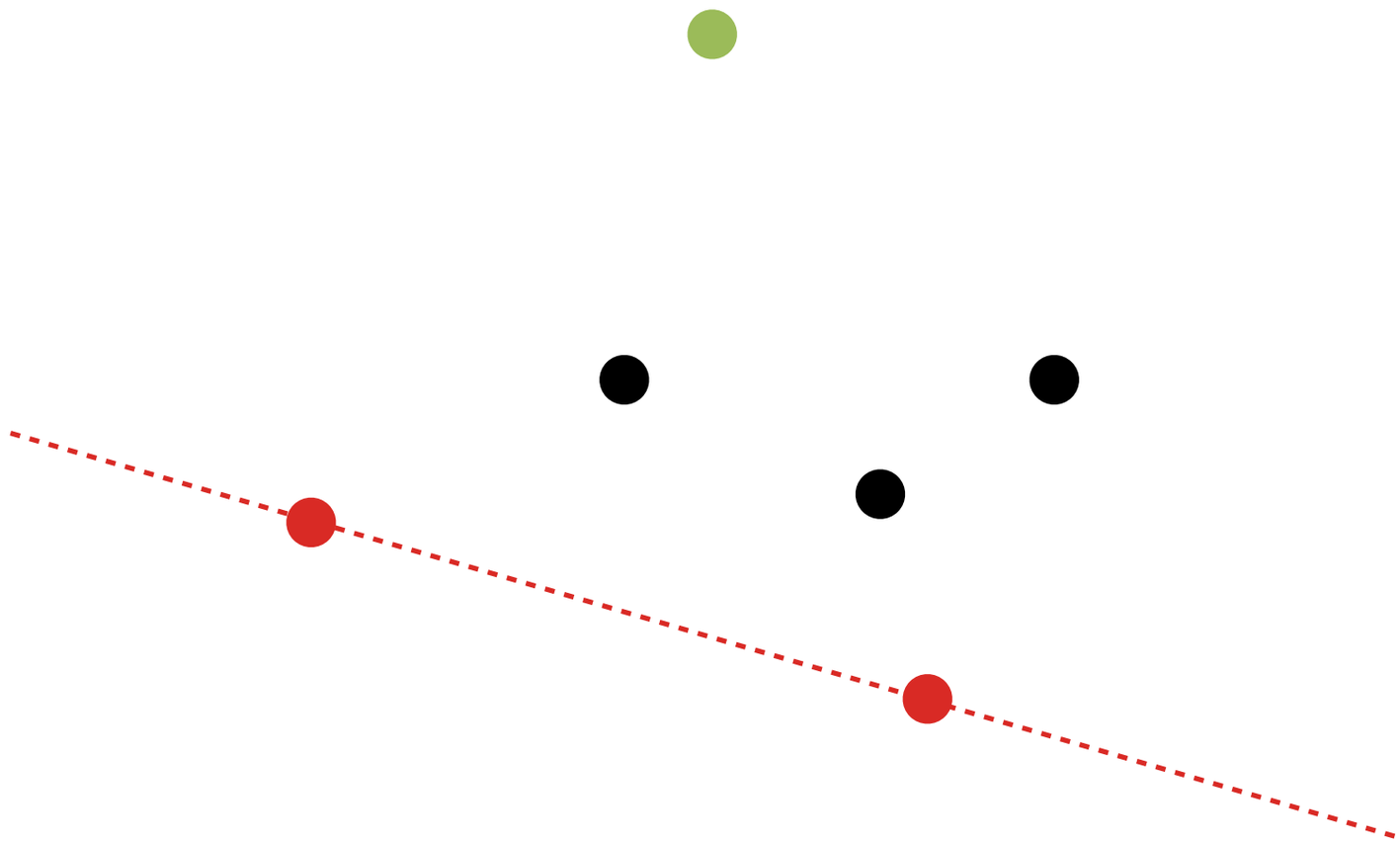
Claim -- Rotating Calipers

There are O(n) antipodal pairs for a point set without 3 points colinear, and they can be enumerated in O(n) time.



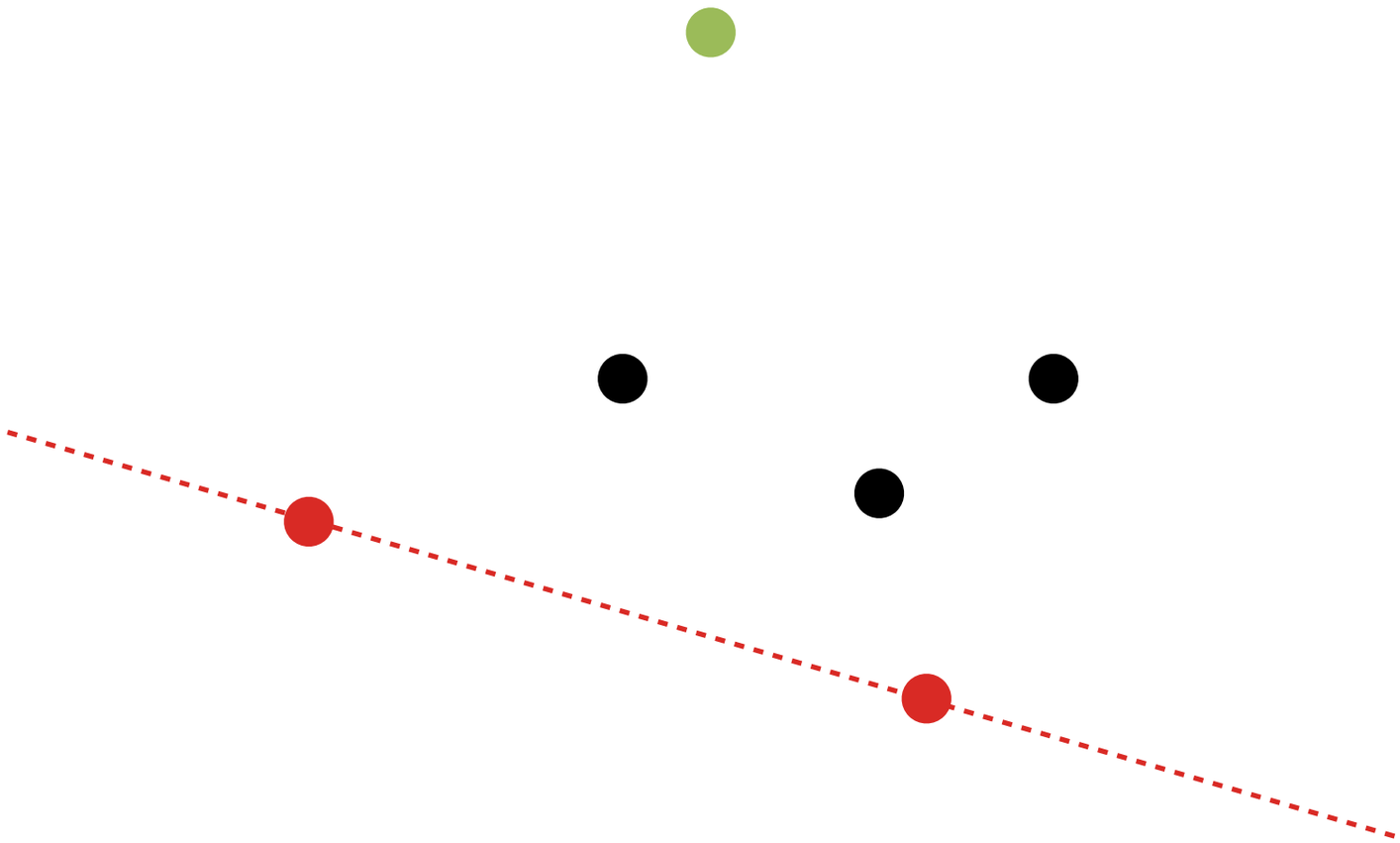
Claim -- Rotating Calipers

There are O(n) antipodal pairs for a point set without 3 points colinear, and they can be enumerated in O(n) time.



Running Time

The red line will iterate over the edges on the convex hull and the green point will iterate over the vertices on the convex hull. The total running time is bounded by O(h) = O(n).



Algorithm to find a farthest pairs

Step 1. Compute the convex hull of the n given points.

Step 2. Enumerate all (i.e. O(n)) antipodal pairs, and one of them is the farthest pair.

This algorithm has running time O(n log n) + O(n) = O(n).

Exercise

Input: a set of n points in the plane.

Output: a rectangle that contains all the points, a.k.a. *the minimum enclosing rectangle*.