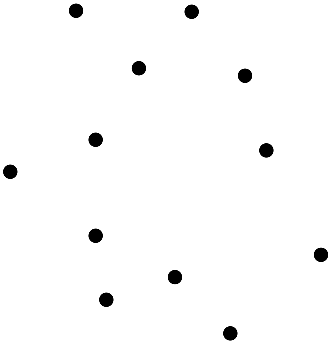
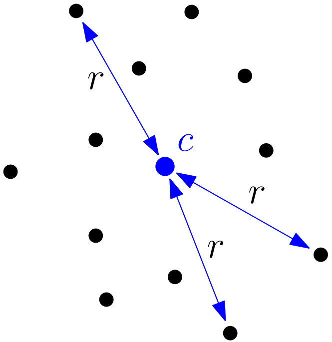
Facility location

Given a set P of n sites in the plane,
find the point c that minimises the distance r to the furthest point(s) of P



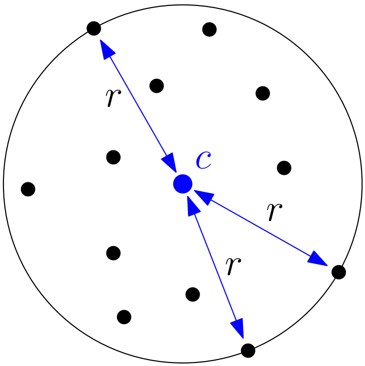
Facility location

Given a set P of n sites in the plane,
find the point c that minimises the distance r to the furthest point(s) of P



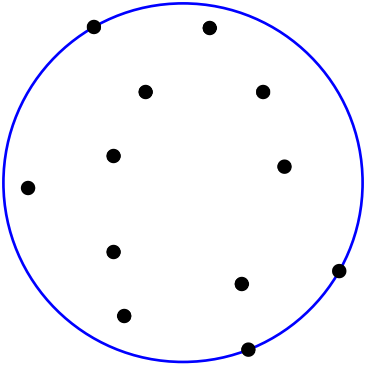
Facility location

Given a set P of n sites in the plane,
find the point c that minimises the distance r to the furthest point(s) of P



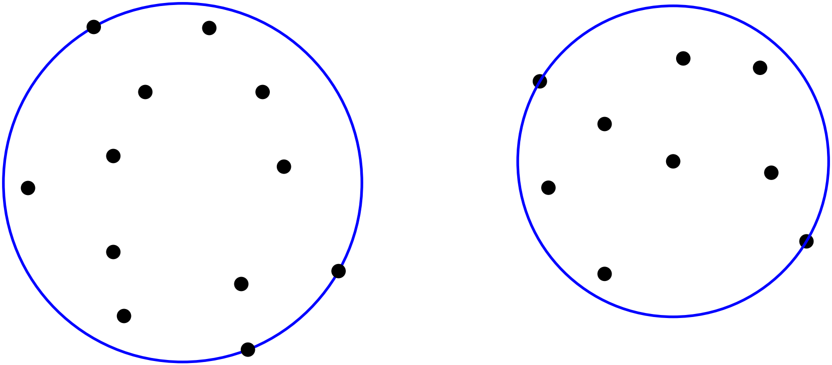
Smallest enclosing circles

Given a set P of n points in the plane,
find the smallest circle $C(P)$ that contains all points of P



Smallest enclosing circles

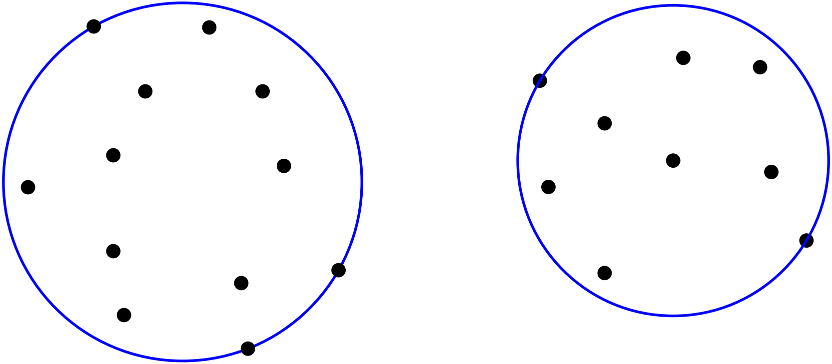
Given a set P of n points in the plane,
find the smallest circle $C(P)$ that contains all points of P



$C(P)$ is either a circle defined by a set $S \subseteq P$ of *three* points on the boundary,
or a circle defined by a set $S \subseteq P$ of *two* diametrically opposite points

Smallest enclosing circles

Given a set P of n points in the plane,
find the smallest circle $C(P)$ that contains all points of P

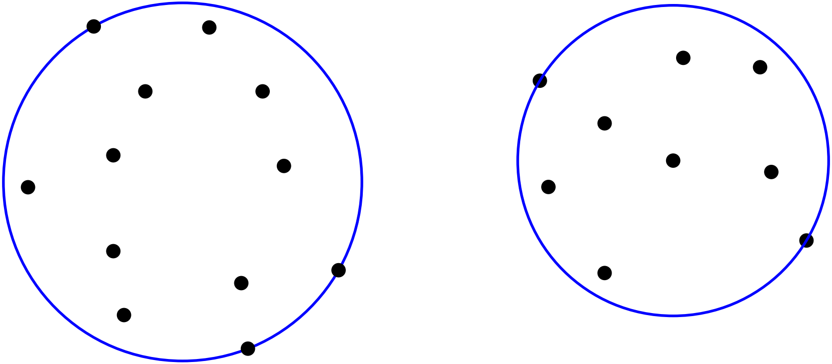


$C(P)$ is either a circle defined by a set $S \subseteq P$ of *three* points on the boundary,
or a circle defined by a set $S \subseteq P$ of *two* diametrically opposite points

Why not more than three points?
Why not less than two points?
Why not two points that are not diametrically opposite?

Smallest enclosing circles

Given a set P of n points in the plane,
find the smallest circle $C(P)$ that contains all points of P

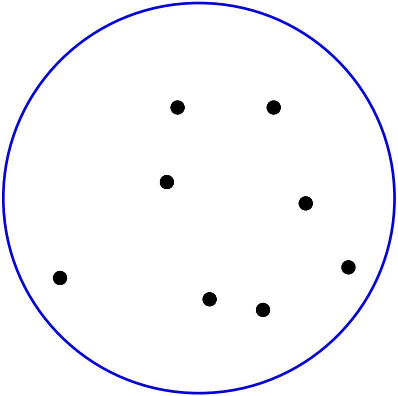


$C(P)$ is either a circle defined by a set $S \subseteq P$ of *three* points on the boundary,
or a circle defined by a set $S \subseteq P$ of *two* diametrically opposite points

Why not more than three points? Three points define a unique circle
Why not less than two points?
Why not two points that are not diametrically opposite?

Smallest enclosing circles

Given a set P of n points in the plane,
find the smallest circle $C(P)$ that contains all points of P

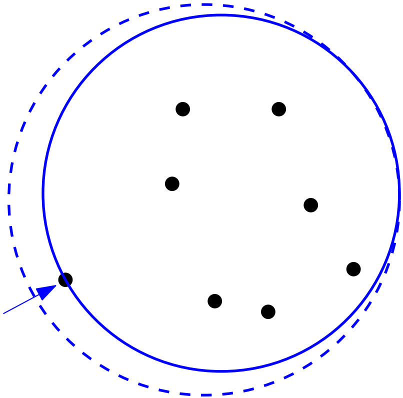


$C(P)$ is either a circle defined by a set $S \subseteq P$ of *three* points on the boundary, or a circle defined by a set $S \subseteq P$ of *two* diametrically opposite points

Why not more than three points? Three points define a unique circle
Why not less than two points? Could make circle smaller until there are two
Why not two points that are not diametrically opposite?

Smallest enclosing circles

Given a set P of n points in the plane,
find the smallest circle $C(P)$ that contains all points of P

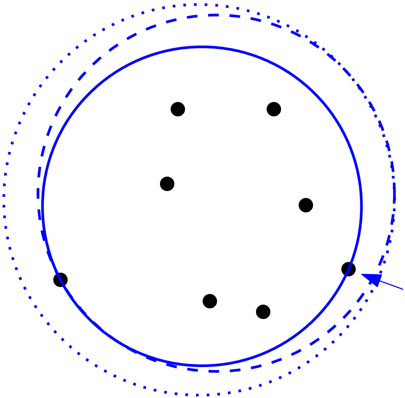


$C(P)$ is either a circle defined by a set $S \subseteq P$ of *three* points on the boundary, or a circle defined by a set $S \subseteq P$ of *two* diametrically opposite points

Why not more than three points? Three points define a unique circle
Why not less than two points? Could make circle smaller until there are two
Why not two points that are not diametrically opposite?

Smallest enclosing circles

Given a set P of n points in the plane,
find the smallest circle $C(P)$ that contains all points of P

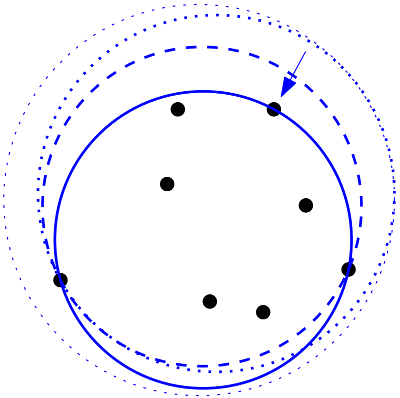


$C(P)$ is either a circle defined by a set $S \subseteq P$ of *three* points on the boundary, or a circle defined by a set $S \subseteq P$ of *two* diametrically opposite points

Why not more than three points? Three points define a unique circle
Why not less than two points? Could make circle smaller until there are two
Why not two points that are not diametrically opposite?

Smallest enclosing circles

Given a set P of n points in the plane,
find the smallest circle $C(P)$ that contains all points of P

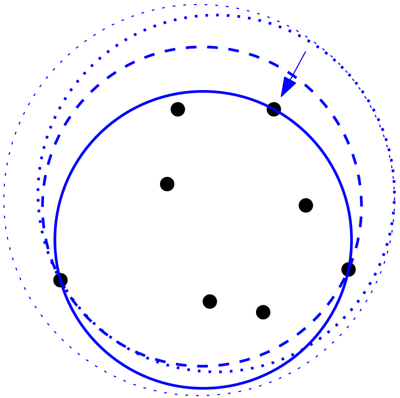


$C(P)$ is either a circle defined by a set $S \subseteq P$ of *three* points on the boundary, or a circle defined by a set $S \subseteq P$ of *two* diametrically opposite points

Why not more than three points? Three points define a unique circle
Why not less than two points? Could make circle smaller until there are two
Why not two points that are not diametrically opposite? Smaller again

Smallest enclosing circles

Given a set P of n points in the plane,
find the smallest circle $C(P)$ that contains all points of P



$C(P)$ is either a circle defined by a set $S \subseteq P$ of *three* points on the boundary,
or a circle defined by a set $S \subseteq P$ of *two* diametrically opposite points

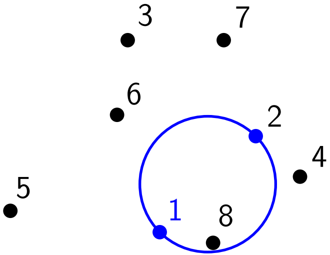
Trivial algorithm: for all $\Theta(n^3)$ pairs and triples of points that define a circle,
test if all other $\Theta(n)$ points lie inside;
choose the smallest circle that passes the test $\rightarrow \Theta(n^4)$ time

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
  then be happy and continue
  else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C

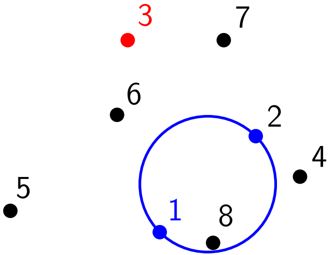


Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

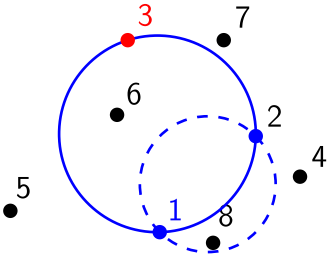
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
  then be happy and continue
  else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C



Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
    then be happy and continue
    else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C

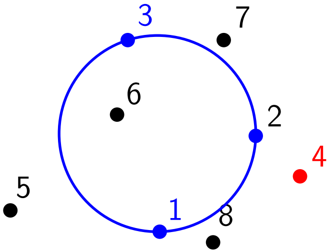


Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
  then be happy and continue
  else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C

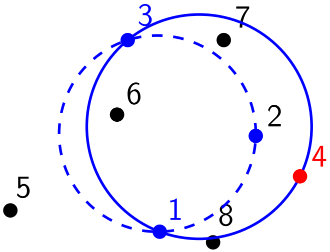


Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
    then be happy and continue
    else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C

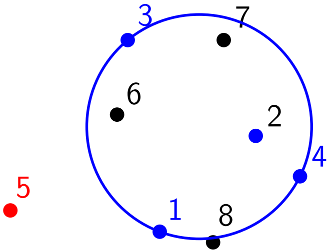


Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

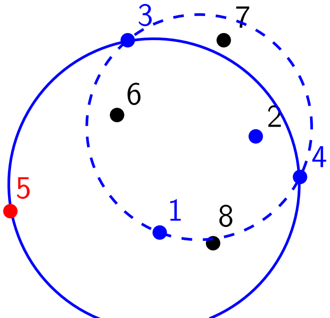
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
  then be happy and continue
  else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C



Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
  then be happy and continue
  else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C

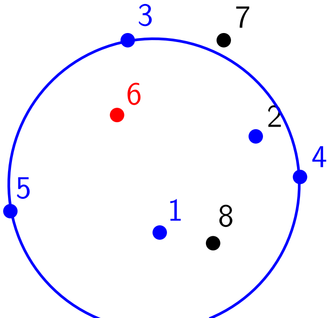


Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
  then be happy and continue
  else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C

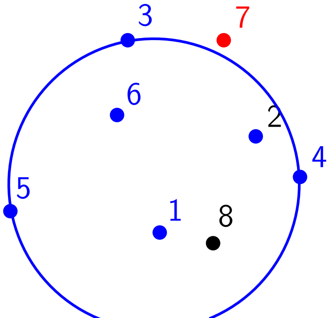


Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
  then be happy and continue
  else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C

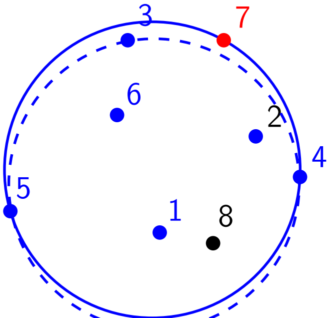


Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
    then be happy and continue
    else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C

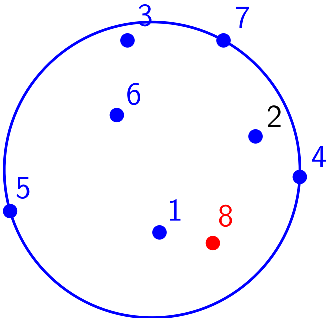


Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
  then be happy and continue
  else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$
return C

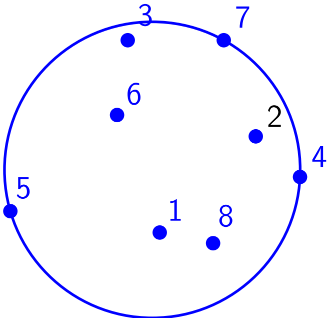


Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
    then be happy and continue
    else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$ with $P[i]$ on boundary
return C

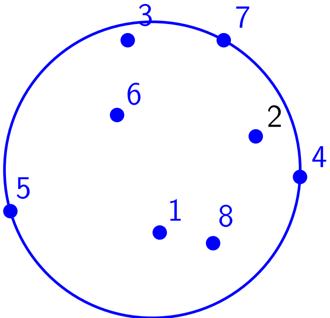


Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
  then be happy and continue
  else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$ with $P[i]$ on boundary
return C



Why does this work? (sketch)

$\mathrm{SEC}(P, i)$ is defined by some set $S \subseteq P[1..i]$ on the boundary of $\mathrm{SEC}(P, i)$; $|S| \leq 3$.

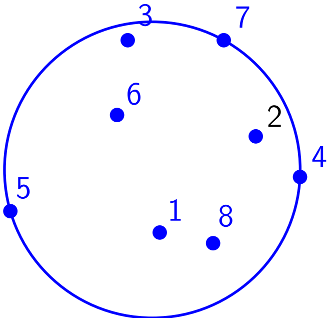
$P[i] \notin S \rightarrow \mathrm{SEC}(P, i) = \mathrm{SEC}(P, i-1)$

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
    then be happy and continue
    else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$ with $P[i]$ on boundary
return C



Why does this work? (sketch)

$\mathrm{SEC}(P, i)$ is defined by some set $S \subseteq P[1..i]$ on the boundary of $\mathrm{SEC}(P, i)$; $|S| \leq 3$.

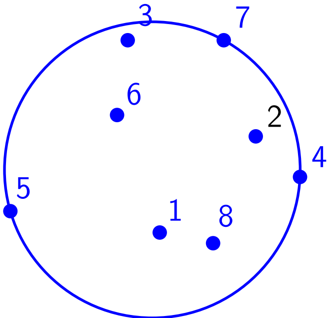
$P[i] \notin S \rightarrow \mathrm{SEC}(P, i) = \mathrm{SEC}(P, i-1)$, so $\mathrm{SEC}(P, i) \neq \mathrm{SEC}(P, i-1) \rightarrow P[i] \in S$;

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ lies in C
    then be happy and continue
    else $C \leftarrow$ the smallest enclosing circle of $P[1..i]$ with $P[i]$ on boundary
return C



Why does this work? (sketch)

$\mathrm{SEC}(P, i)$ is defined by some set $S \subseteq P[1..i]$ on the boundary of $\mathrm{SEC}(P, i)$; $|S| \leq 3$.

$P[i] \notin S \rightarrow \mathrm{SEC}(P, i) = \mathrm{SEC}(P, i-1)$,
so $\mathrm{SEC}(P, i) \neq \mathrm{SEC}(P, i-1) \rightarrow P[i] \in S$;
in other words:
$P[i] \notin \mathrm{SEC}(P, i-1) \rightarrow P[i] \in S$

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow$ s.e.c. of $P[1..i]$ with $P[i]$ on boundary
return C

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow$ s.e.c. of $P[1..i]$ with $P[i]$ on boundary
return C

Algorithm $\mathrm{SEC1}(P, n, p)$

Input: an array P with points in the plane; a number $n \geq 1$
Output: smallest circle containing all points of $P[1..n]$ with p on boundary

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C
    then $C \leftarrow$ s.e.c. of $P[1..i]$ with $P[i]$ and p on boundary
return C

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Input: an array P with points in the plane; a number $n \geq 2$
Output: the smallest circle that contains all points of $P[1..n]$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Algorithm $\mathrm{SEC1}(P, n, p)$

Input: an array P with points in the plane; a number $n \geq 1$
Output: smallest circle containing all points of $P[1..n]$ with p on boundary

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C
    then $C \leftarrow$ s.e.c. of $P[1..i]$ with $P[i]$ and p on boundary
return C

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$ *(smallest circle enclosing $P[1..n]$)*

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Algorithm $\mathrm{SEC1}(P, n, p)$ *(small. circle encl. $P[1..n]$ with p on boundary)*

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Algorithm $\mathrm{SEC2}(P, n, p, q)$ *(sm. circ. encl. $P[1..n]$ with p, q on bound.)*

$C \leftarrow$ circle defined by p and q
for $i \leftarrow 1$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow$ circle defined by p, q and $P[i]$
return C

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Algorithm $\mathrm{SEC1}(P, n, p)$

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Algorithm $\mathrm{SEC2}(P, n, p, q)$

Worst-case running time: $\Theta(n)$

$C \leftarrow$ circle defined by p and q
for $i \leftarrow 1$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow$ circle defined by p, q and $P[i]$
return C

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Algorithm $\mathrm{SEC1}(P, n, p)$

Worst-case running time: $\Theta(1) + \sum_{i=2}^{n} O(i) = O(n^2)$

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Algorithm $\mathrm{SEC2}(P, n, p, q)$

Worst-case running time: $\Theta(n)$

$C \leftarrow$ circle defined by p and q
for $i \leftarrow 1$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow$ circle defined by p, q and $P[i]$
return C

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Worst-case running time:
$\Theta(1) + \sum_{i=3}^{n} O(i^2) = O(n^3)$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Algorithm $\mathrm{SEC1}(P, n, p)$

Worst-case running time:
$\Theta(1) + \sum_{i=2}^{n} O(i) = O(n^2)$

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Algorithm $\mathrm{SEC2}(P, n, p, q)$

Worst-case running time: $\Theta(n)$

$C \leftarrow$ circle defined by p and q
for $i \leftarrow 1$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow$ circle defined by p, q and $P[i]$
return C

Smallest enclosing circles: an incremental algorithm

Algorithm $\text{SEC}(P, n)$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \text{SEC1}(P, i-1, P[i])$
return C

Worst-case running time:
$\Theta(1) + \sum_{i=3}^{n} O(i^2) = O(n^3)$

Algorithm $\text{SEC1}(P, n, p)$

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \text{SEC2}(P, i-1, P[i], p)$
return C

Worst-case running time:
$\Theta(1) + \sum_{i=2}^{n} O(i) = O(n^2)$

Worst case takes into account that line 3 of SEC may call SEC1 for every i, and line 3 of SEC1 may call SEC2 for every i.
But this does not always happen, only if $P[i]$ does not lie in C.

What about *average expected* running time?

Smallest enclosing circles: an incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Worst-case running time:
$\Theta(1) + \sum_{i=3}^{n} O(i^2) = O(n^3)$

Algorithm $\mathrm{SEC1}(P, n, p)$

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Worst-case running time:
$\Theta(1) + \sum_{i=2}^{n} O(i) = O(n^2)$

Worst case takes into account that line 3 of SEC may call SEC1 for every i, and line 3 of SEC1 may call SEC2 for every i.
But this does not always happen, only if $P[i]$ does not lie in C.

What about *average expected* running time? IMPOSSIBLE TO SAY: maybe the data in our application has some structure that often brings out the worst-case behaviour (e.g. with points on a line from left to right: SEC calls SEC1 for every i)

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$ *(smallest circle enclosing $P[1..n]$)*

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$ *(smallest circle enclosing $P[1..n]$)*

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

randomly choose a permutation π of $P[1..n]$ according to a uniform probability distribution on all $n!$ permutations of $P[1..n]$; apply π to $P[1..n]$

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$ *(smallest circle enclosing $P[1..n]$)*

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Algorithm $\mathrm{SEC1}(P, n, p)$ *(small. circle encl. $P[1..n]$ with p on boundary)*

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Algorithm $\mathrm{SEC2}(P, n, p, q)$ *(unchanged, still running in $\Theta(n)$ time)*

...

...

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$ *(smallest circle enclosing* $P[1..n]$*)*

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Algorithm $\mathrm{SEC1}(P, n, p)$ *(small. circle encl.* $P[1..n]$ *with* p *on boundary)*

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Only happens if $P[i]$ is one of the one or two points in $P[1..i]$ that, together with p, define $\mathrm{SEC1}(P, i, p)$. What is the probability q that this happens?

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$ *(smallest circle enclosing $P[1..n]$)*

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Algorithm $\mathrm{SEC1}(P, n, p)$ *(small. circle encl. $P[1..n]$ with p on boundary)*

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Only happens if $P[i]$ is one of the one or two points in $P[1..i]$ that, together with p, define $\mathrm{SEC1}(P, i, p)$. What is the probability q that this happens?

Answer: if all permutations of $P[1..n]$ are equally likely, then $q \leq 2/i$

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$ *(smallest circle enclosing $P[1..n]$)*

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Expected running time:

$\Theta(n) + \sum_{i=2}^{n}(\Theta(1) + (2/i)O(i))$

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

Algorithm $\mathrm{SEC1}(P, n, p)$

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Only happens if $P[i]$ is one of the one or two points in $P[1..i]$ that, together with p, define $\mathrm{SEC1}(P, i, p)$. What is the probability q that this happens?

Answer: if all permutations of $P[1..n]$ are equally likely, then $q \leq 2/i$

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC1}(P, i-1, P[i])$
return C

Algorithm $\mathrm{SEC1}(P, n, p)$

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** n
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Expected running time $\Theta(n)$ if all permutations of $P[1..n]$ are equally likely

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $j \leftarrow i - 1$; $C \leftarrow \mathrm{SEC1}(P, j, P[i])$
return C

Algorithm $\mathrm{SEC1}(P, j, p)$

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** j
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i - 1, P[i], p)$
return C

Expected running time $\Theta(j)$
if all permutations of $P[1..j]$
are equally likely

Smallest enclosing circles: a randomised incremental algorithm

Expected running time:

$\Theta(n) + \sum_{i=2}^{n}(\Theta(1) + (3/i)O(i))$
$= \Theta(n) + \sum_{i=2}^{n}\Theta(1) = \Theta(n)$

Algorithm $\mathrm{SEC}(P, n)$

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $j \leftarrow i - 1$; $C \leftarrow \mathrm{SEC1}(P, j, P[i])$
return C

Only happens if $P[i]$ is one of the two or three points in $P[1..i]$ that define $\mathrm{SEC}(P, i)$. What is the probability that this happens?

Answer: at most $3/i$ (because $P[1..n]$ was randomly permuted)

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $j \leftarrow i - 1$; $C \leftarrow \mathrm{SEC1}(P, j, P[i])$
return C

Expected running time:

$\Theta(n) + \sum_{i=2}^{n}(\Theta(1) + (3/i)O(i))$
$= \Theta(n) + \sum_{i=2}^{n}\Theta(1) = \Theta(n)$

Only happens if $P[i]$ is one of the two or three points in $P[1..i]$ that define $\mathrm{SEC}(P, i)$. What is the probability that this happens?

Answer: at most $3/i$ (because $P[1..n]$ was randomly permuted)

Let $p = P[i]$; let P' be the set of points in $P[1..j]$.

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $j \leftarrow i - 1$; $C \leftarrow \mathrm{SEC1}(P, j, P[i])$
return C

Expected running time:

$\Theta(n) + \sum_{i=2}^{n}(\Theta(1) + (3/i)O(i))$
$= \Theta(n) + \sum_{i=2}^{n}\Theta(1) = \Theta(n)$

Only happens if $P[i]$ is one of the two or three points in $P[1..i]$ that define $\mathrm{SEC}(P, i)$. What is the probability that this happens?

Answer: at most $3/i$ (because $P[1..n]$ was randomly permuted)

Let $p = P[i]$; let P' be the set of points in $P[1..j]$.
Let $R =$ set of permutations with $P[i] = p$ and P' in $P[1..j]$; $|R| = j!(n-i)!$.

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $j \leftarrow i - 1$; $C \leftarrow \mathrm{SEC1}(P, j, P[i])$
return C

Expected running time:

$\Theta(n) + \sum_{i=2}^{n}(\Theta(1) + (3/i)O(i))$
$= \Theta(n) + \sum_{i=2}^{n}\Theta(1) = \Theta(n)$

Only happens if $P[i]$ is one of the two or three points in $P[1..i]$ that define $\mathrm{SEC}(P, i)$. What is the probability that this happens?

Answer: at most $3/i$ (because $P[1..n]$ was randomly permuted)

Let $p = P[i]$; let P' be the set of points in $P[1..j]$.
Let $R =$ set of permutations with $P[i] = p$ and P' in $P[1..j]$; $|R| = j!(n-i)!$.
If p does not lie in C, then this happens for all permutations in R.

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $j \leftarrow i-1$; $C \leftarrow \mathrm{SEC1}(P, j, P[i])$
return C

Expected running time:

$\Theta(n) + \sum_{i=2}^{n}(\Theta(1) + (3/i)O(i))$
$= \Theta(n) + \sum_{i=2}^{n}\Theta(1) = \Theta(n)$

Only happens if $P[i]$ is one of the two or three points in $P[1..i]$ that define $\mathrm{SEC}(P, i)$. What is the probability that this happens?

Answer: at most $3/i$ (because $P[1..n]$ was randomly permuted)

Let $p = P[i]$; let P' be the set of points in $P[1..j]$.
Let R = set of permutations with $P[i] = p$ and P' in $P[1..j]$; $|R| = j!(n-i)!$.
If p does not lie in C, then this happens for all permutations in R.
For every permutation of $P[1..j]$, there are $(n-i)!$ permutations in R such that $P[1..j]$ is permuted this way.

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $j \leftarrow i - 1$; $C \leftarrow \mathrm{SEC1}(P, j, P[i])$
return C

Expected running time:

$\Theta(n) + \sum_{i=2}^{n}(\Theta(1) + (3/i)O(i))$
$= \Theta(n) + \sum_{i=2}^{n}\Theta(1) = \Theta(n)$

Only happens if $P[i]$ is one of the two or three points in $P[1..i]$ that define $\mathrm{SEC}(P, i)$. What is the probability that this happens?

Answer: at most $3/i$ (because $P[1..n]$ was randomly permuted)

Let $p = P[i]$; let P' be the set of points in $P[1..j]$.
Let $R =$ set of permutations with $P[i] = p$ and P' in $P[1..j]$; $|R| = j!(n-i)!$.
If p does not lie in C, then this happens for all permutations in R.
For every permutation of $P[1..j]$, there are $(n-i)!$ permutations in R such that $P[1..j]$ is permuted this way.
So, for fixed p and P', every permutation of $P[1..j]$ occurs with probability $(n-i)!/(j!(n-i)!) = 1/j!$.

Smallest enclosing circles: a randomised incremental algorithm

Expected running time:

$\Theta(n) + \sum_{i=2}^{n}(\Theta(1) + (3/i)O(i))$
$= \Theta(n) + \sum_{i=2}^{n}\Theta(1) = \Theta(n)$

Algorithm $\mathrm{SEC}(P, n)$

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $j \leftarrow i - 1$; $C \leftarrow \mathrm{SEC1}(P, j, P[i])$
return C

Only happens if $P[i]$ is one of the two or three points in $P[1..i]$ that define $\mathrm{SEC}(P, i)$. What is the probability that this happens?

Answer: at most $3/i$ (because $P[1..n]$ was randomly permuted)

If it happens, it happens for all permutations with this point $P[i]$, and this set of points in $P[1..j]$; $\rightarrow$ all permutations of $P[1..j]$ are equally likely

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $j \leftarrow i - 1$; $C \leftarrow \mathrm{SEC1}(P, j, P[i])$
return C

Expected running time:

$\Theta(n) + \sum_{i=2}^{n}(\Theta(1) + (3/i)O(i))$
$= \Theta(n) + \sum_{i=2}^{n}\Theta(1) = \Theta(n)$

Only happens if $P[i]$ is one of the two or three points in $P[1..i]$ that define $\mathrm{SEC}(P, i)$. What is the probability that this happens?

Answer: at most $3/i$ (because $P[1..n]$ was randomly permuted)

If it happens, it happens for all permutations with this point $P[i]$, and this set of points in $P[1..j]$; $\rightarrow$ all permutations of $P[1..j]$ are equally likely

Algorithm $\mathrm{SEC1}(P, j, p)$

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** j
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C

Expected running time $\Theta(j)$ if all permutations of $P[1..j]$ are equally likely

Smallest enclosing circles: a randomised incremental algorithm

Algorithm $\mathrm{SEC}(P, n)$

Expected running time:

$\Theta(n) + \sum_{i=2}^{n}(\Theta(1) + (3/i)O(i))$

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

randomly permute the points in $P[1..n]$
$C \leftarrow$ circle defined by $P[1]$ and $P[2]$
for $i \leftarrow 3$ **to** n
do if $P[i]$ does not lie in C **then** $j \leftarrow i-1$; $C \leftarrow \mathrm{SEC1}(P, j, P[i])$
return C

Only happens if $P[i]$ is one of the two or three points in $P[1..i]$ that define $\mathrm{SEC}(P, i)$. What is the probability that this happens?

Answer: at most $3/i$ (because $P[1..n]$ was randomly permuted)

If it happens, it happens for all permutations with this point $P[i]$, and this set of points in $P[1..j]$; $\rightarrow$ all permutations of $P[1..j]$ are equally likely

Algorithm $\mathrm{SEC1}(P, j, p)$

Expected running time $\Theta(j)$ if all permutations of $P[1..j]$ are equally likely

$C \leftarrow$ circle defined by p and $P[1]$
for $i \leftarrow 2$ **to** j
do if $P[i]$ does not lie in C **then** $C \leftarrow \mathrm{SEC2}(P, i-1, P[i], p)$
return C