Introduction to Computational Geometry

Sasanka Roy
Chennai Mathematical Institute

Organization of the Talk

Organization of the Talk

1. Preliminaries, Generic definition and Literature

2. Some technical details of easy versions

3. Conclusion

Organization of the Talk

1. Preliminaries, Generic definition and Literature

2. Some technical details of easy versions

3. Conclusion

What are we going to talk about?

What are we going to talk about?

We have some data

What are we going to talk about?

We have some data

Geometric Data

What are we going to talk about?

We have some data

Geometric Data ????

Geometric Data

What are we going to talk about?

We have some data

Geometric Data ????

Geometric Data

What do I mean ????

What are we going to talk about?

We have some data

Geometric Data ????

I mean: we have

Geometric Data

What do I mean ????

What are we going to talk about?

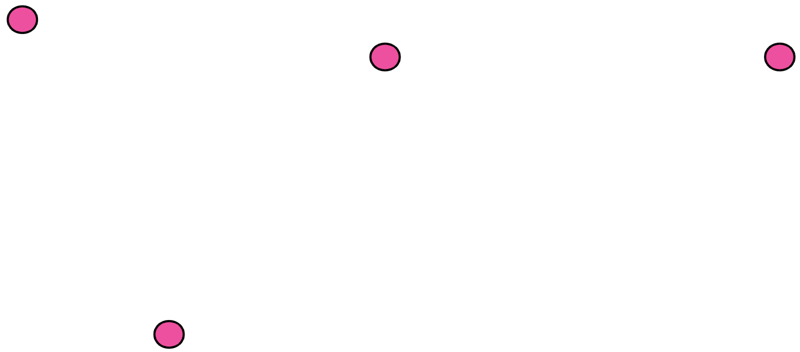
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points,



What are we going to talk about?

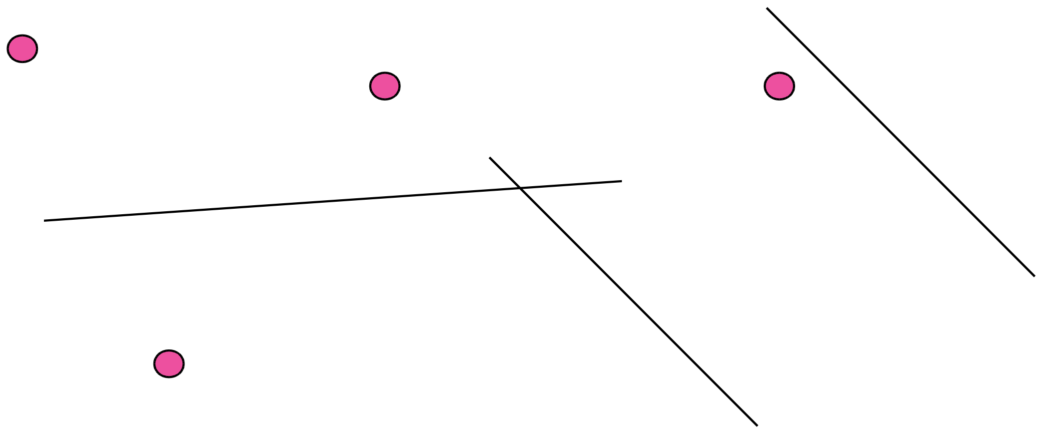
We have some data

Geometric Data

Geometric Data ????

What do I mean ????

I mean: we have points, line segments,



What are we going to talk about?

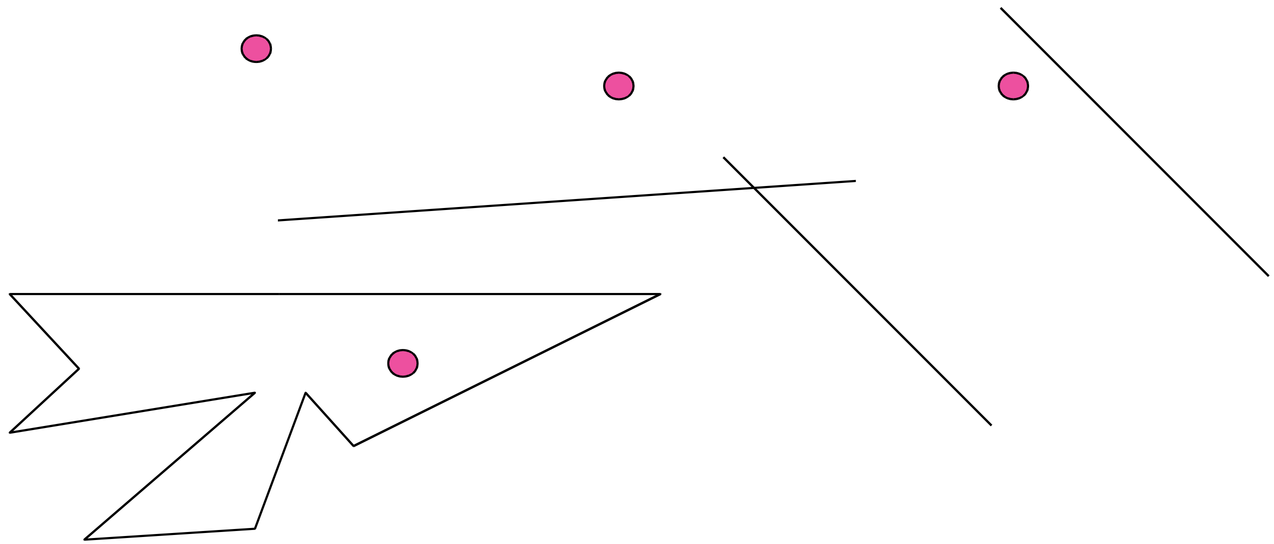
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



What are we going to talk about?

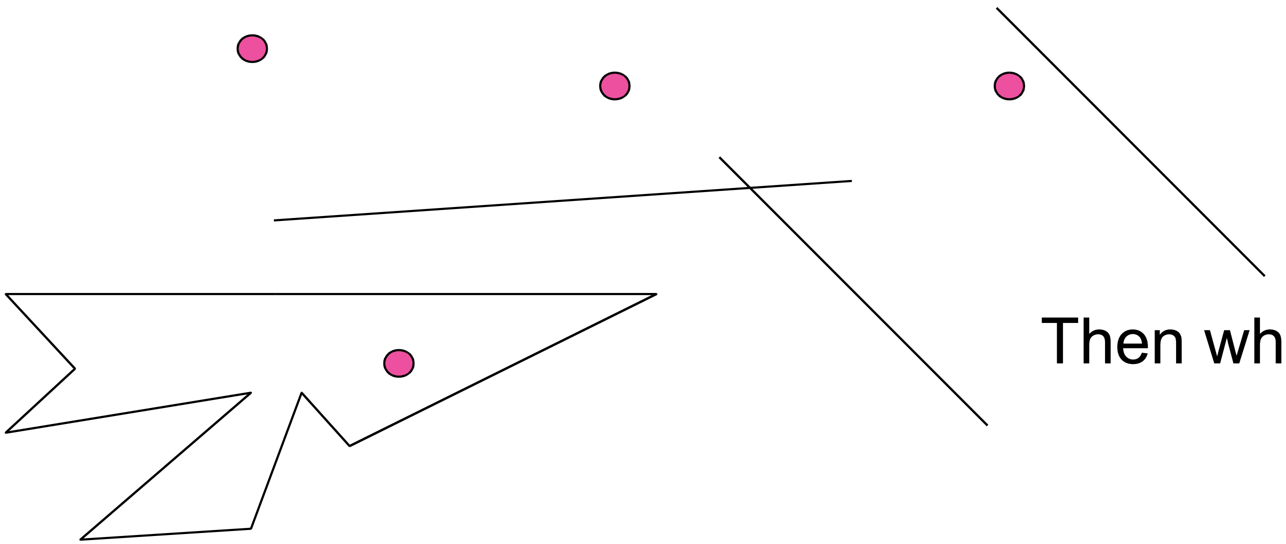
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

What are we going to talk about?

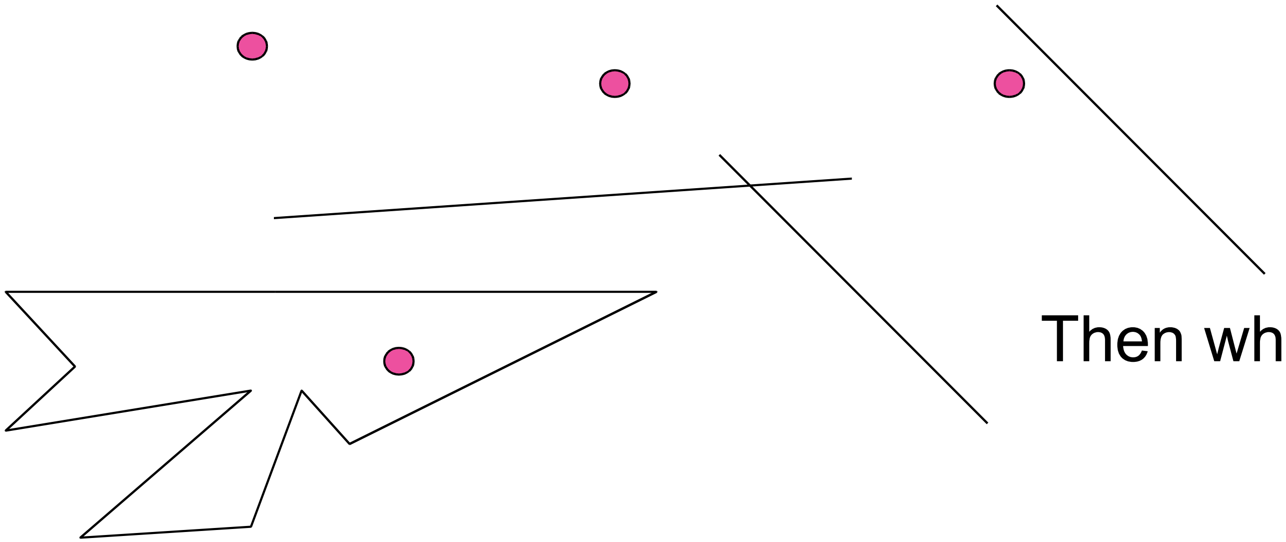
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

What are we going to talk about?

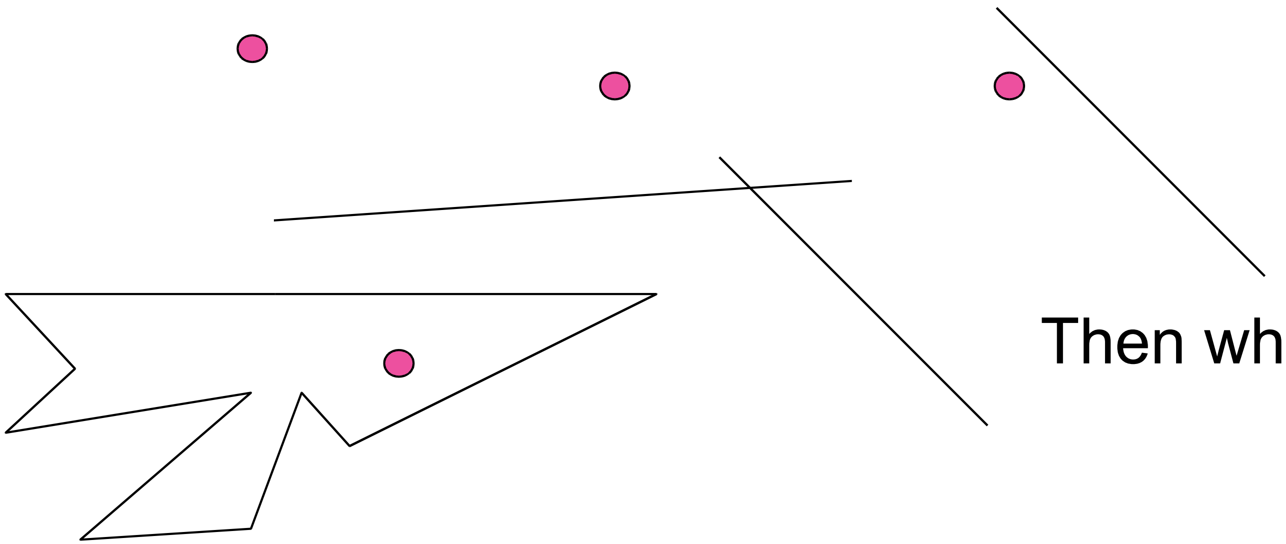
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

Closest points to the line segments

What are we going to talk about?

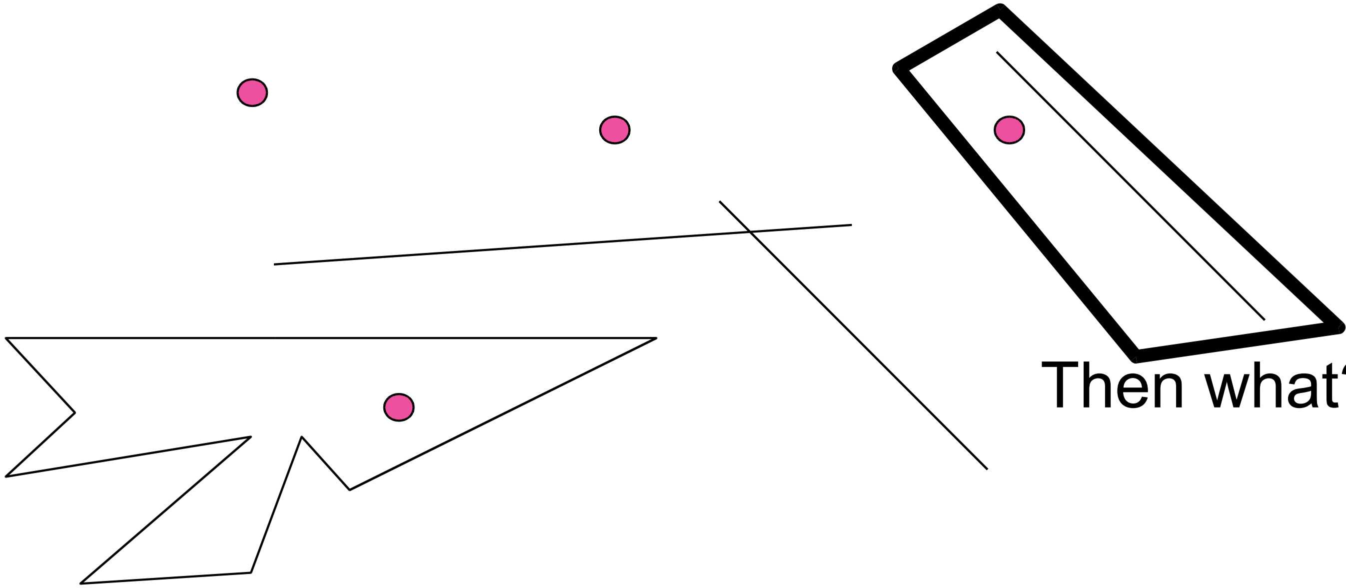
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

Closest points to the line segments

What are we going to talk about?

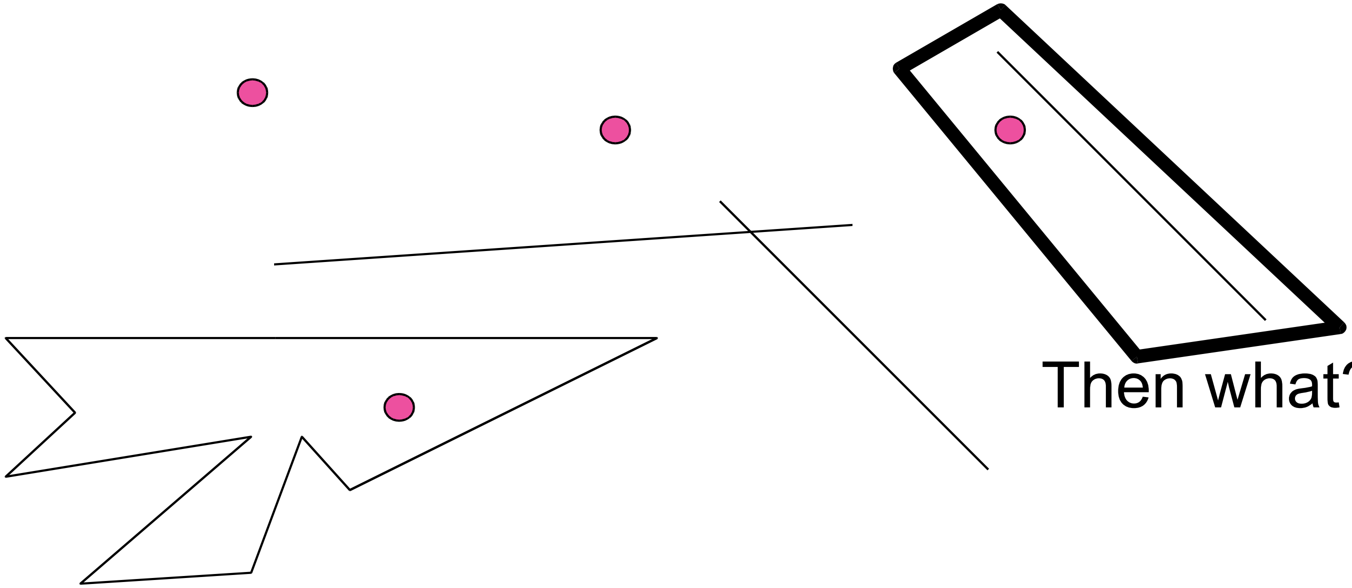
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

Closest points to the line segments

Point inside the simple polygon

What are we going to talk about?

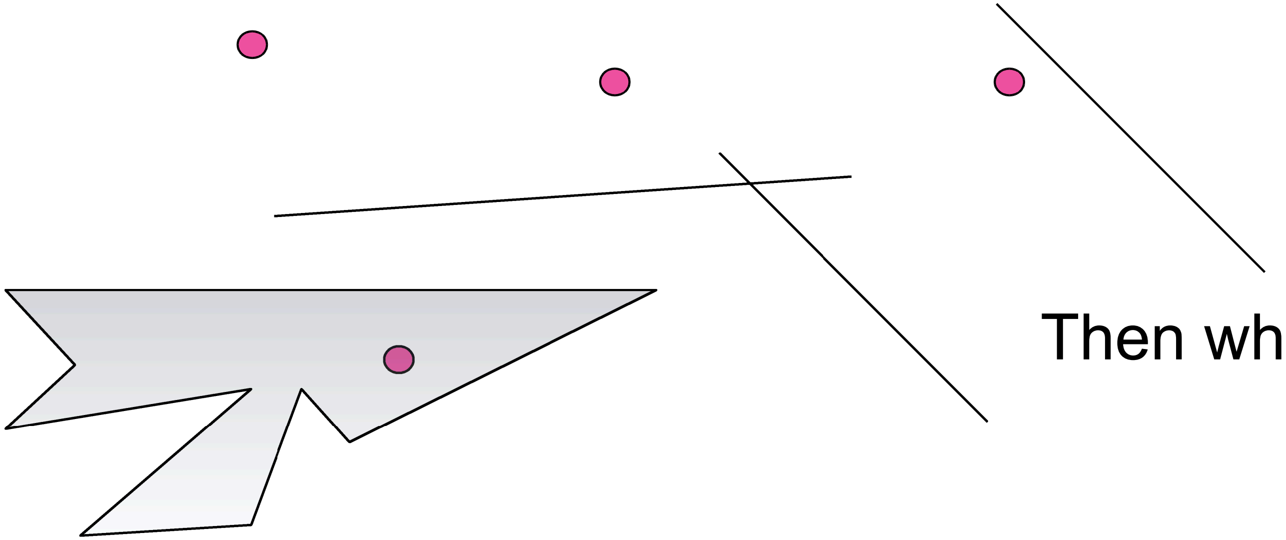
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

Closest points to the line segments

Point inside the simple polygon

What are we going to talk about?

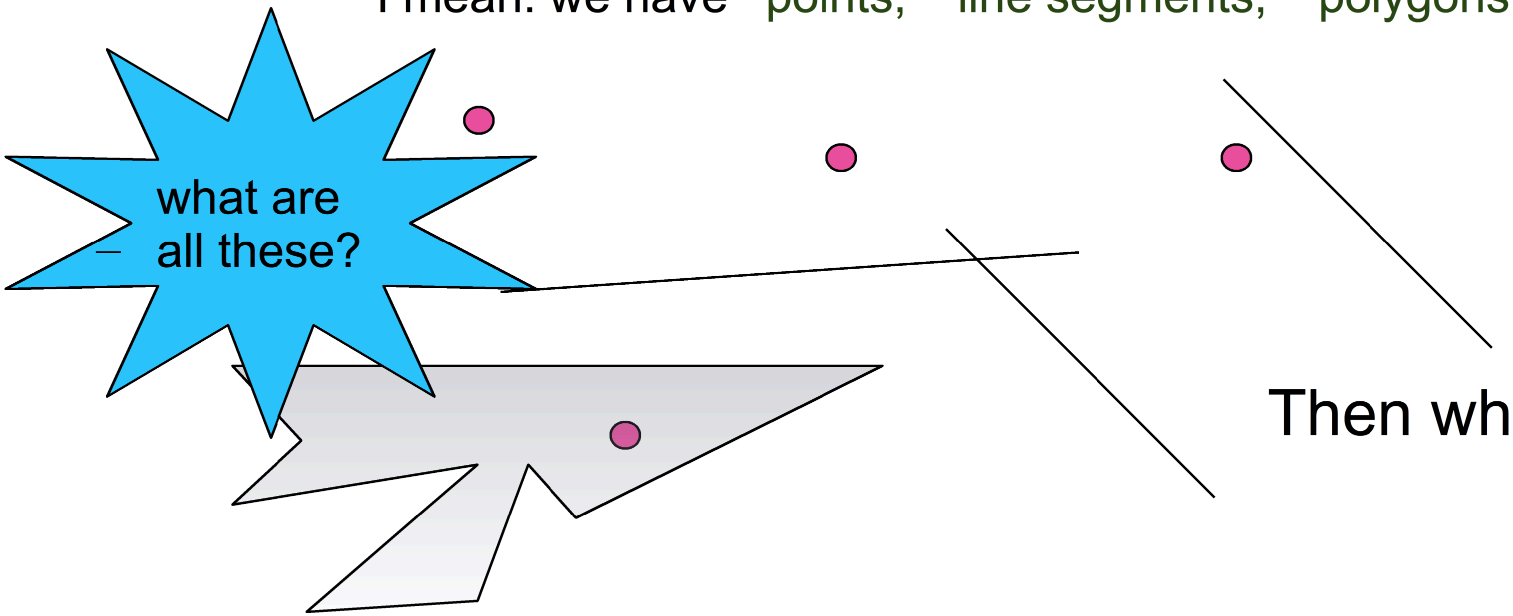
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

Closest points to the line segments

Point inside the simple polygon

Can you be a bit Practical??

Planar Point Location

Planar Point Location

Which state has the site/point with

Latitude=   28° 38' N

Longitude= 72° 12' E

Planar Point Location

Which state has the site/point with

Latitude= 28° 38' N

Longitude= 72° 12' E

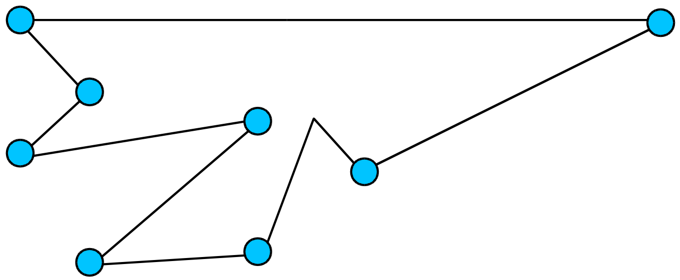
Planar Point Location

Which state has the site/point with

Latitude= 28° 38' N

Longitude= 72° 12' E

Can we view States as simple polygon?



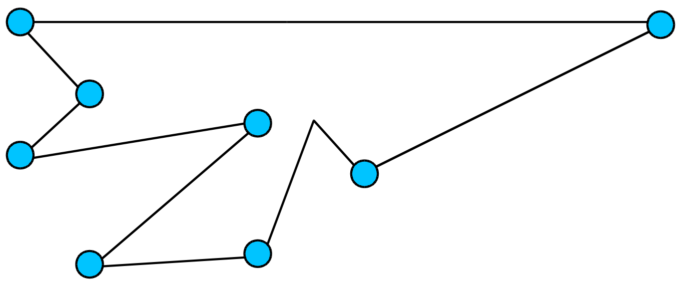
Planar Point Location

Which state has the site/point with

Latitude= 13° 08' 10'' N

Longitude= 80° 27' 40'' E

Can we view States as simple polygon?



simple polygon: Closed region whose boundary is formed by non-intersecting line segments

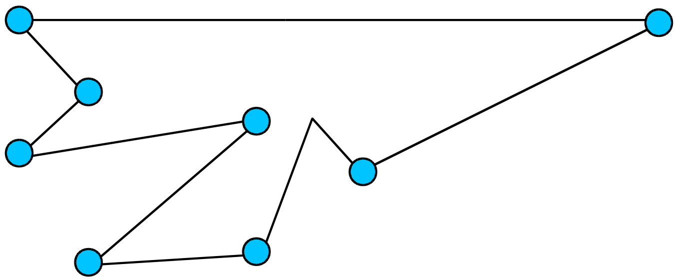
Planar Point Location

Which state has the site/point with

Latitude= 13° 08' 10'' N

Longitude= 80° 27' 40'' E

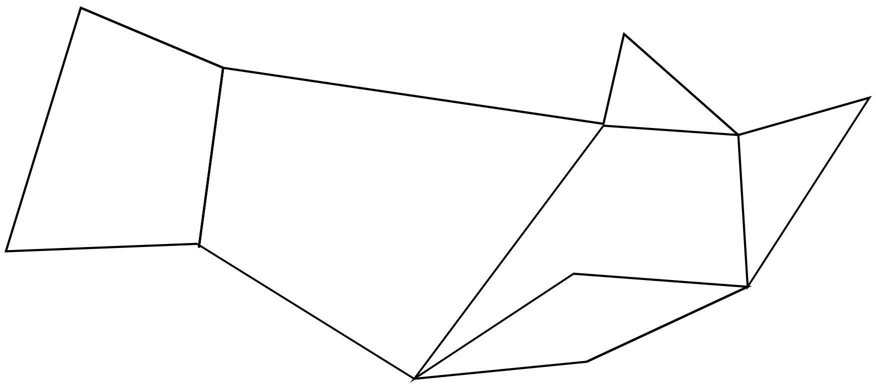
Can we view States as simple polygon? Yes



simple polygon: Closed region whose boundary is formed by non-intersecting line segments

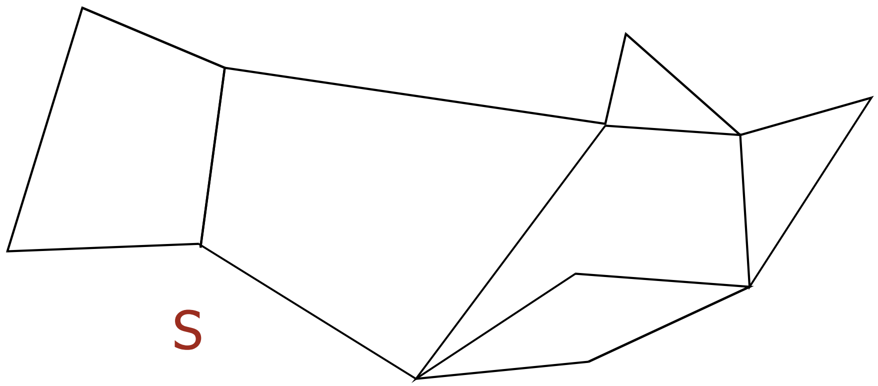
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



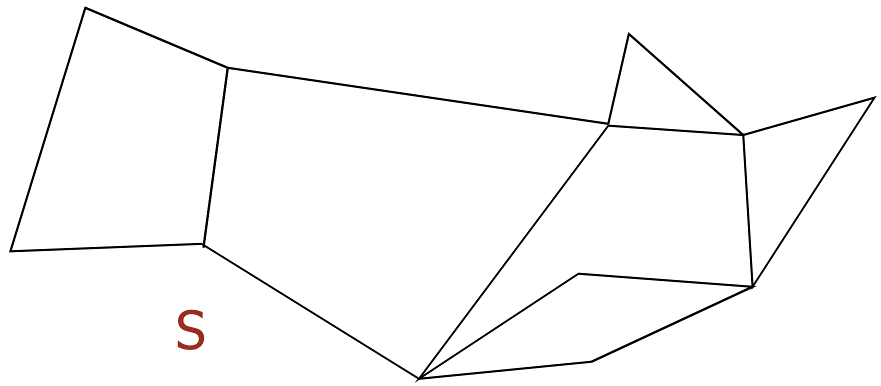
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



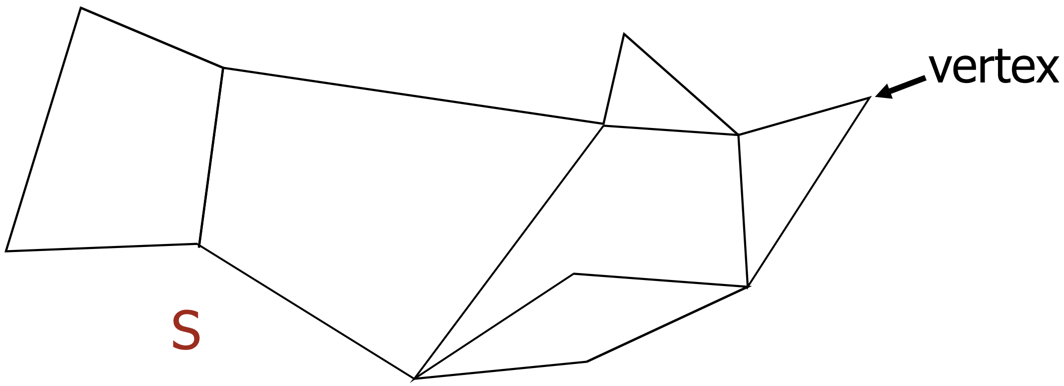
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



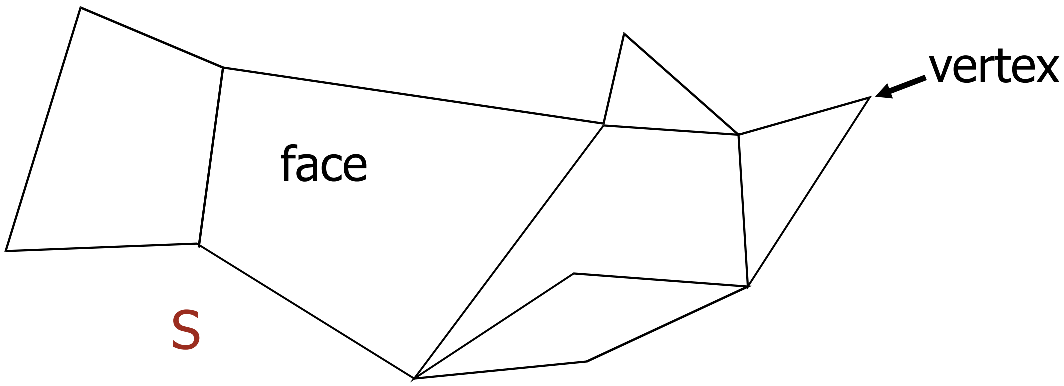
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



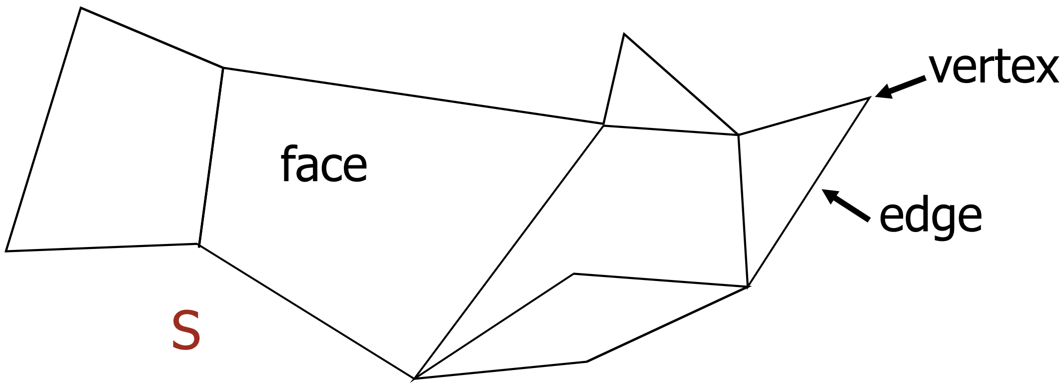
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



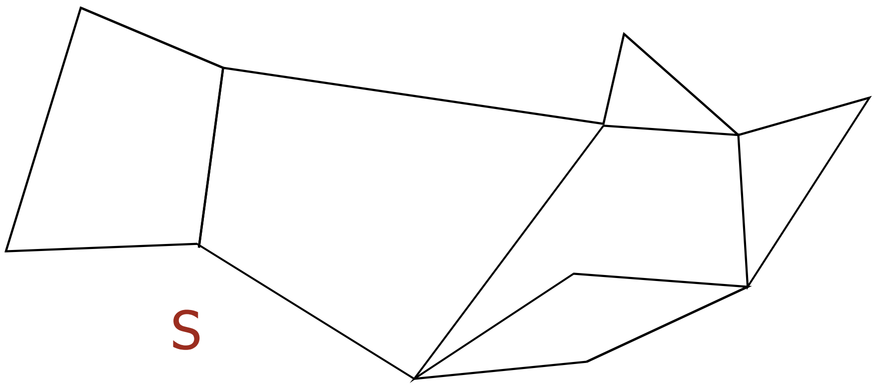
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



Formally Planar Point Location

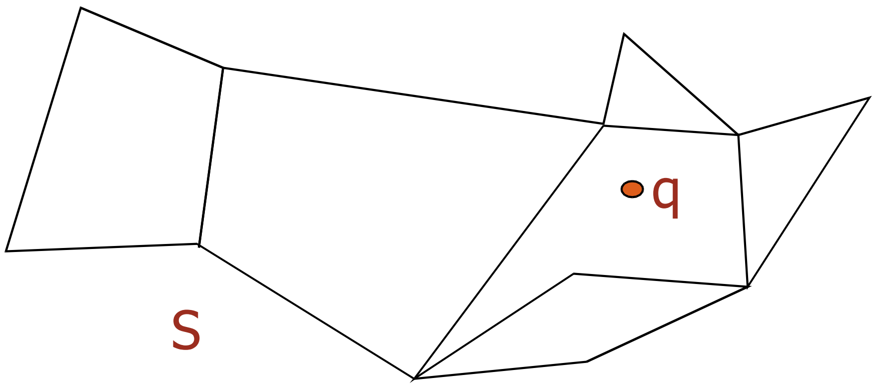
Given a planar subdivision S



Preprocess S such that:

Formally Planar Point Location

Given a planar subdivision S

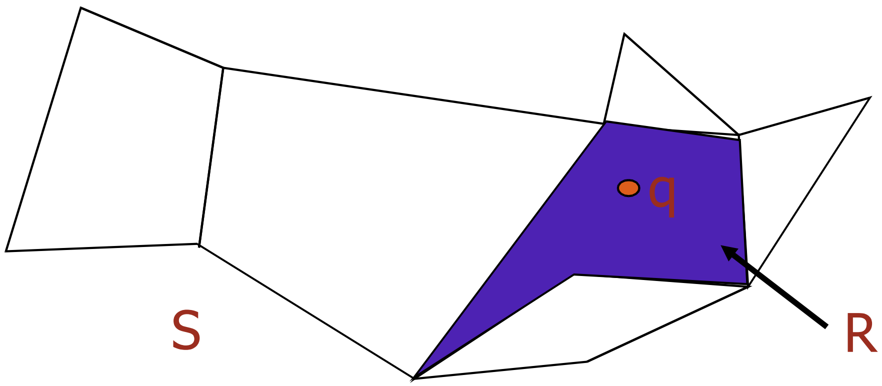


Preprocess S such that:

For any query point q,

Formally Planar Point Location

Given a planar subdivision S

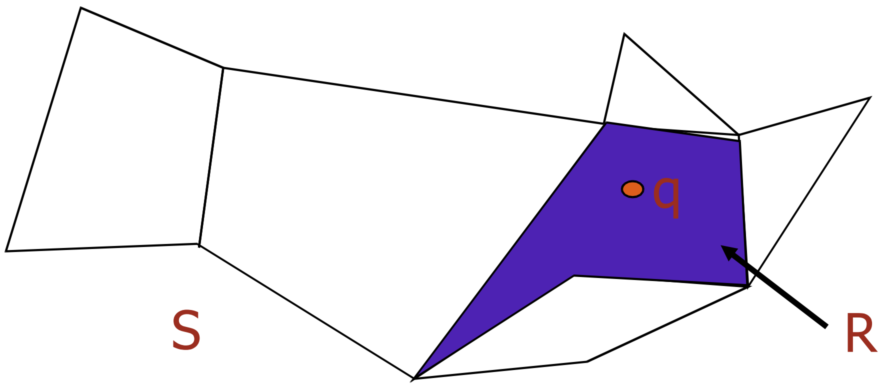


Preprocess S such that:

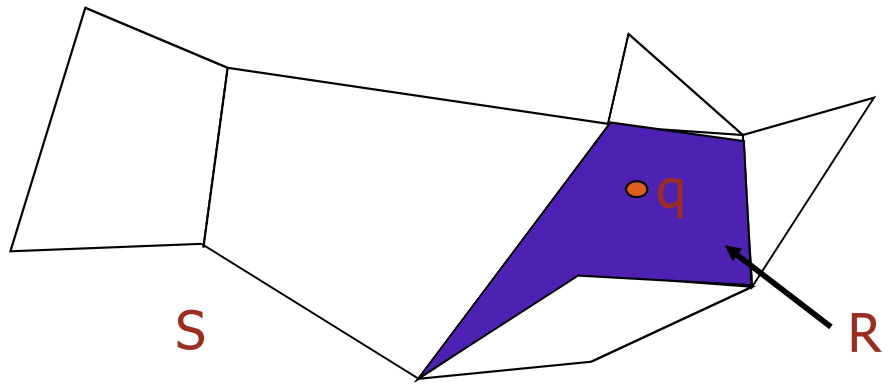
For any query point q,

The region/face R containing q can be reported efficiently.

Questions?

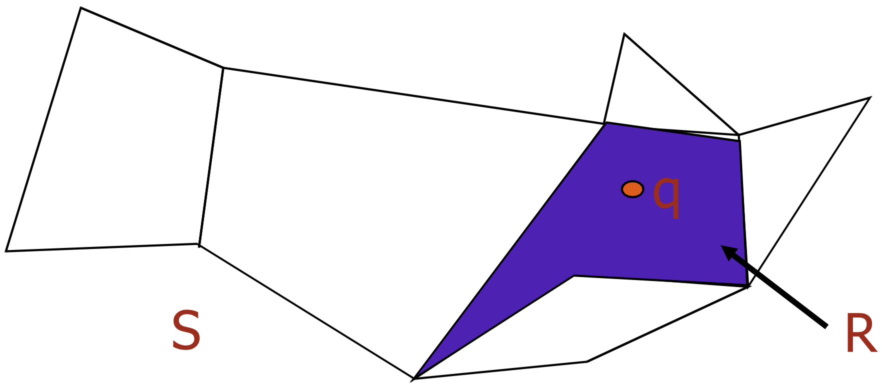


Questions?



1. How much is the preprocessing Time?

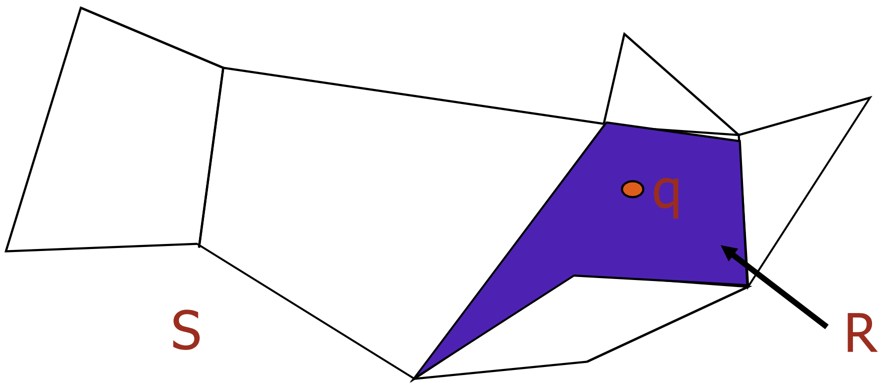
Questions?



1. How much is the preprocessing Time?

2. How much space is required?

Questions?



1. How much is the preprocessing Time?
2. How much space is required?
3. How much is the query time?

Organization of the Talk

1. Preliminaries, Generic definition and Literature

2. Some technical details of easy versions

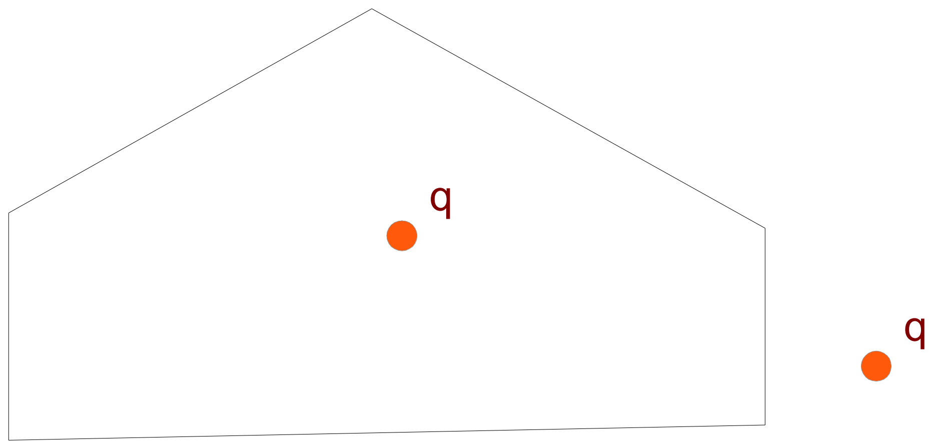
3. Conclusion

For convex polygon

P $\rightarrow$ The Geometric Objects is a Convex Polygon with n edges

For convex polygon

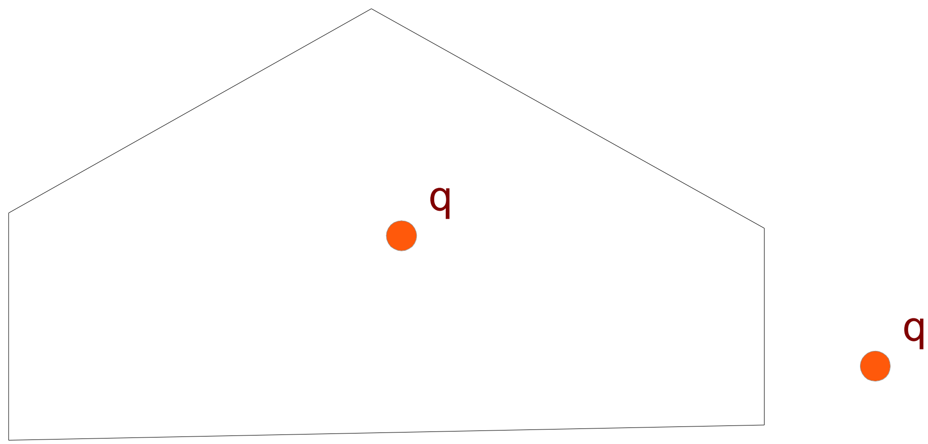
P → The Geometric Objects is a Convex Polygon with n edges



For convex polygon

P $\rightarrow$ The Geometric Objects is a Convex Polygon with n edges

Whether q is inside or outside?

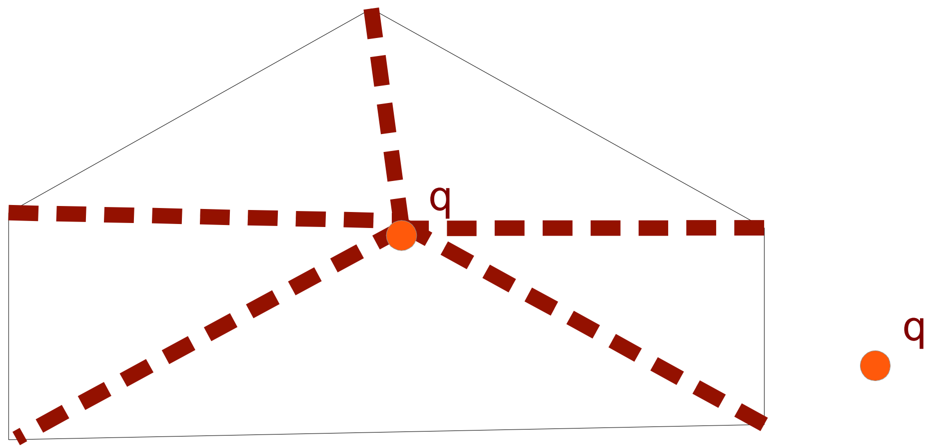


For convex polygon

P $\rightarrow$ The Geometric Objects is a Convex Polygon with n edges

Whether q is inside or outside?

Clue!! from area

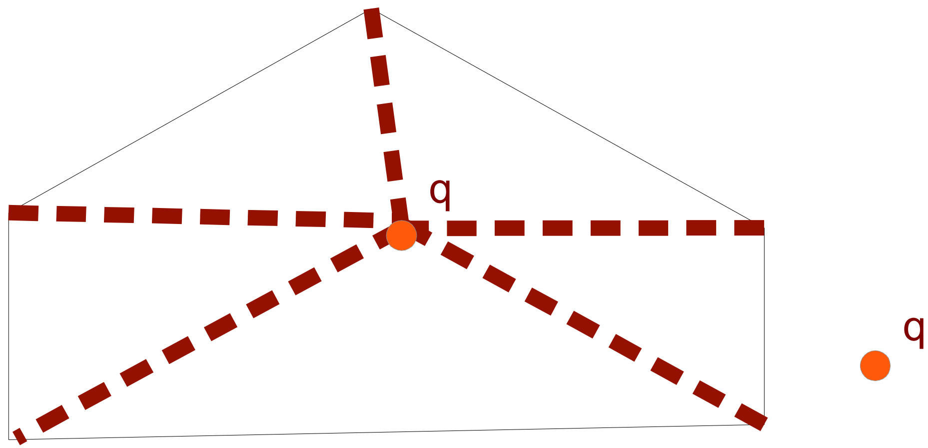


For convex polygon

P → The Geometric Objects is a Convex Polygon with n edges

Whether q is inside or outside?

Clue!! from area



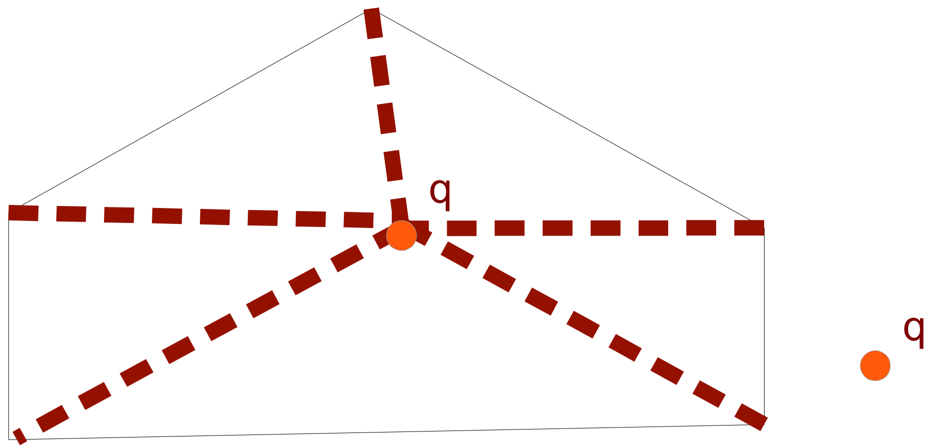
Complexity:

For convex polygon

P → The Geometric Objects is a Convex Polygon with n edges

Whether q is inside or outside?

Clue!! from area

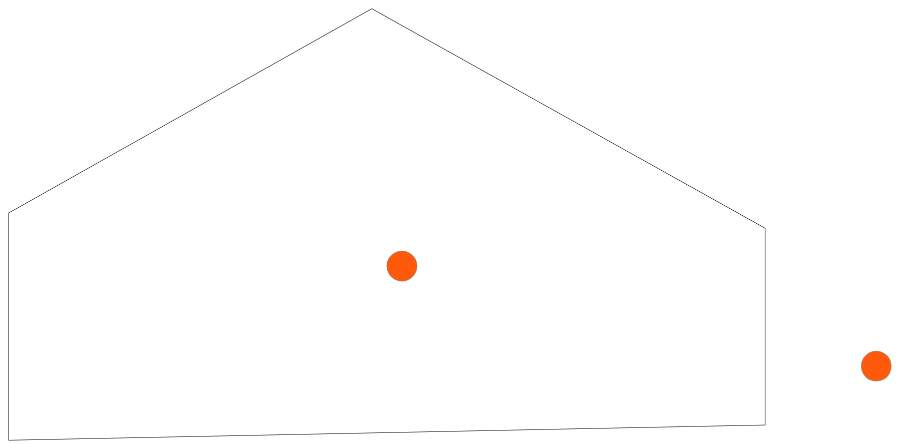


Complexity: O(n)

For convex polygon

P $\rightarrow$ The Geometric Objects is a Convex Polygon with n edges

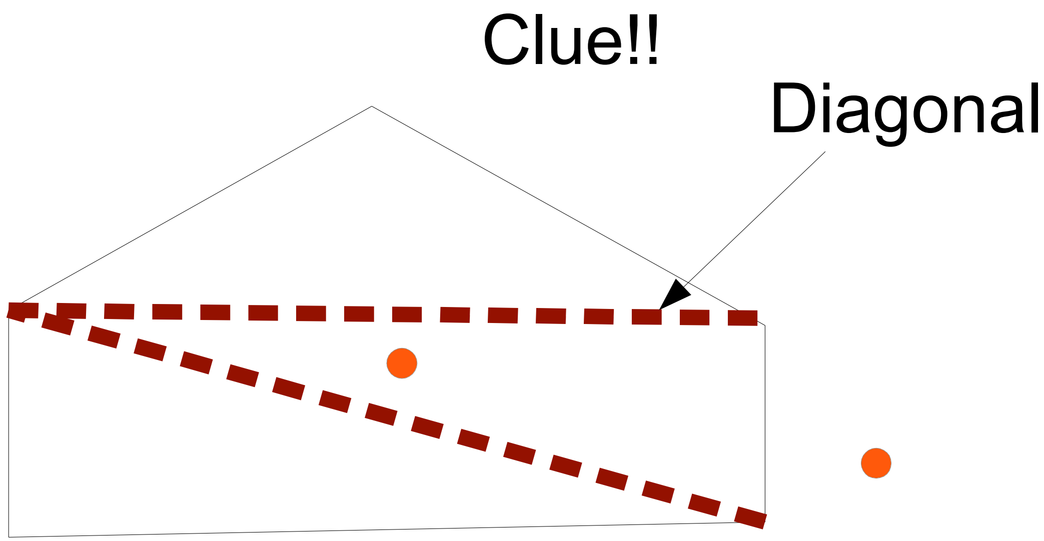
Now preprocess P to answer the question quickly



For convex polygon

P → The Geometric Objects is a Convex Polygon with n edges

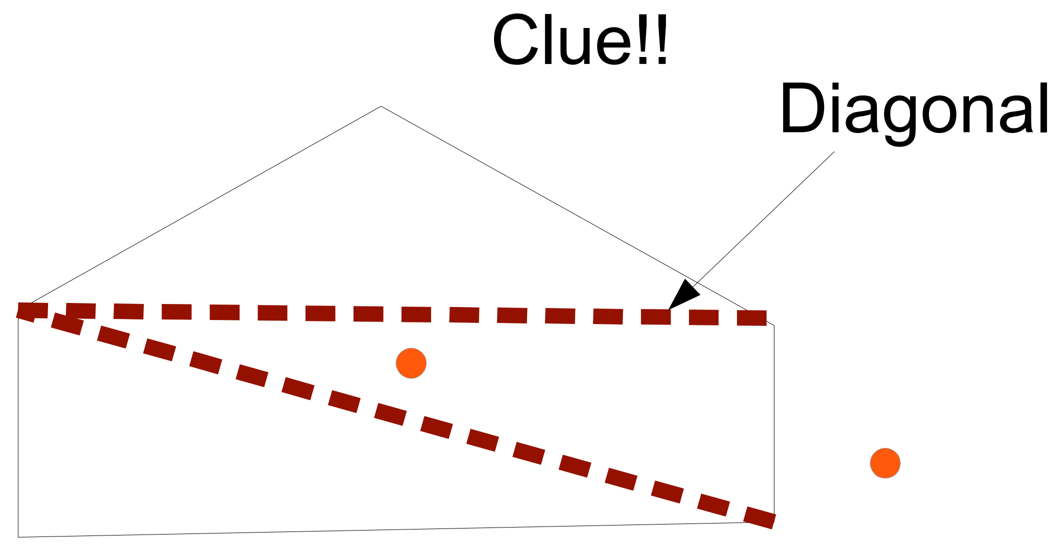
Now preprocess P to answer the question quickly



For convex polygon

P → The Geometric Objects is a Convex Polygon with n edges

Now preprocess P to answer the question quickly



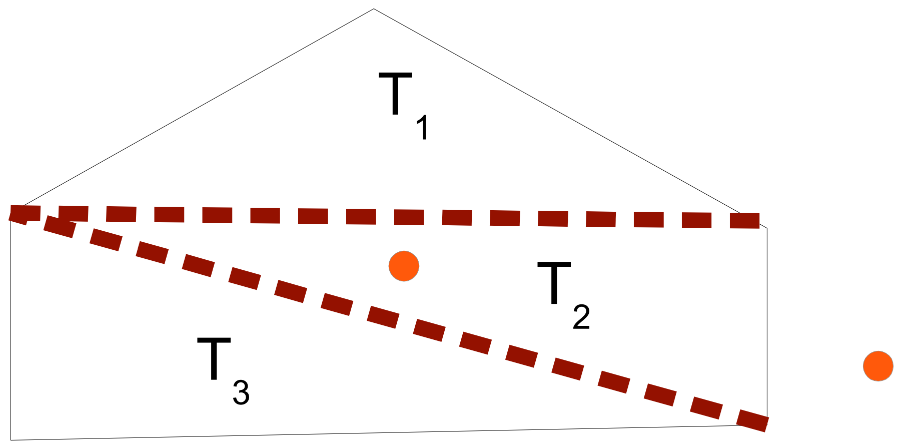
Triangulated Convex polygon

For convex polygon

P $\rightarrow$ The Geometric Objects is a Convex Polygon with n edges

Now preprocess P to answer the question quickly

Clue!!

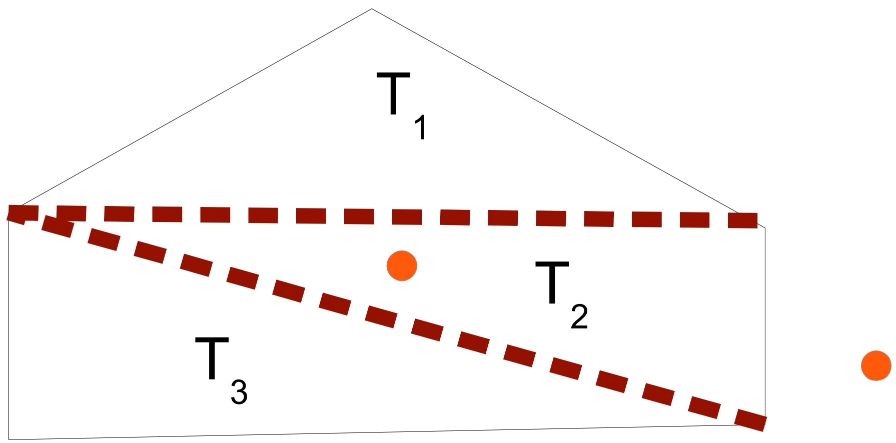


For convex polygon

P → The Geometric Objects is a Convex Polygon with n edges

Now preprocess P to answer the question quickly

Clue!!



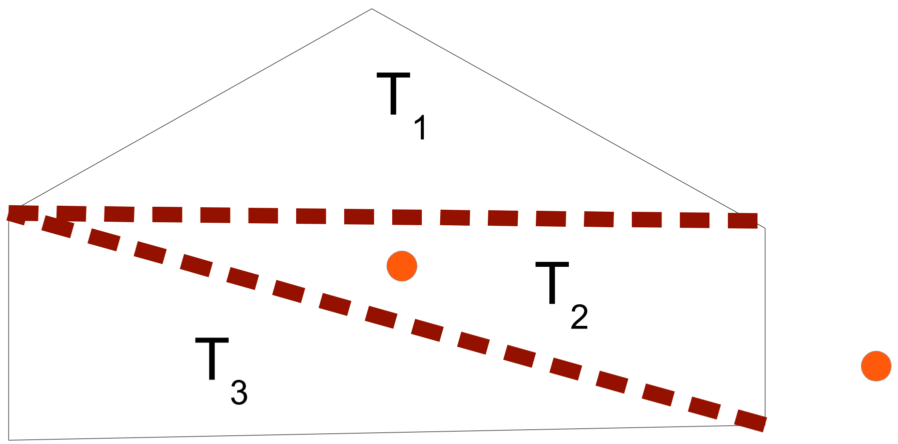
May we have an angular order of the triangles please!!

For convex polygon

P $\rightarrow$ The Geometric Objects is a Convex Polygon with n edges

Now preprocess P to answer the question quickly

Clue!!



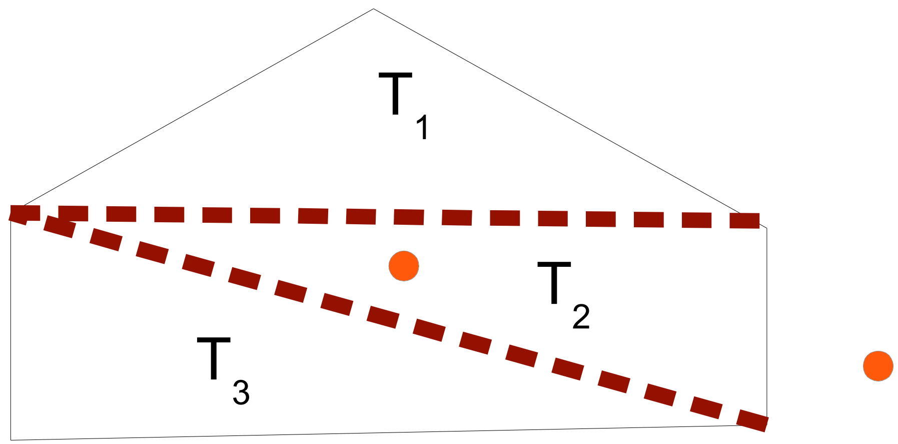
May we have an angular order of the triangles please!!

And then!!

Results

P → The Geometric Objects is a Convex Polygon with n edges

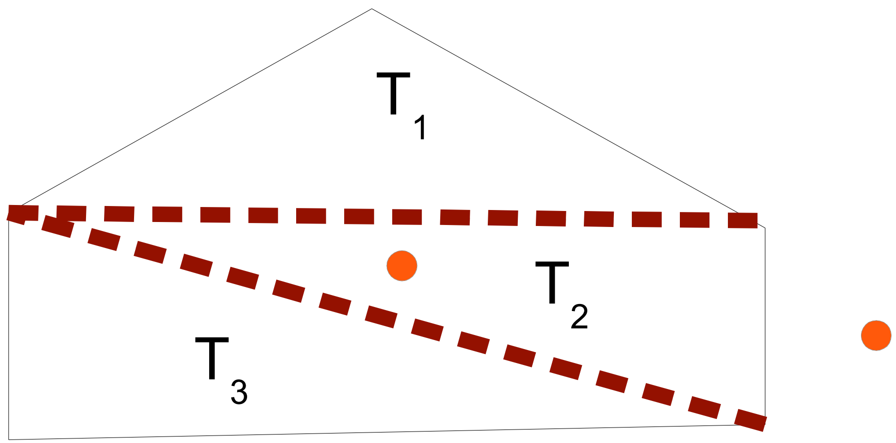
Now preprocess P to answer the question quickly



Results

P → The Geometric Objects is a Convex Polygon with n edges

Now preprocess P to answer the question quickly



Preprocessing Time: O(n)

Preprocessing space requirement: O(n)

Query Time: O(log n)

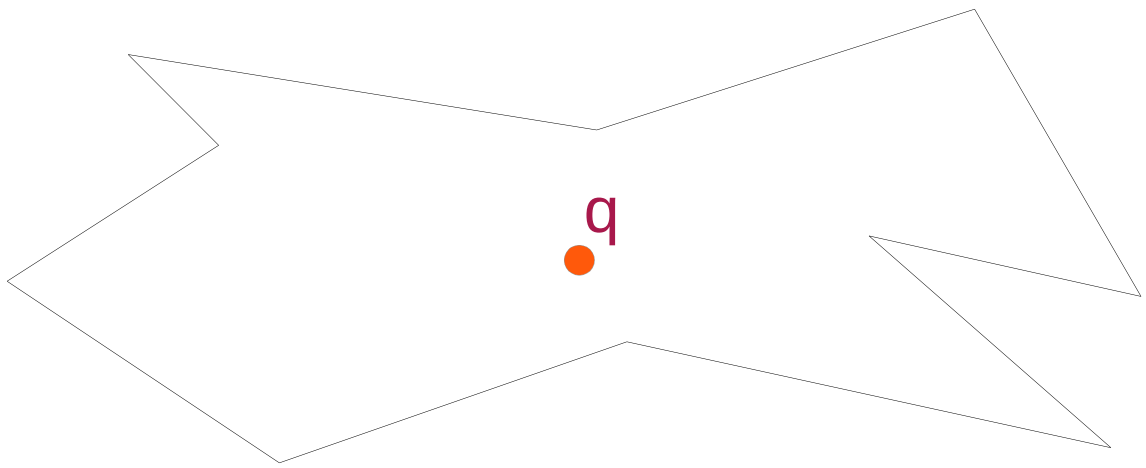
For simple polygon

P $\rightarrow$ The Geometric Objects is a simple Polygon with n edges

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Whether q is inside or outside?

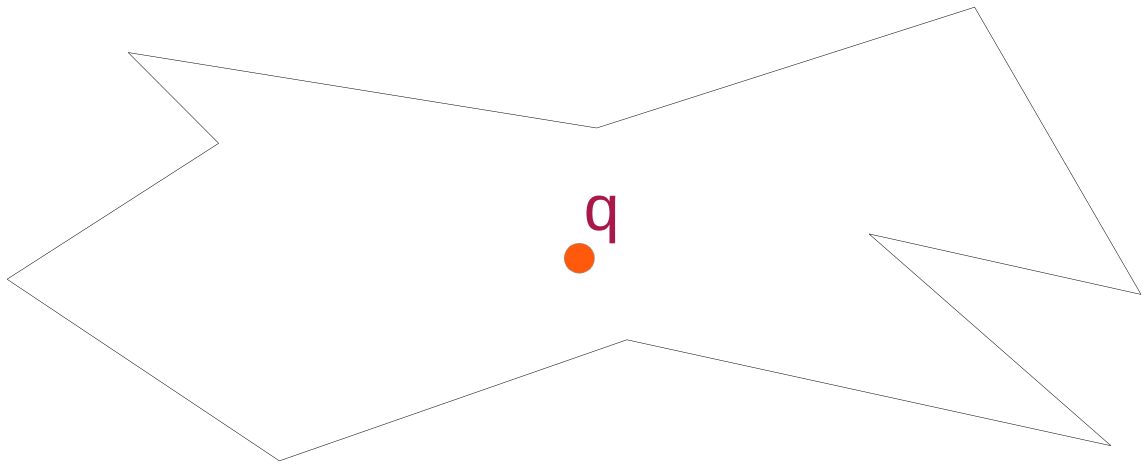


For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Whether q is inside or outside?

What about Brute Force!!

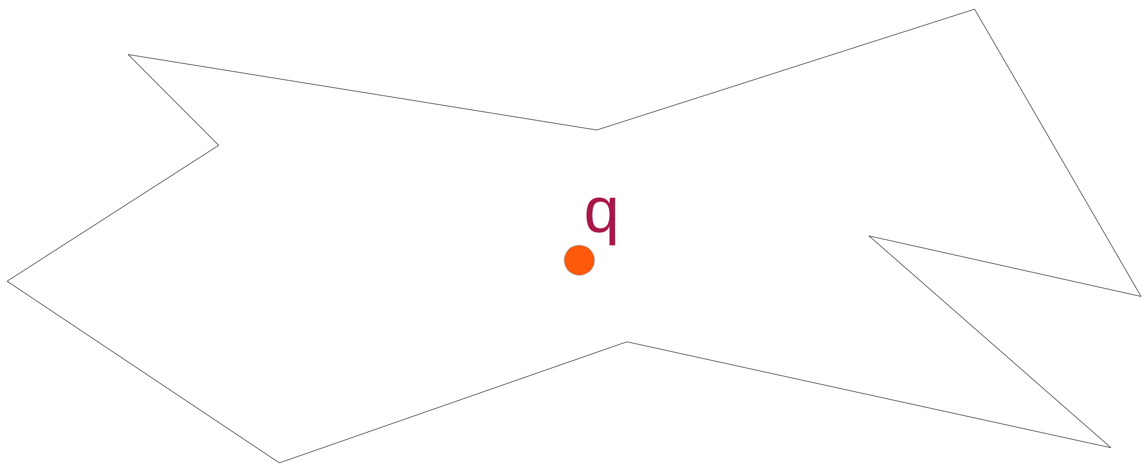


For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Whether q is inside or outside?

What about Brute Force!!



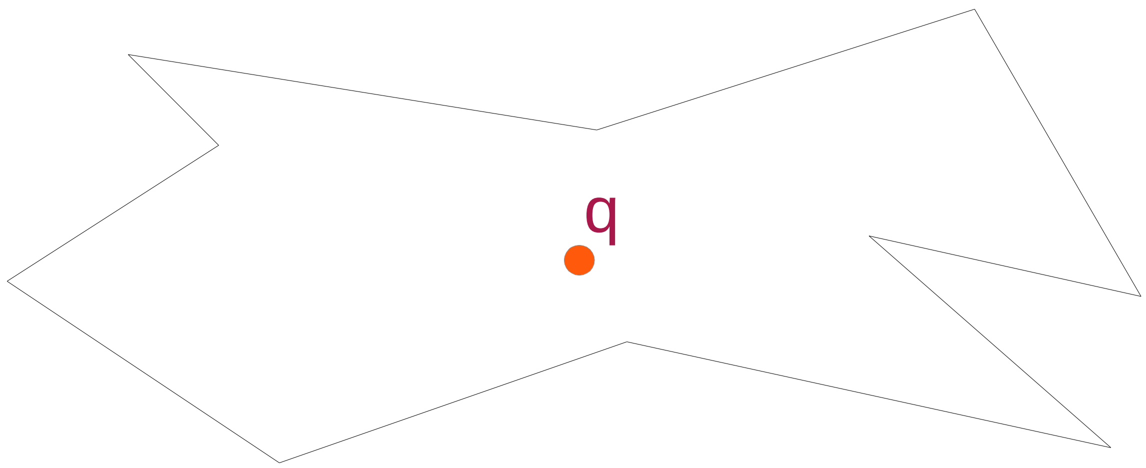
Can we use the convex polygon method/ area method?

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Whether q is inside or outside?

What about Brute Force!!



Can we use the convex polygon method/ area method?

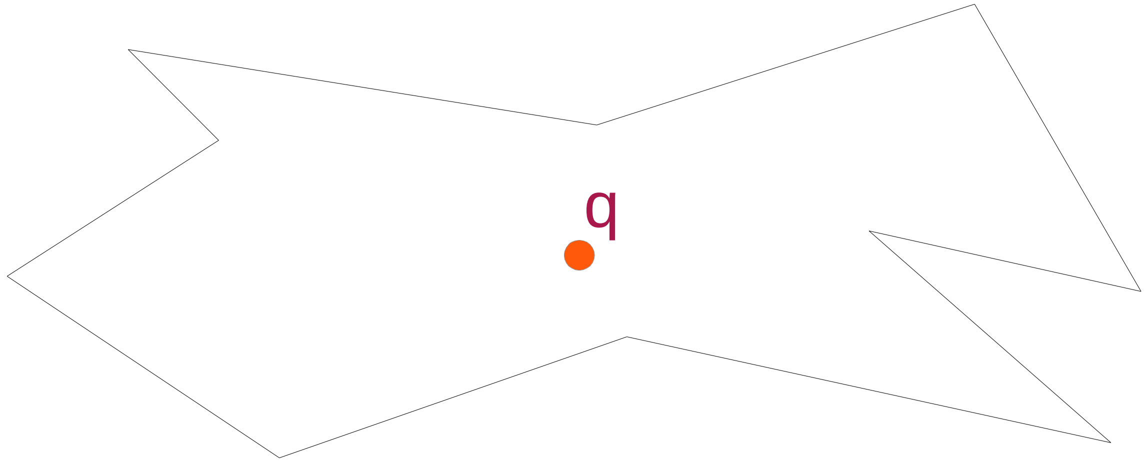
Not easily!!

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Whether q is inside or outside?

What about Brute Force!!



Can we use the convex polygon method/ area method?

Not easily!!

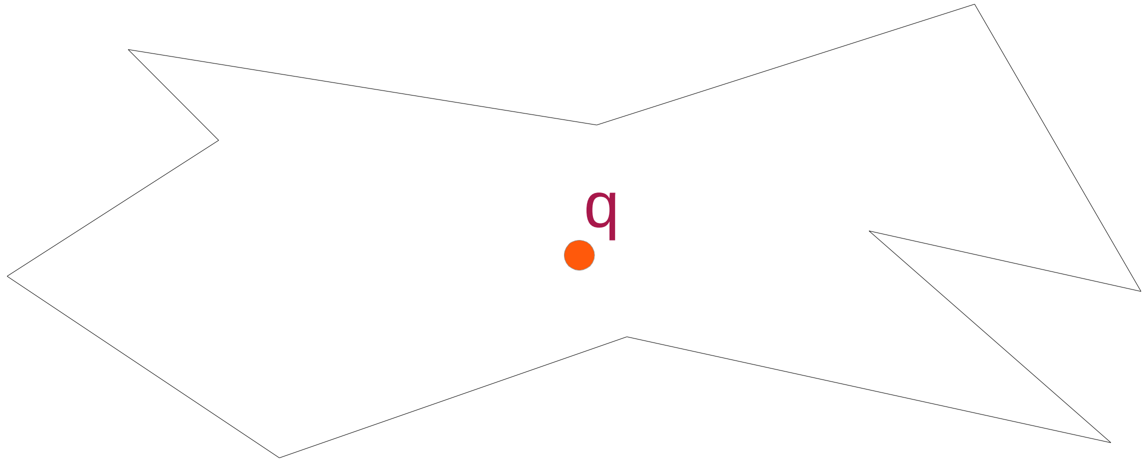
Then how do we solve??

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Whether q is inside or outside?

What about Brute Force!!



Can we use the convex polygon method/ area method?

Not easily!!

Then how do we solve??

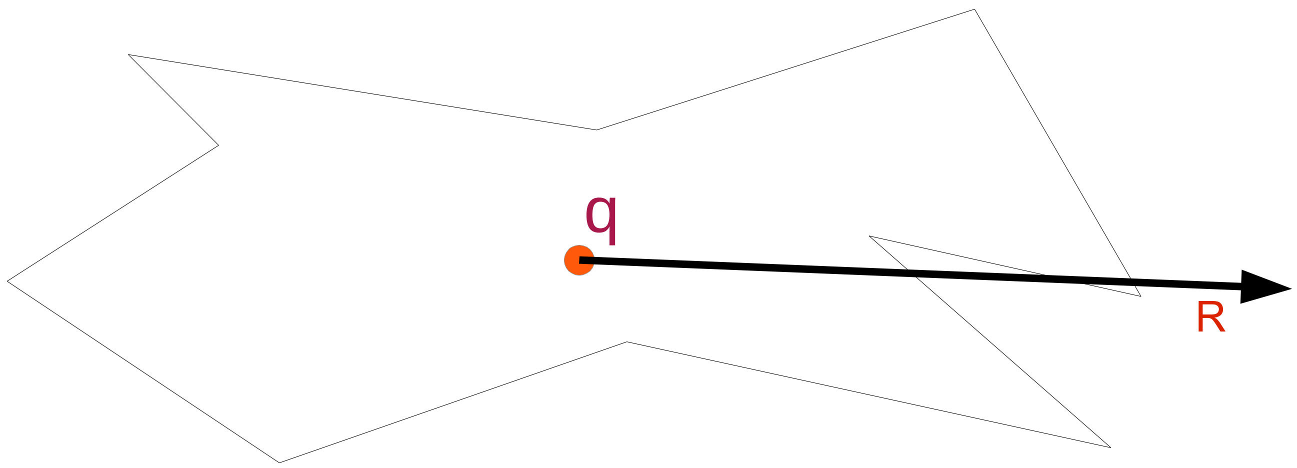
Shoot a horizontal ray R from q towards positive x-direction

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Whether q is inside or outside?

What about Brute Force!!



Can we use the convex polygon method/ area method?

Not easily!!

Then how do we solve??

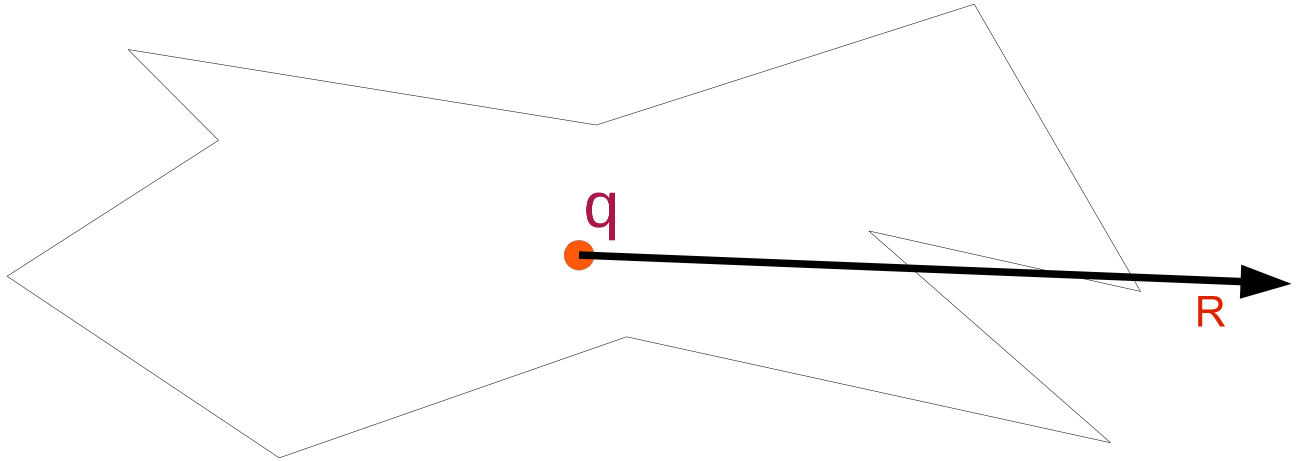
Shoot a horizontal ray R from q towards positive x-direction

Odd number of intersection between R and P implies ==>>

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Whether q is inside or outside? What about Brute Force!!



Can we use the convex polygon method/ area method? Not easily!!

Then how do we solve??

Shoot a horizontal ray R from q towards positive x-direction

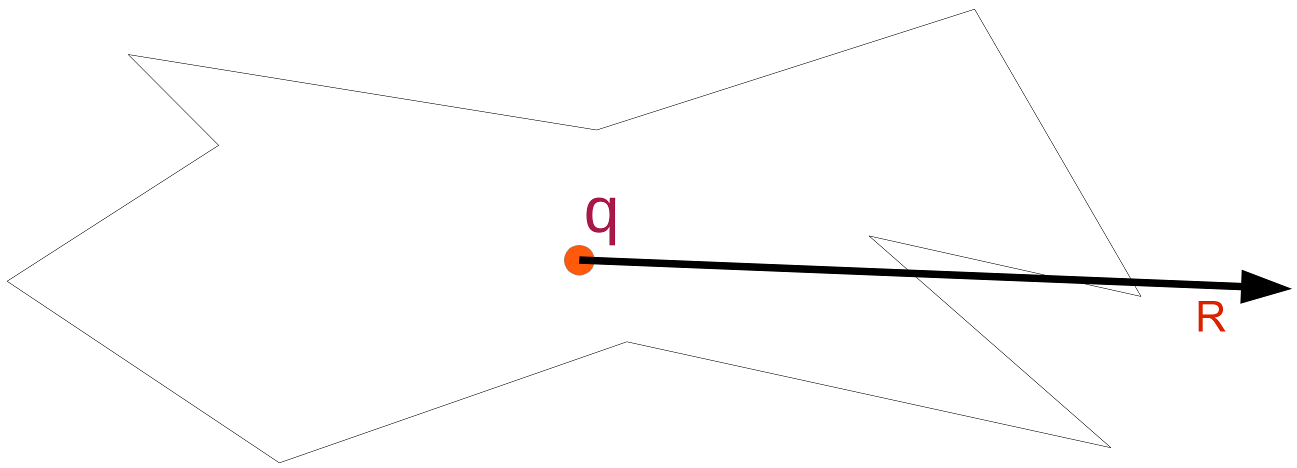
Odd number of intersection between R and P implies ==>> q is inside P

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Whether q is inside or outside?

What about Brute Force!!



Can we use the convex polygon method/ area method?

Not easily!!

Then how do we solve??

Shoot a horizontal ray R from q towards positive x-direction

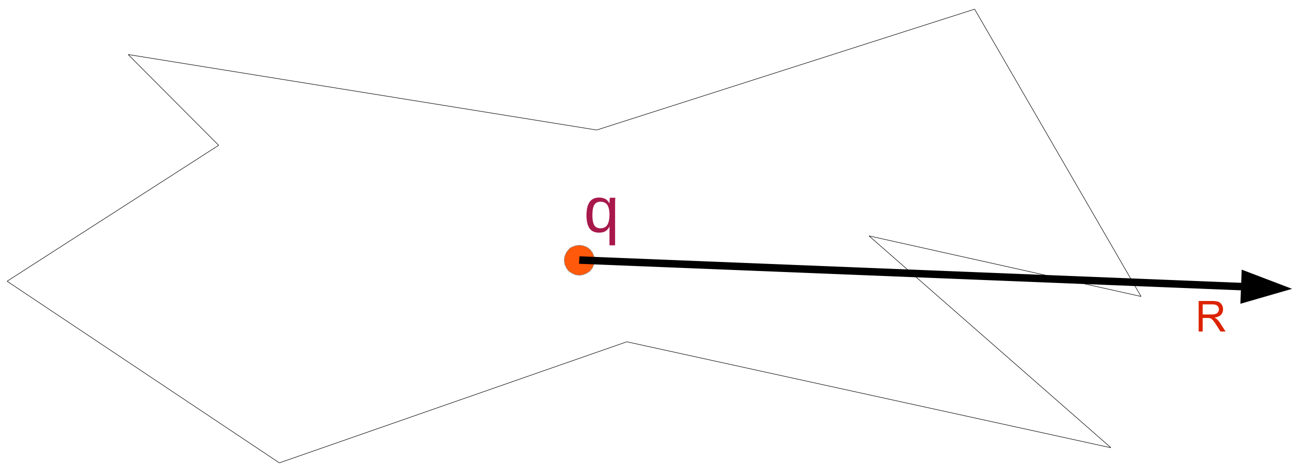
Odd number of intersection between R and P implies ==>>

Complexity:

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Whether q is inside or outside? What about Brute Force!!



Can we use the convex polygon method/ area method? Not easily!!

Then how do we solve??

Shoot a horizontal ray R from q towards positive x-direction

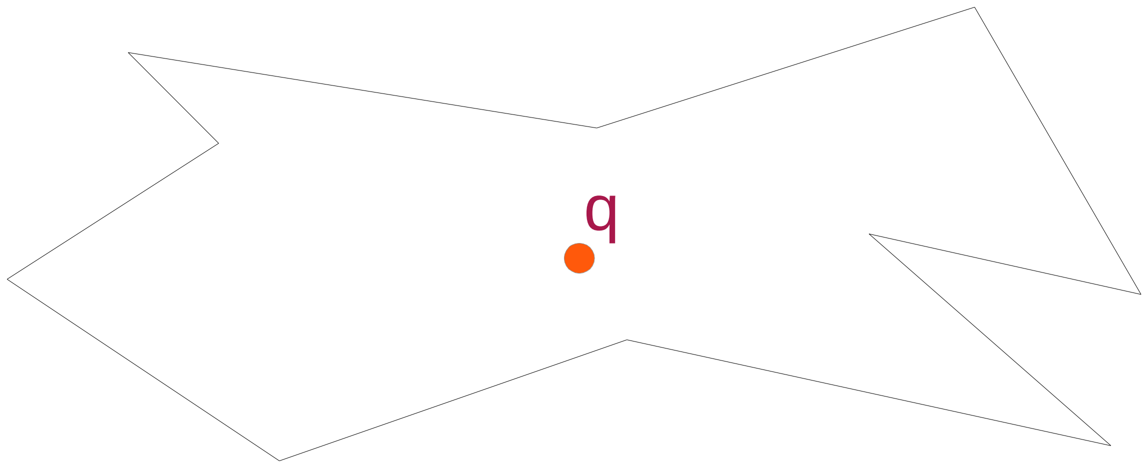
Odd number of intersection between R and P implies ==>>

Complexity: O(n)

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

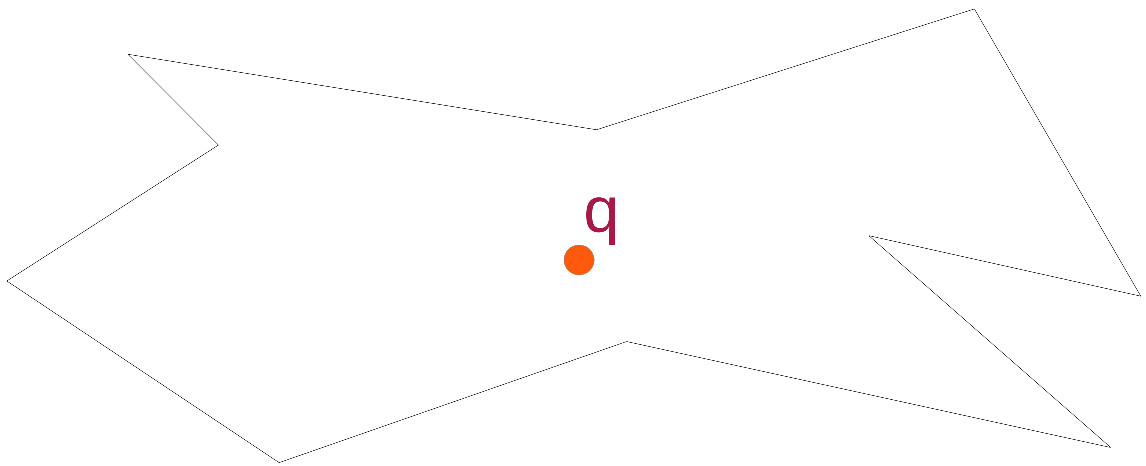
Now preprocess P to answer the question quickly



For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Now preprocess P to answer the question quickly

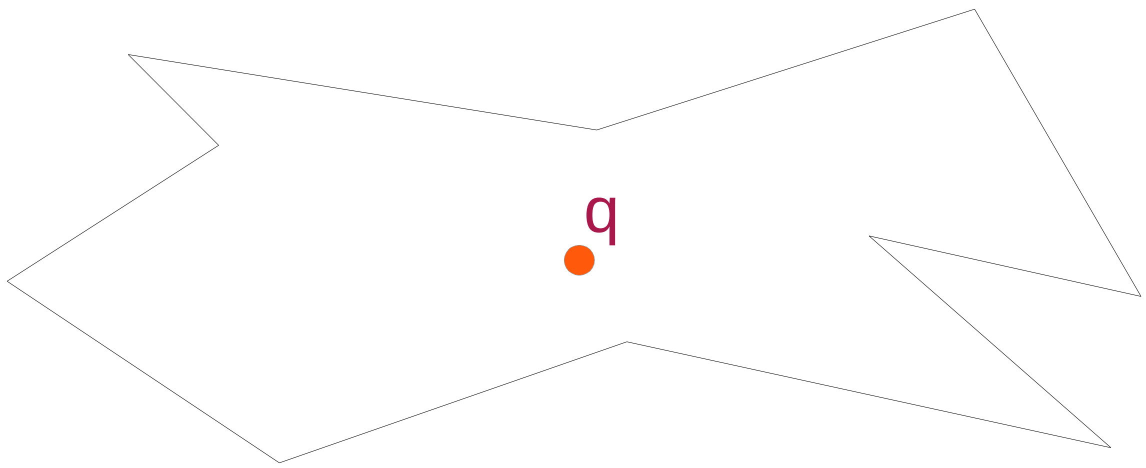


Can we use the convex polygon triangulation method?

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Now preprocess P to answer the question quickly

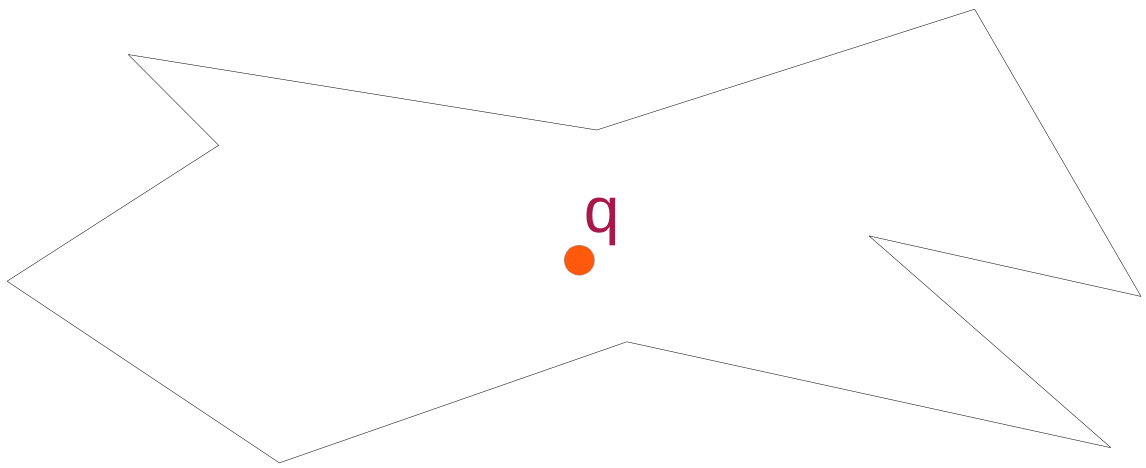


Can we use the convex polygon triangulation method? May be yes!!

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Now preprocess P to answer the question quickly



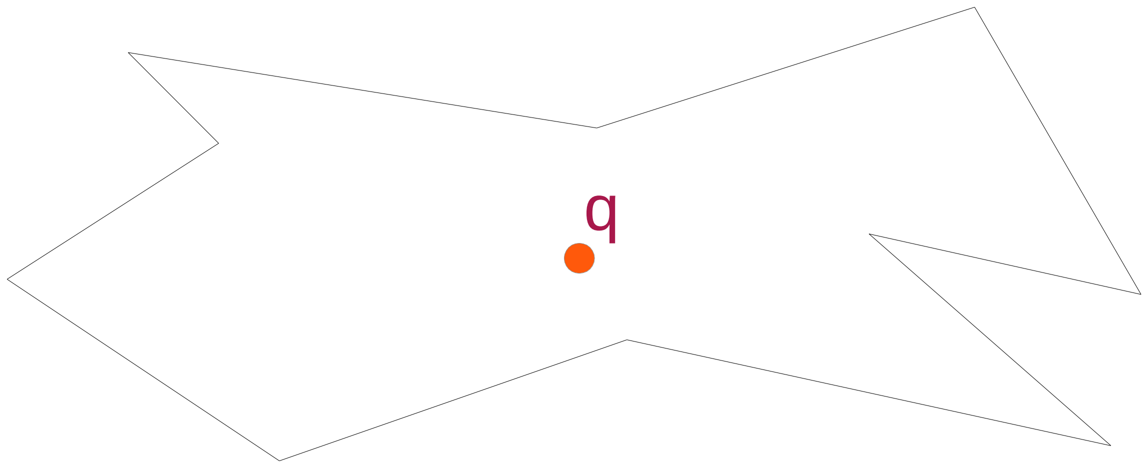
Can we use the convex polygon triangulation method? May be yes!!

But how do we triangulate??

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Now preprocess P to answer the question quickly



Can we use the convex polygon triangulation method? May be yes!!

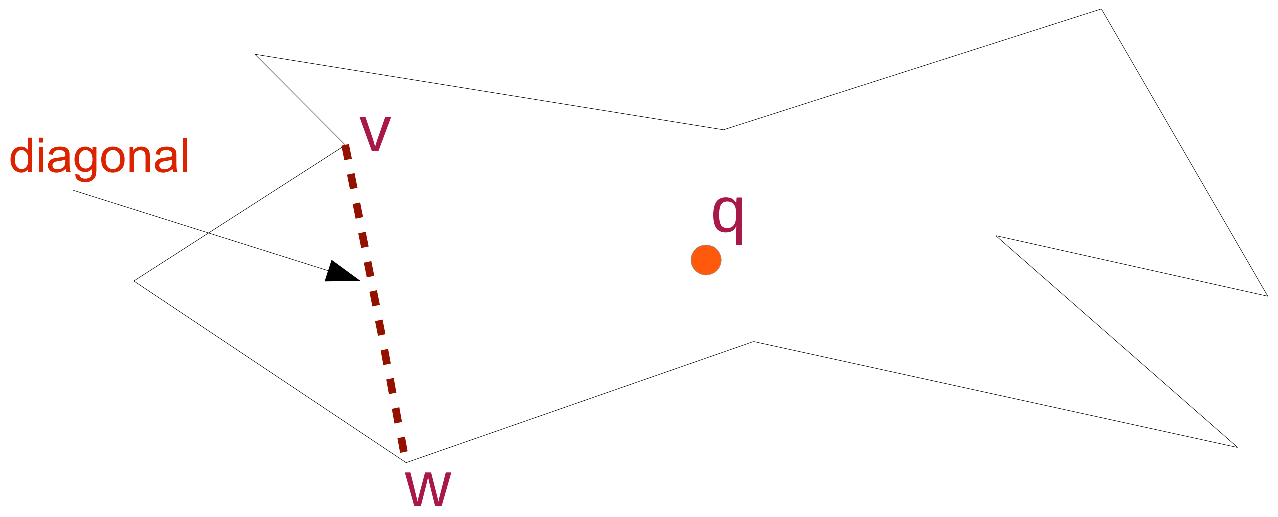
But how do we triangulate??

How easily can we draw a diagonal??

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Now preprocess P to answer the question quickly



Can we use the convex polygon triangulation method? May be yes!!

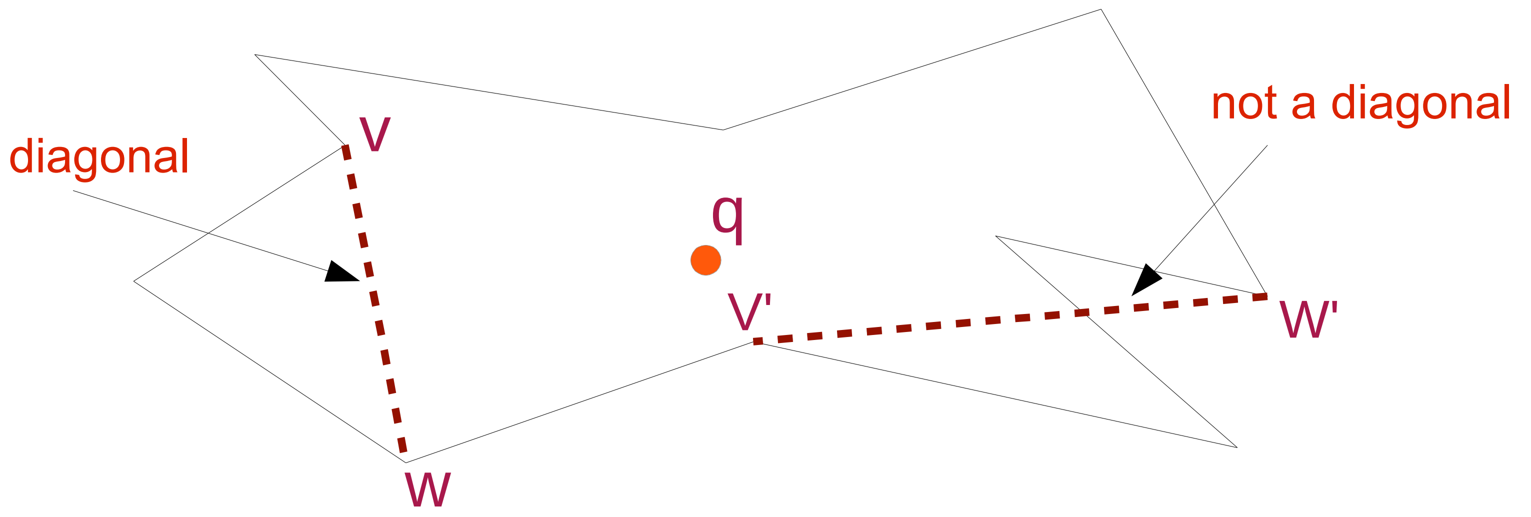
But how do we triangulate??

How easily can we draw a diagonal??

For simple polygon

P → The Geometric Objects is a simple Polygon with n edges

Now preprocess P to answer the question quickly



Can we use the convex polygon triangulation method? May be yes!!

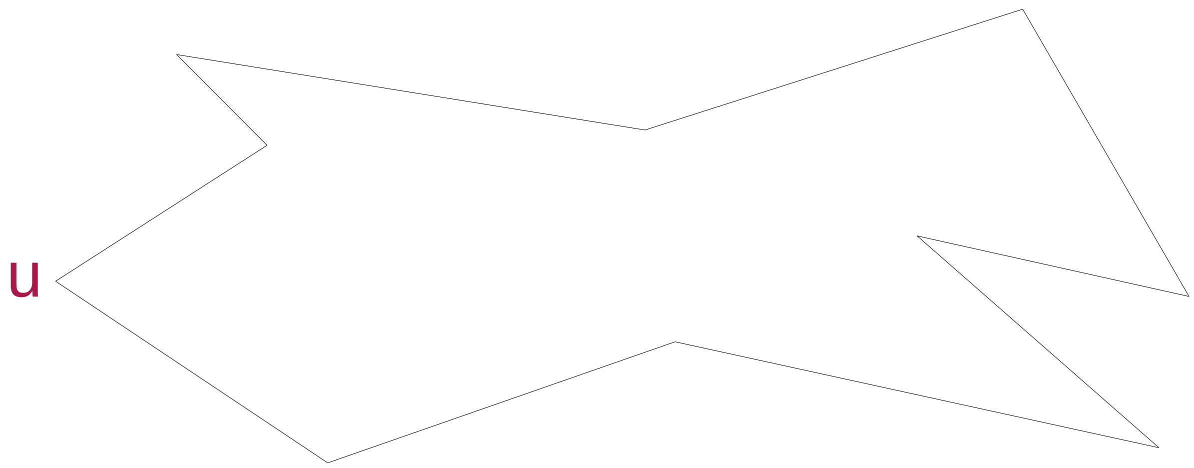
But how do we triangulate??

How easily can we draw a diagonal??

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

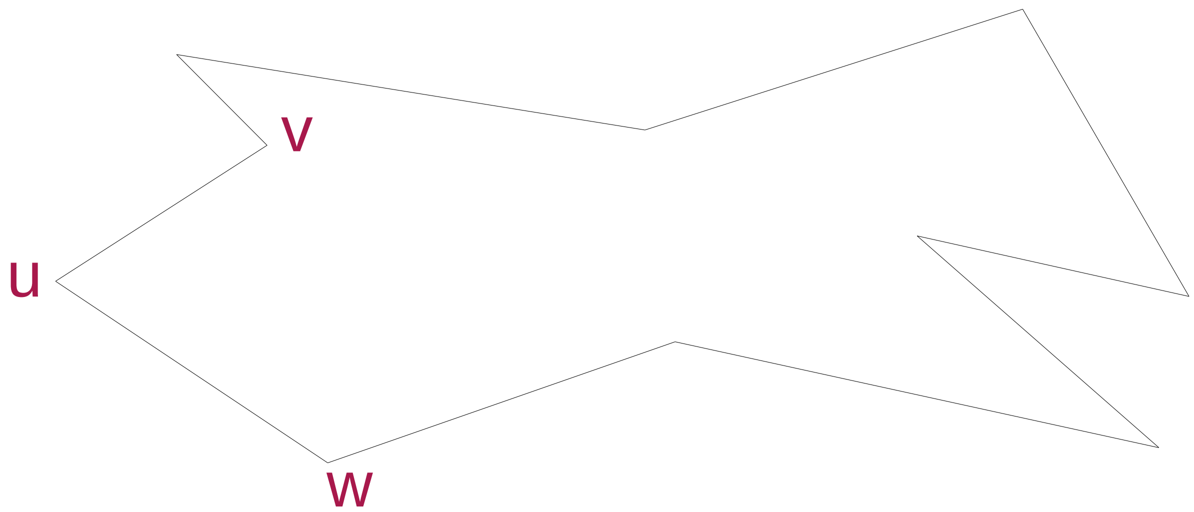
Leftmost vertex u of P is a convex vertex (interior angle less than 180°)



Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Leftmost vertex u of P is a convex vertex (interior angle less than 180°)

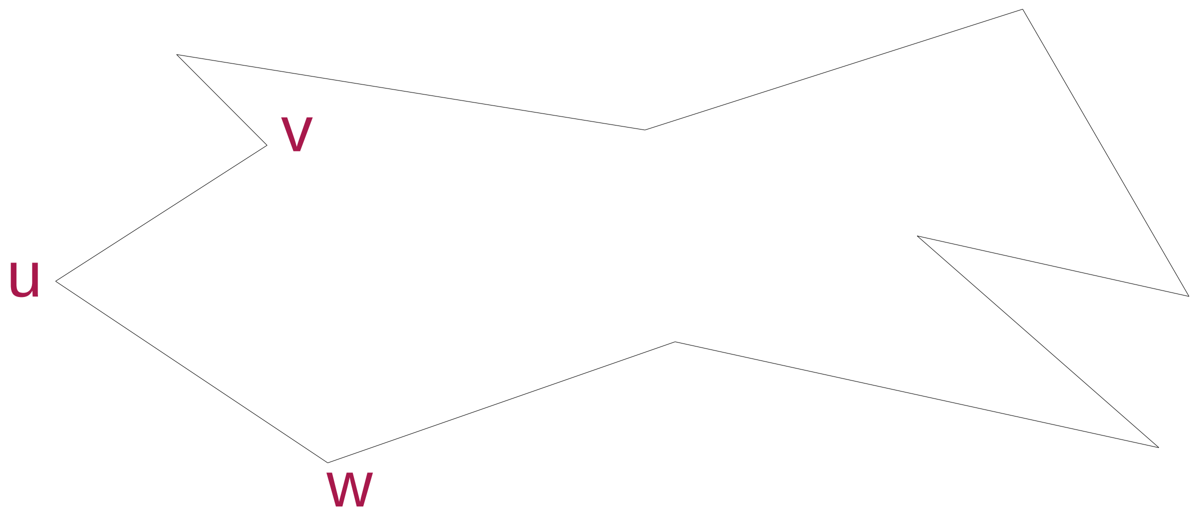


Let v and w be adjacent vertices of vertex u

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Leftmost vertex u of P is a convex vertex (interior angle less than 180°)



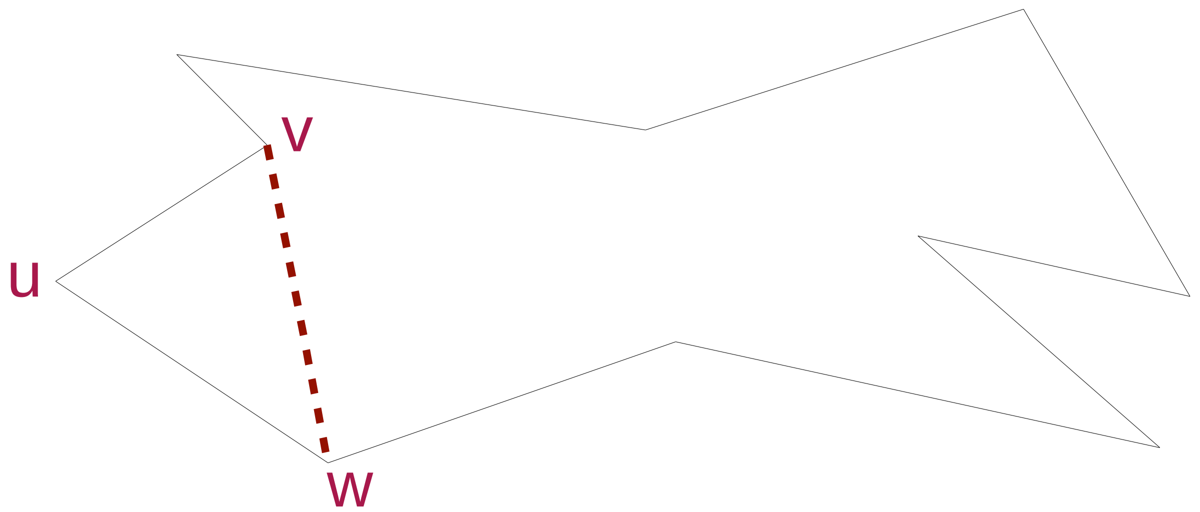
Let v and w be adjacent vertices of vertex u

If triangle uvw does not contain any other vertex u' then vw is a diagonal

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Leftmost vertex u of P is a convex vertex (interior angle less than 180°)



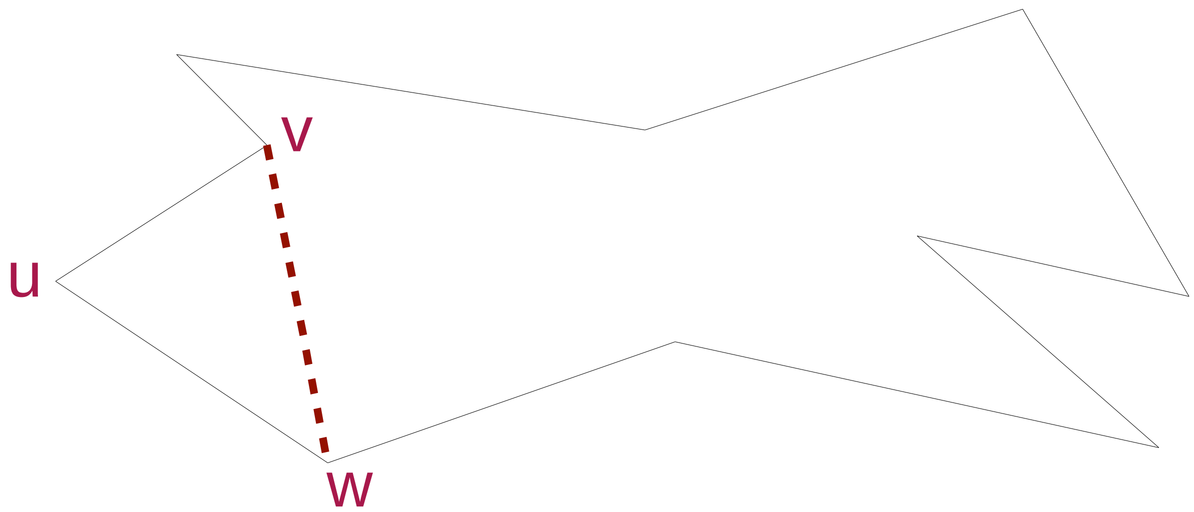
Let v and w be adjacent vertices of vertex u

If triangle uvw does not contain any other vertex u' then vw is a diagonal

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Leftmost vertex u of P is a convex vertex (interior angle less than 180°)



Let v and w be adjacent vertices of vertex u

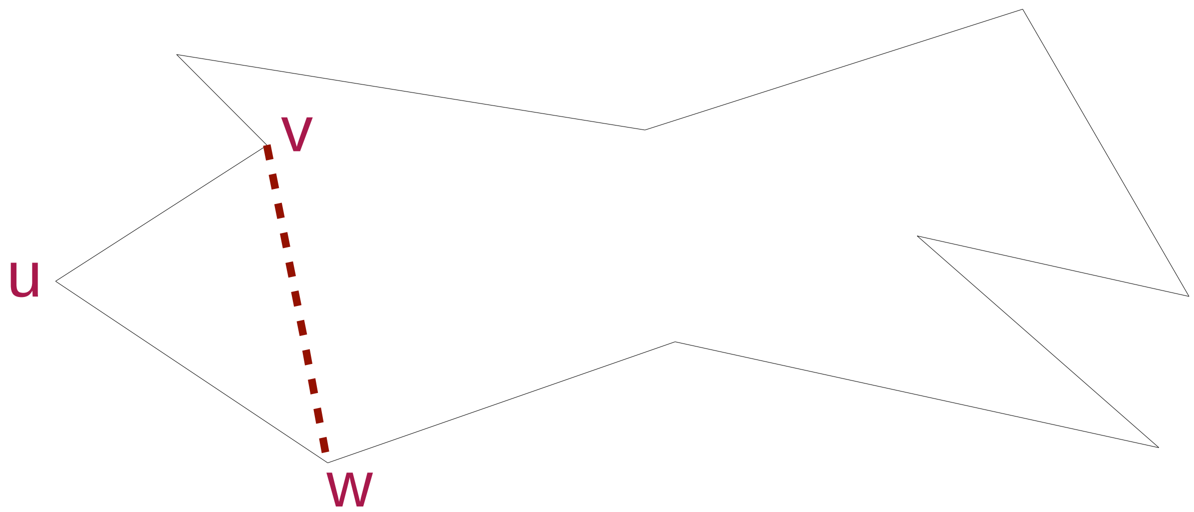
If triangle uvw does not contain any other vertex u' then vw is a diagonal

Will triangle uvw always be empty?

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Leftmost vertex u of P is a convex vertex (interior angle less than 180°)



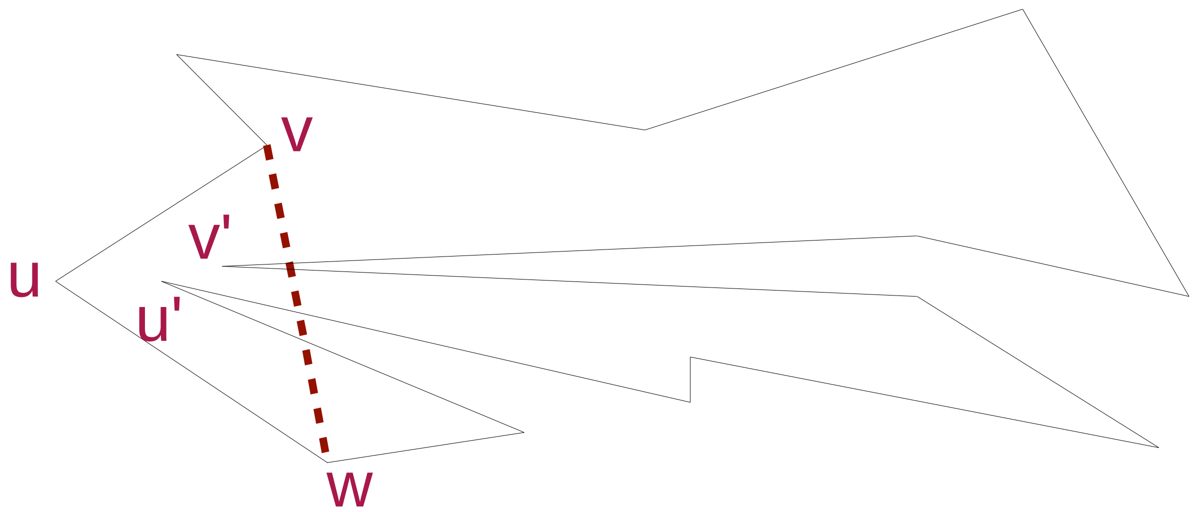
Let v and w be adjacent vertices of vertex u

If triangle uvw does not contain any other vertex u' then vw is a diagonal

Will triangle uvw always be empty? No

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

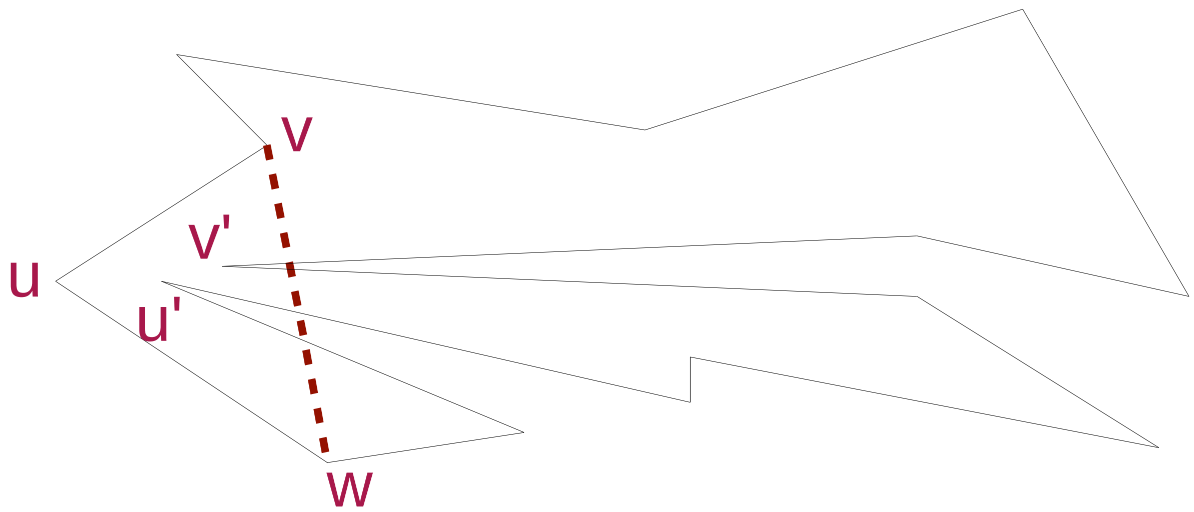


What should be the characteristic of u' so that uu' is a diagonal

Will triangle uvw always be empty? No

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges



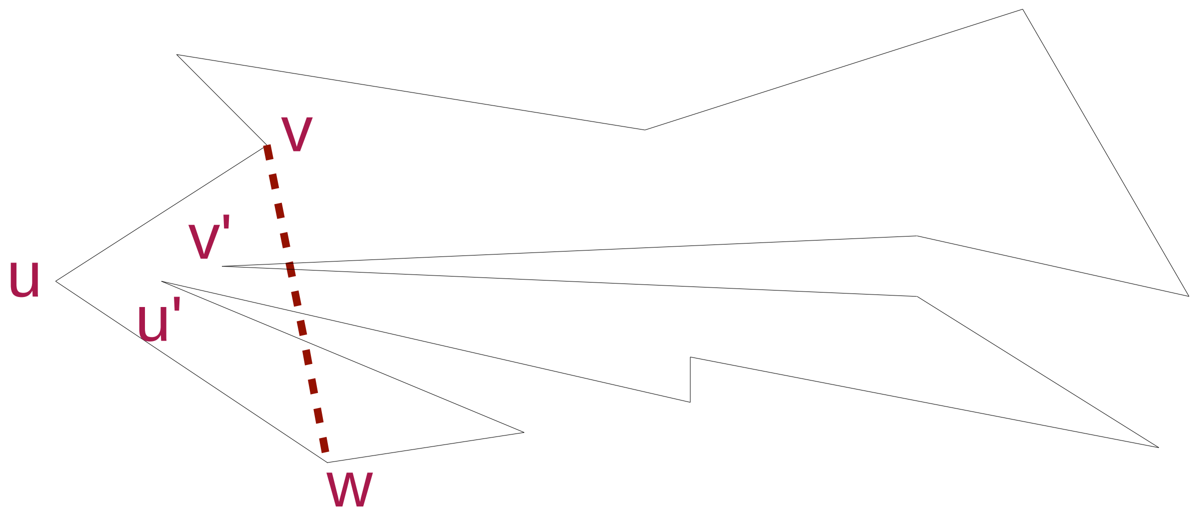
What should be the characteristic of u' so that uu' is a diagonal

Will triangle uvw always be empty? No

Then how do we solve??

Finding a diagonal in linear time

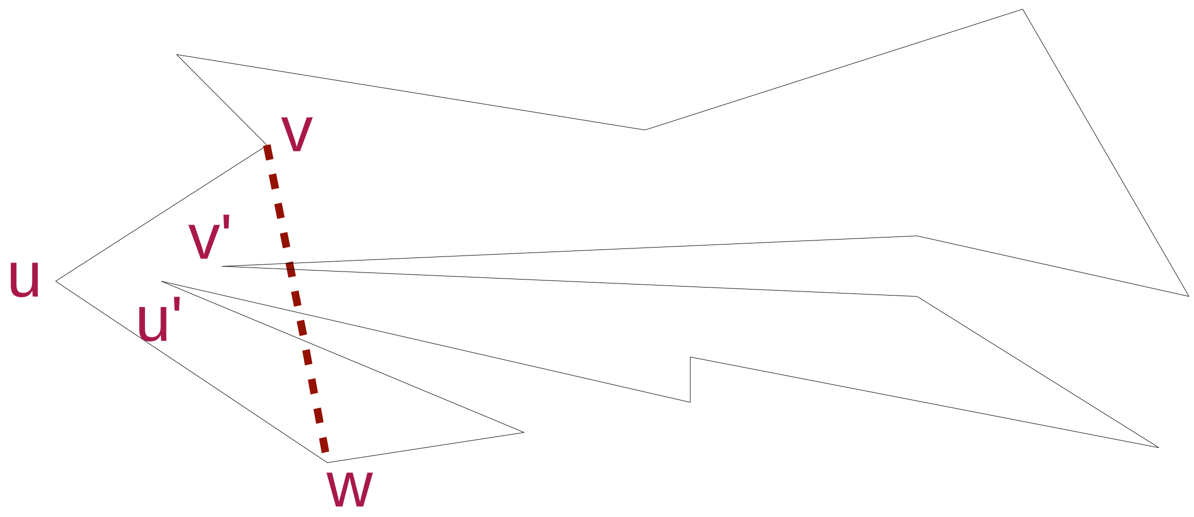
P → The Geometric Objects is a simple Polygon with n edges



Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

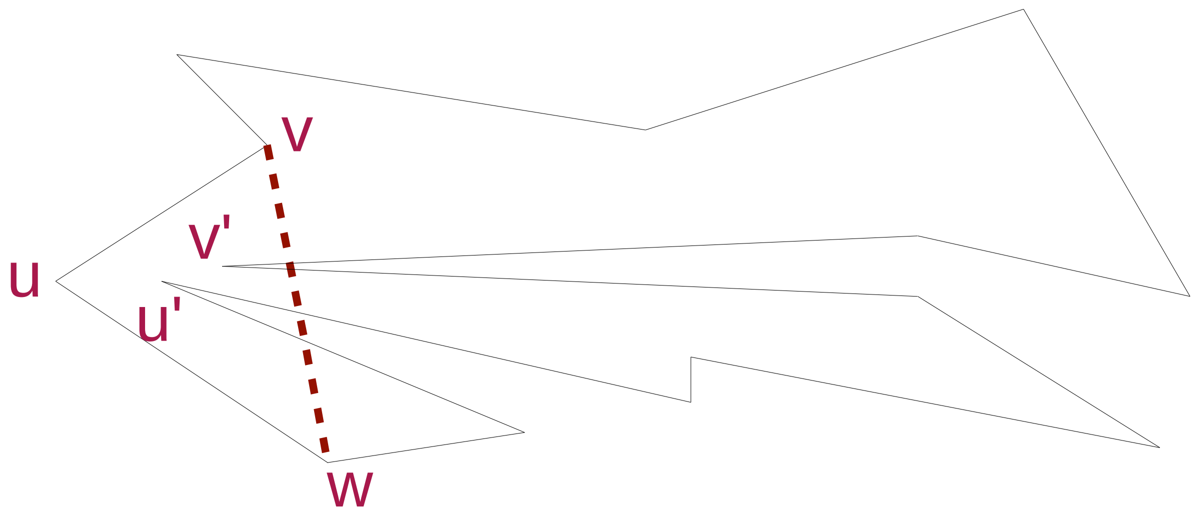
Can we find a vertex u' inside triangle uvw such that uu' is a diagonal?



Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

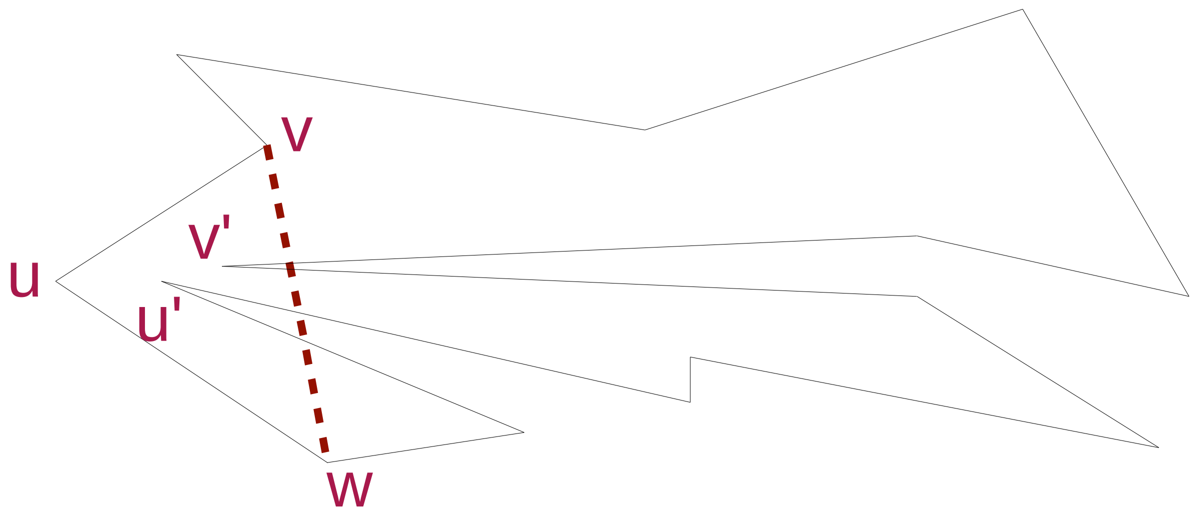
Can we find a vertex u' inside triangle uvw such that uu' is a diagonal? Yes



Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Can we find a vertex u' inside triangle uvw such that uu' is a diagonal?

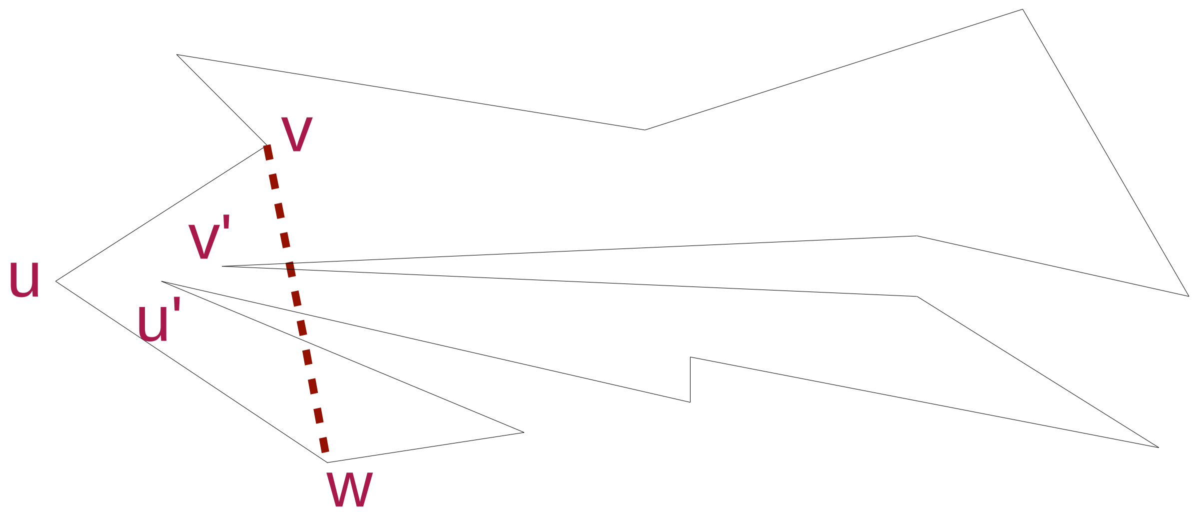


What should be the characteristic of u' so that uu' is a diagonal

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Can we find a vertex u' inside triangle uvw such that uu' is a diagonal



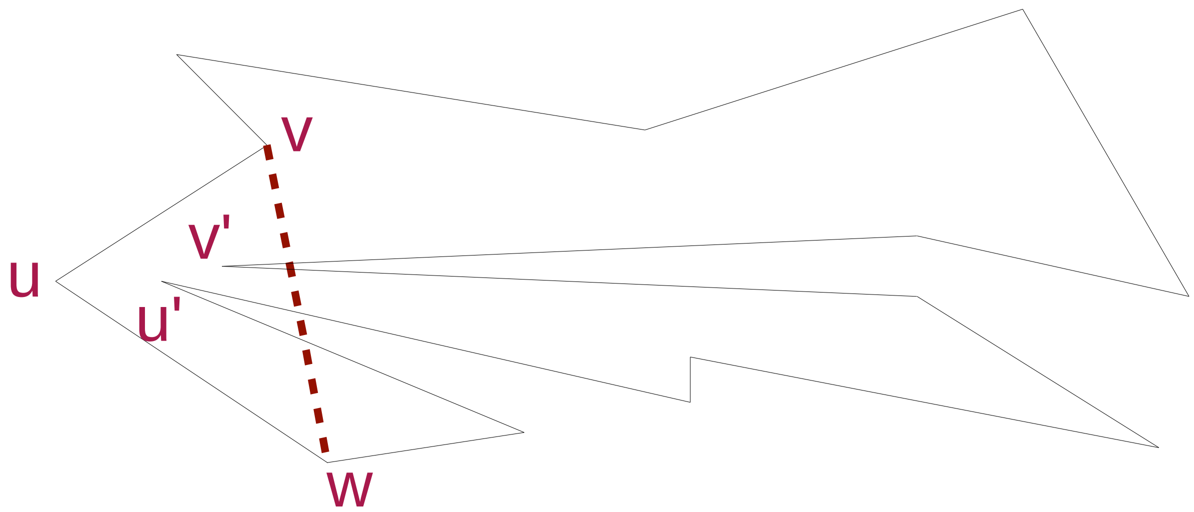
What should be the characteristic of u' so that uu' is a diagonal

What about closest vertex u' of u

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Can we find a vertex u' inside triangle uvw such that uu' is a diagonal



What should be the characteristic of u' so that uu' is a diagonal

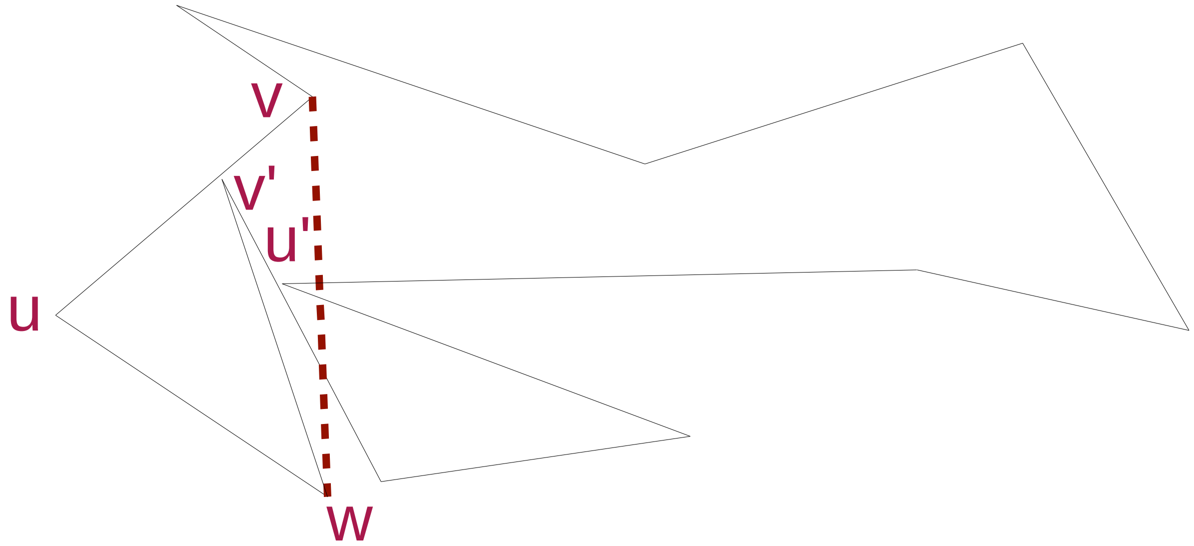
What about closest vertex u' of u

Wrong answer !!

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Can we find a vertex u' inside triangle uvw such that uu' is a diagonal



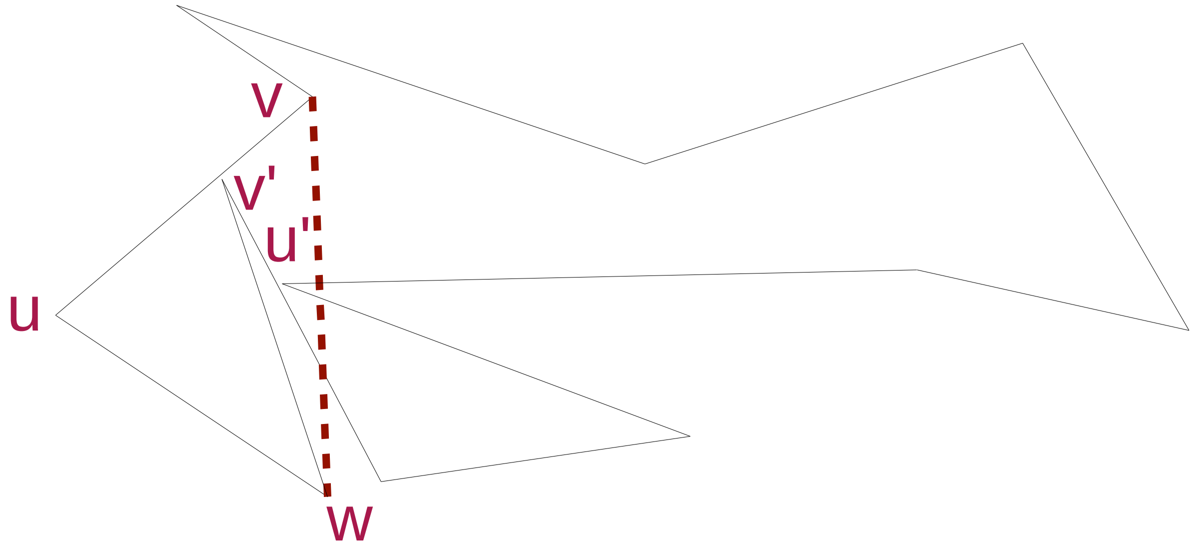
What should be the characteristic of u' so that uu' is a diagonal

What about closest vertex u' of u   Wrong answer !!

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Can we find a vertex u' inside triangle uvw such that uu' is a diagonal



What should be the characteristic of u' so that uu' is a diagonal

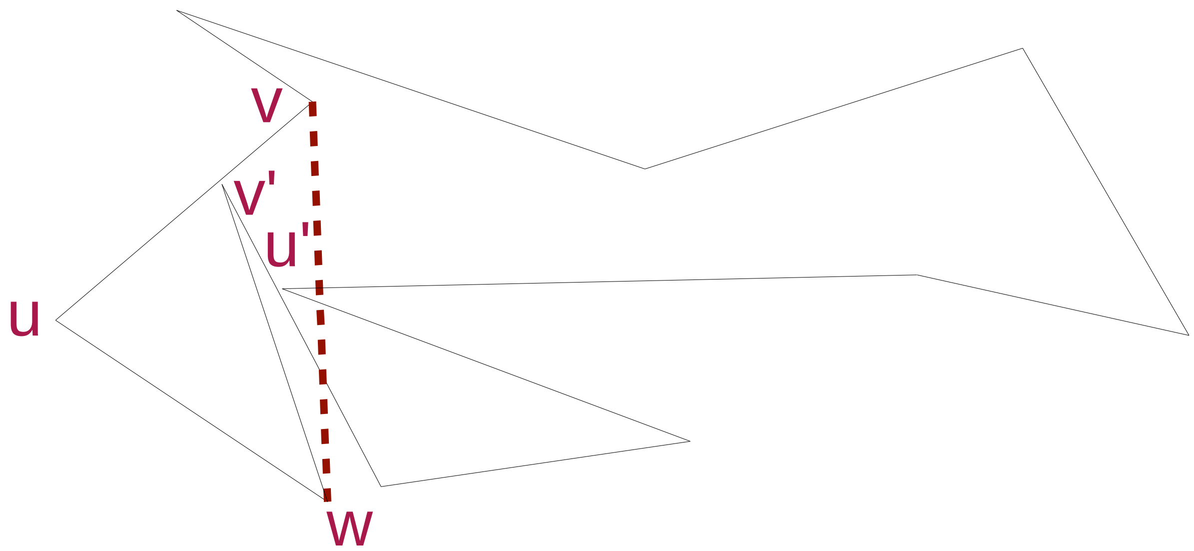
What about closest vertex u' of u Wrong answer !!

Correct answer is:

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Can we find a vertex u' inside triangle uvw such that uu' is a diagonal



What should be the characteristic of u' so that uu' is a diagonal

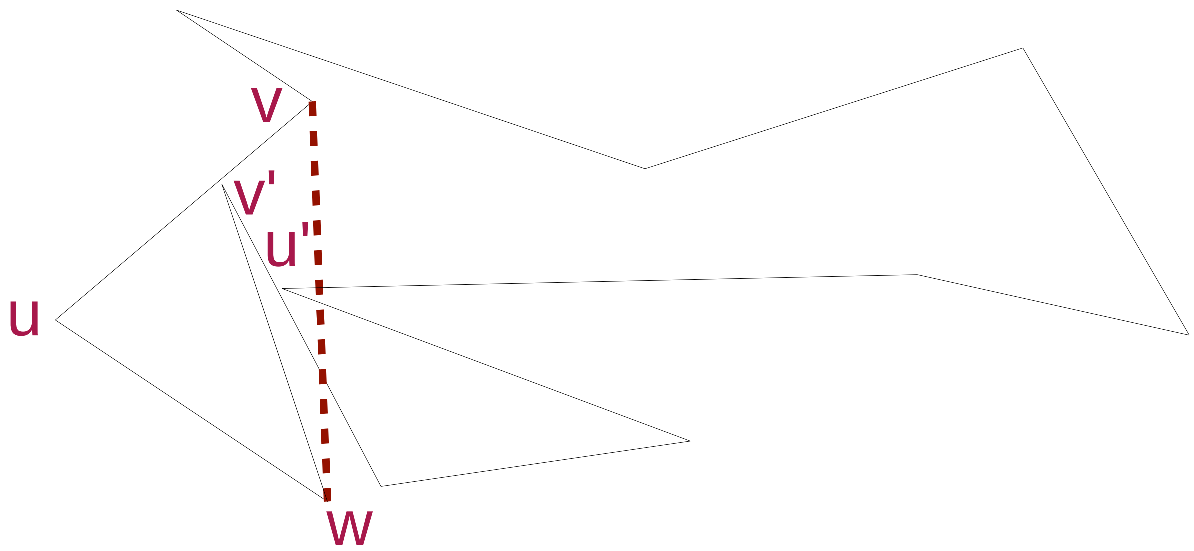
What about closest vertex u' of u   Wrong answer !!

Correct answer is: Leftmost vertex inside triangle uvw is u'

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Can we find a vertex u' inside triangle uvw such that uu' is a diagonal



What should be the characteristic of u' so that uu' is a diagonal

What about closest vertex u' of u   Wrong answer !!

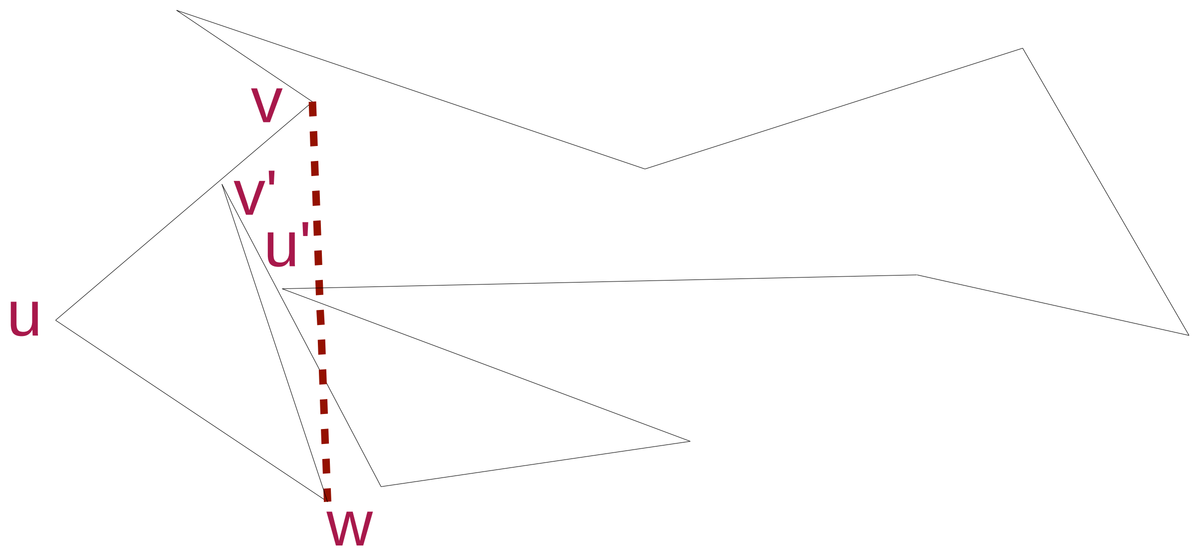
Correct answer is: Leftmost vertex inside triangle uvw is u'

Complexity:

Finding a diagonal in linear time

P → The Geometric Objects is a simple Polygon with n edges

Can we find a vertex u' inside triangle uvw such that uu' is a diagonal



What should be the characteristic of u' so that uu' is a diagonal

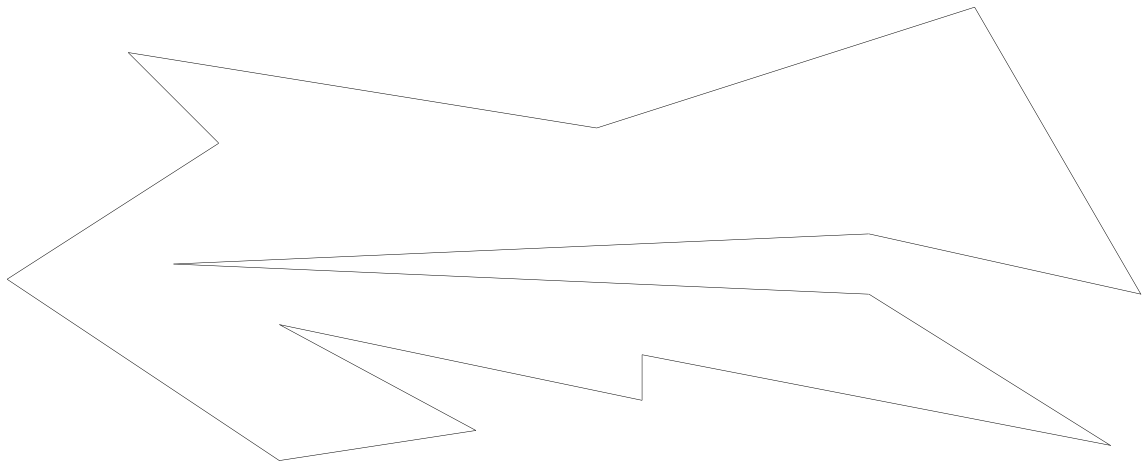
What about closest vertex u' of u   Wrong answer !!

Correct answer is: Leftmost vertex inside triangle uvw is u'

Complexity: O(n)

Triangulation in quadratic time

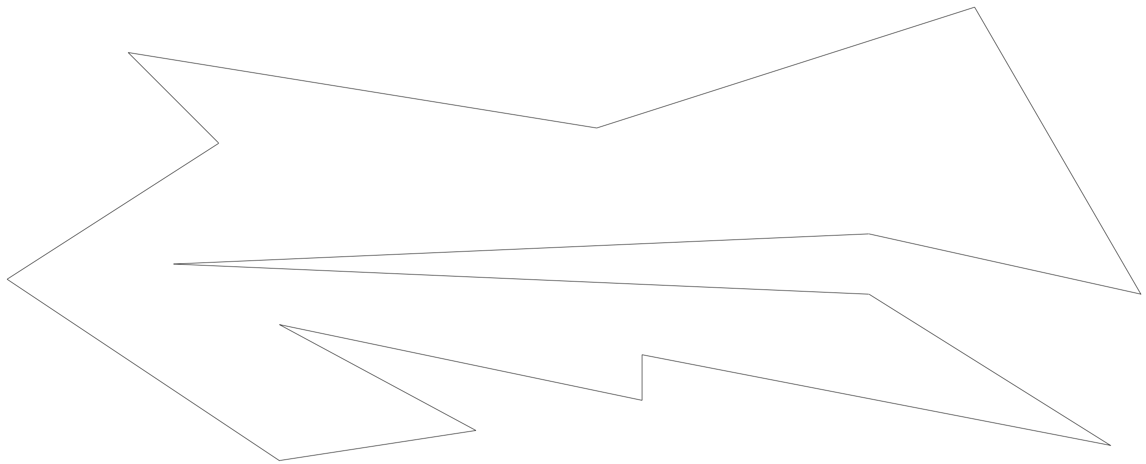
P → The Geometric Objects is a simple Polygon with n edges



Triangulation in quadratic time

P → The Geometric Objects is a simple Polygon with n edges

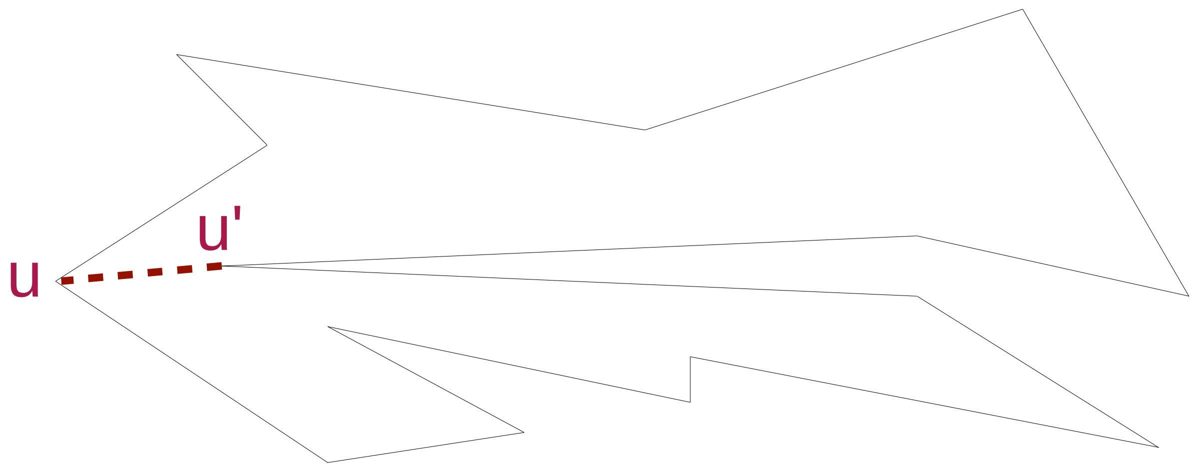
Find a diagonal uu' in linear time



Triangulation in quadratic time

P → The Geometric Objects is a simple Polygon with n edges

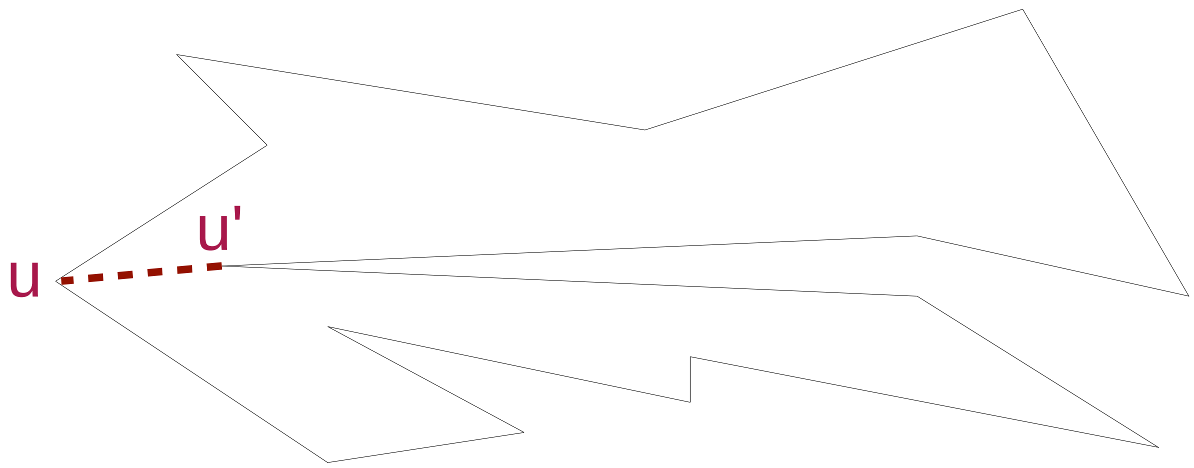
Find a diagonal uu' in linear time



Triangulation in quadratic time

P → The Geometric Objects is a simple Polygon with n edges

Find a diagonal uu' in linear time

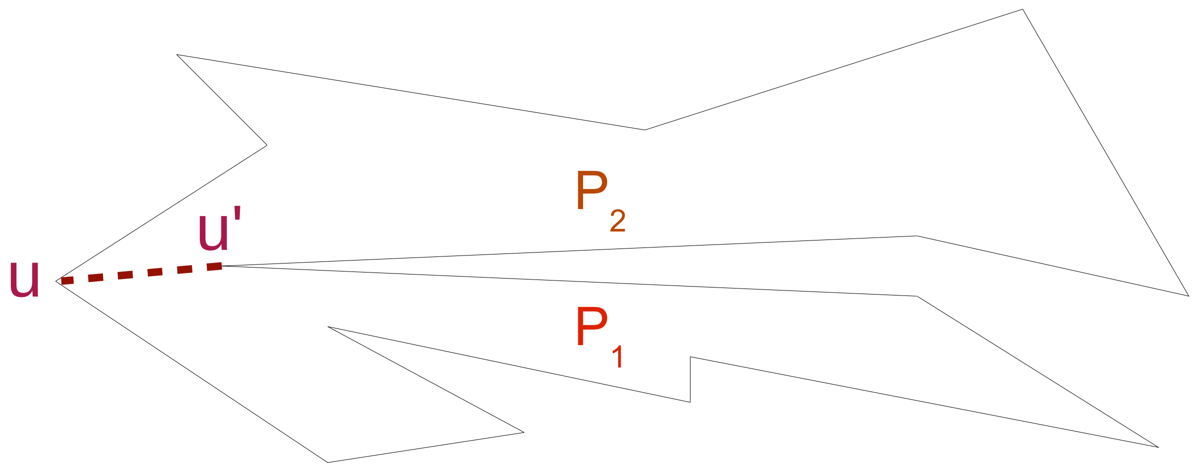


Diagonal uu' partitions P in two smaller size polygon, say P_1 and P_2

Triangulation in quadratic time

P → The Geometric Objects is a simple Polygon with n edges

Find a diagonal uu' in linear time

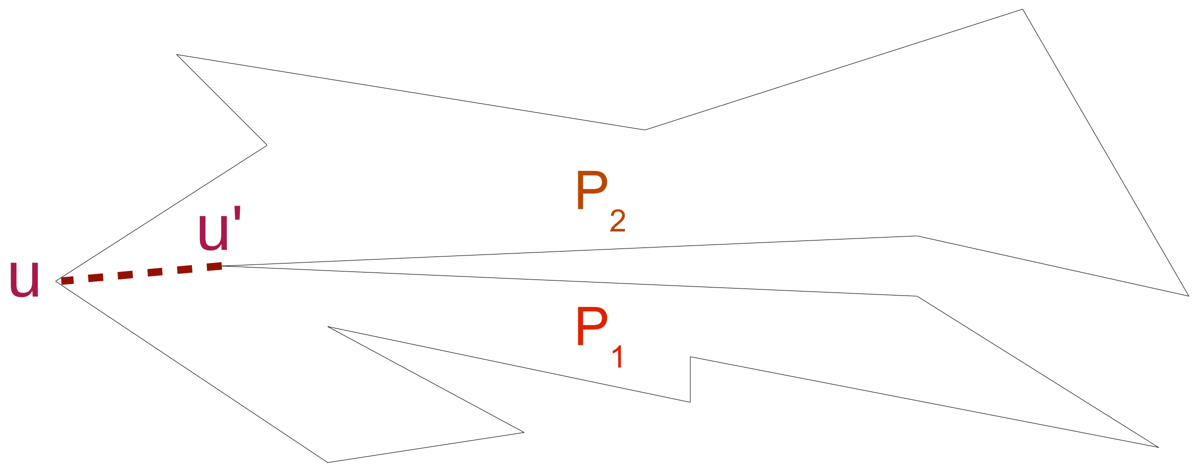


Diagonal uu' partitions P in two smaller size polygon, say P_1 and P_2

Triangulation in quadratic time

P → The Geometric Objects is a simple Polygon with n edges

Find a diagonal uu' in linear time



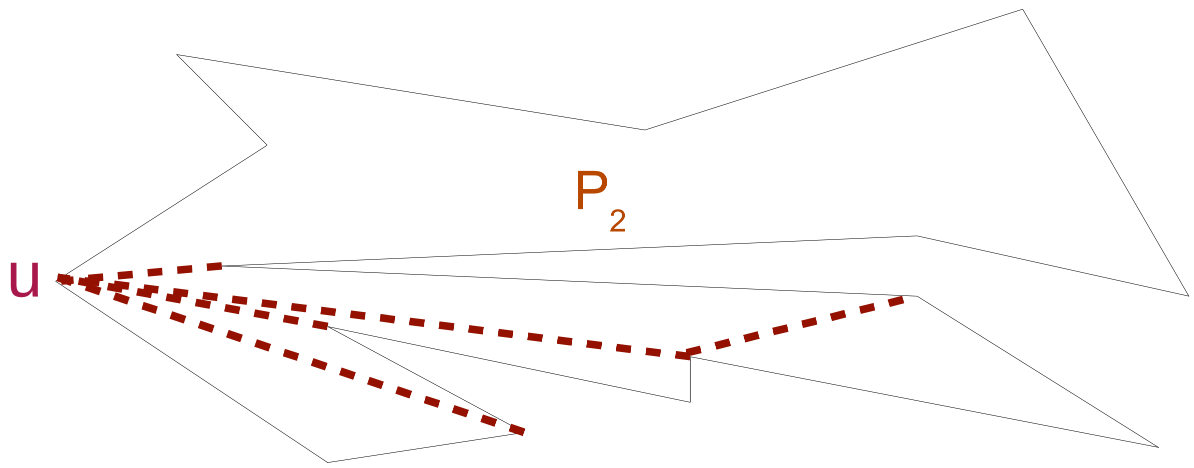
Diagonal uu' partitions P in two smaller size polygon, say P_1 and P_2

Recursively find diagonals of P_1 and P_2 until each smaller partition is a triangle

Triangulation in quadratic time

P $\rightarrow$ The Geometric Objects is a simple Polygon with n edges

Find a diagonal uu' in linear time



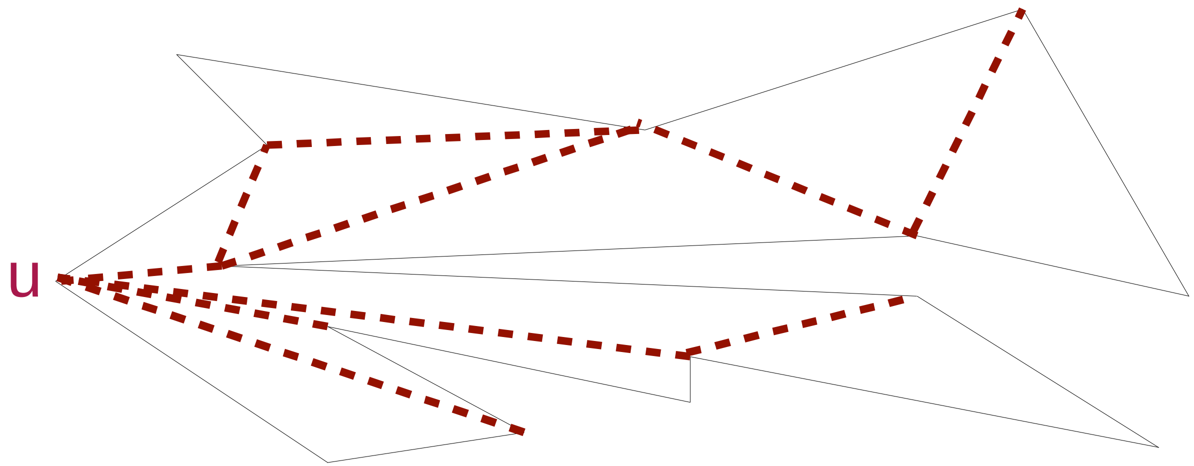
Diagonal uu' partitions P in two smaller size polygon, say P_1 and P_2

Recursively find diagonals of P_1 and P_2 until each smaller partition is a triangle

Triangulation in quadratic time

P → The Geometric Objects is a simple Polygon with n edges

Find a diagonal uu' in linear time



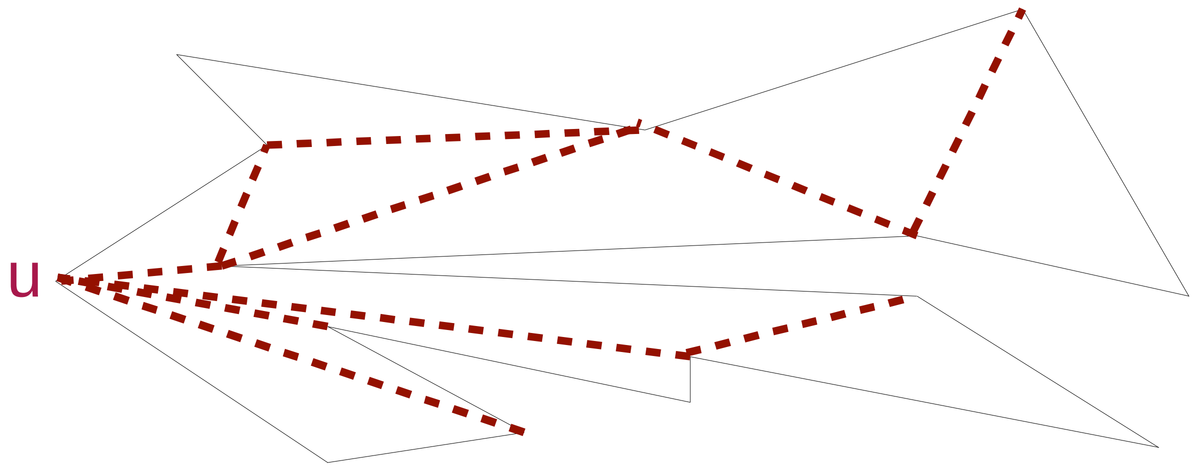
Diagonal uu' partitions P in two smaller size polygon, say P_1 and P_2

Recursively find diagonals of P_1 and P_2 until each smaller partition is a triangle

Triangulation in quadratic time

P → The Geometric Objects is a simple Polygon with n edges

Find a diagonal uu' in linear time



Diagonal uu' partitions P in two smaller size polygon, say P_1 and P_2

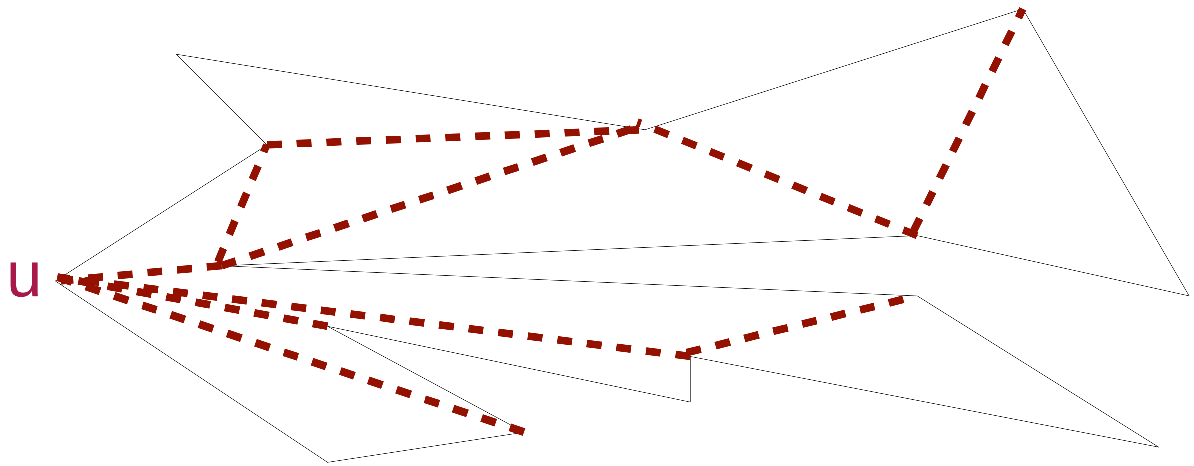
Recursively find diagonals of P_1 and P_2 until each smaller partition is a triangle

Complexity:

Triangulation in quadratic time

P → The Geometric Objects is a simple Polygon with n edges

Find a diagonal uu' in linear time



Diagonal uu' partitions P in two smaller size polygon, say P_1 and P_2

Recursively find diagonals of P_1 and P_2 until each smaller partition is a triangle

Complexity: $O(n^2)$

Some results on triangulation

Some results on triangulation

Chazelle proposed a linear time algorithm for triangulation

Bernard Chazelle: Triangulating a Simple Polygon in Linear Time.
Discrete & Computational Geometry 6: 485-524 (1991)

Some results on triangulation

Chazelle proposed a linear time algorithm for triangulation

Bernard Chazelle: Triangulating a Simple Polygon in Linear Time.
Discrete & Computational Geometry 6: 485-524 (1991)

This is possible the hardest algorithm to implement.
No one till date possible dare to implement it !!! :(

Some results on triangulation

Chazelle proposed a linear time algorithm for triangulation

Bernard Chazelle: Triangulating a Simple Polygon in Linear Time.
Discrete & Computational Geometry 6: 485-524 (1991)

This is possible the hardest algorithm to implement.
No one till date possible dare to implement it !!! :(

There exists simple O(n log n) time implementable algorithms :)

Some results on triangulation

Chazelle proposed a linear time algorithm for triangulation

Bernard Chazelle: Triangulating a Simple Polygon in Linear Time.
Discrete & Computational Geometry 6: 485-524 (1991)

This is possible the hardest algorithm to implement.
No one till date possible dare to implement it !!! :(

There exists simple O(n log n) time implementable algorithms :)

Preparata, and Shamos: Computational Geometry - An Introduction (Book)

Some results on triangulation

Chazelle proposed a linear time algorithm for triangulation

Bernard Chazelle: Triangulating a Simple Polygon in Linear Time.
Discrete & Computational Geometry 6: 485-524 (1991)

This is possible the hardest algorithm to implement.
No one till date possible dare to implement it !!! :(

There exists simple O(n log n) time implementable algorithms :)

Preparata, and Shamos: Computational Geometry - An Introduction (Book)

People are still looking for implementable linear time algorithms

Some results on triangulation

Chazelle proposed a linear time algorithm for triangulation

Bernard Chazelle: Triangulating a Simple Polygon in Linear Time.
Discrete & Computational Geometry 6: 485-524 (1991)

This is possible the hardest algorithm to implement.
No one till date possible dare to implement it !!! :(

There exists simple O(n log n) time implementable algorithms :)

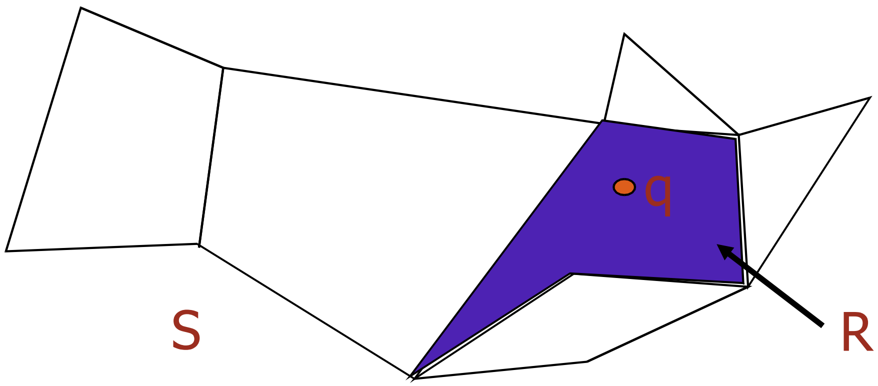
Preparata, and Shamos: Computational Geometry - An Introduction (Book)

People are still looking for implementable linear time algorithms

But we want planar point location not triangulation !!

Planar Point Location

Given a planar subdivision S



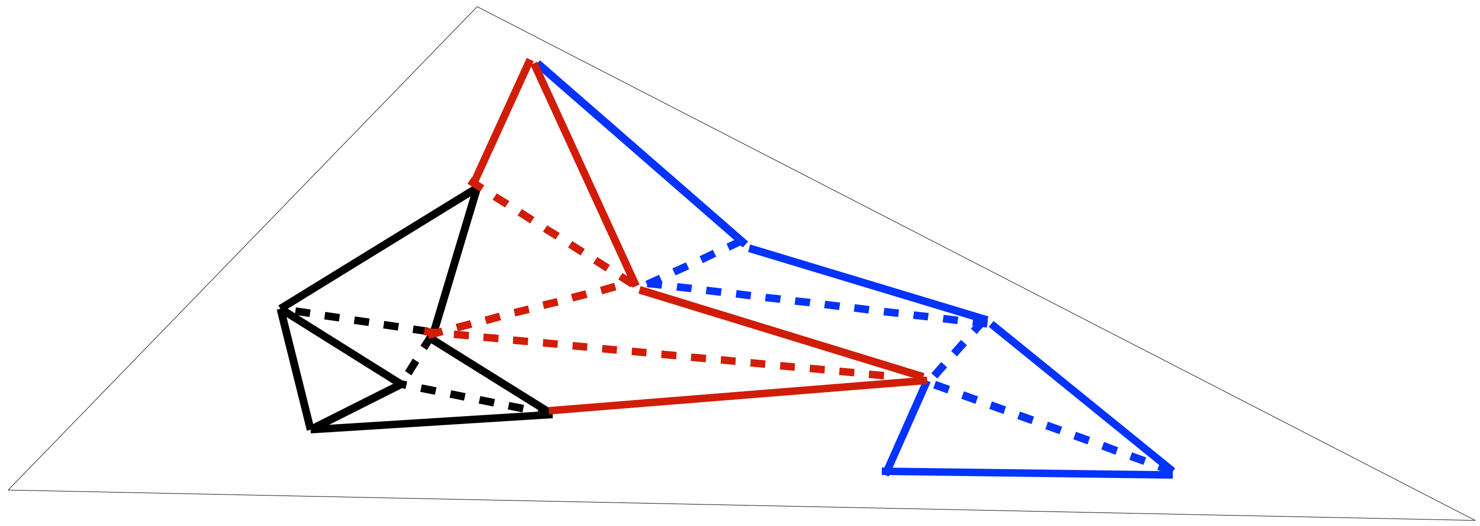
Preprocess S such that:

For any query point q,

The region/face R containing q can be reported efficiently.

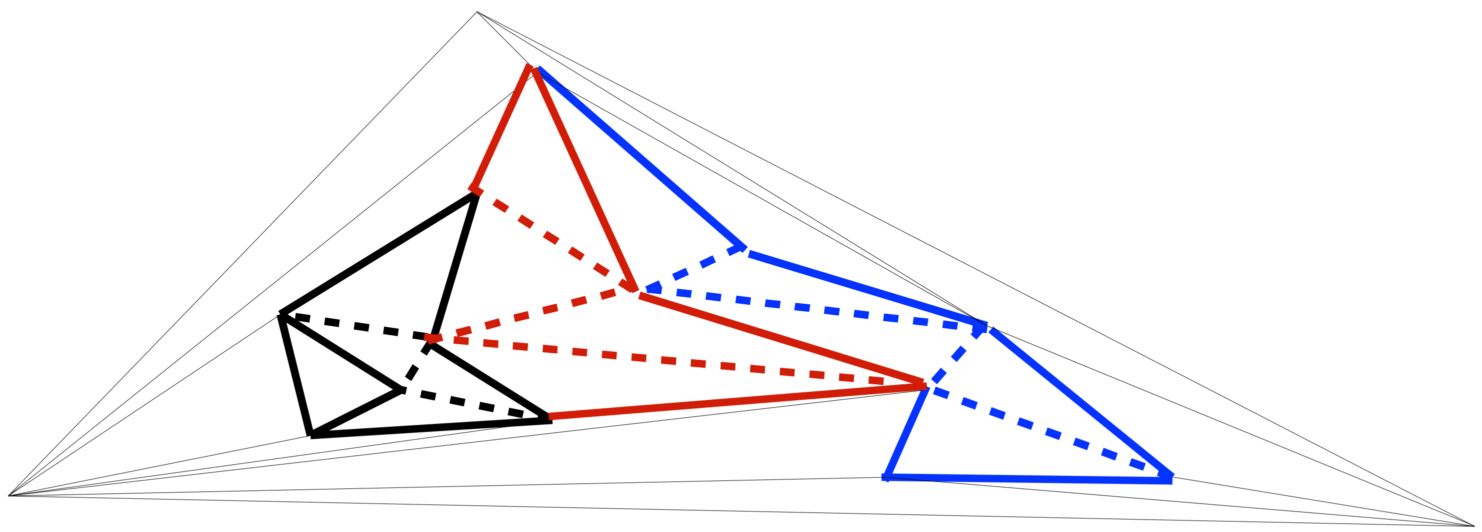
Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle



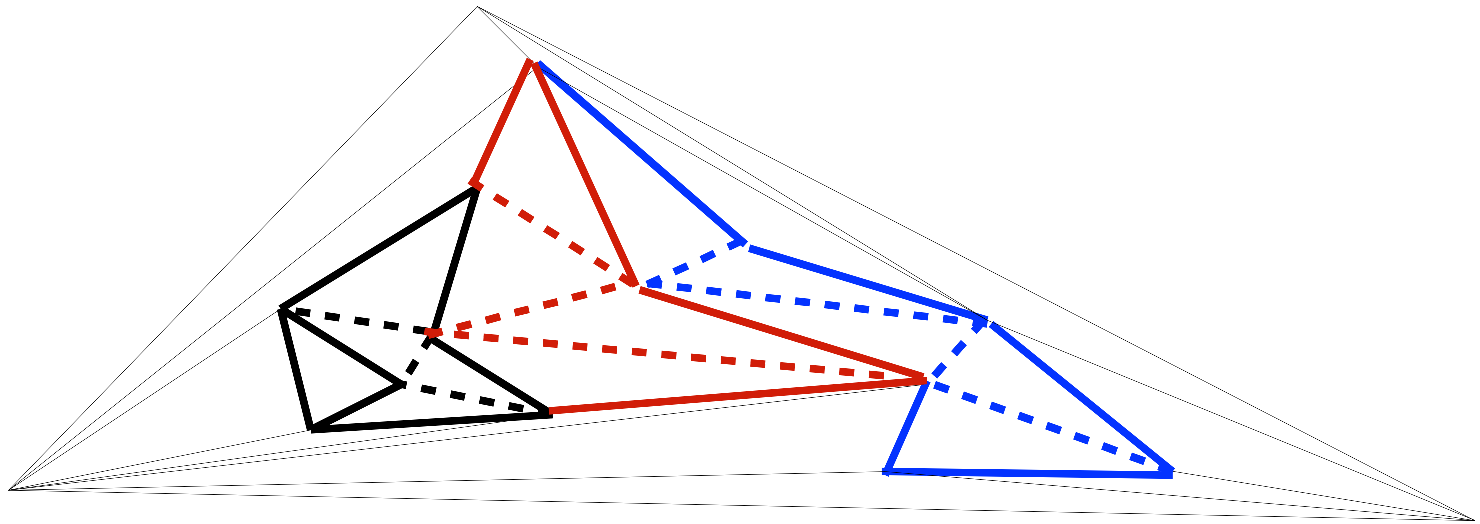
Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle



Outline of Kirkpatrick Planar Point Location

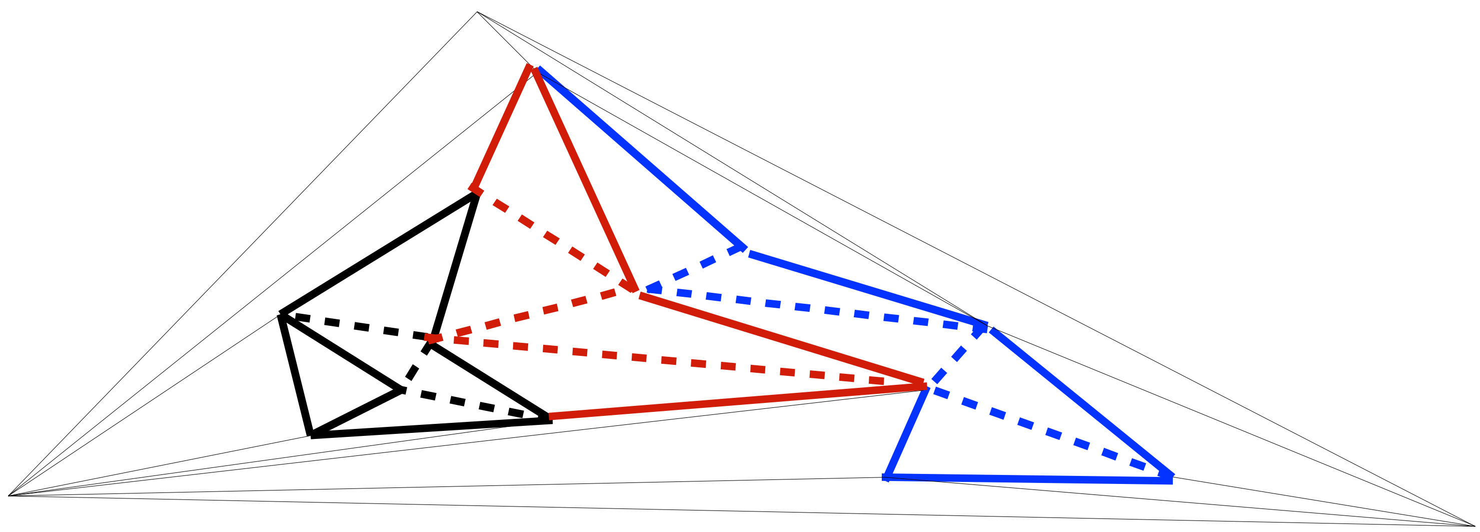
Given a triangulated planar subdivision S inside a bounded triangle



Fact : S has n/2 vertices with degree at most 9 [because S is planar]

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle



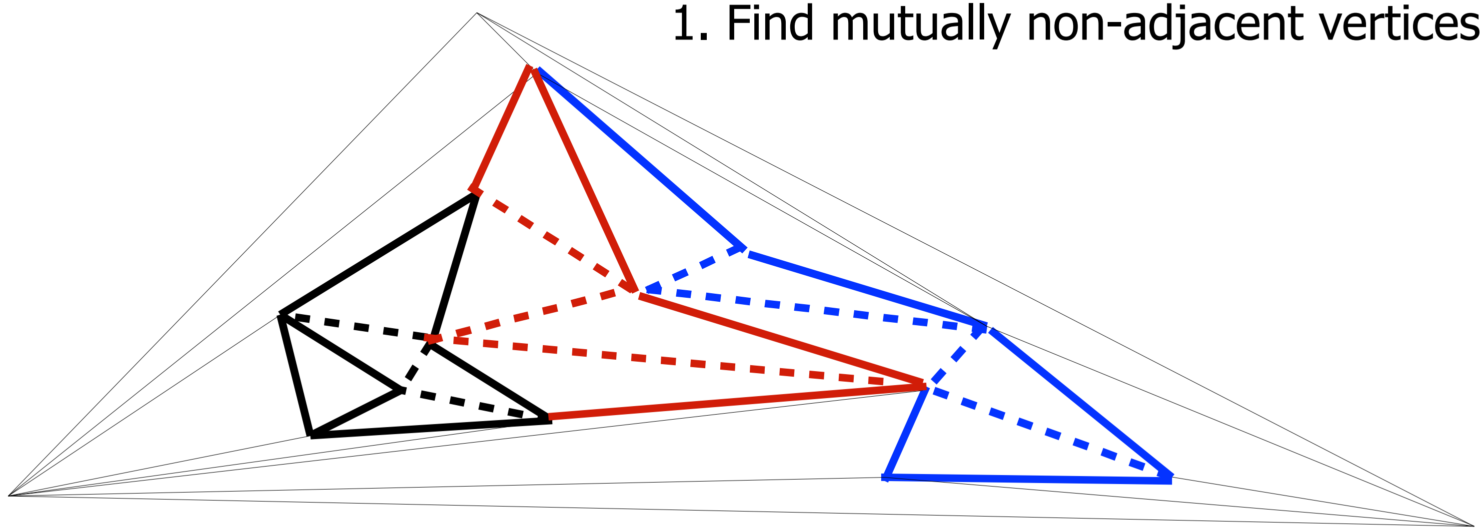
Fact : S has n/2 vertices with degree at most 9 [because S is planar]

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H

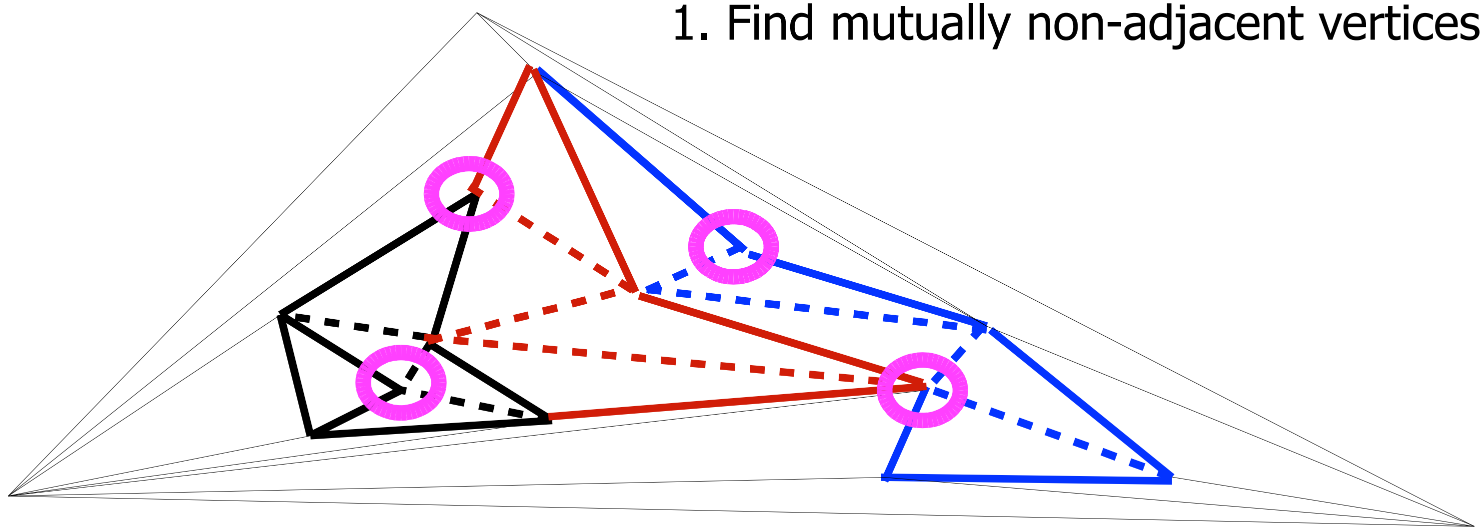


Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H

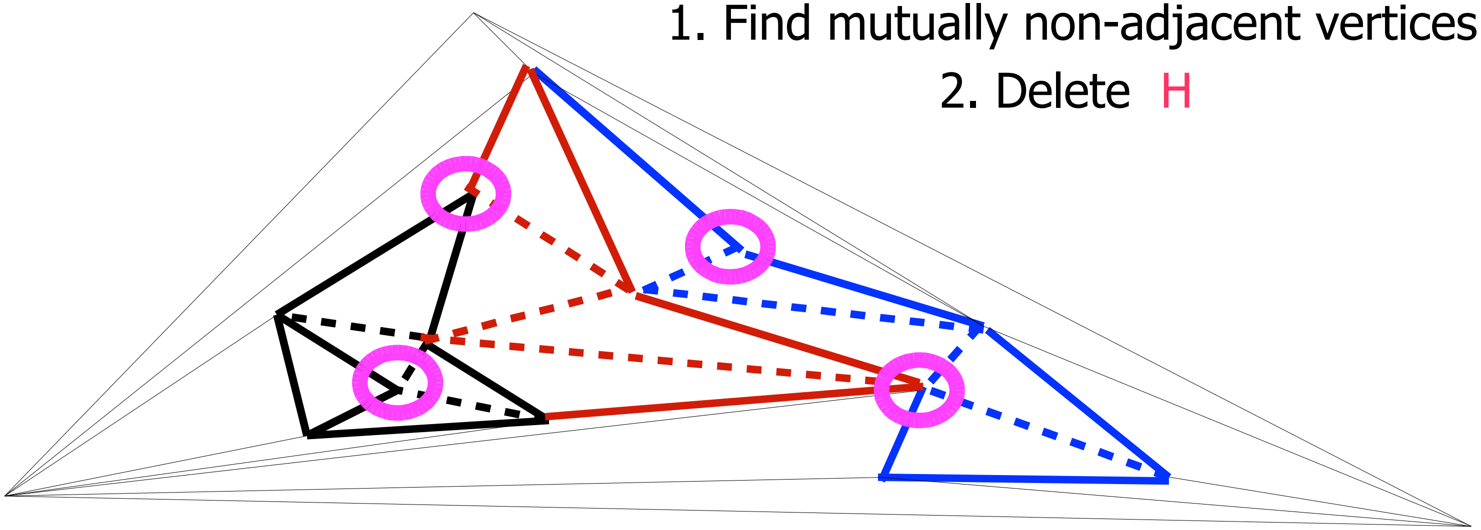


Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H
2. Delete H



Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

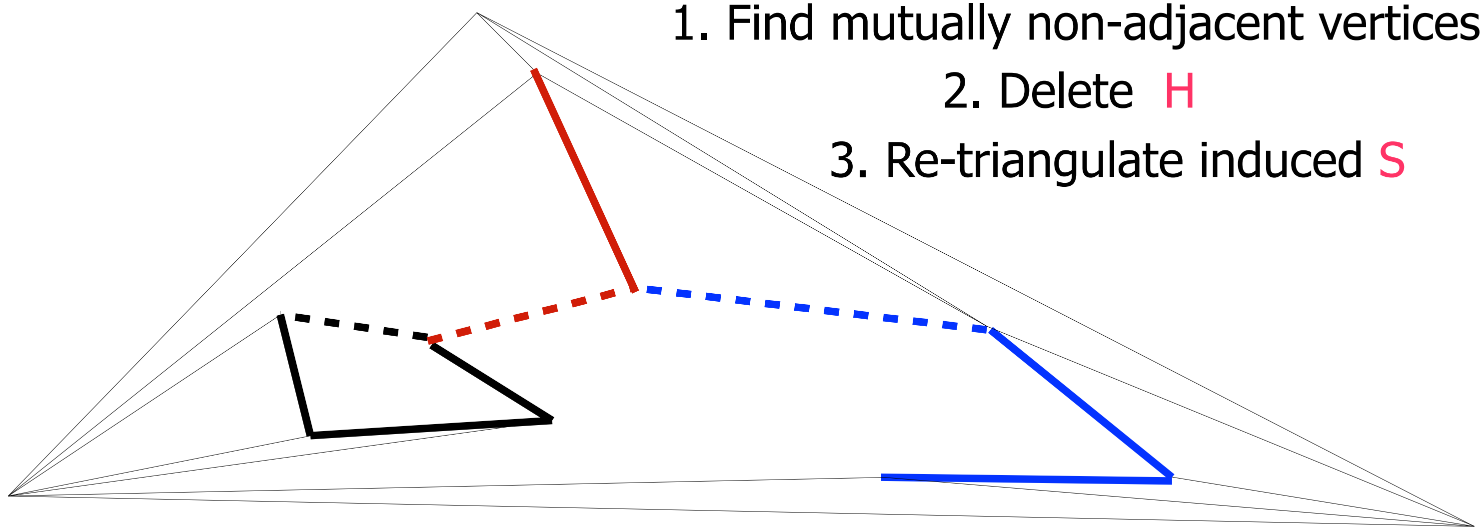
Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H

2. Delete H

3. Re-triangulate induced S



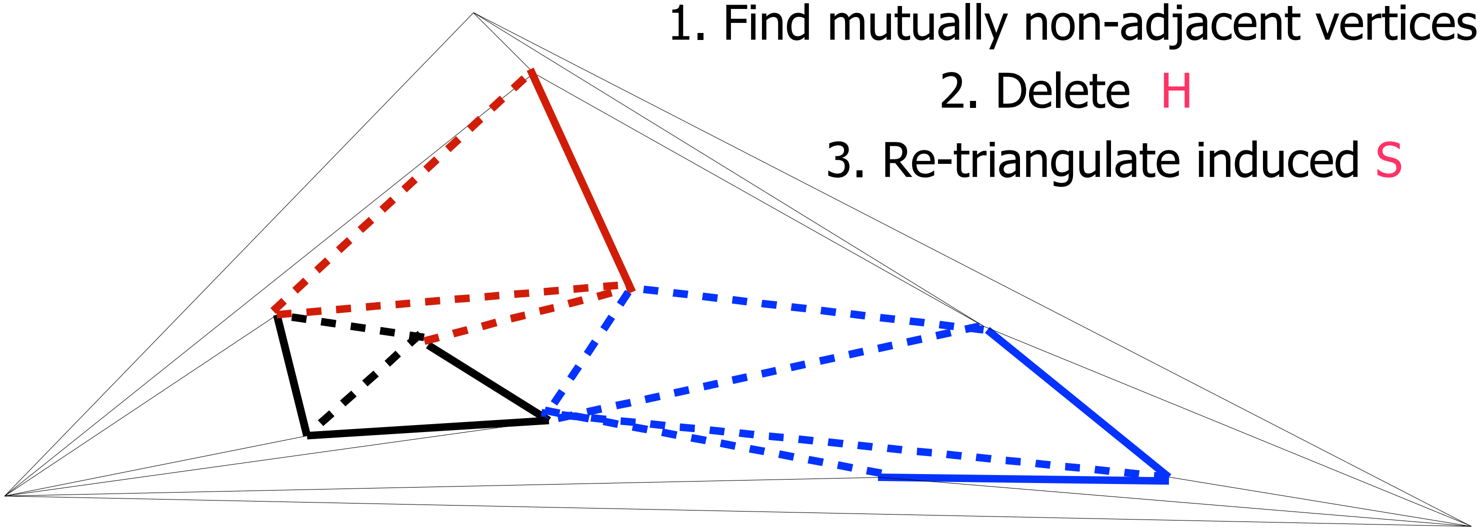
Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H

2. Delete H

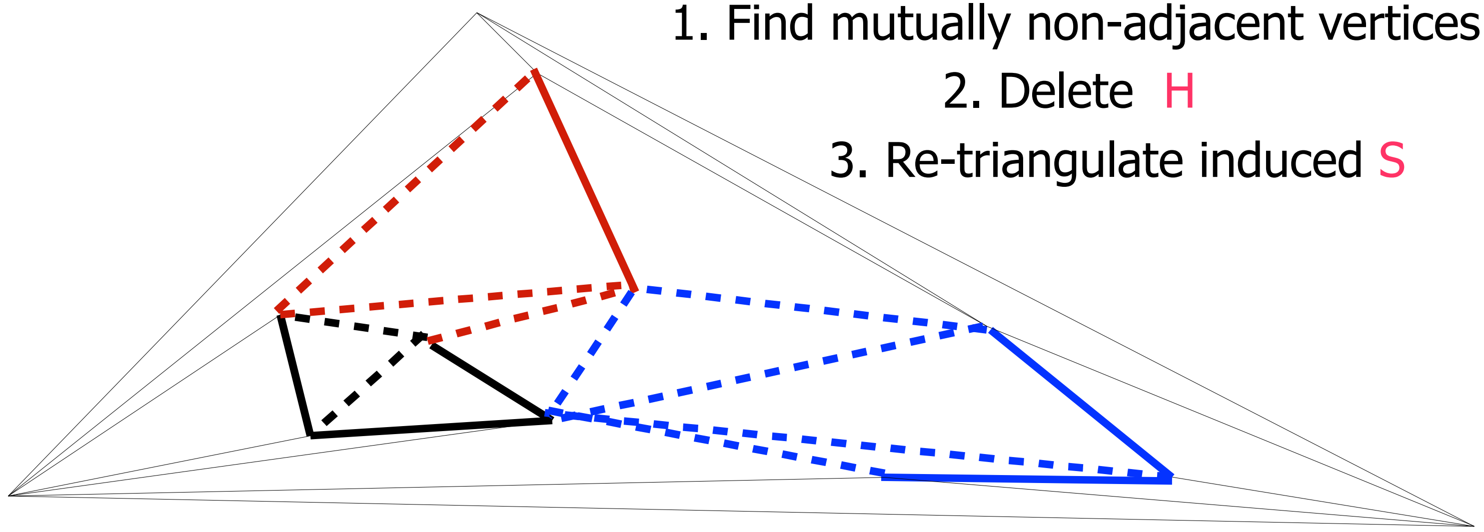
3. Re-triangulate induced S



Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S



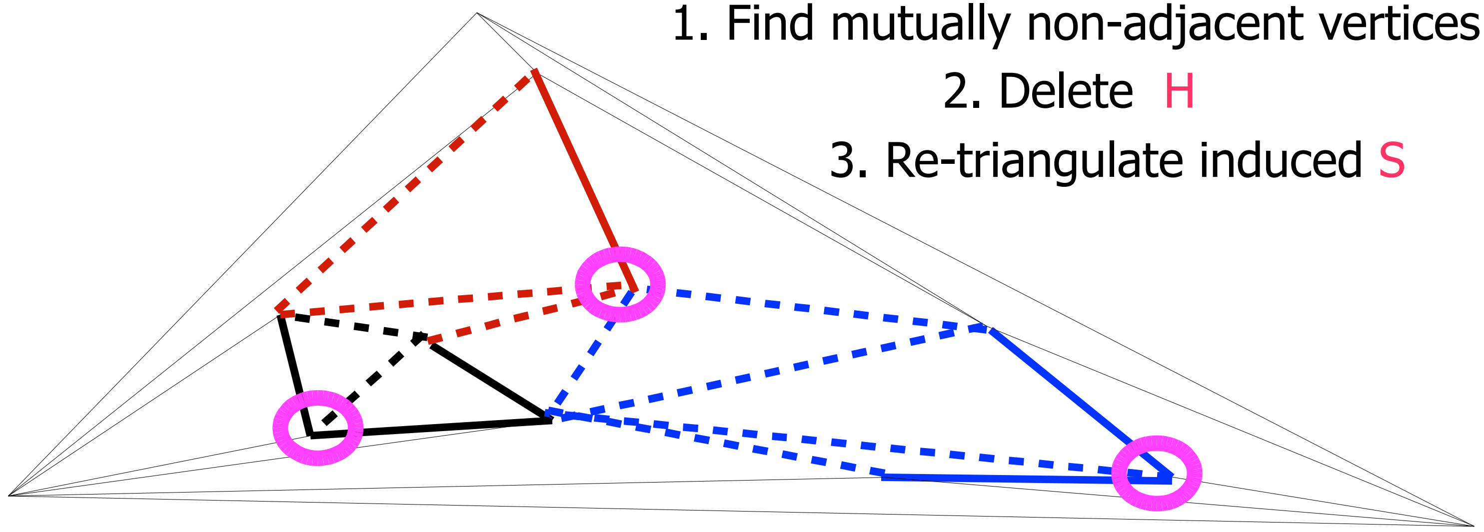
Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S



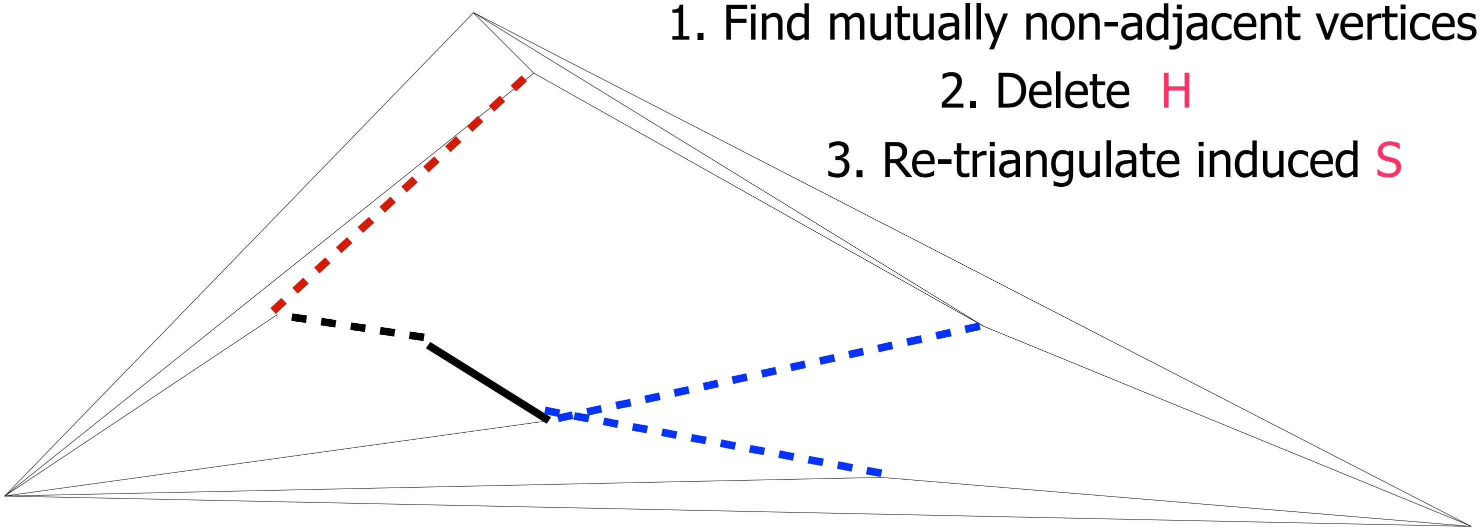
Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S



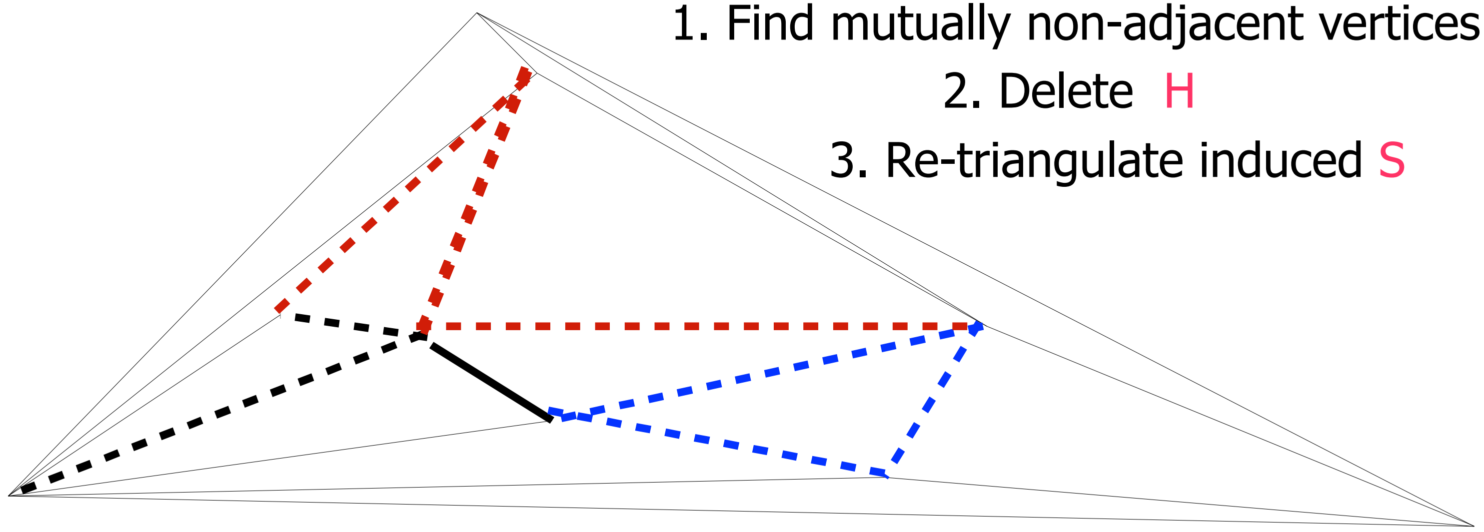
Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S



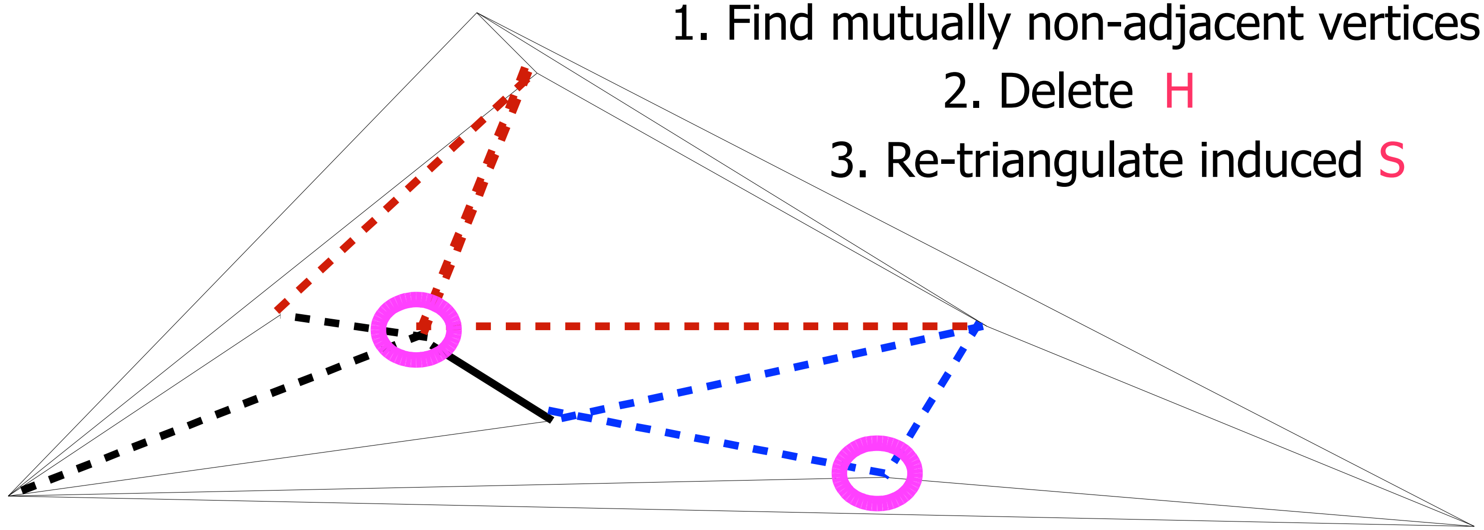
Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S

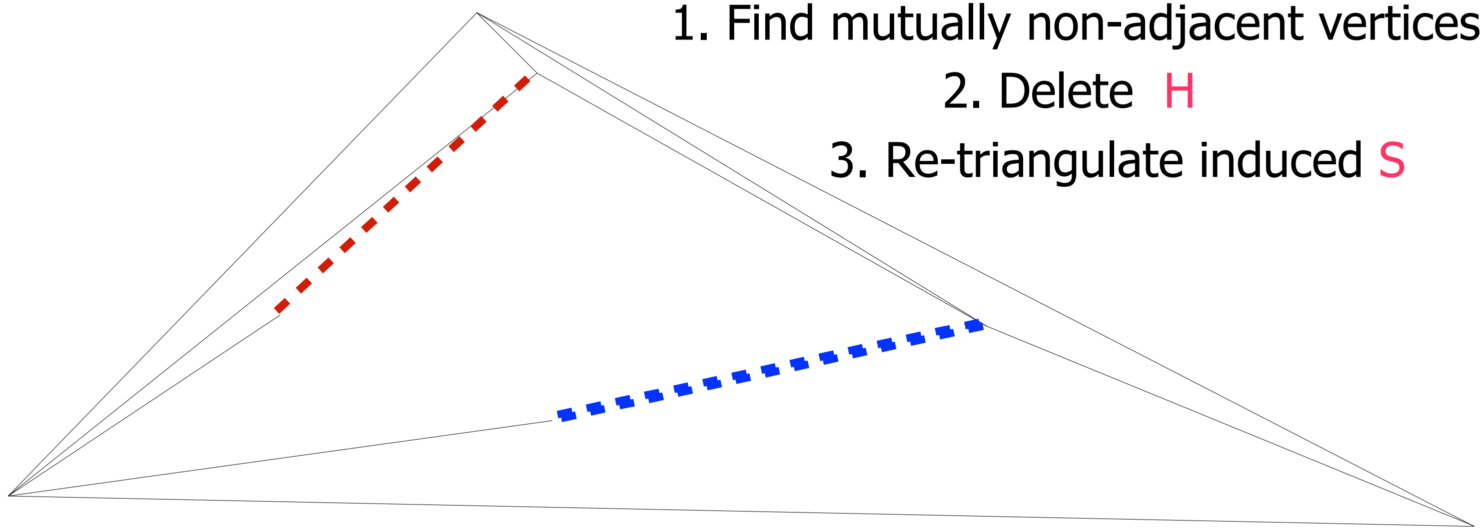


Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle



1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S

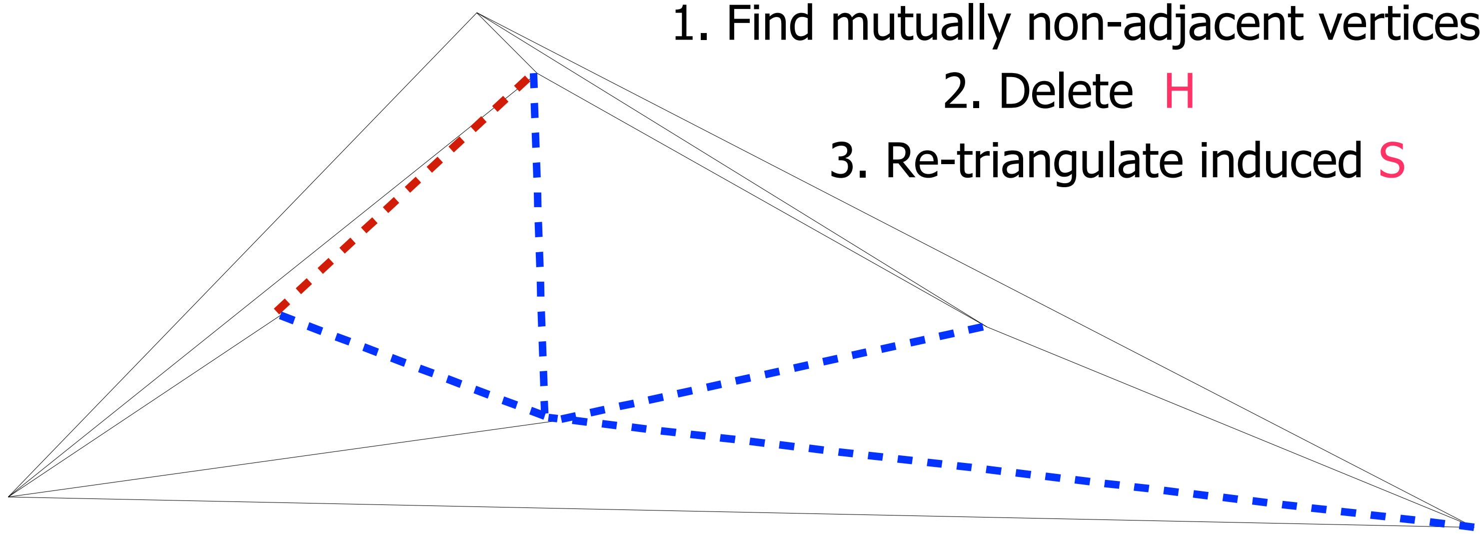
Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S



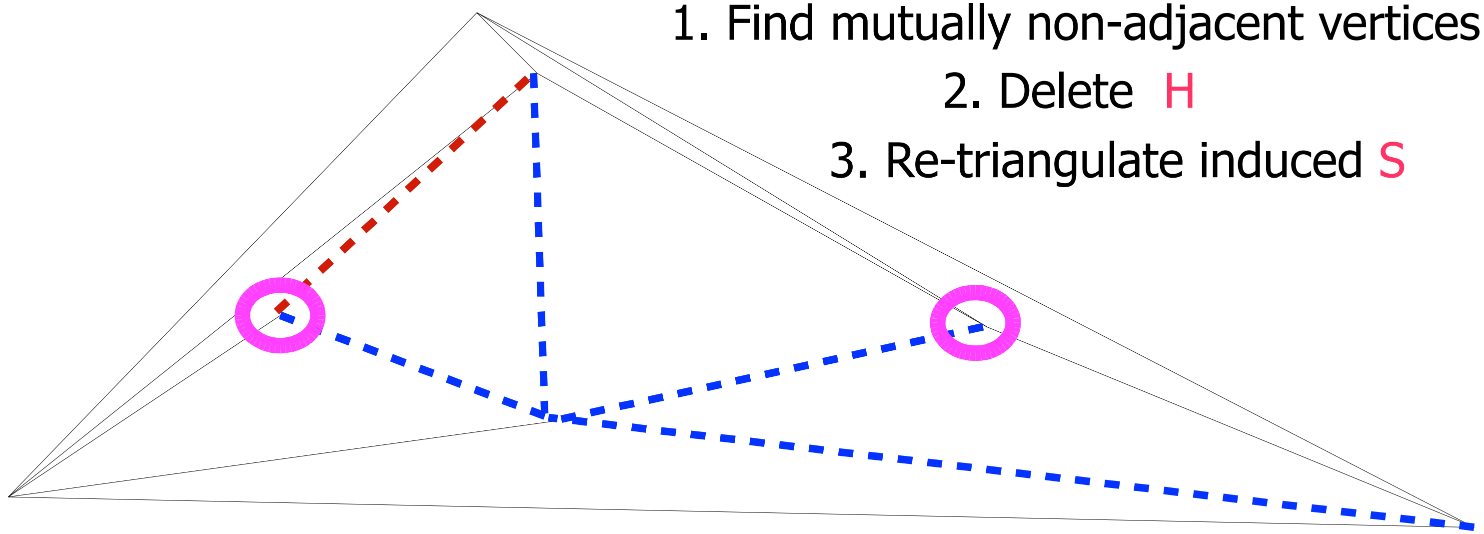
Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S

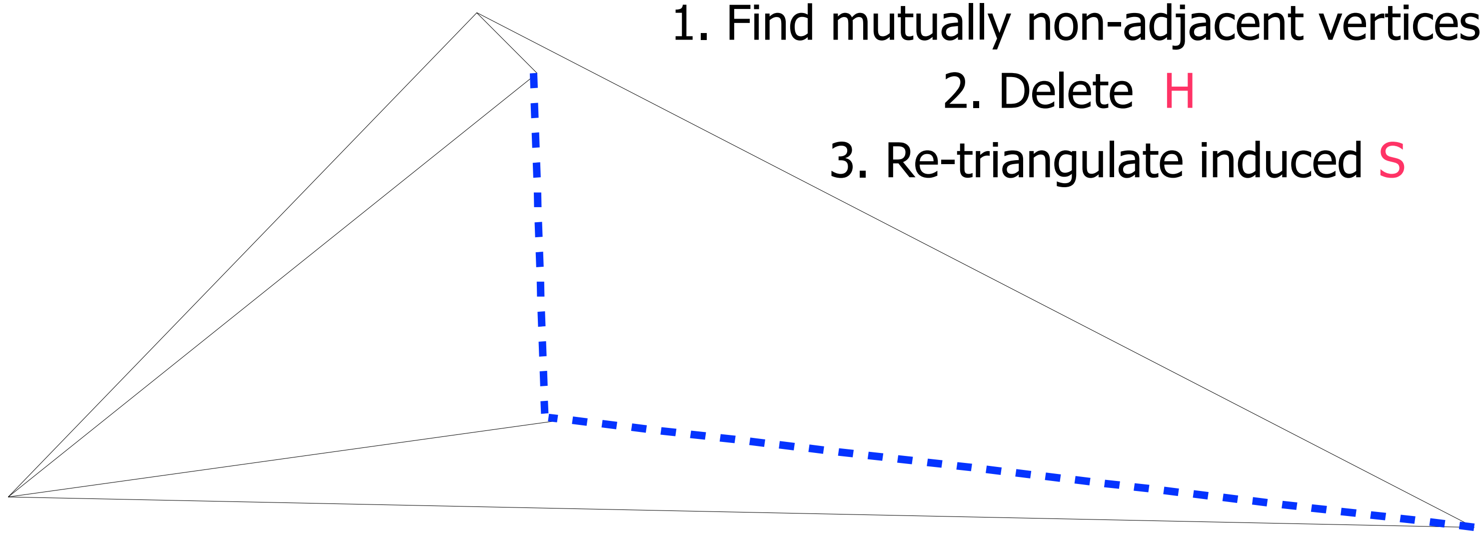


Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle



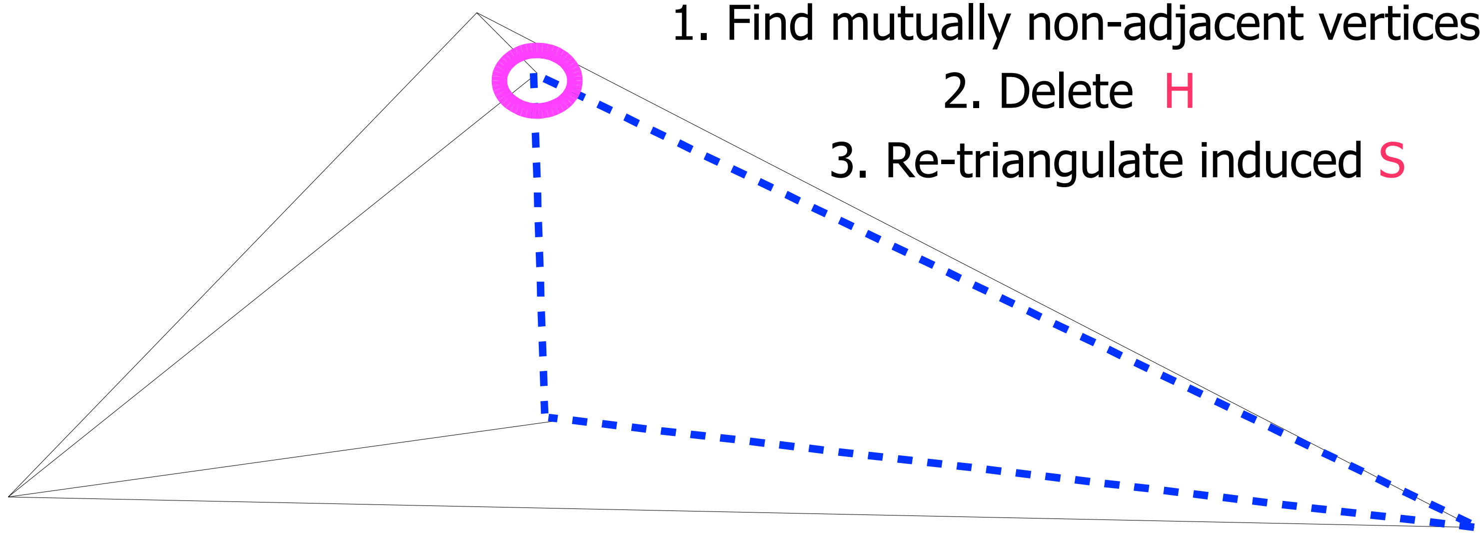
1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S

Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle



1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S

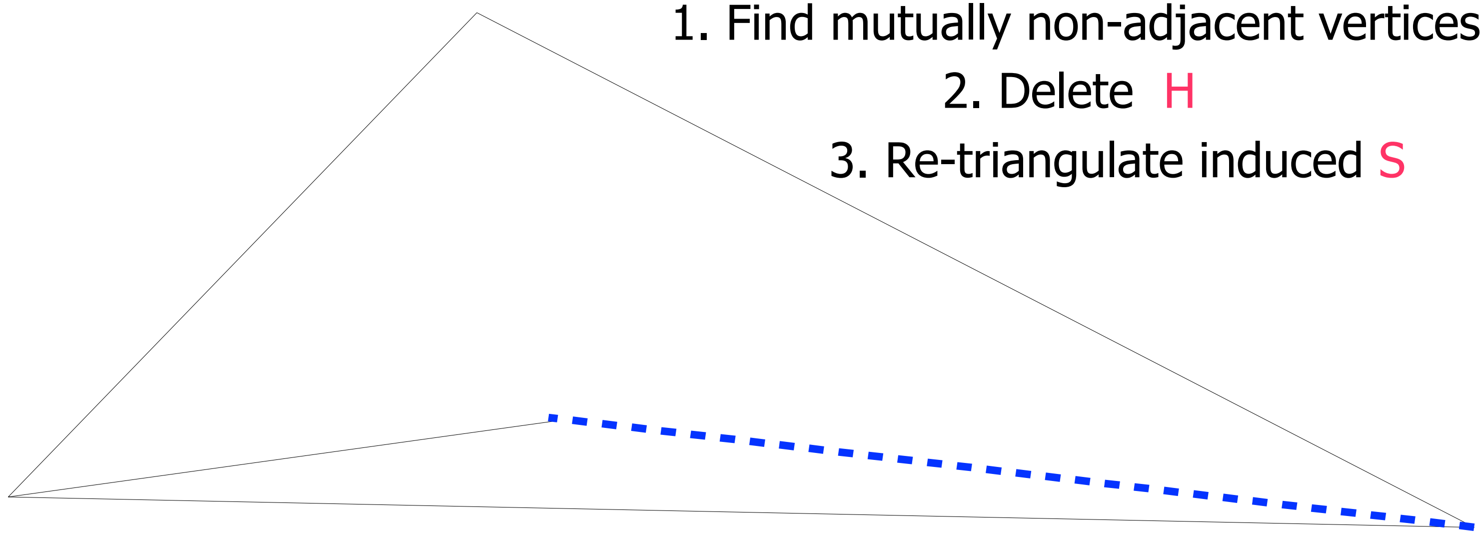
Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S



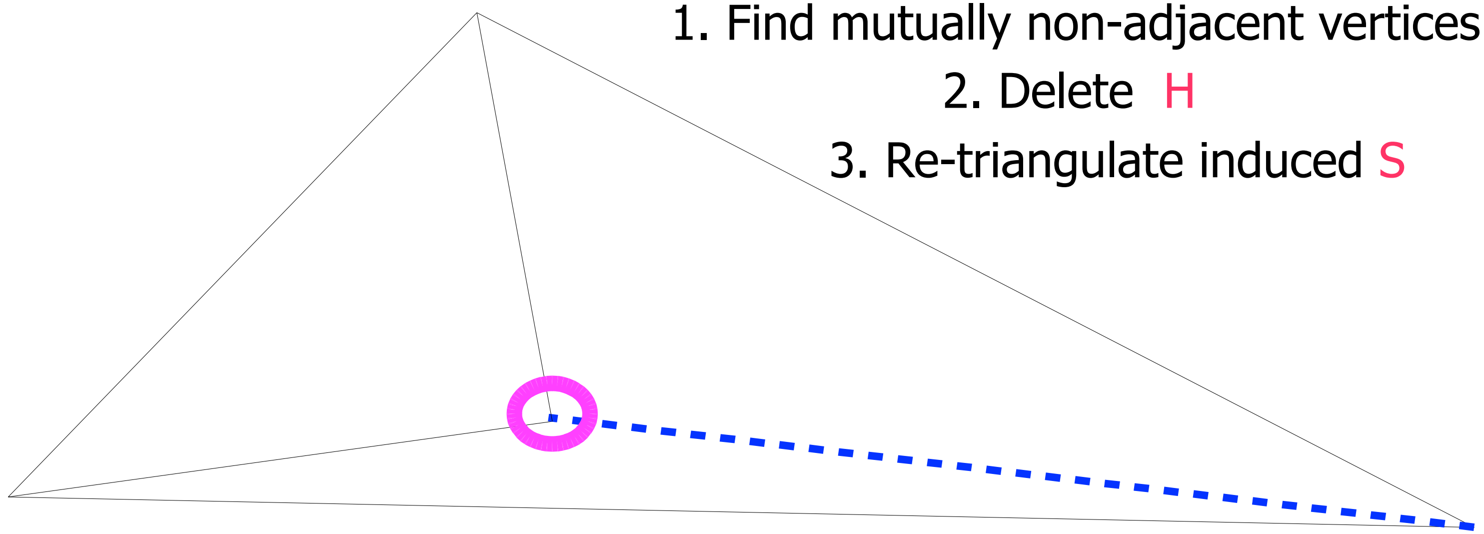
Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle

1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S

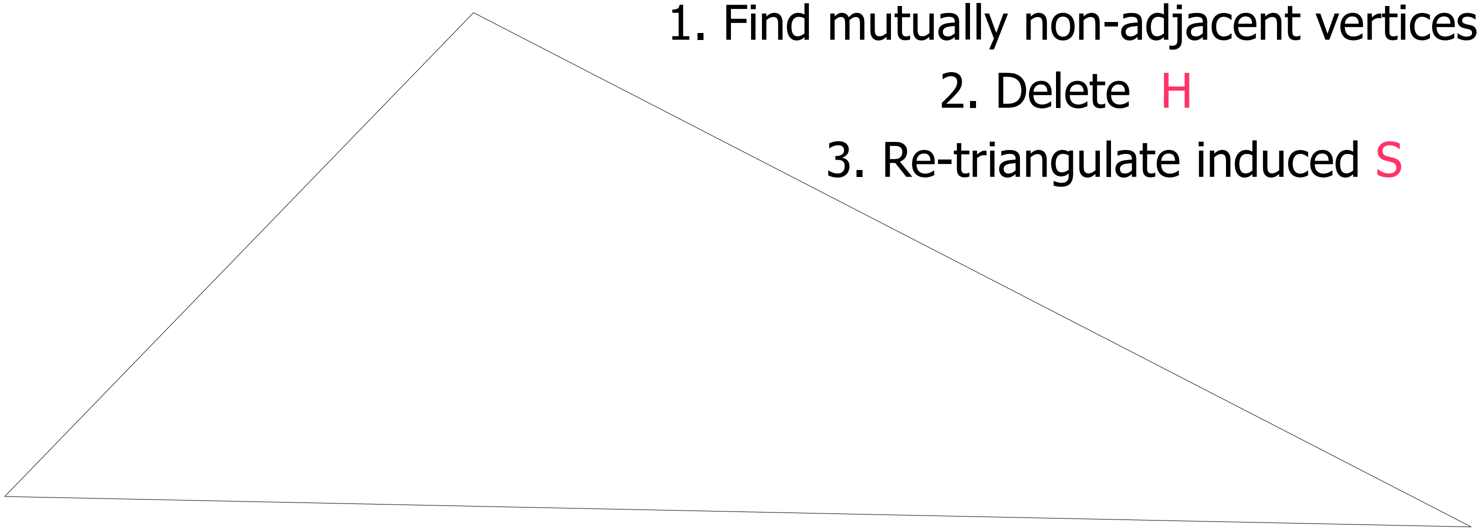


Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

Outline of Kirkpatrick Planar Point Location

Given a triangulated planar subdivision S inside a bounded triangle



1. Find mutually non-adjacent vertices H
2. Delete H
3. Re-triangulate induced S

Recursively do the same until only the three outside vertices are left

Corollary : S has n/18 mutually non-adjacent vertices H with degree at most 9

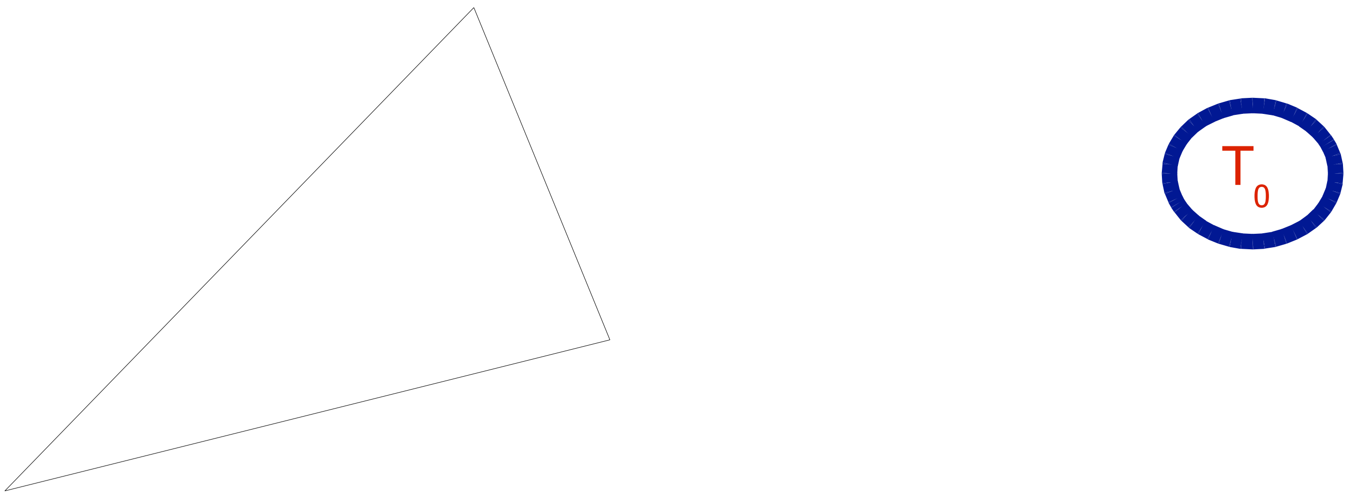
How the data structure looks like

How the data structure looks like

Finally we have the bounded triangle T_0 as the root

How the data structure looks like

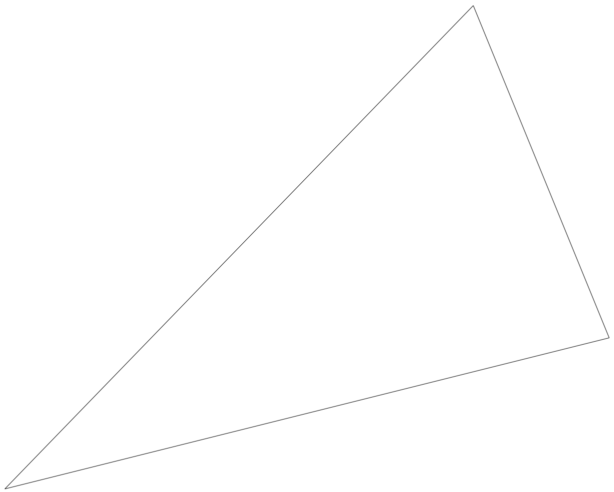
Finally we have the bounded triangle T_0 as the root

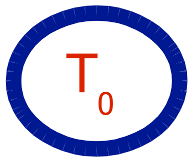


How the data structure looks like

Finally we have the bounded triangle T_0 as the root

1. Three triangles in previous step

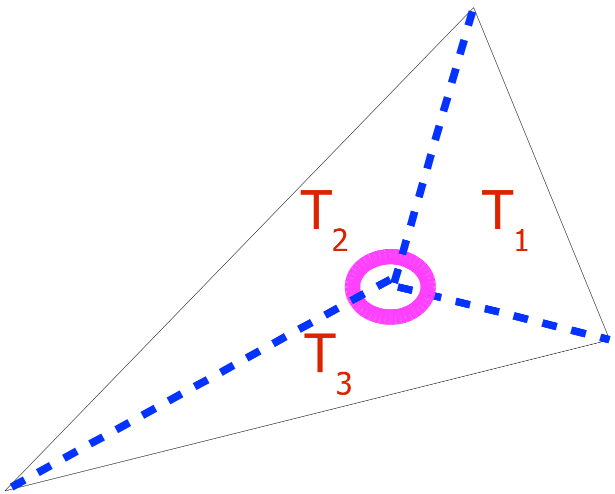


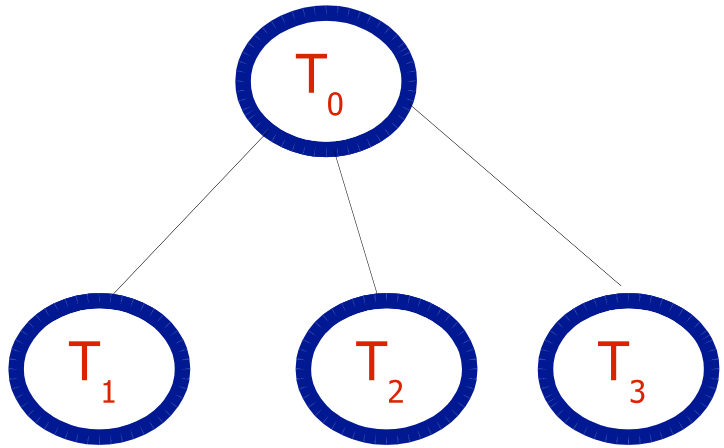


How the data structure looks like

Finally we have the bounded triangle T_0 as the root

1. Three triangles in previous step



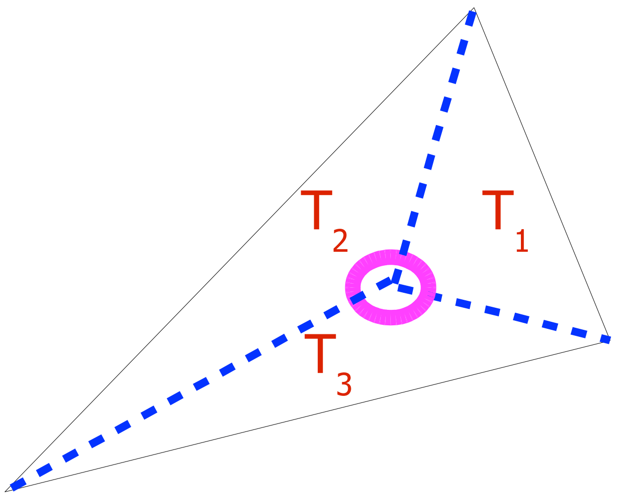


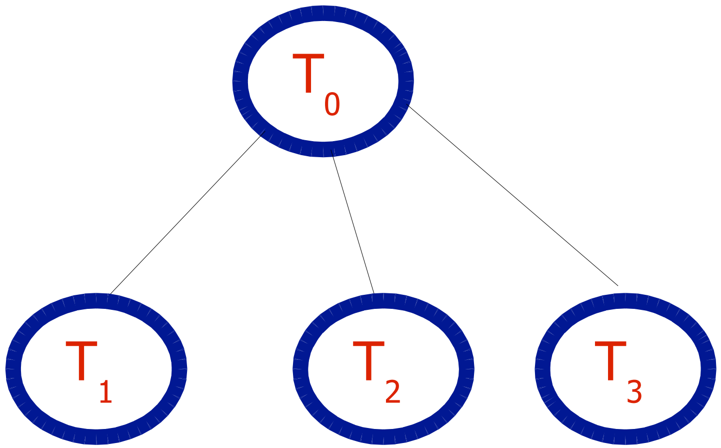
How the data structure looks like

Finally we have the bounded triangle T_0 as the root

1. Three triangles in previous step

2. Few more triangles interacting with previous triangles



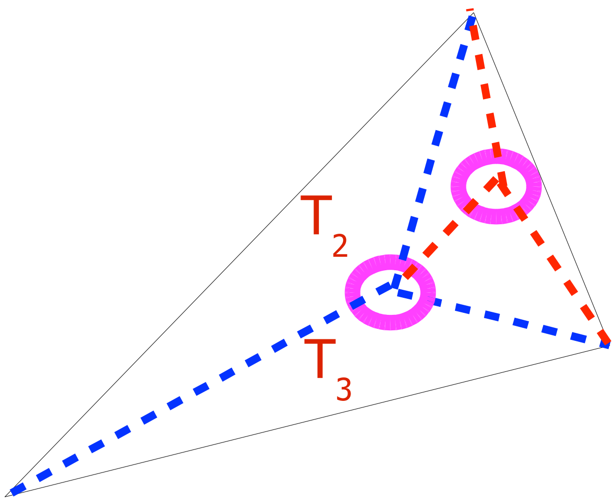


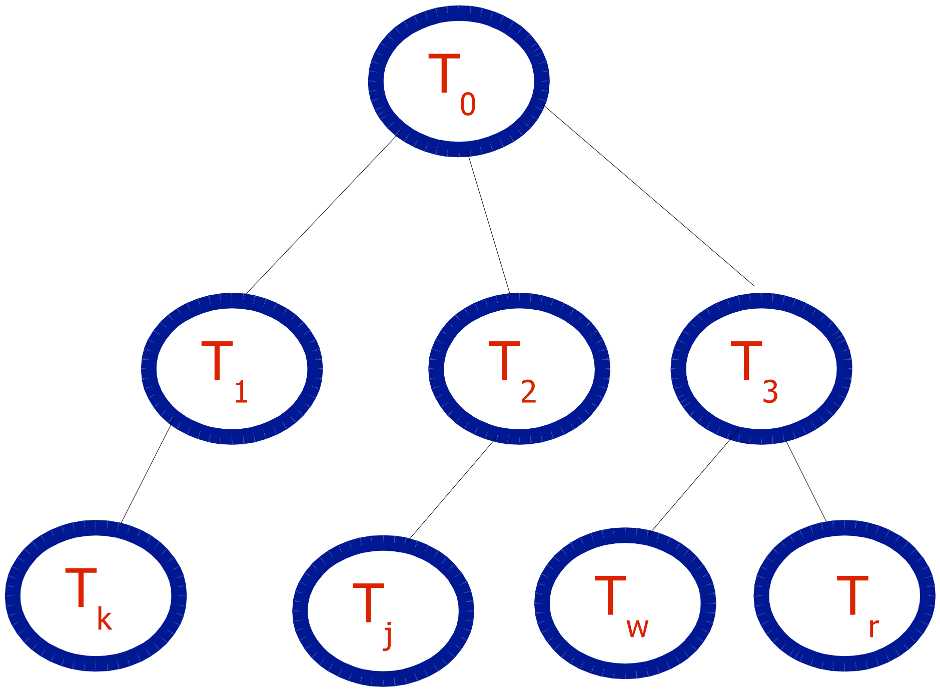
How the data structure looks like

Finally we have the bounded triangle T_0 as the root

1. Three triangles in previous step

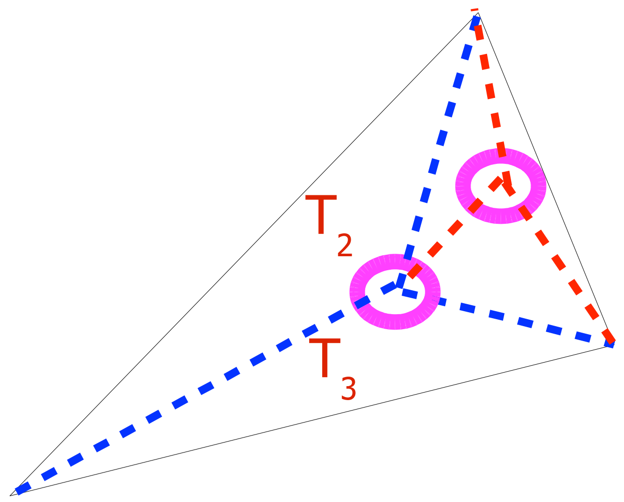
2. Few more triangles interacting with previous triangles





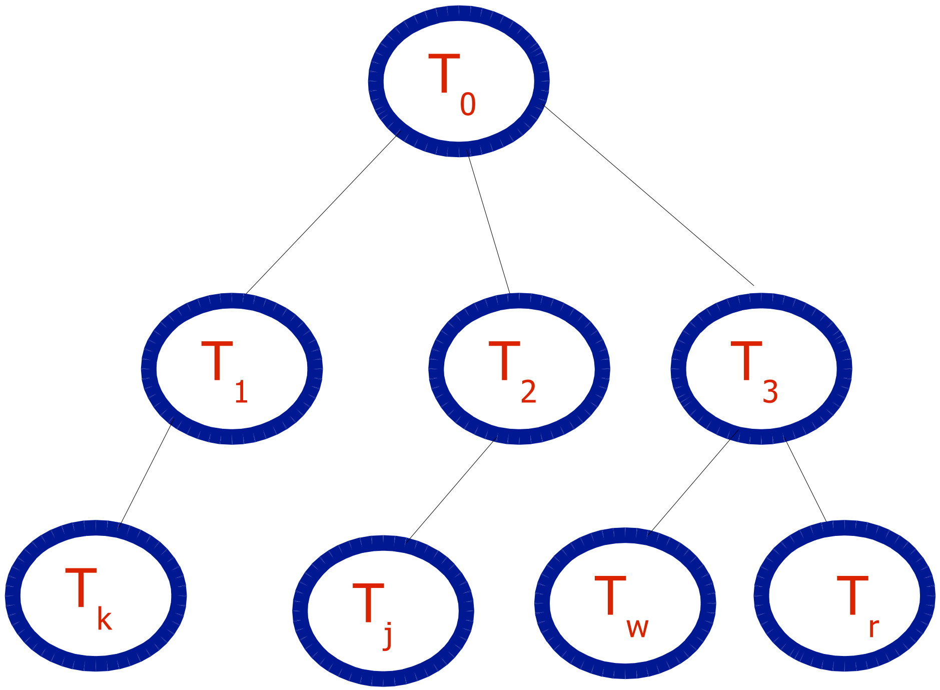
How the data structure looks like

Finally we have the bounded triangle T_0 as the root



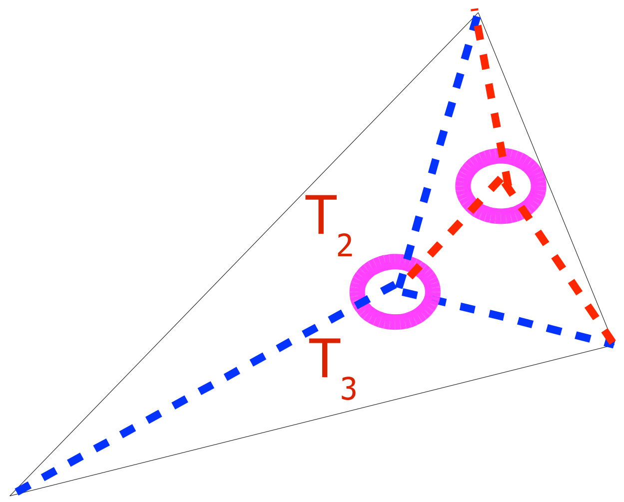
1. Three triangles in previous step
2. Few more triangles interacting with previous triangles

Looks like a:

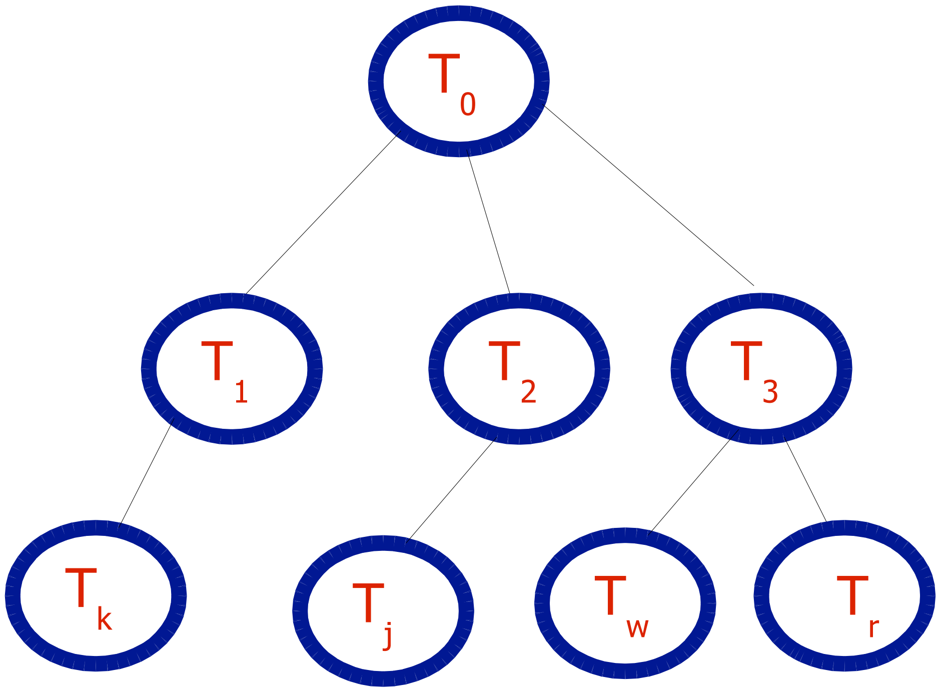


How the data structure looks like

Finally we have the bounded triangle T_0 as the root



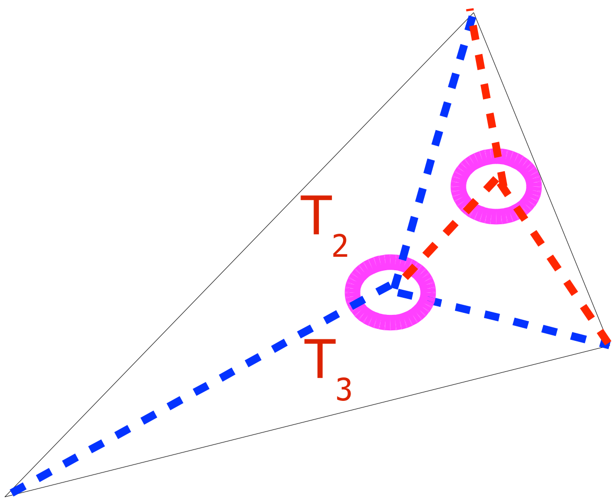
1. Three triangles in previous step
2. Few more triangles interacting with previous triangles



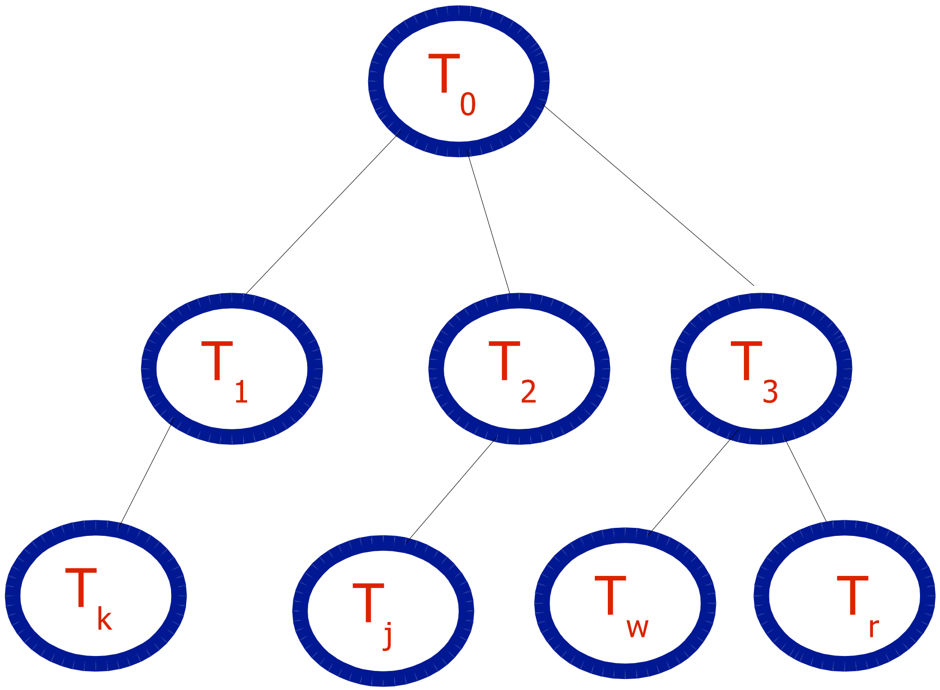
Looks like a: Tree

How the data structure looks like

Finally we have the bounded triangle T_0 as the root



1. Three triangles in previous step
2. Few more triangles interacting with previous triangles



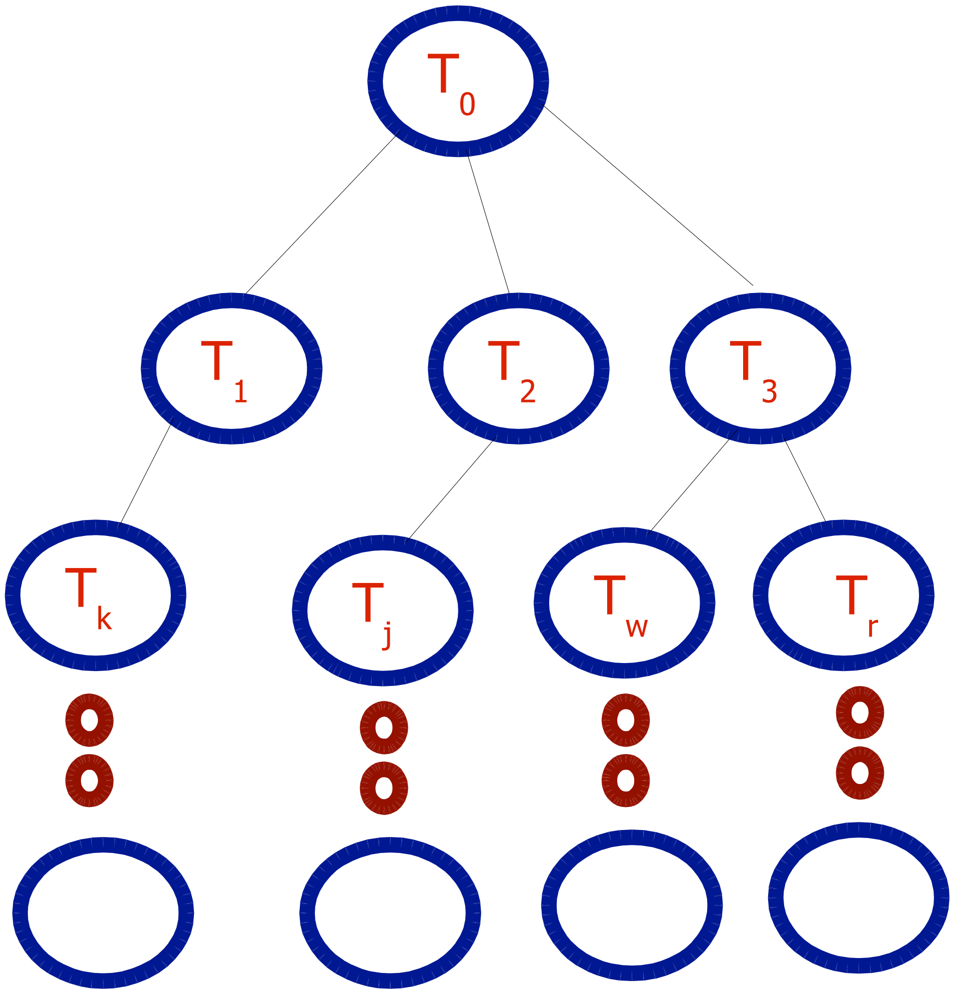
Looks like a: Tree

We name it :

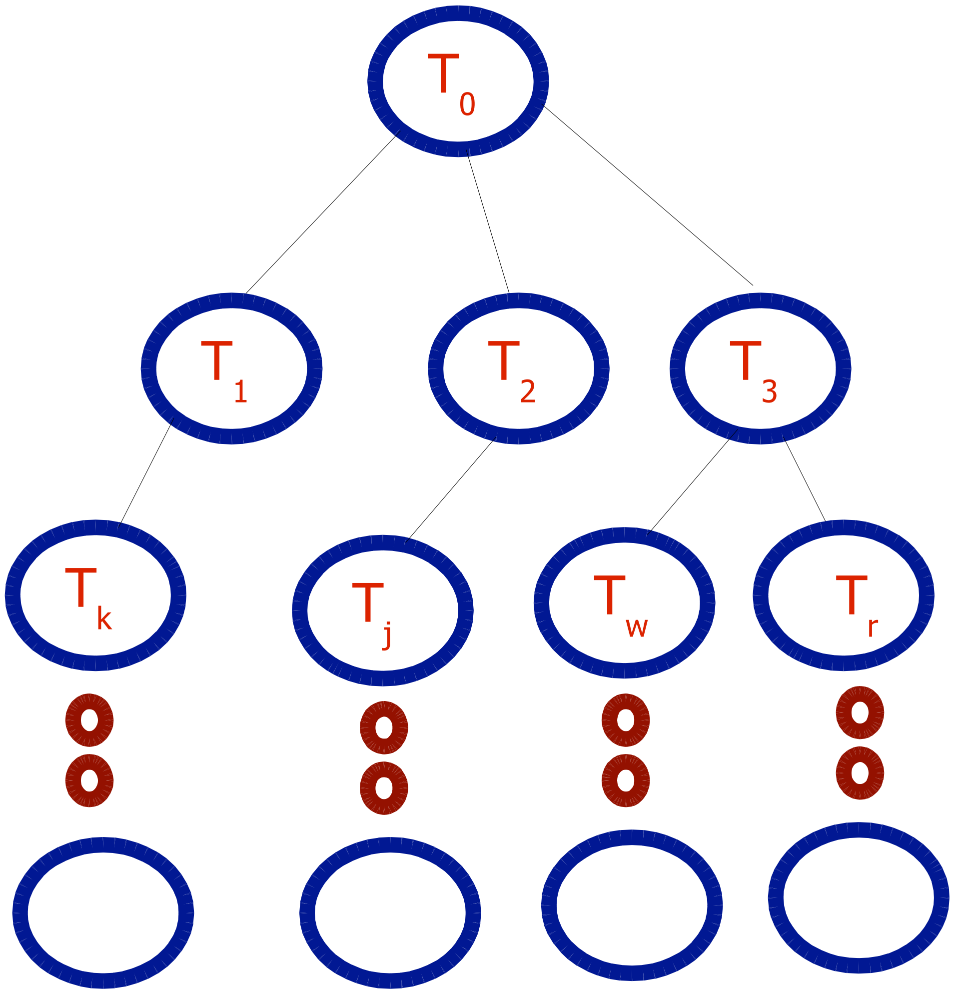
Point Location Tree (PLT)

Characterization of PLT

Characterization of PLT

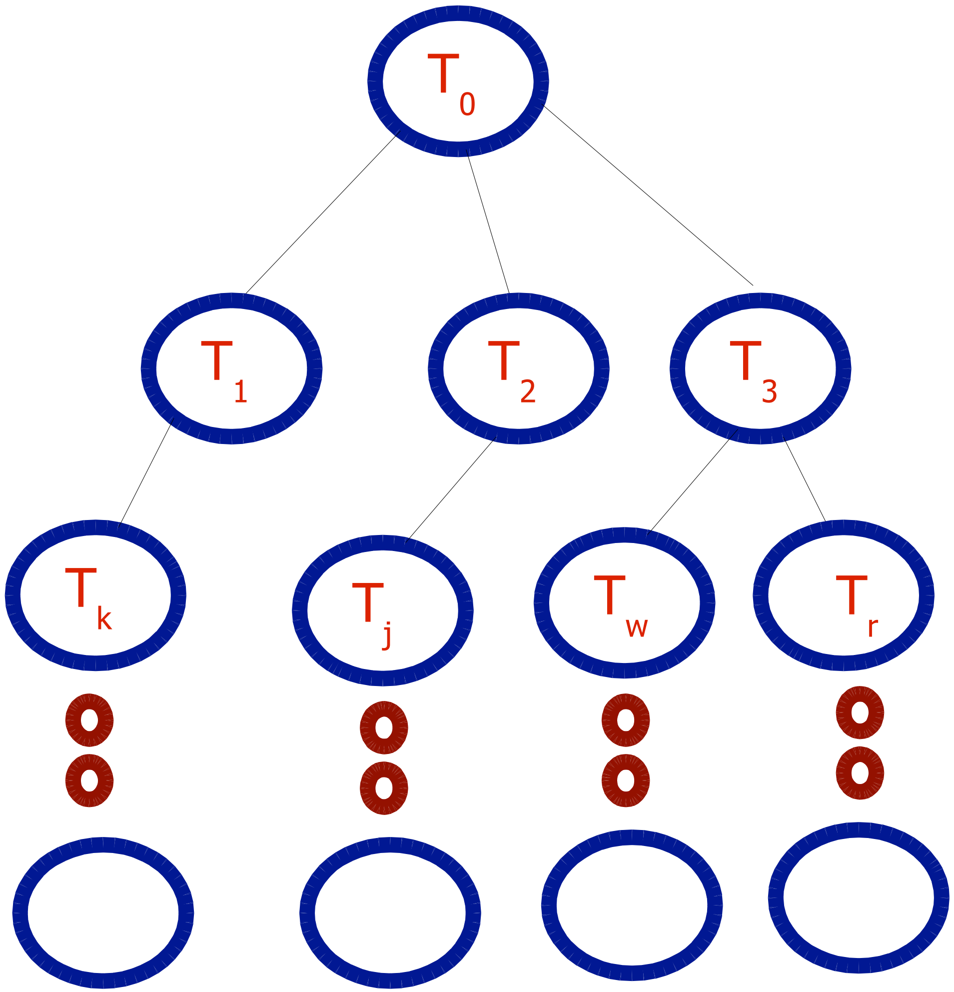


Characterization of PLT



1. Each nodes of PLT is a triangles interacting with O(1) leaf nodes which are also triangles

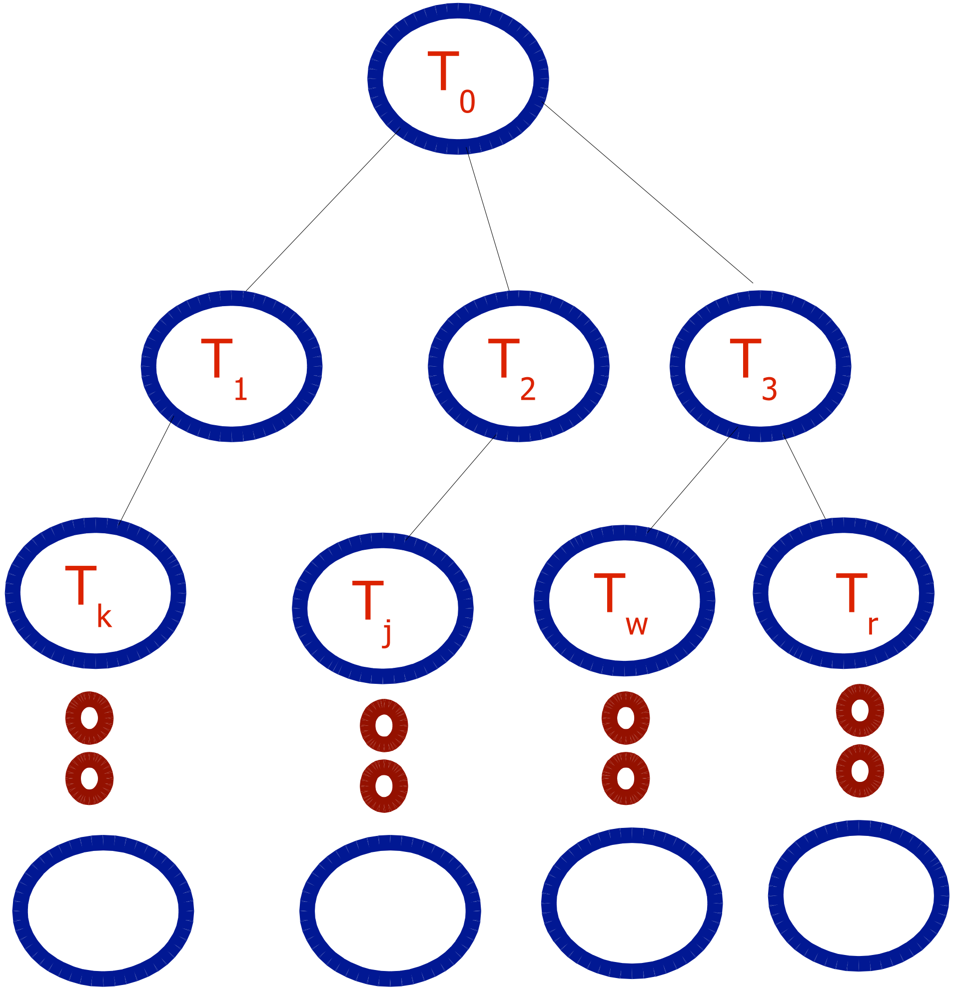
Characterization of PLT



1. Each nodes of PLT is a triangles interacting with O(1) leaf nodes which are also triangles

2. Leaf nodes of the tree are the original triangles of the planar subdivision we started with.

Characterization of PLT

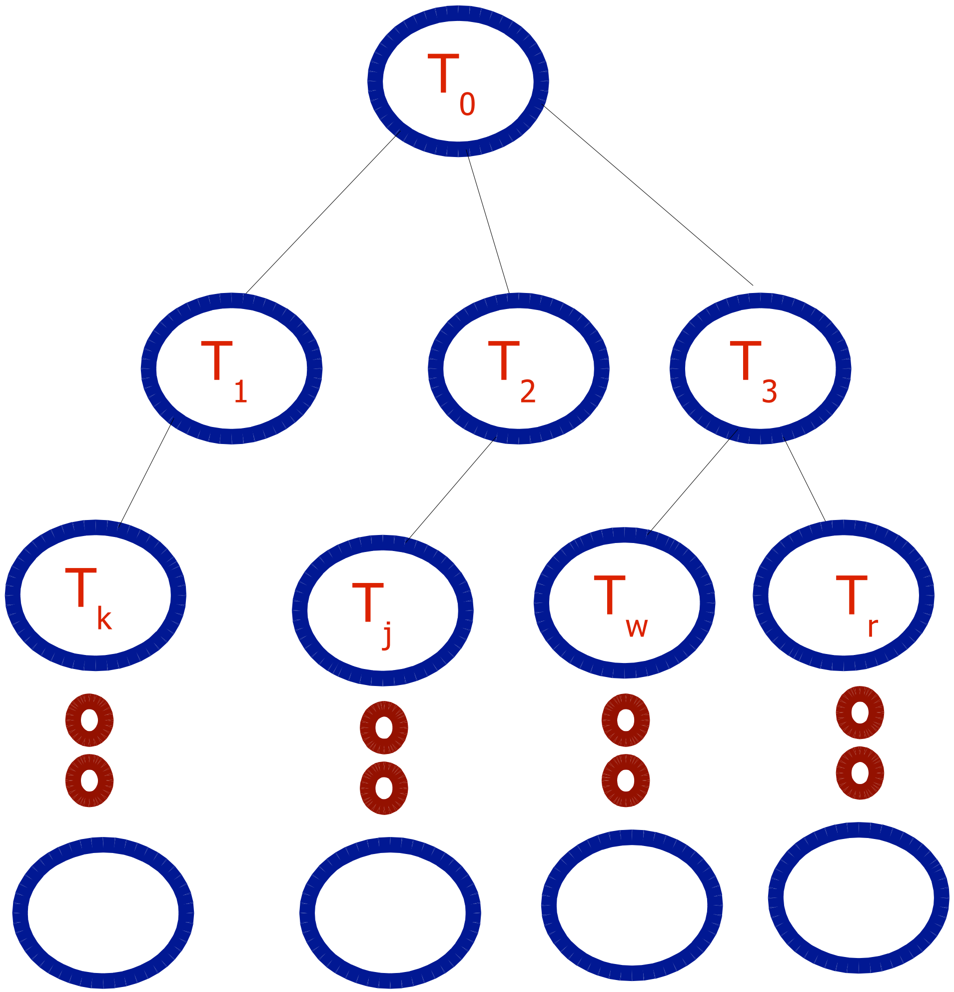


1. Each nodes of PLT is a triangles interacting with O(1) leaf nodes which are also triangles

2. Leaf nodes of the tree are the original triangles of the planar subdivision we started with.

3. Depth of PLT is $O(\log n)$

Characterization of PLT



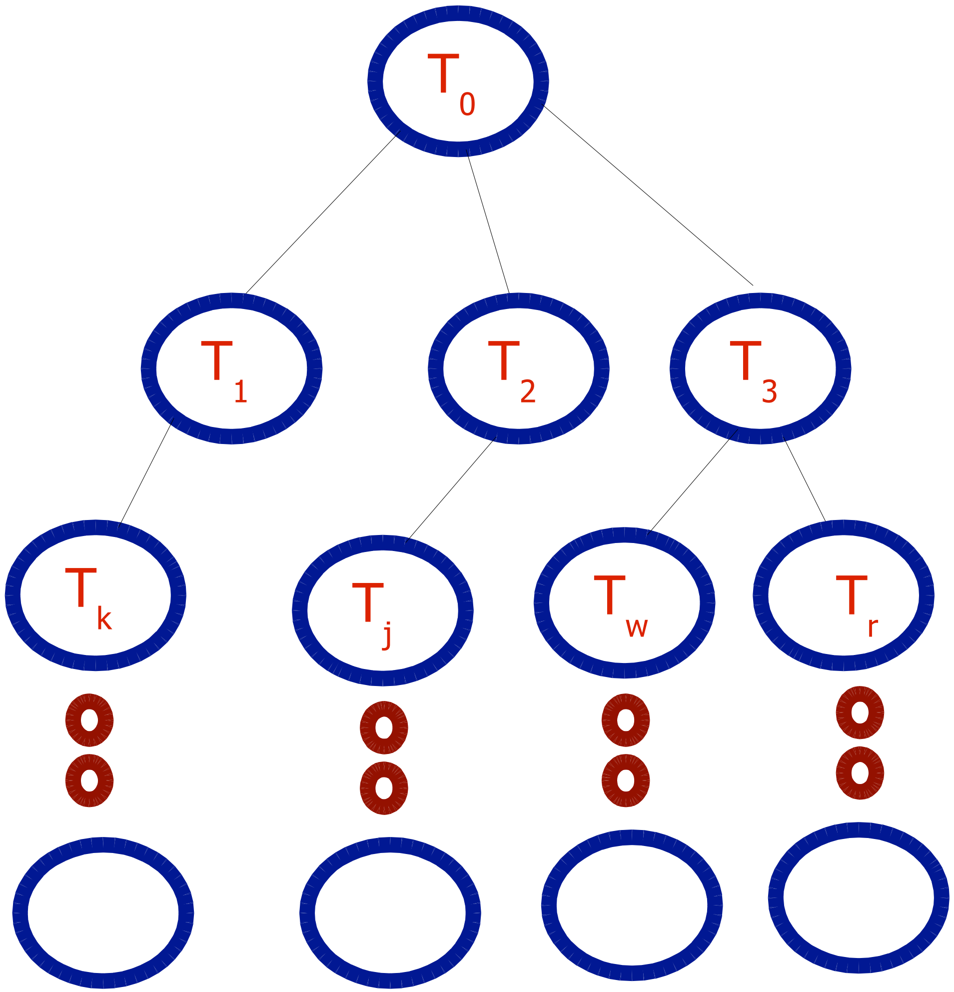
1. Each nodes of PLT is a triangles interacting with O(1) leaf nodes which are also triangles

2. Leaf nodes of the tree are the original triangles of the planar subdivision we started with.

3. Depth of PLT is $O(\log n)$

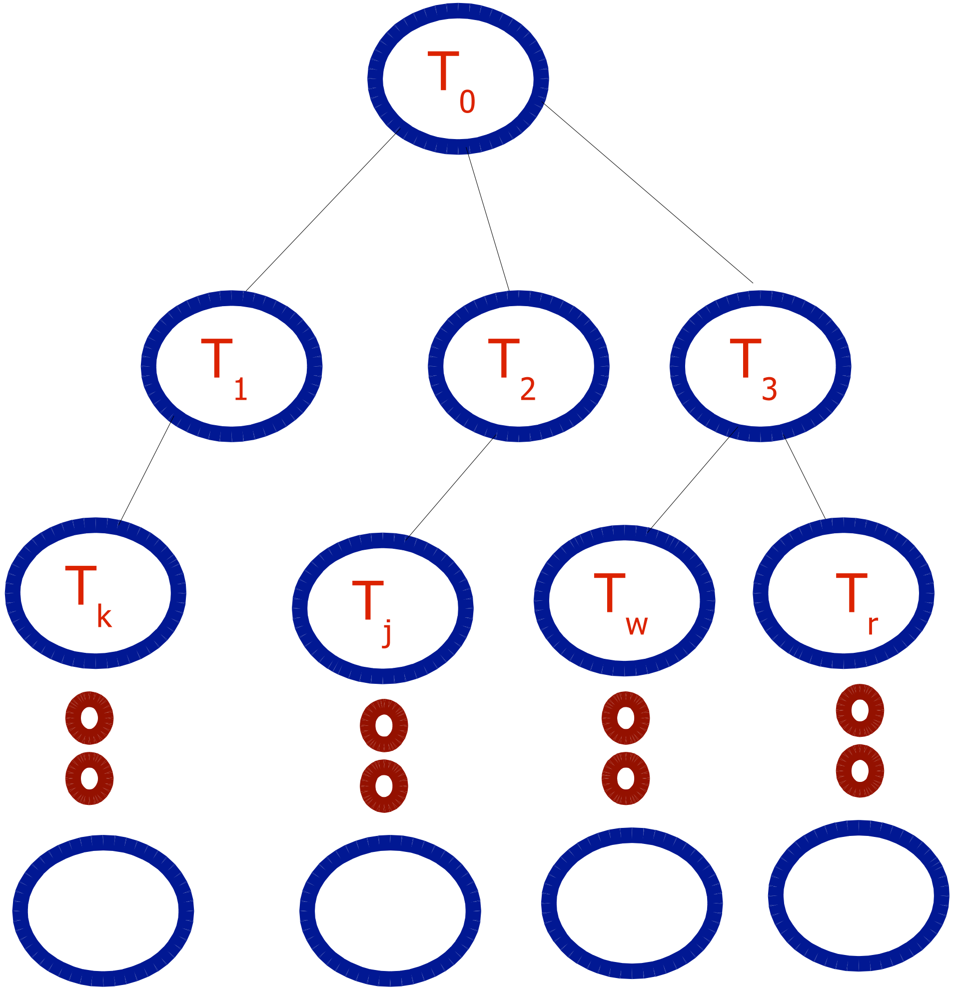
4. Construction time:

Characterization of PLT



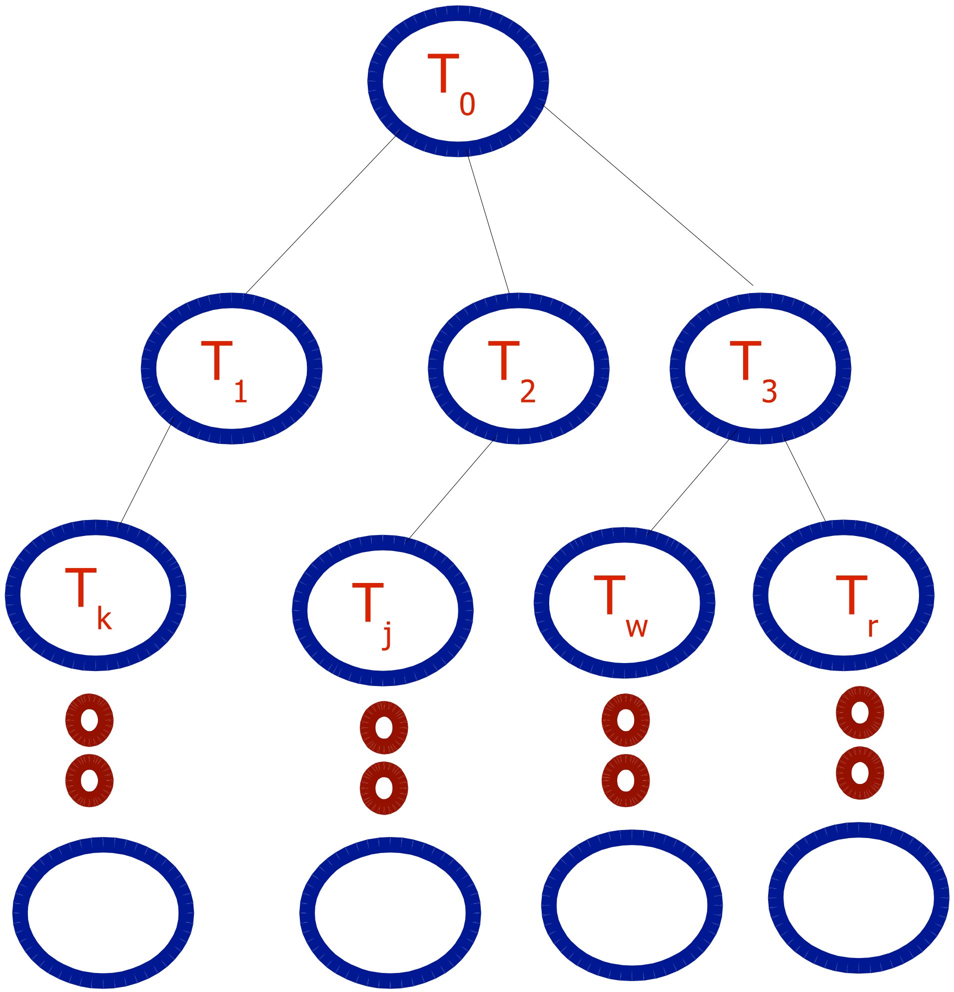
1. Each nodes of PLT is a triangles interacting with O(1) leaf nodes which are also triangles
2. Leaf nodes of the tree are the original triangles of the planar subdivision we started with.
3. Depth of PLT is $O(\log n)$
4. Construction time: $O(n)$

Characterization of PLT



1. Each nodes of PLT is a triangles interacting with O(1) leaf nodes which are also triangles
2. Leaf nodes of the tree are the original triangles of the planar subdivision we started with.
3. Depth of PLT is O(log n)
4. Construction time: O(n)
5. Space requirement:

Characterization of PLT



1. Each nodes of PLT is a triangles interacting with O(1) leaf nodes which are also triangles

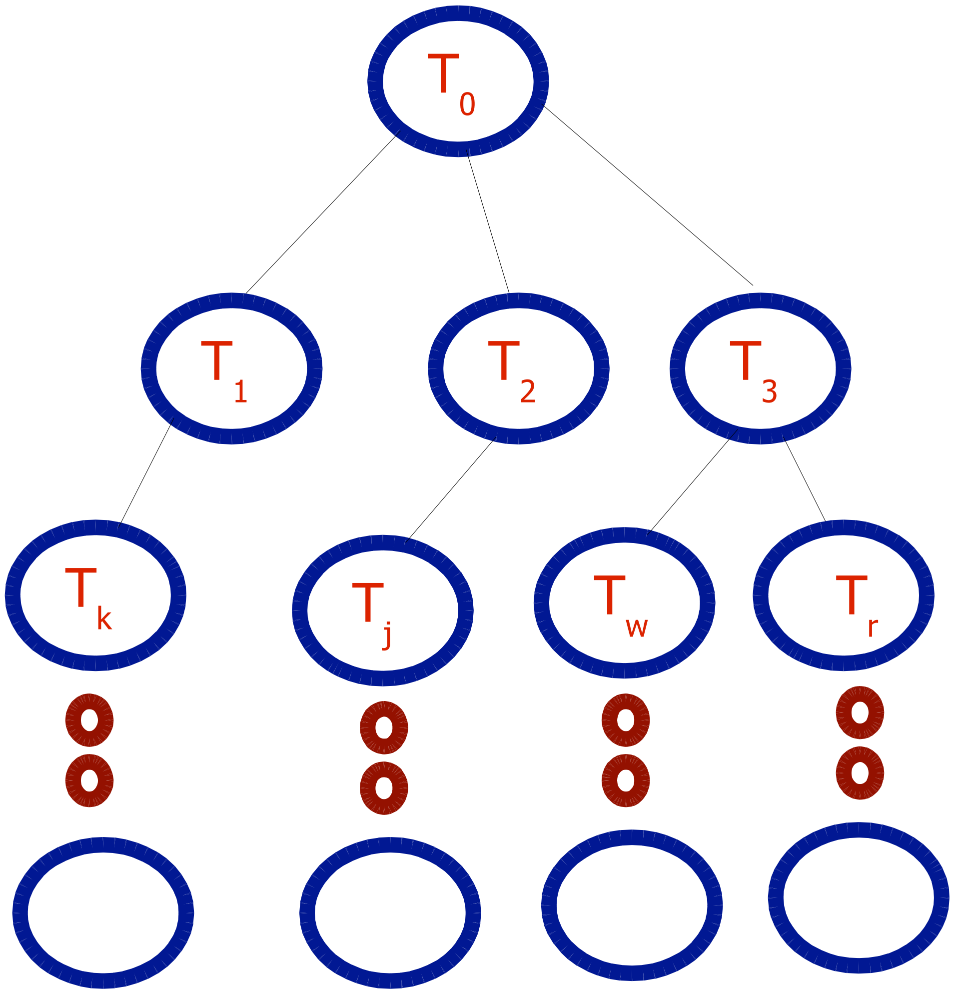
2. Leaf nodes of the tree are the original triangles of the planar subdivision we started with.

3. Depth of PLT is O(log n)

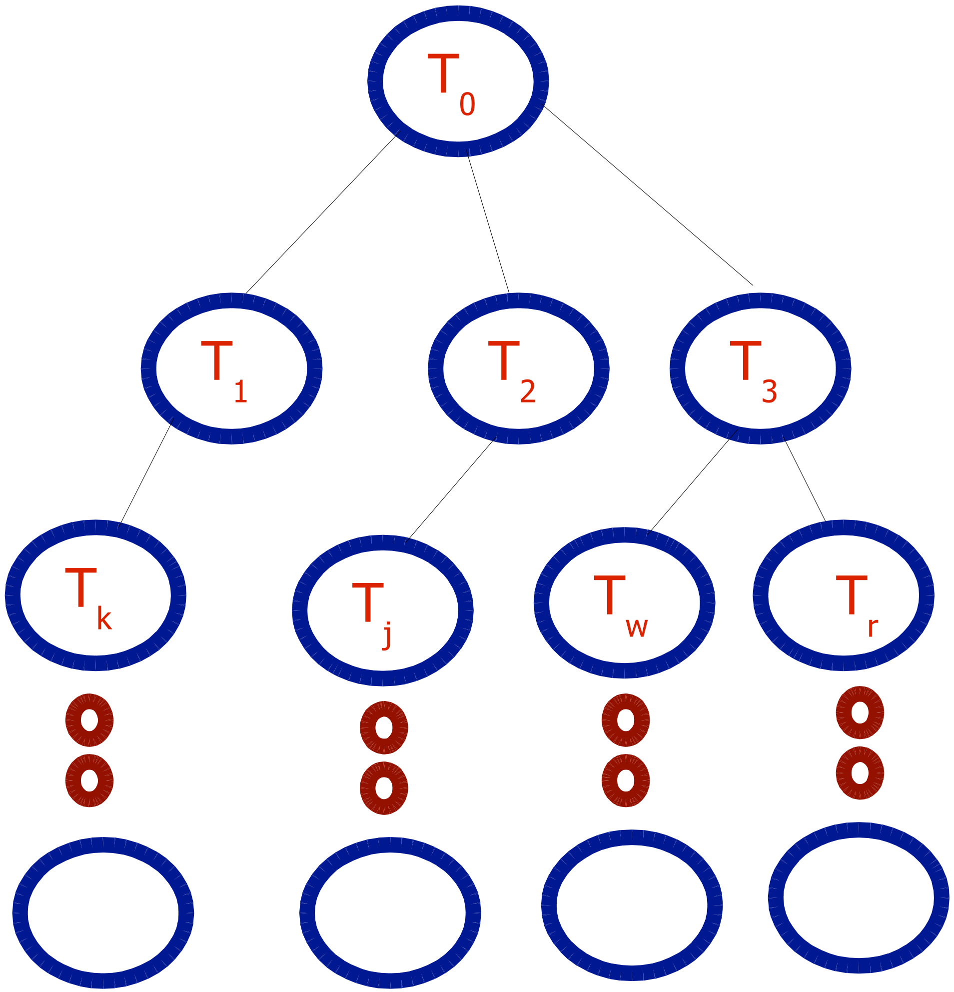
4. Construction time: O(n)

5. Space requirement: O(n)

Searching a query point q in PLT

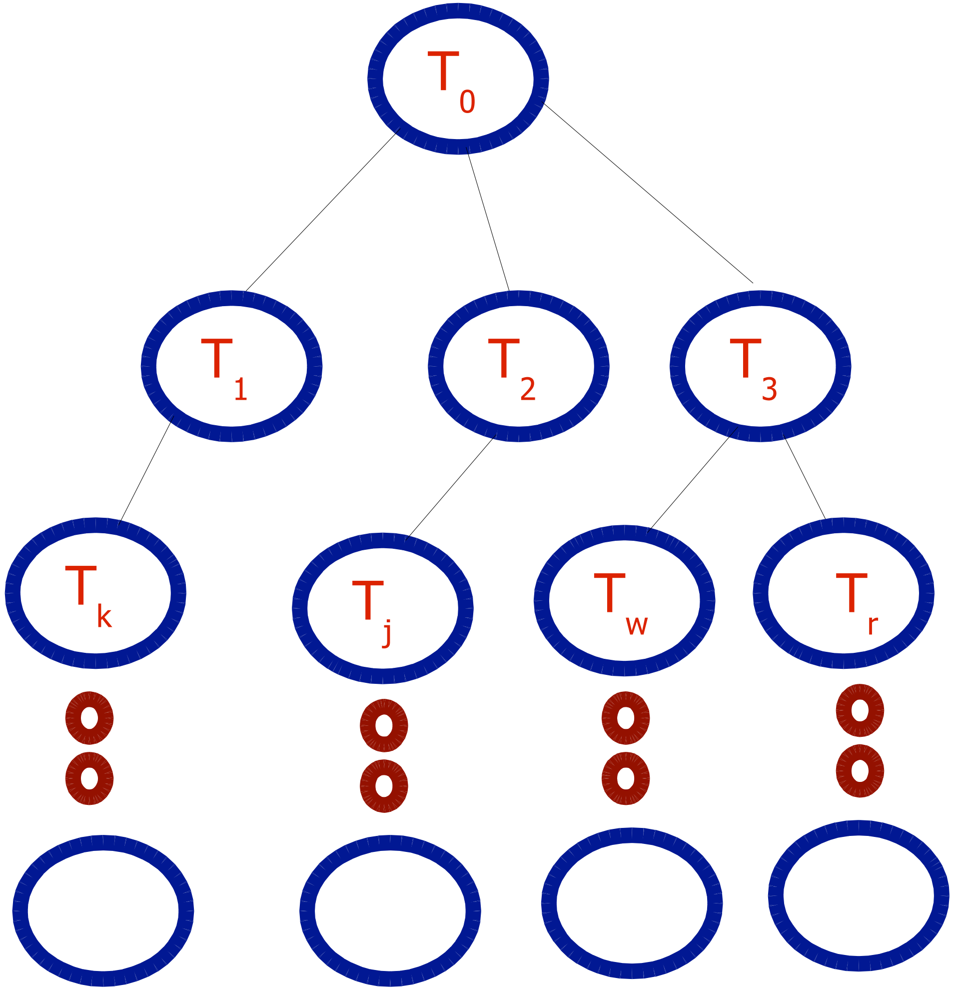


Searching a query point q in PLT



1. Find if T_0 contains q.

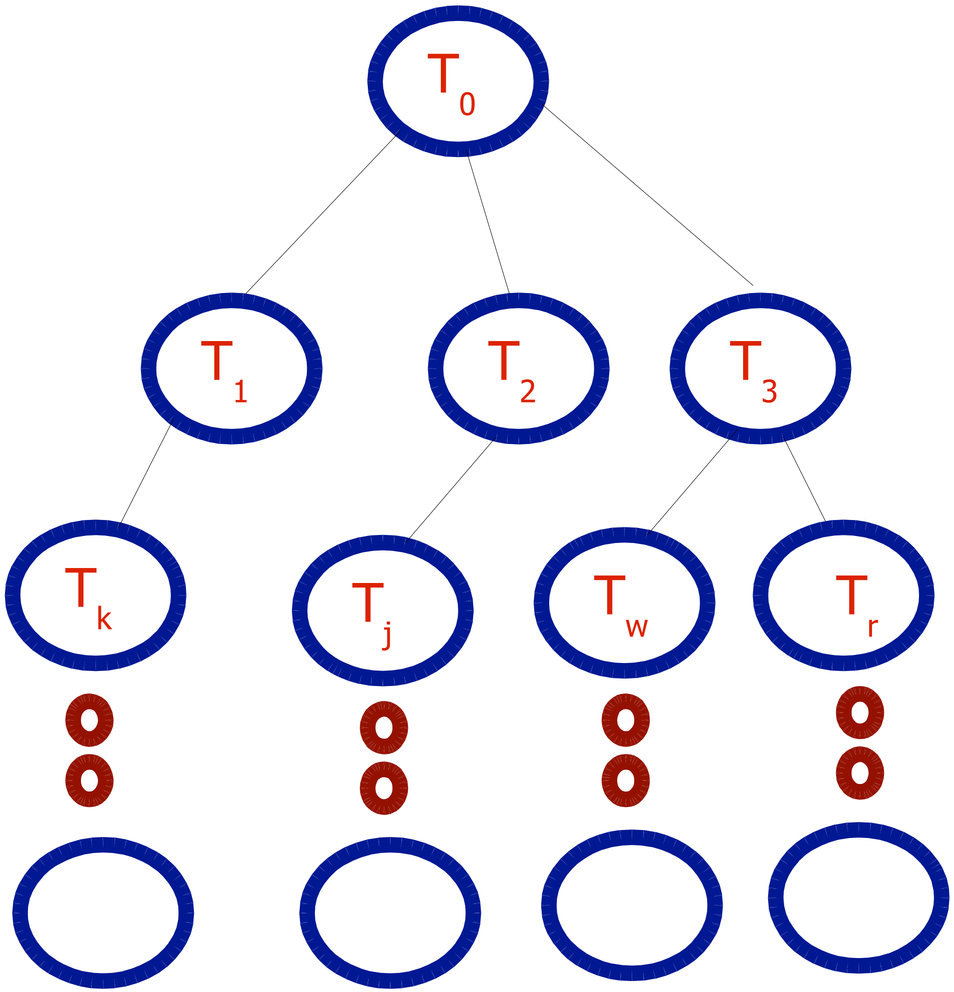
Searching a query point q in PLT



1. Find if T_0 contains q.

2. If no, then report q is out of range.

Searching a query point q in PLT

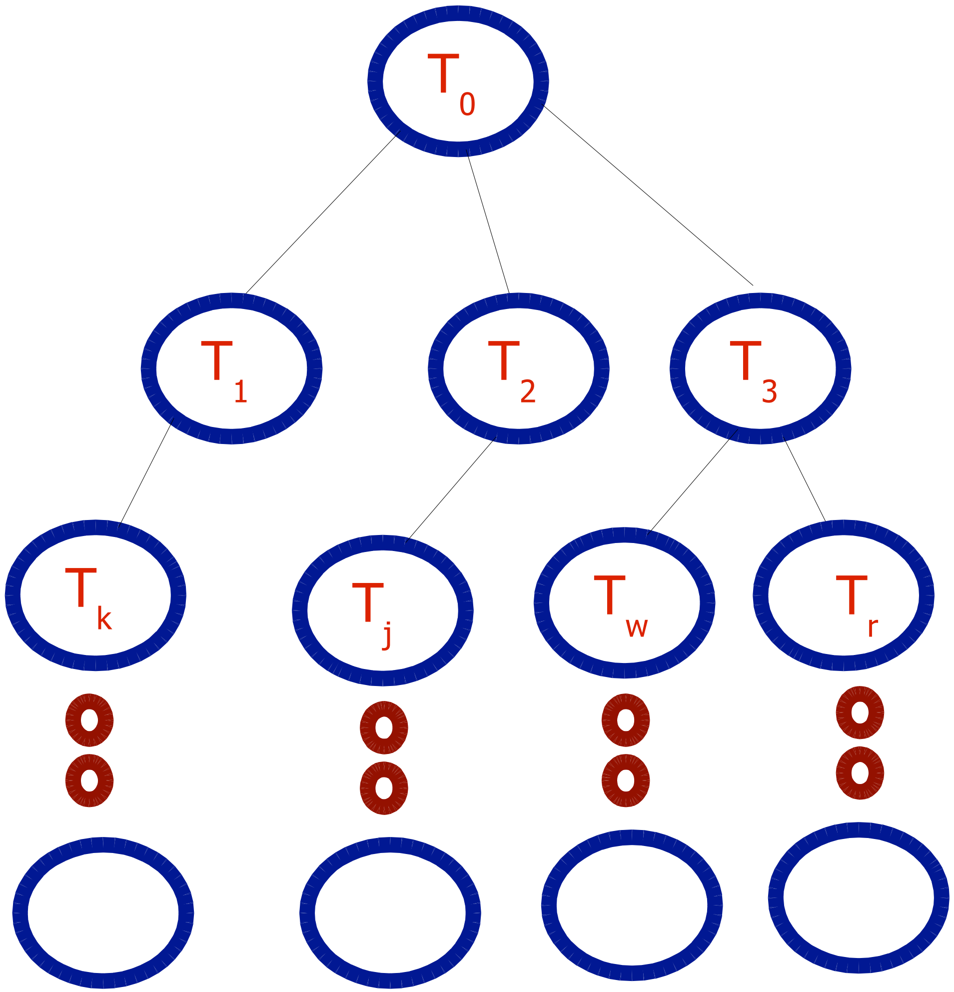


1. Find if T_0 contains q.

2. If no, then report q is out of range.

3. If yes, then find the leaf node of T_0 containing q recursively until we reach the correct leaf nodes

Searching a query point q in PLT



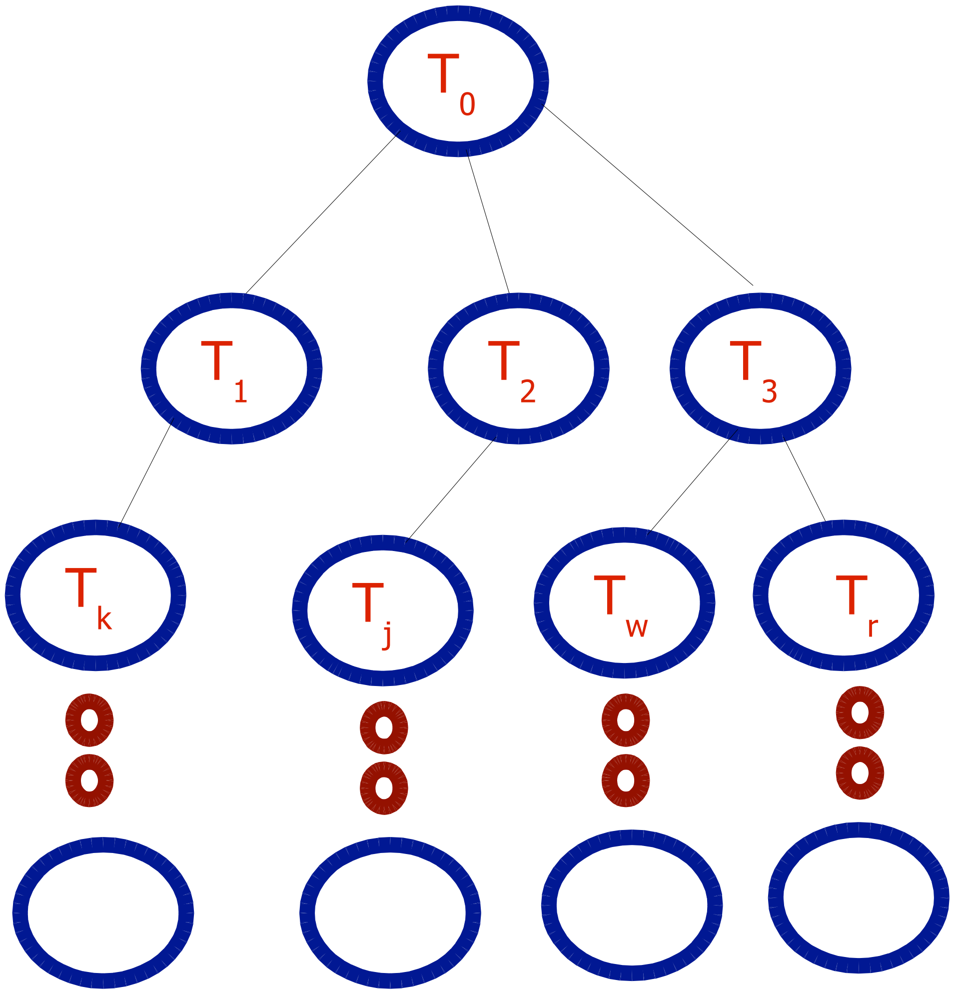
1. Find if T_0 contains q.

2. If no, then report q is out of range.

3. If yes, then find the leaf node of T_0 containing q recursively until we reach the correct leaf nodes

Query time complexity:

Searching a query point q in PLT



1. Find if T_0 contains q.

2. If no, then report q is out of range.

3. If yes, then find the leaf node of T_0 containing q recursively until we reach the correct leaf nodes

Query time complexity: $O(\log n)$

Organization of the Talk

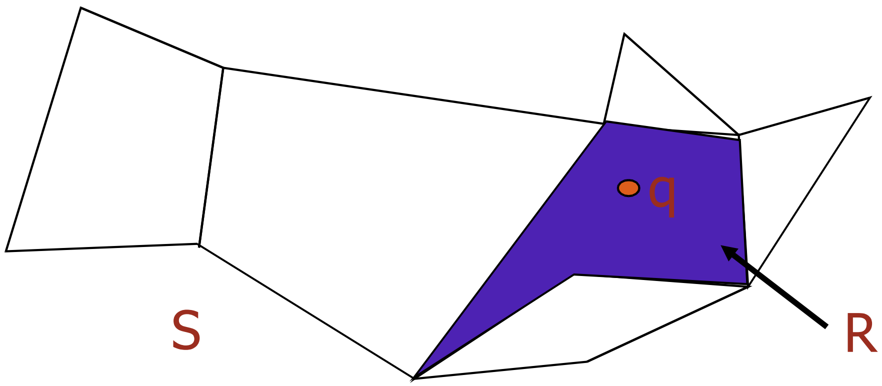
1. Preliminaries, Generic definition and Literature

2. Some technical details of easy versions

3. Conclusion

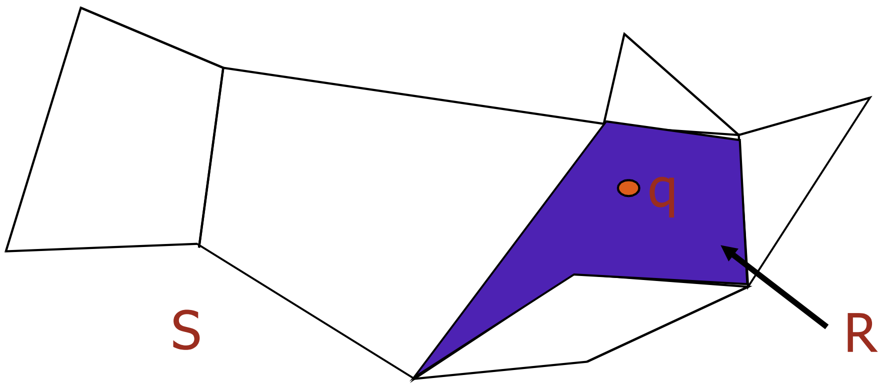
Planar Point Location

For a given a planar subdivision S



Planar Point Location

For a given a planar subdivision S

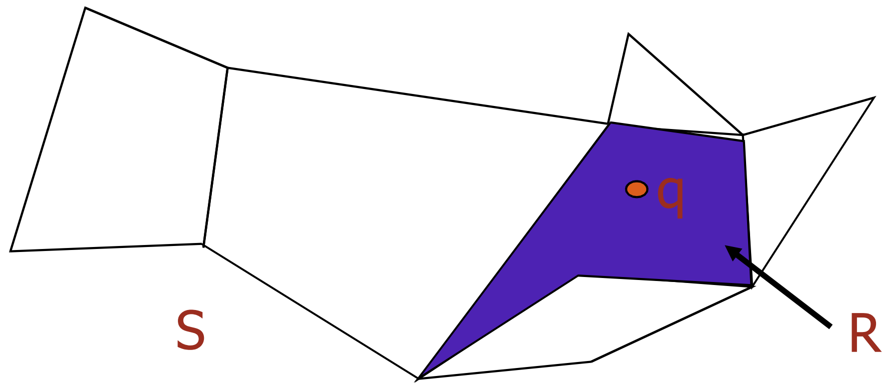


S can be preprocessed such that:

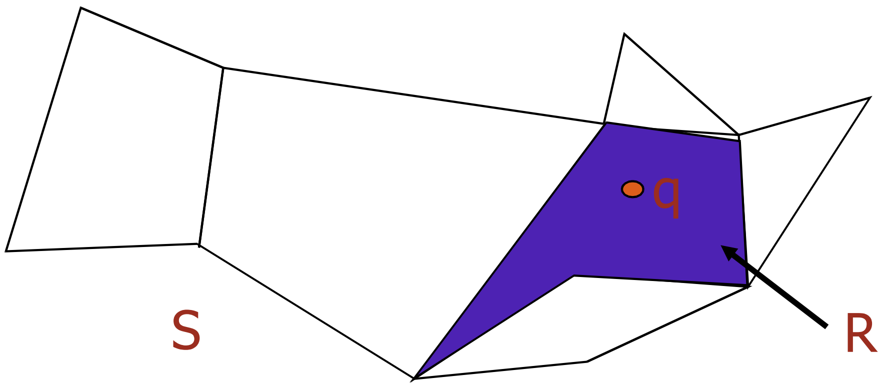
For any query point q,

The region/face R containing q can be reported efficiently.

Results

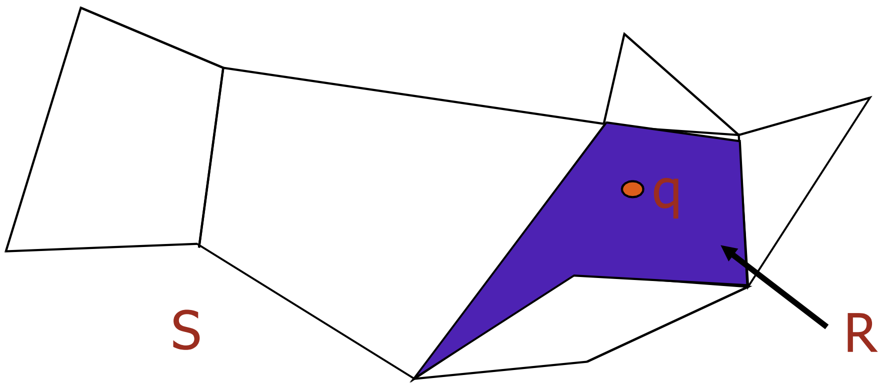


Results



Preprocessing Time: O(n)

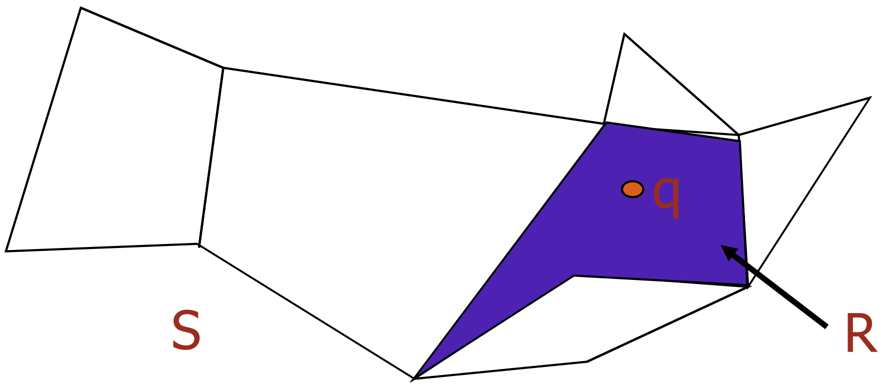
Results



Preprocessing Time: $O(n)$

Preprocessing space requirement: $O(n)$

Results



Preprocessing Time: O(n)

Preprocessing space requirement: O(n)

Query Time: O(log n)

Thank You