Smallest enclosing circles and more

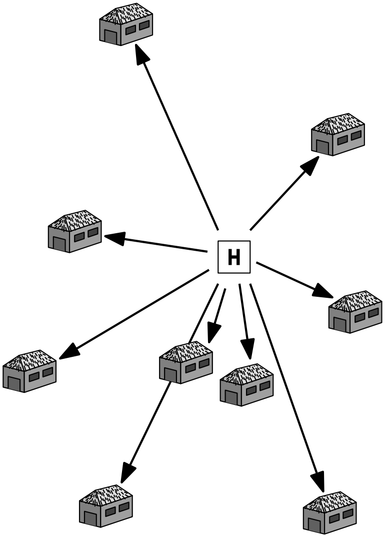
Computational Geometry

Lecture 6: Smallest enclosing circles and more

Facility location

Given a set of houses and farms in an isolated area. Can we place a helicopter ambulance post so that each house and farm can be reached within 15 minutes?

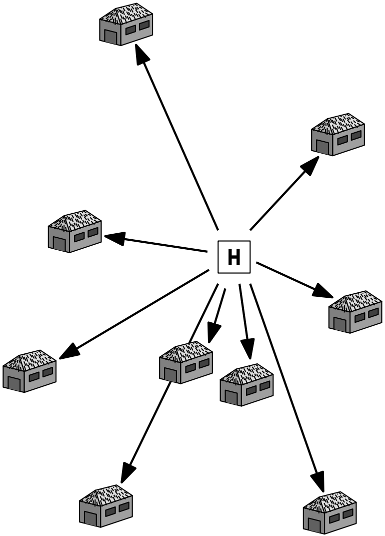
Where should we place an antenna so that a number of locations have maximum reception?



Facility location in geometric terms

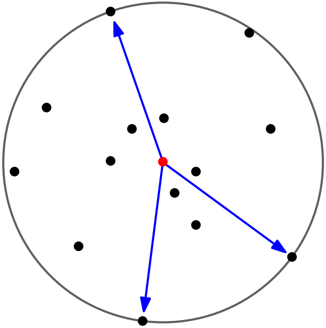
Given a set of points in the plane. Is there any point that is within a certain distance of these points?

Where do we place a point that minimizes the maximum distance to a set of points?



Facility location in geometric terms

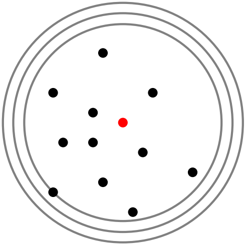
Given a set of points in the plane, compute the smallest enclosing circle

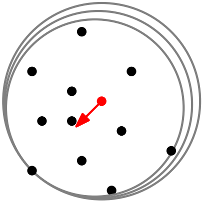


Smallest enclosing circle

Observation: It must pass through some points, or else it cannot be smallest

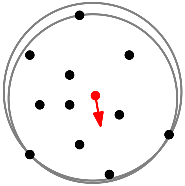
- Take any circle that encloses the points, and reduce its radius until it contains a point p
- Move center towards p while reducing the radius further, until the circle contains another point q

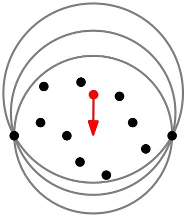




Smallest enclosing circle

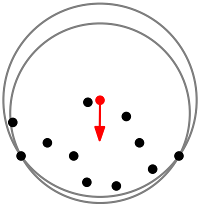
- Move center on the bisector of p and q towards their midpoint, until:
  (i) the circle contains a third point, or
  (ii) the center reaches the midpoint of p and q





Smallest enclosing circle

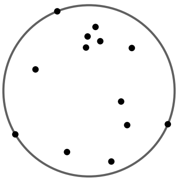
Question: Does the “algorithm” of the previous slide work?

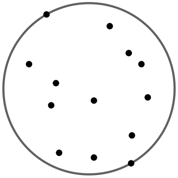


Smallest enclosing circle

Observe: A smallest enclosing circle has (at least) three points on its boundary, or only two in which case they are diametrally opposite

Question: What is the extra property when there are three points on the boundary?





Randomized incremental construction

Construction by randomized incremental construction

incremental construction: Add points one by one and maintain the solution so far

randomized: Use a random order to add the points

Adding a point

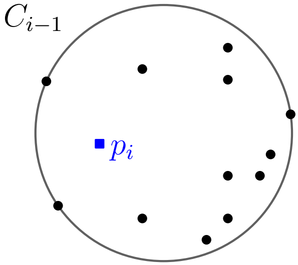
Let $p_1, \ldots, p_n$ be the points in random order

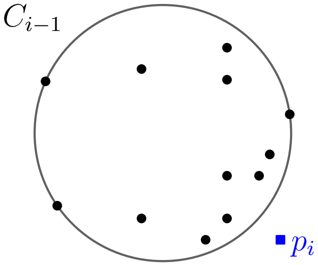
Let C_i be the smallest enclosing circle for $p_1, \ldots, p_i$

Suppose we know C_{i-1} and we want to add p_i

- If p_i is inside C_{i-1}, then $C_i = C_{i-1}$
- If p_i is outside C_{i-1}, then C_i will have p_i on its boundary

Adding a point

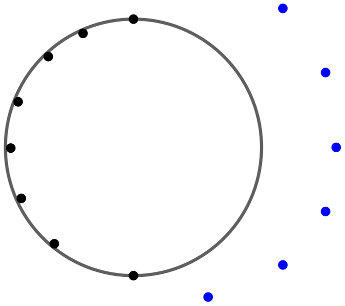




Adding a point

Question: Suppose we remembered not only C_{i-1}, but also the two or three points defining it. It looks like if p_i is outside C_{i-1}, the new circle C_i is defined by p_i and some points that defined C_{i-1}. Why is this false?

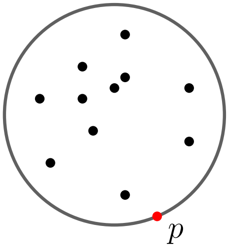
Adding a point



Adding a point

How do we find the smallest enclosing circle of $p_1 \ldots, p_{i-1}$ with p_i on the boundary?

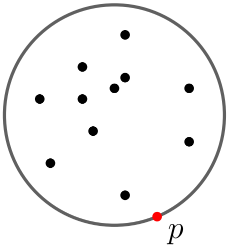
We study the *new(!)* geometric problem of computing the smallest enclosing circle with a given point p on its boundary



Smallest enclosing circle with point

Given a set P of points and one special point p, determine the smallest enclosing circle of P that must have p on the boundary

Question: How do we solve it?



Randomized incremental construction

Construction by randomized incremental construction

incremental construction: Add points one by one and maintain the solution so far

randomized: Use a random order to add the points

Adding a point

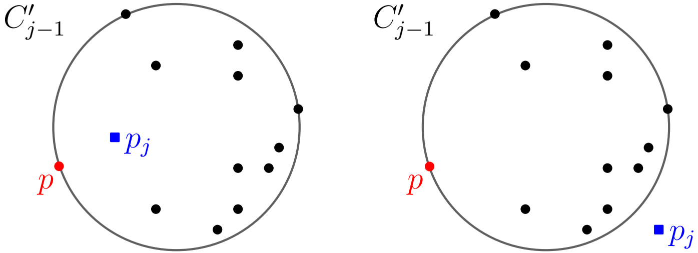
Let $p_1, \ldots, p_{i-1}$ be the points in random order

Let C'_j be the smallest enclosing circle for $p_1, \ldots, p_j$ $(j \leq i-1)$ and with p on the boundary

Suppose we know C'_{j-1} and we want to add p_j

- If p_j is inside C'_{j-1}, then $C'_j = C'_{j-1}$
- If p_j is outside C'_{j-1}, then C'_j will have p_j on its boundary (and also p of course!)

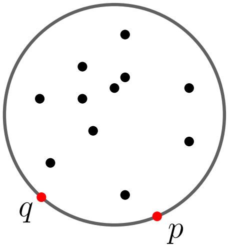
Adding a point



Adding a point

How do we find the smallest enclosing circle of $p_1 \ldots, p_{j-1}$ with p and p_j on the boundary?

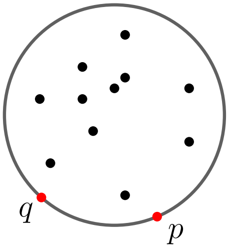
We study the *new(!)* geometric problem of computing the smallest enclosing circle with two given points on its boundary



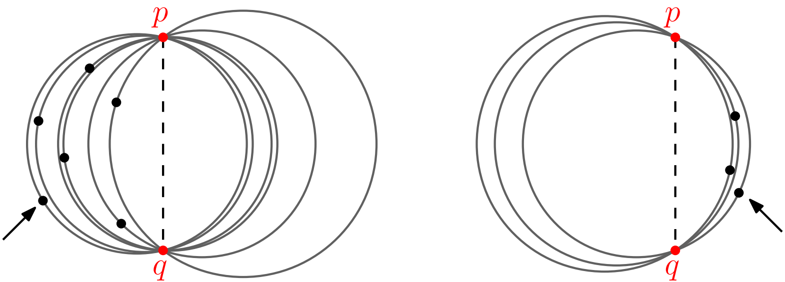
Smallest enclosing circle with two points

Given a set P of points and two special points p and q, determine the smallest enclosing circle of P that must have p and q on the boundary

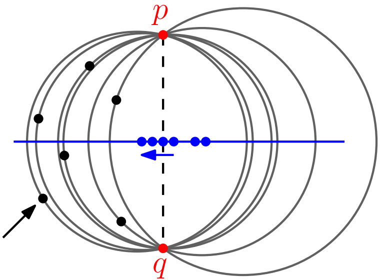
Question: How do we solve it?

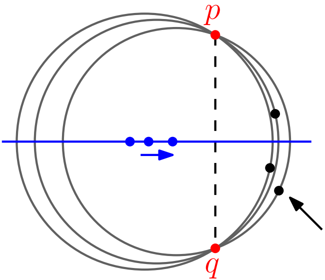


Two points known



Two points known





Algorithm: two points known

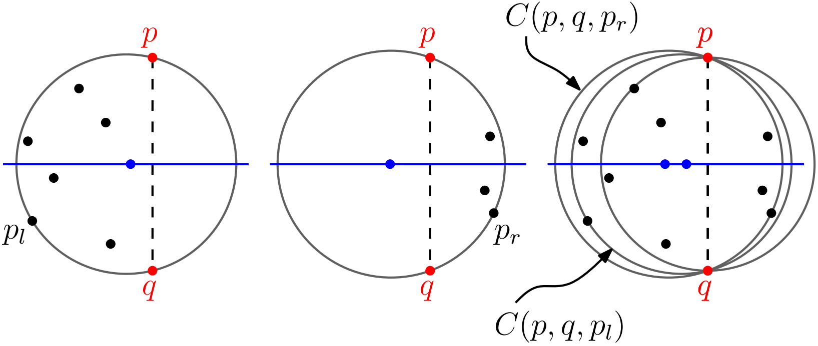
Assume w.lo.g. that p and q lie on a vertical line. Let ℓ be the line through p and q and let ℓ' be their bisector

For all points left of ℓ, find the one that, together with p and q, defines a circle whose center is leftmost $\rightarrow p_l$

For all points right of ℓ, find the one that, together with p and q, defines a circle whose center is rightmost $\rightarrow p_r$

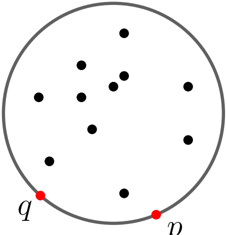
Decide if $C(p,q,p_l)$ or $C(p,q,p_r)$ or $C(p,q)$ is the smallest enclosing circle

Two points known



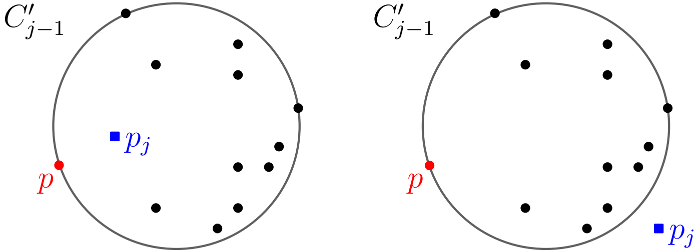
Analysis: two points known

Smallest enclosing circle for n points with two points already known takes $O(n)$ time, worst case



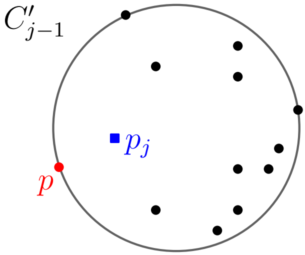
Algorithm: one point known

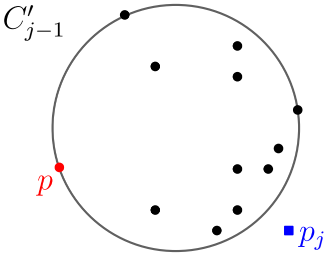
- Use a random order for $p_1,\ldots,p_n$; start with $C_1 = C(p,p_1)$
- **for** $j \leftarrow 2$ **to** n **do**
  If p_j in or on C_{j-1} then $C_j = C_{j-1}$; otherwise, solve smallest enclosing circle for $p_1,\ldots,p_{j-1}$ with two points known (p and p_j)



Analysis: one point known

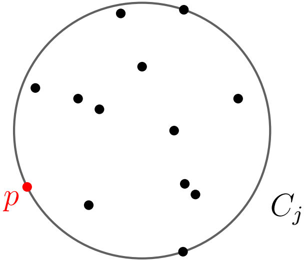
If only one point is known, we used randomized incremental construction, so we need an *expected time analysis*

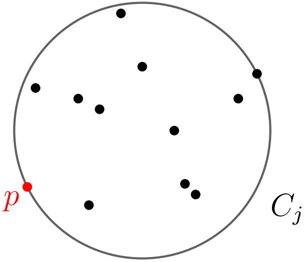




Analysis: one point known

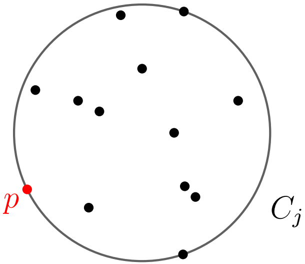
Backwards analysis: Consider the situation *after* adding p_j, so we have computed C_j

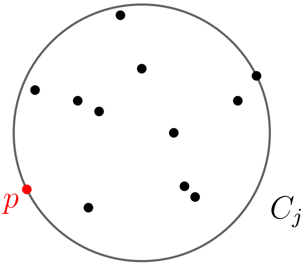




Analysis: one point known

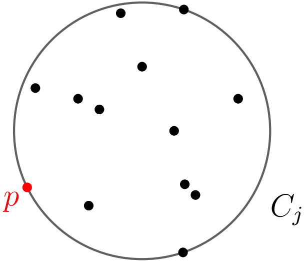
The probability that the j-th addition was expensive is the same as the probability that the smallest enclosing circle changes (decreases in size) if we remove a random point from the j points

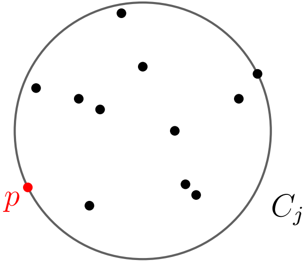




Analysis: one point known

This probability is $2/j$ in the left situation and $1/j$ in the right situation





Analysis: one point known

The expected time for the j-th addition of a point is

$$\frac{j-2}{j} \cdot \Theta(1) + \frac{2}{j} \cdot \Theta(j) = O(1)$$

or

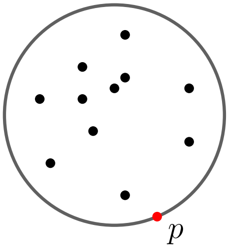
$$\frac{j-1}{j} \cdot \Theta(1) + \frac{1}{j} \cdot \Theta(j) = O(1)$$

The expected running time of the algorithm for n points is:

$$\Theta(n) + \sum_{j=2}^{n} \Theta(1) = \Theta(n)$$

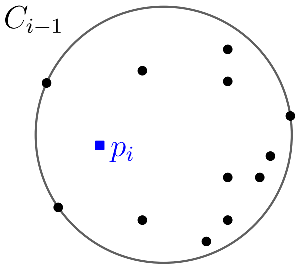
Analysis: one point known

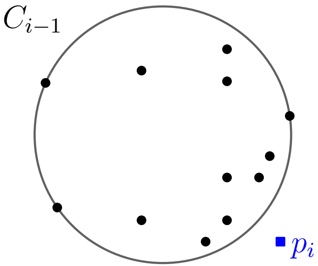
Smallest enclosing circle for n points with one point already known takes $\Theta(n)$ time, expected



Algorithm: smallest enclosing circle

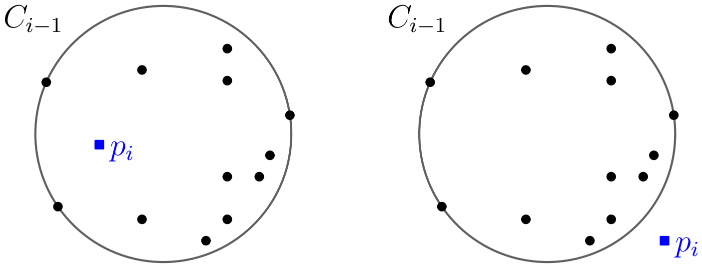
- Use a random order for $p_1,\ldots,p_n$; start with $C_2 = C(p_1,p_2)$
- **for** $i \leftarrow 3$ **to** n **do**
  If p_i in or on C_{i-1} then $C_i = C_{i-1}$; otherwise, solve smallest enclosing circle for $p_1,\ldots,p_{i-1}$ with one point known (p_i)





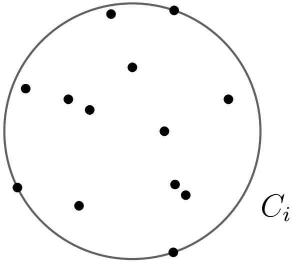
Analysis: smallest enclosing circle

For smallest enclosing circle, we used randomized incremental construction, so we need an *expected time analysis*



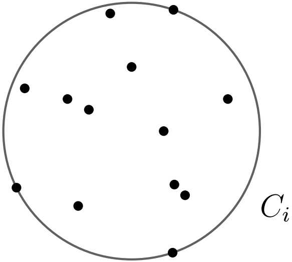
Analysis: smallest enclosing circle

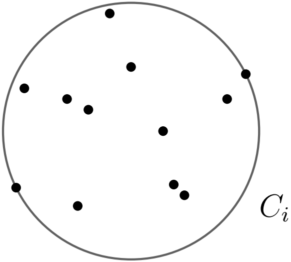
Backwards analysis: Consider the situation *after* adding p_i, so we have computed C_i



Analysis: smallest enclosing circle

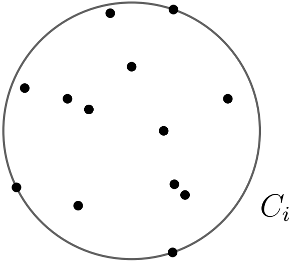
The probability that the i-th addition was expensive is the same as the probability that the smallest enclosing circle changes (decreases in size) if we remove a random point from the i points

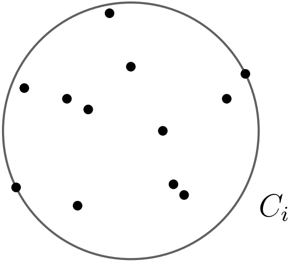




Analysis: smallest enclosing circle

This probability is $3/i$ in the left situation and $2/i$ in the right situation





Analysis: smallest enclosing circle

The expected time for the i-th addition of a point is

$$\frac{i-3}{i} \cdot \Theta(1) + \frac{3}{i} \cdot \Theta(i) = O(1)$$

or

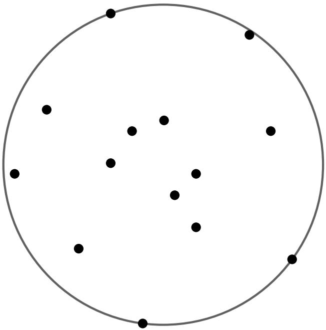
$$\frac{i-2}{i} \cdot \Theta(1) + \frac{2}{i} \cdot \Theta(i) = O(1)$$

The expected running time of the algorithm for n points is:

$$\Theta(n) + \sum_{i=3}^{n} \Theta(1) = \Theta(n)$$

Result: smallest enclosing circle

Theorem The smallest enclosing circle for n points in plane can be computed in $O(n)$ expected time



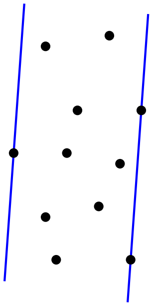
When does it work?

Randomized incremental construction algorithms of this sort (compute an 'optimal' thing) work if:

- The test whether the next input object violates the current optimum must be possible and fast
- If the next input object violates the current optimum, finding the new optimum must be an *easier* problem than the general problem
- The thing must already be defined by $O(1)$ of the input objects
- Ultimately: the analysis must work out

Width

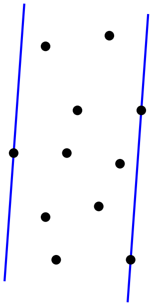
Width: Given a set of n points in the plane, compute the smallest distance between two parallel lines that contain the points (narrowest strip)



Width

Width: Given a set of n points in the plane, compute the smallest distance between two parallel lines that contain the points (narrowest strip)

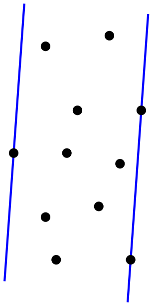
Theorem: The width of a set of n points can be computed in $O(n \log n)$ time.



Width by RIC?

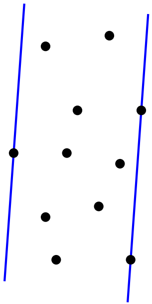
Property: The width is always determined by three points of the set

Idea: Maintain the two lines defining the width to have a fast test for violation.

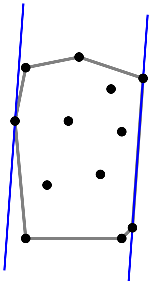


Adding a point

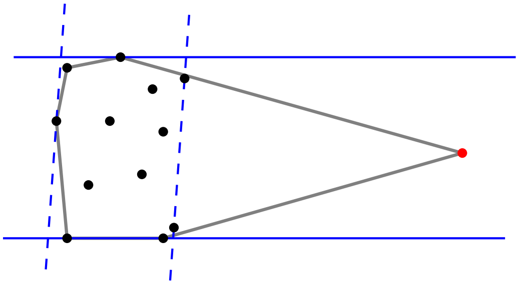
Question: How about adding a point? If the new point lies inside the narrowest strip we are fine, but what if it lies outside?



Adding a point

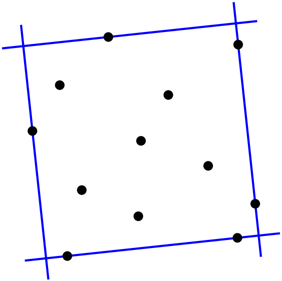


Adding a point



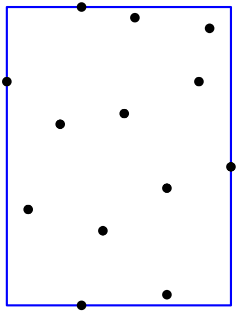
Width

A good reason to be very suspicious of randomized incremental construction as a working approach is *non-uniqueness* of a solution



Minimum bounding box

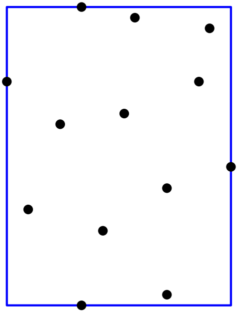
Question: Can we compute the minimum axis-parallel bounding box by randomized incremental construction?



Minimum bounding box

Yes, in $O(n)$ expected time

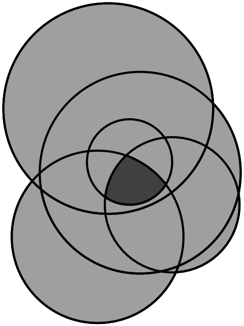
... but a normal incremental algorithm does it in $O(n)$ worst case time



Lowest point in circles

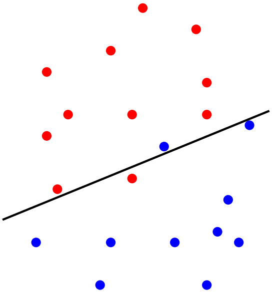
Problem 1: Given n disks in the plane, can we compute the lowest point in their common intersection efficiently by randomized incremental construction?

Problem 2: Given n disks in the plane, can we compute the lowest point in their union efficiently by randomized incremental construction?



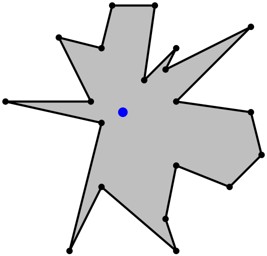
Red-blue separation

Problem: Given a set of n red and blue points in the plane, can we decide efficiently if they have a separating line?



One-guardable polygons

Problem: Given a simple polygon with n vertices, can we decide efficiently if one guard is enough?



One-guardable polygons

It can easily happen that a problem is an instance of linear programming

Then don't devise a new algorithm, just explain how to transform it, and show that it is correct (that your problem is really solved that way)