# Touring a Sequence of Polygons


Moshe Dror University of Arizona

Alon Efrat University of Arizona

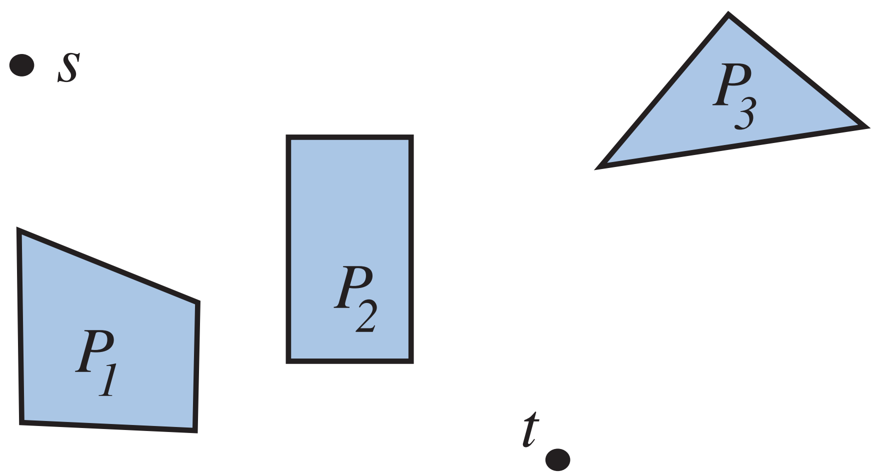
Anna Lubiw University of Waterloo

Joe Mitchell Stony Brook University

# Touring Polygons Problem

Given: a sequence of convex polygons, a start point *s* and a target point *t*

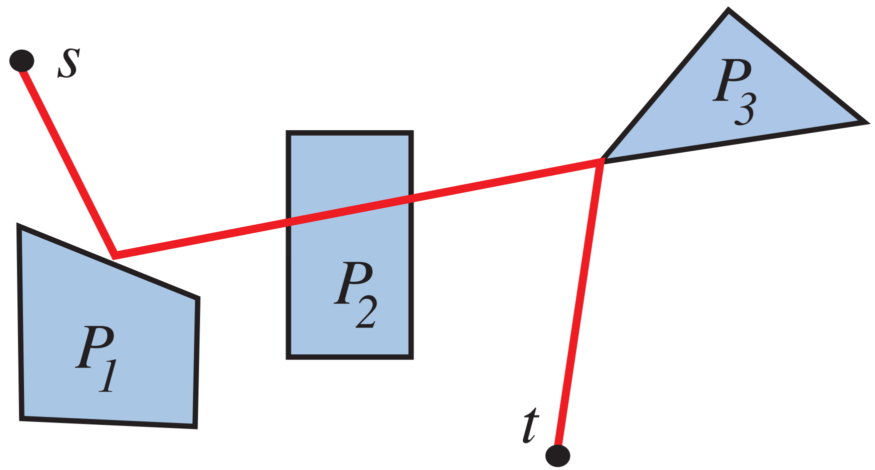
Find: a shortest path that starts at *s*, visits the polygons in sequence, and ends at *t*



# Touring Polygons Problem

Given: a sequence of convex polygons, a start point *s* and a target point *t*

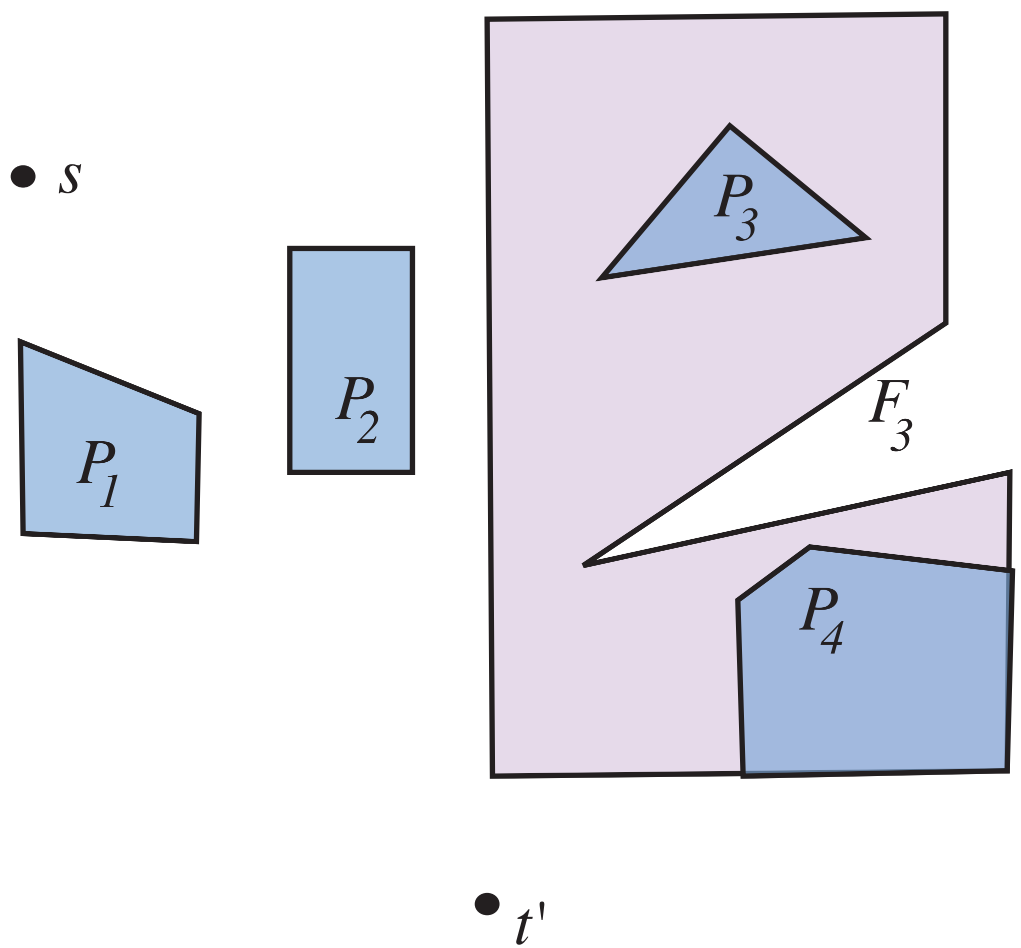
Find: a shortest path that starts at *s*, visits the polygons in sequence, and ends at *t*



# Touring Polygons Problem

Given: a sequence of convex polygons, a start point *s* and a target point *t*

Find: a shortest path that starts at *s*, visits the polygons in sequence, and ends at *t*



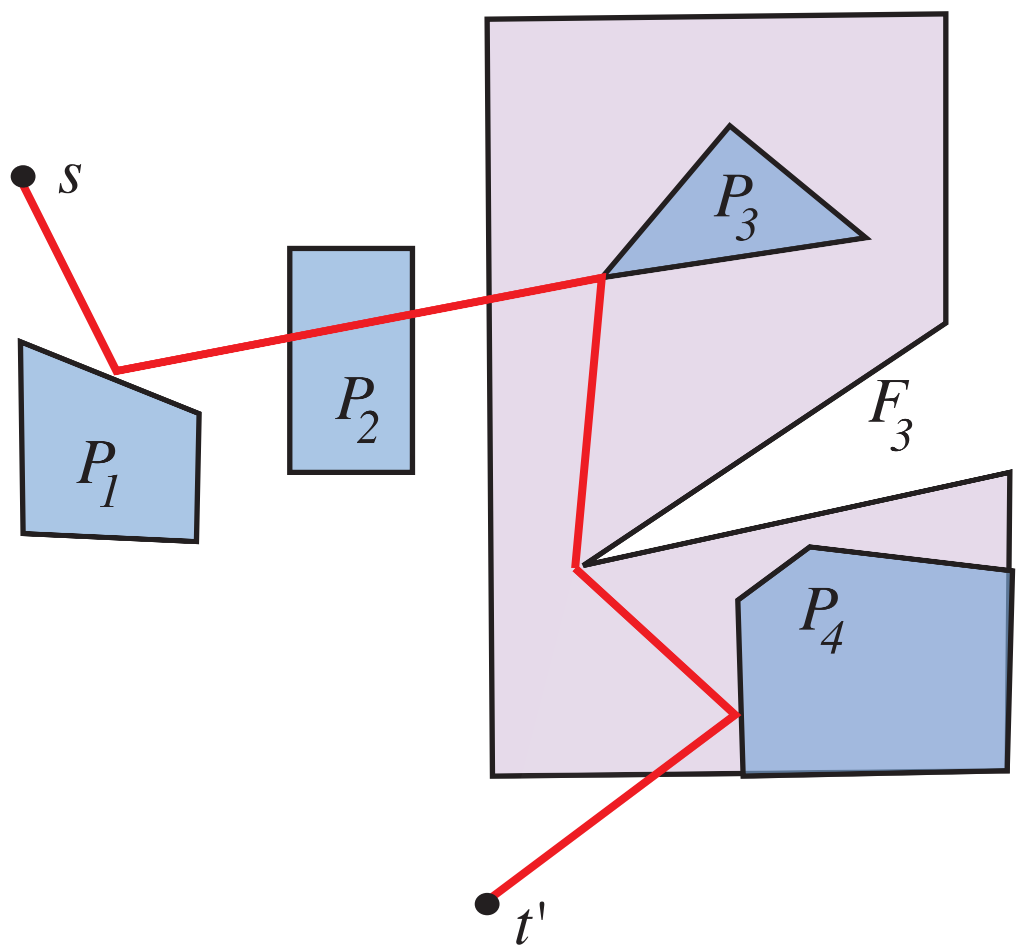

- the path may be constrained by *fences*

# Touring Polygons Problem

Given: a sequence of convex polygons, a start point *s* and a target point *t*

Find: a shortest path that starts at *s*, visits the polygons in sequence, and ends at *t*

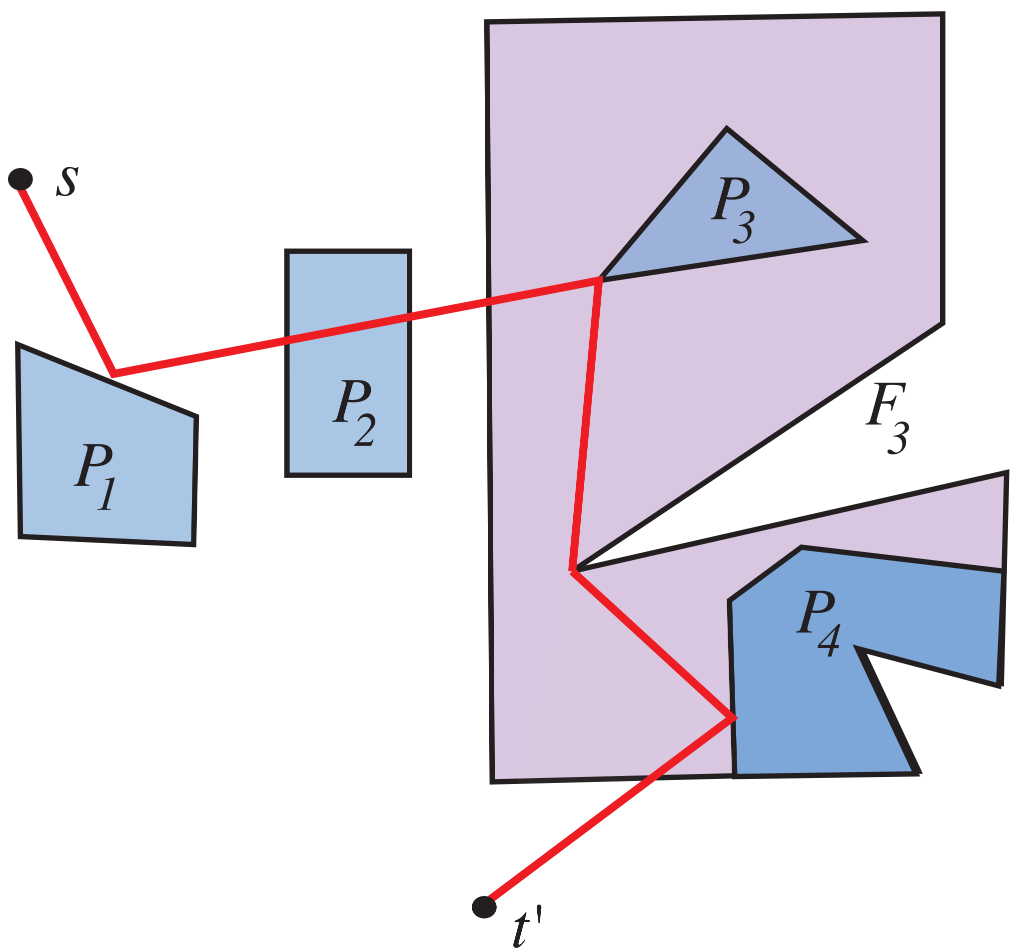
- the path may be constrained by *fences*



# Touring Polygons Problem

Given: a sequence of convex polygons, a start point *s* and a target point *t*

Find: a shortest path that starts at *s*, visits the polygons in sequence, and ends at *t*



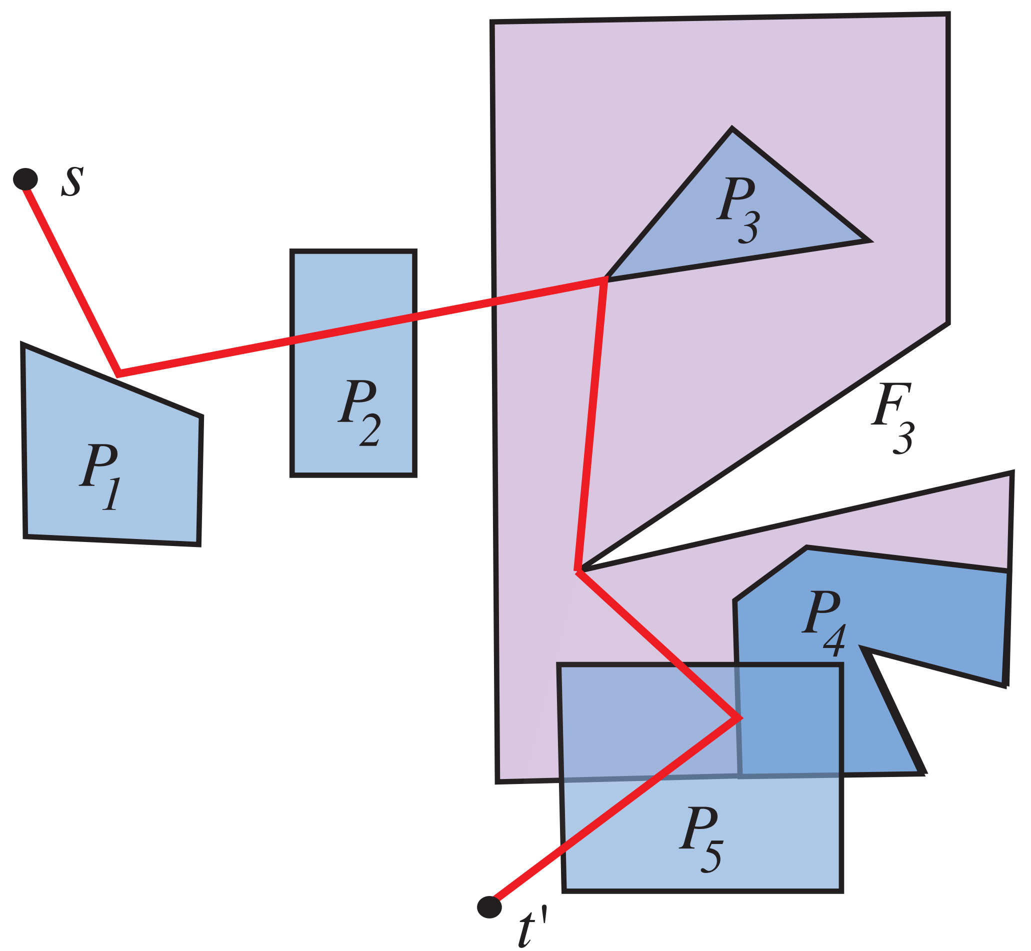

- the path may be constrained by *fences*

- only polygon *facade* must be convex

# Touring Polygons Problem

Given: a sequence of convex polygons, a start point *s* and a target point *t*

Find: a shortest path that starts at *s*, visits the polygons in sequence, and ends at *t*

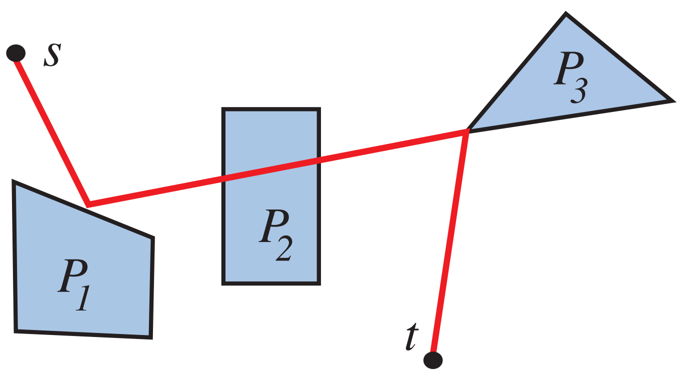



- the path may be constrained by *fences*
- only polygon *facade* must be convex
- polygons may intersect

# Our Algorithm

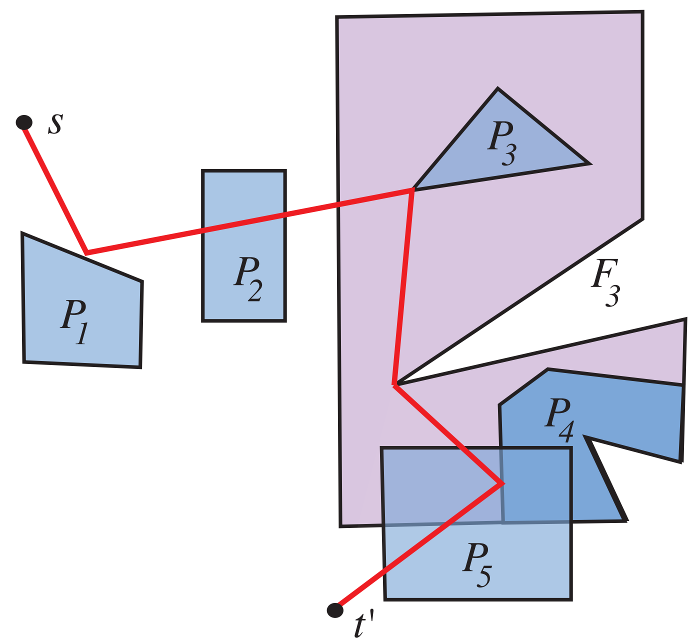
*n* = size of polygons and fences    *k* = number of polygons

★ unconstrained Touring Polygons Problem (TPP) with disjoint convex polygons




$O(kn \log n)$

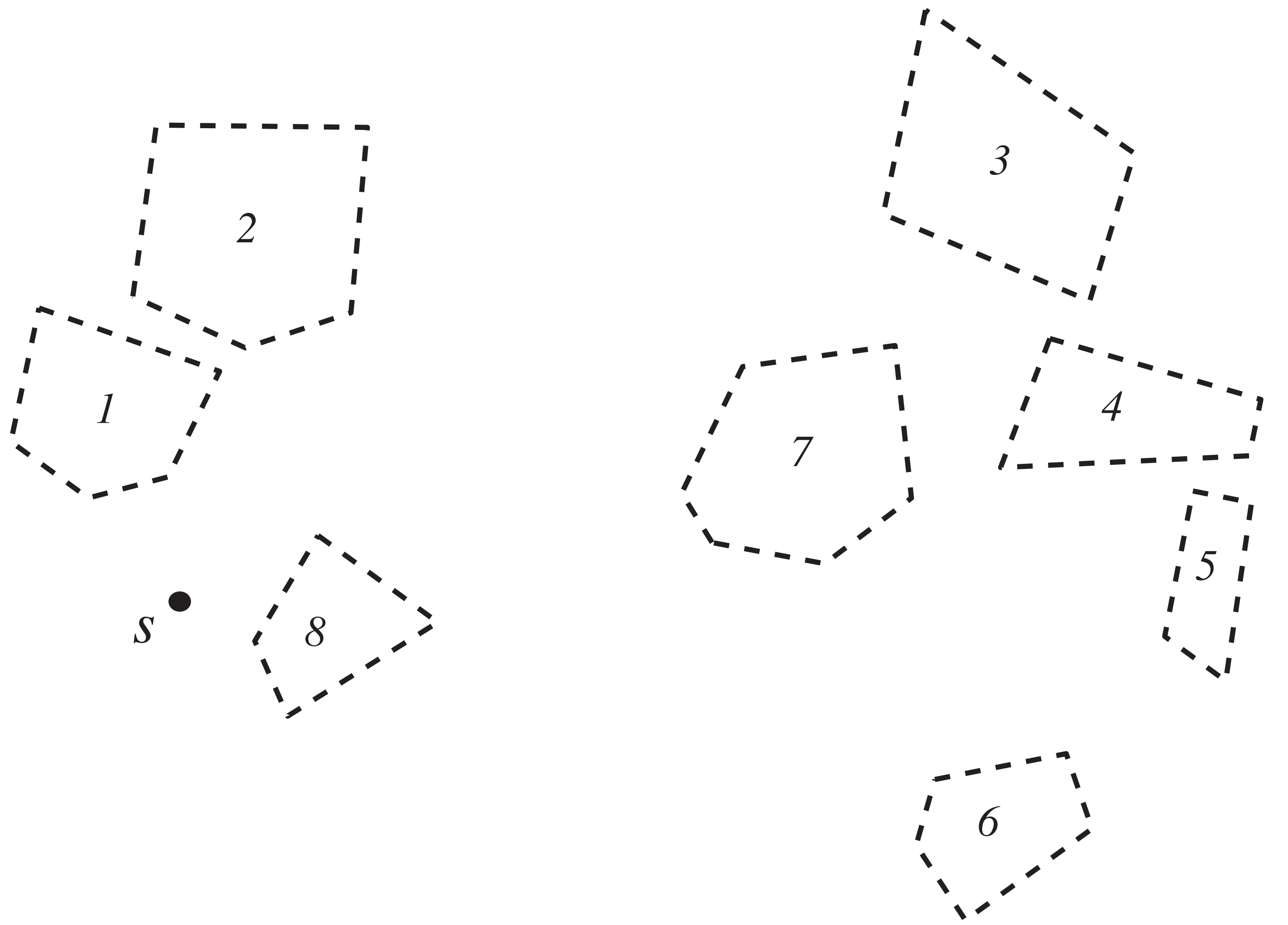
★ general TPP



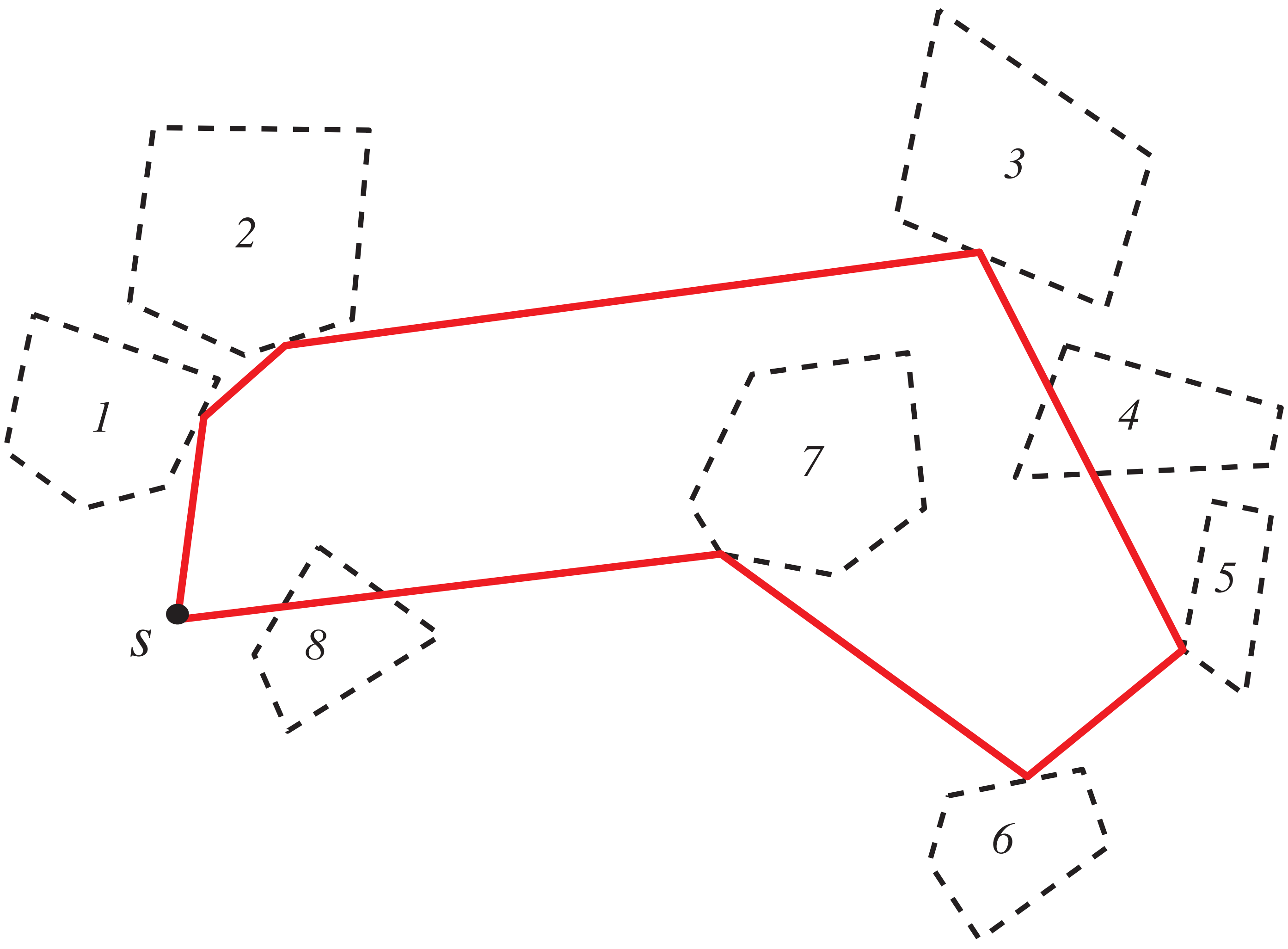

$O(k^2 n \log n)$

for fixed s, shortest path queries take $O(k \log n + \text{output-size})$

# Application: parts cutting

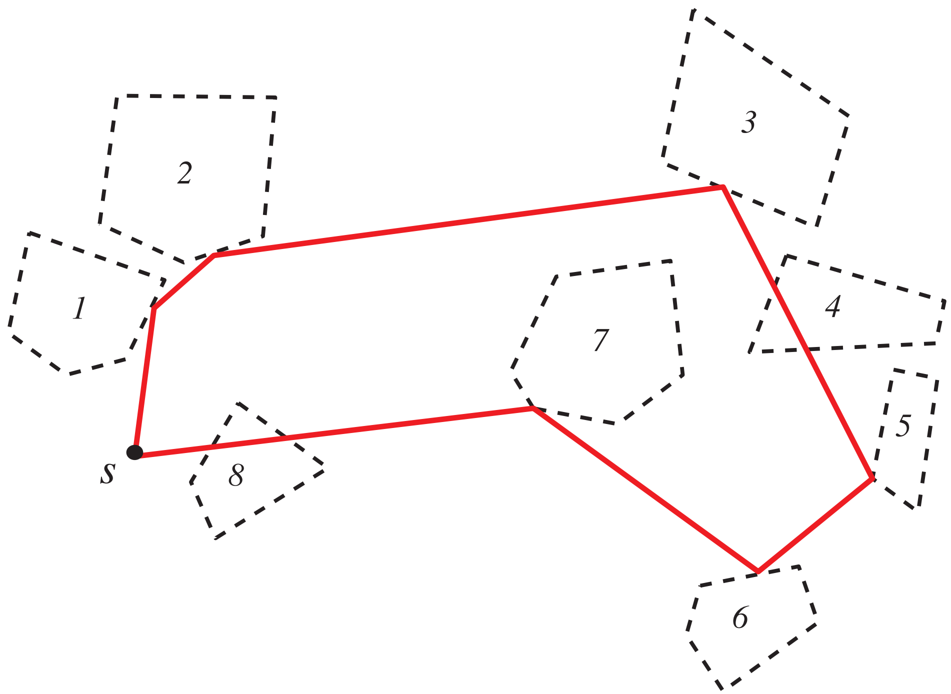


# Application: parts cutting




TPP – disjoint convex polygons
– no fences ("unconstrained")

# Application: parts cutting

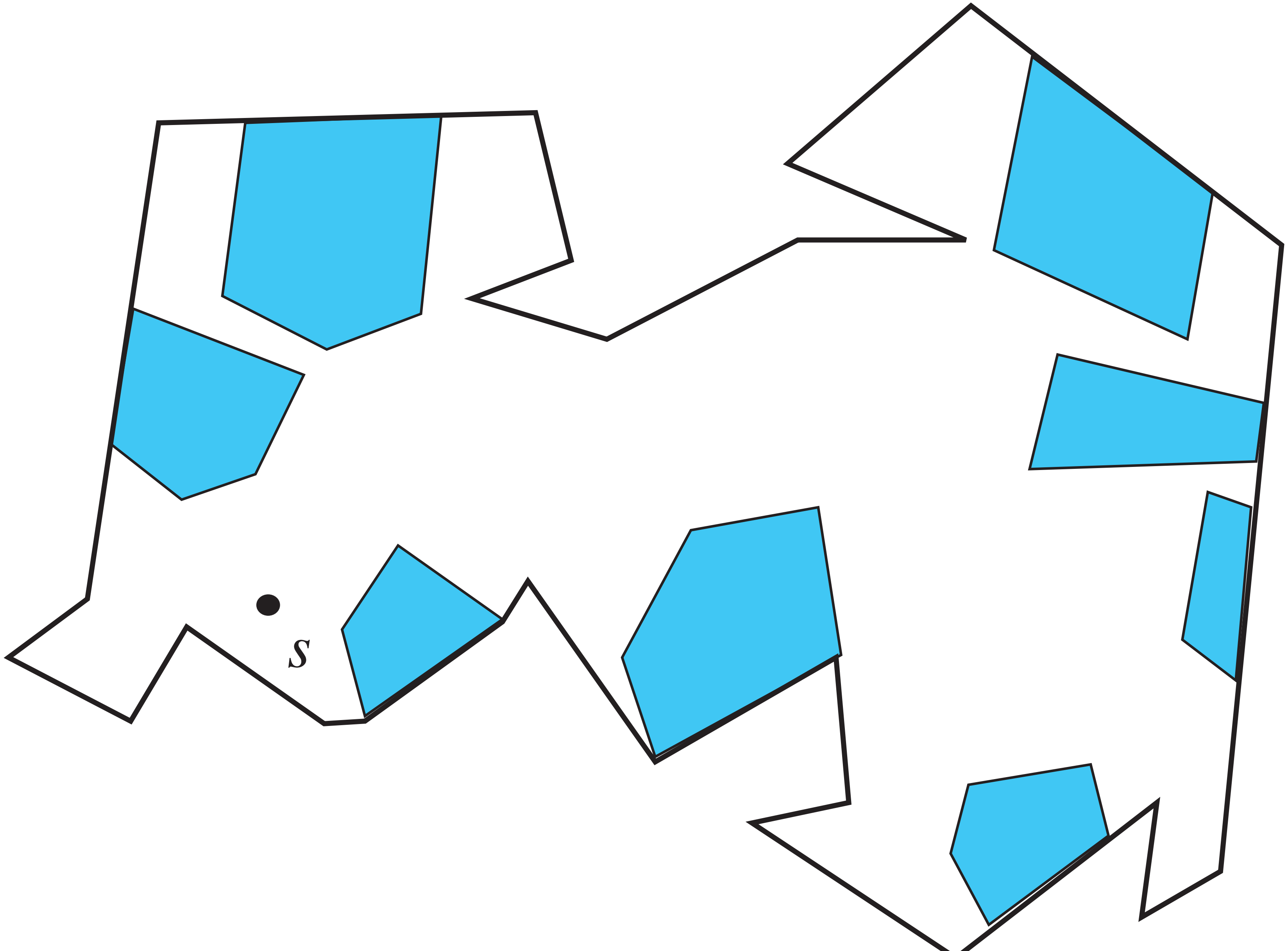



$O(kn \log n)$

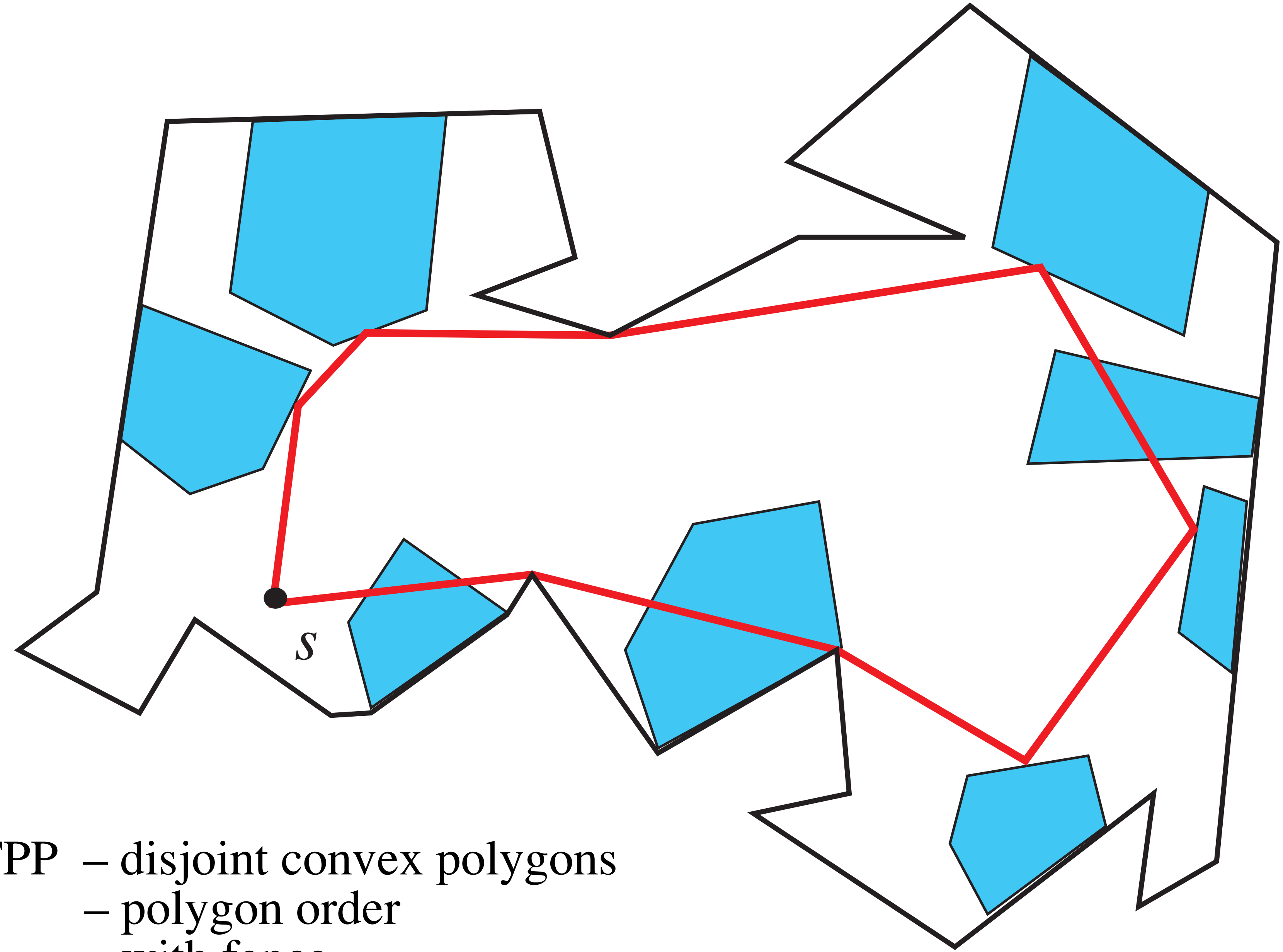
$k$ = number of polygons

$n$ = total size

# Application: safari problem

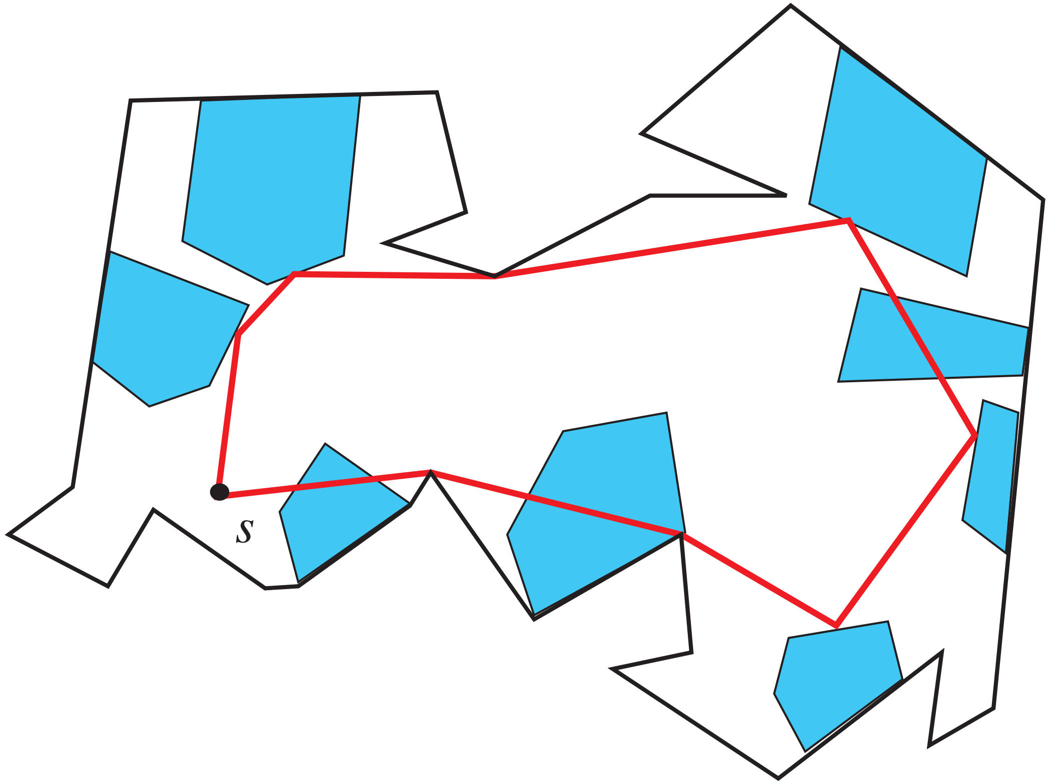


# Application: safari problem




TPP – disjoint convex polygons
– polygon order
– with fence

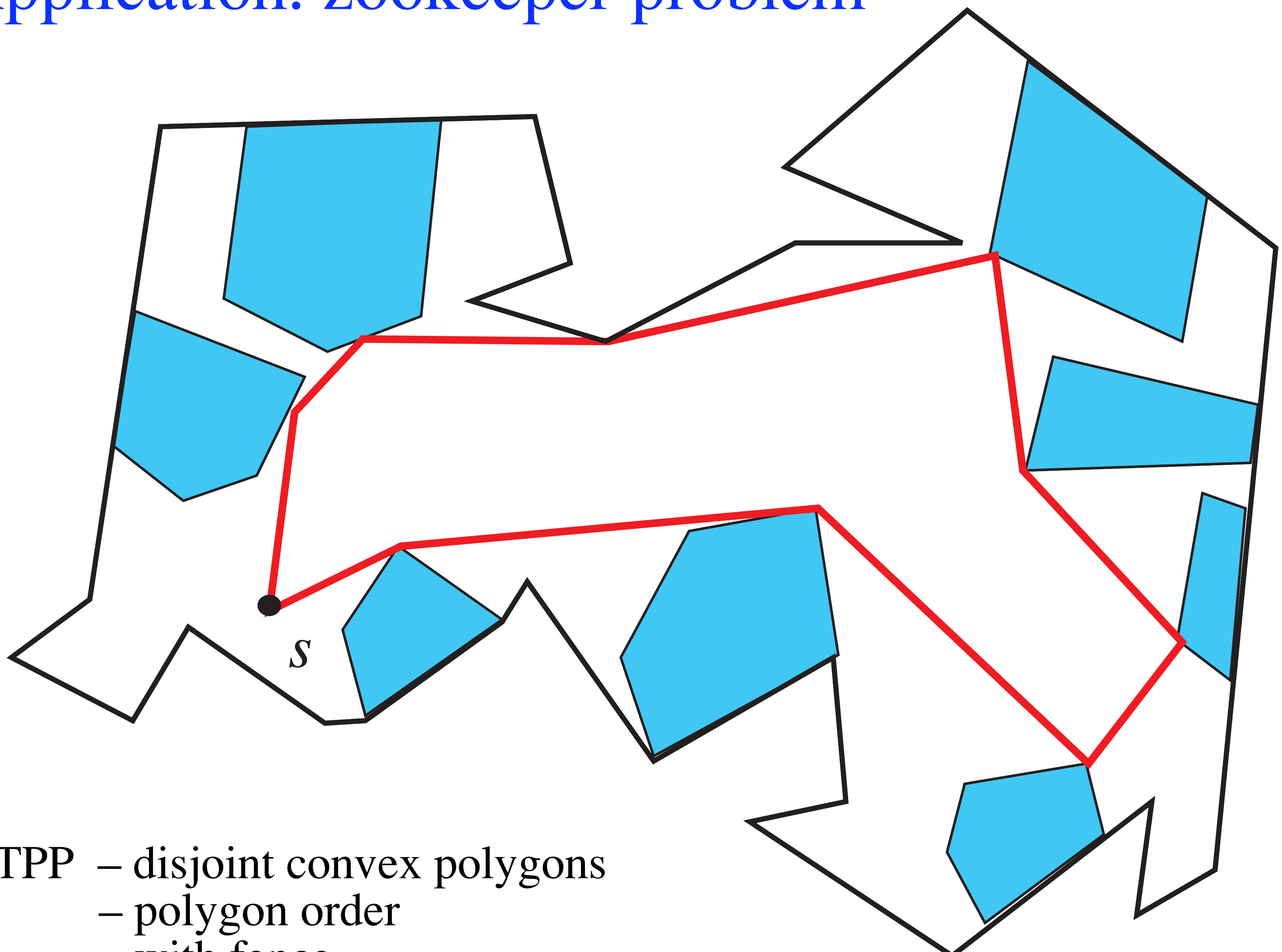
# Application: safari problem




- problem from Ntafos `92
- $O(n^3)$ `92
- $O(n^2)$ `94
- $O(n^3)$ Tan and Hirata `01

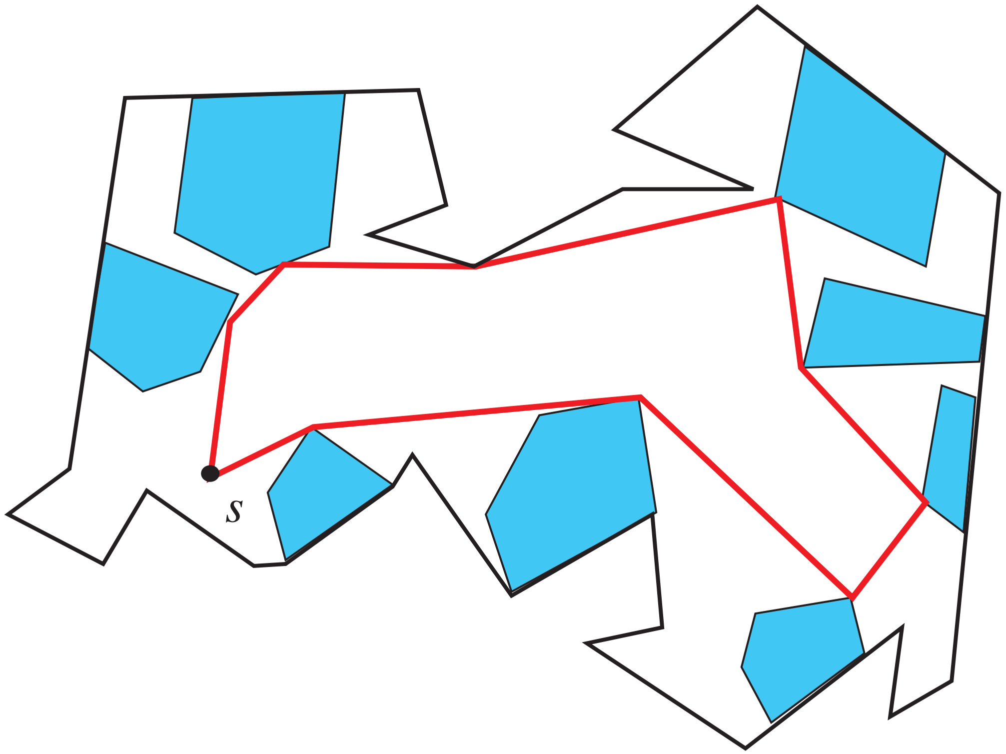
$$O(n^2 \log n)$$

# Application: zookeeper problem



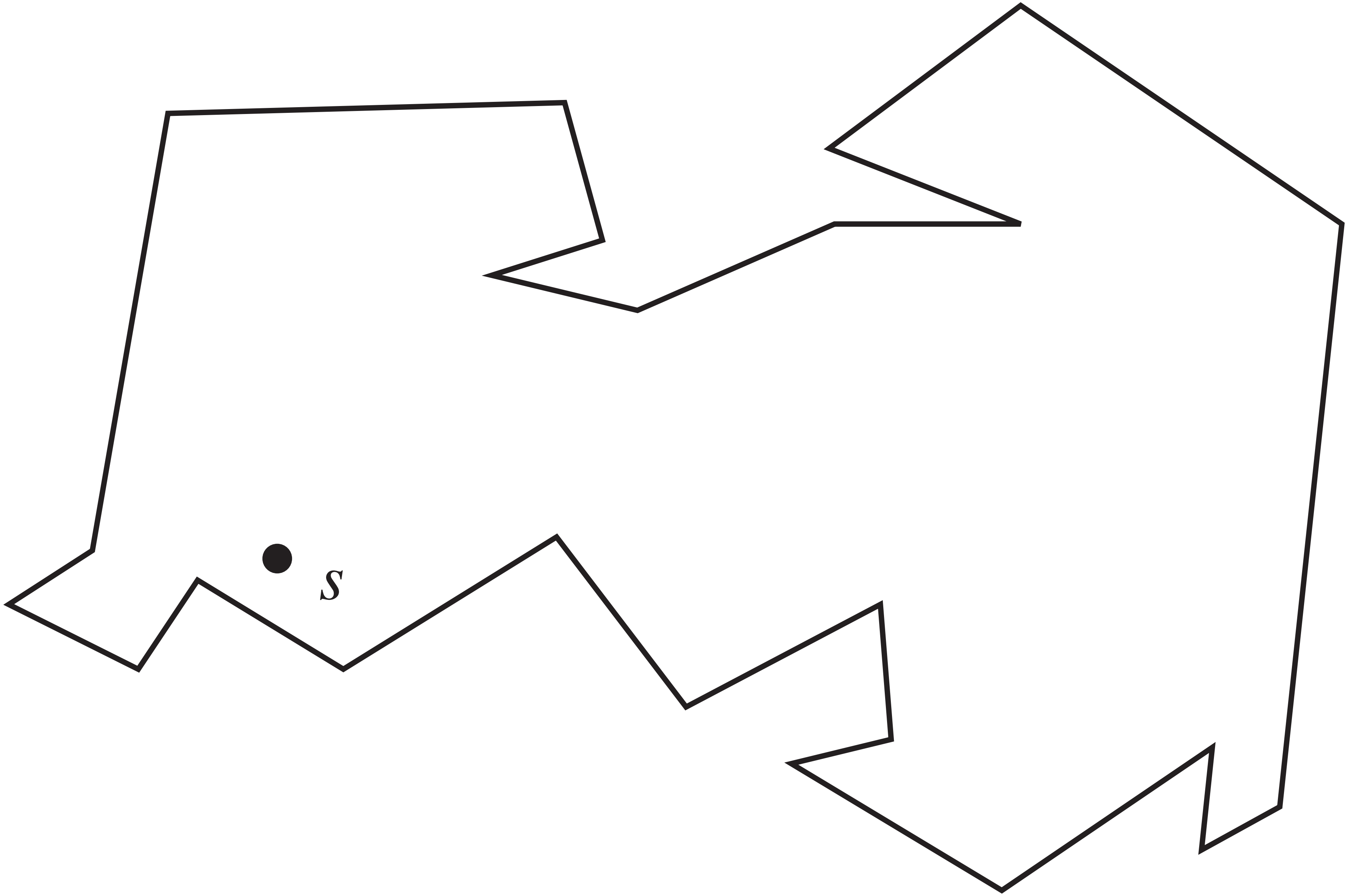

TPP – disjoint convex polygons
– polygon order
– with fence

# Application: zookeeper problem

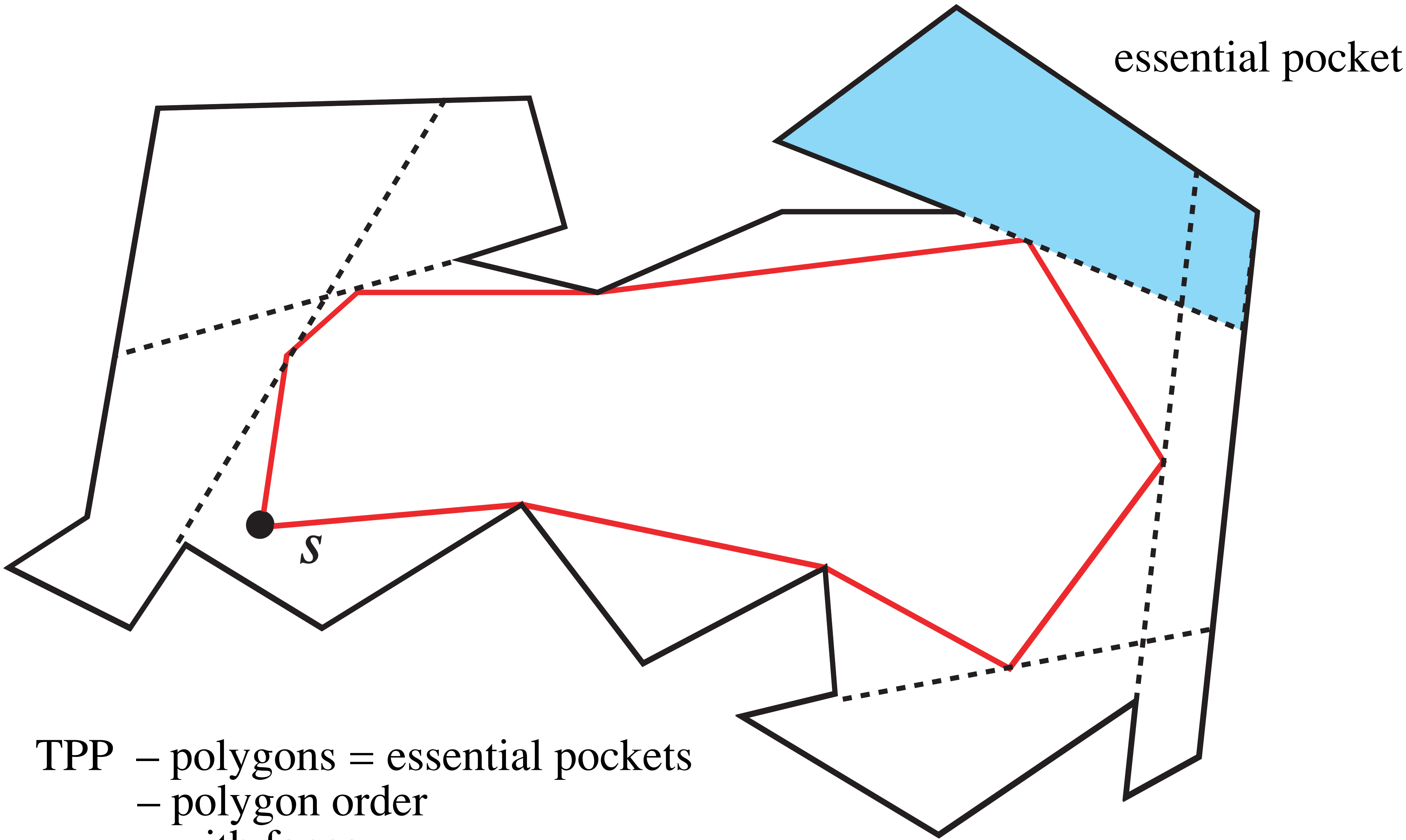



- problem from Chin and Ntafos `92
- O($n$ log $n$) Bespamyatnikh `02

# Application: watchman route problem

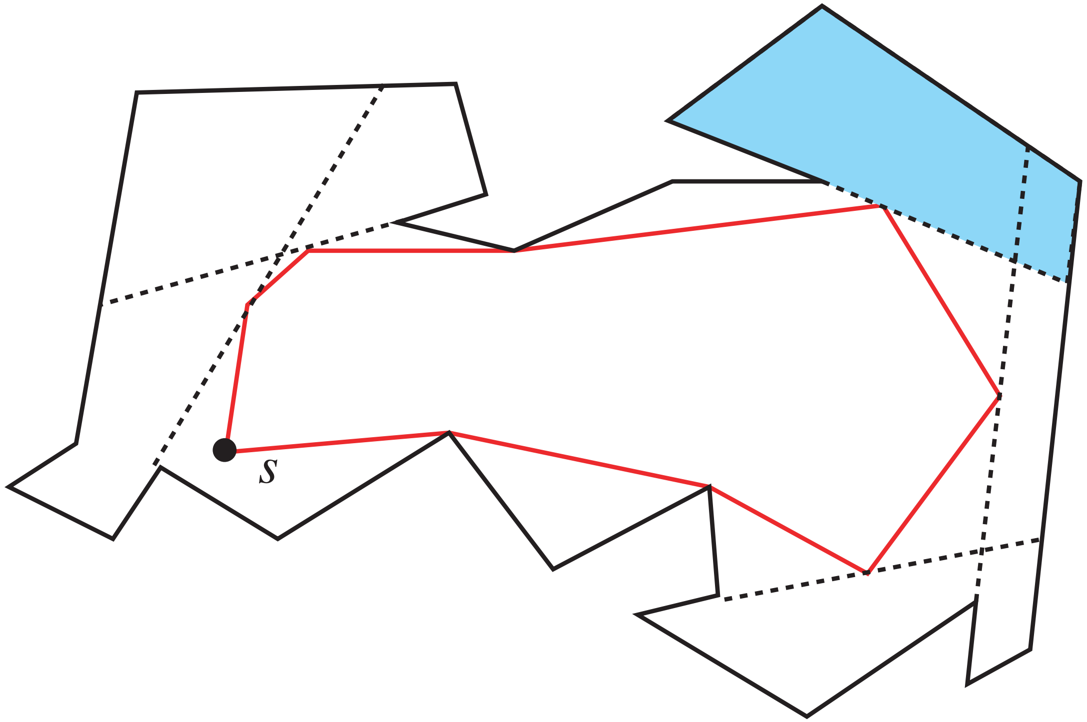


# Application: watchman route problem




TPP – polygons = essential pockets
– polygon order
– with fence

# Application: watchman route problem

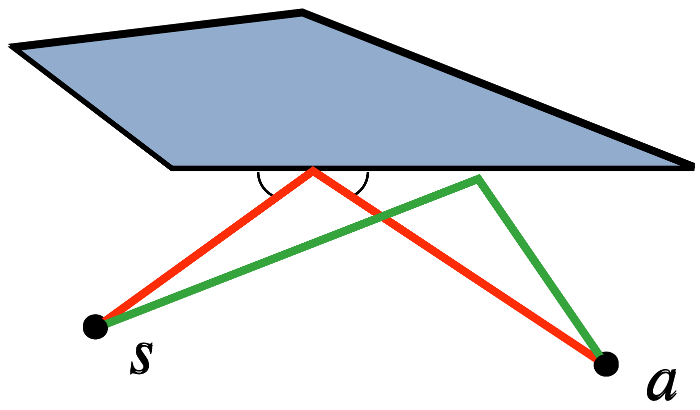



- problem from Chin and Ntafos `91
- ~~$O(n^4)$~~ `91
- $O(n^4)$ Tan, Hirata, Inagaki `99

$$O(n^3 \log n)$$

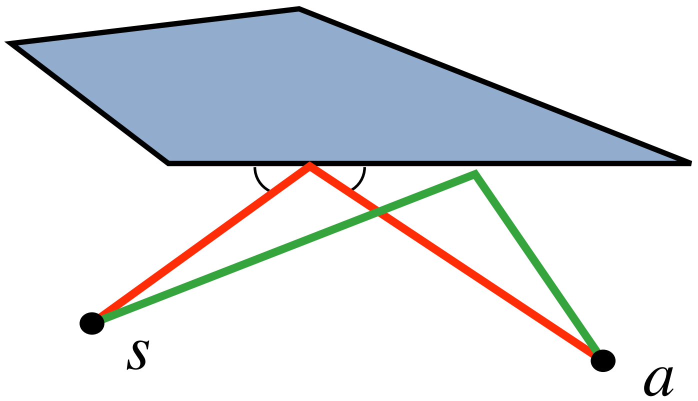
# Ideas of Algorithm: (1) Local Optimality

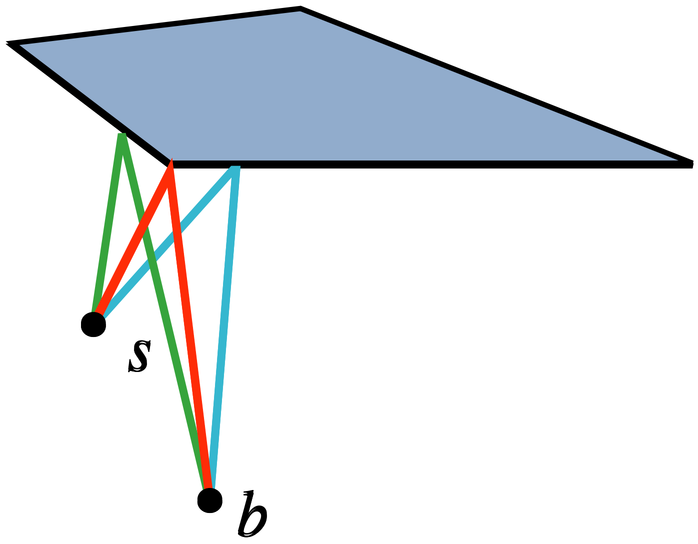
a path is locally optimal if moving any one bend of the path does not improve it



# Ideas of Algorithm: (1) Local Optimality

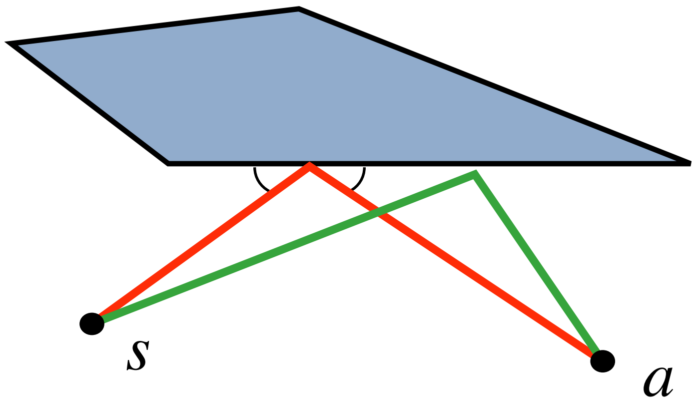
a path is locally optimal if moving any one bend of the path does not improve it

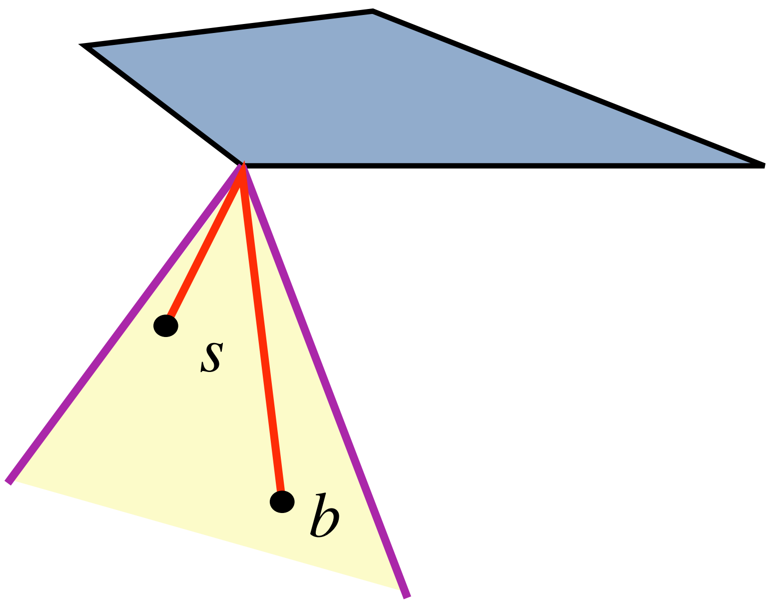


# Ideas of Algorithm:  (1) Local Optimality

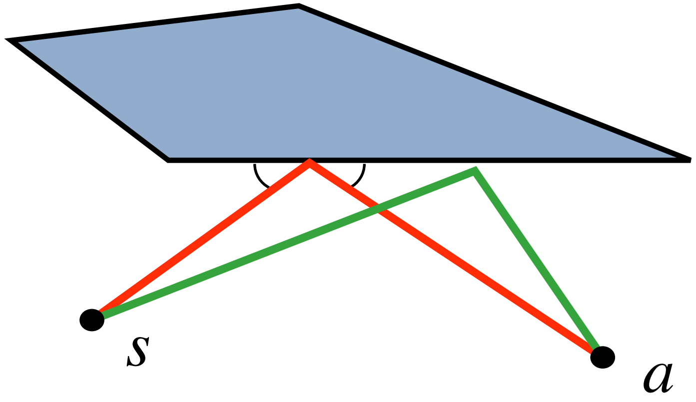
a path is locally optimal if moving any one bend of the path does not improve it

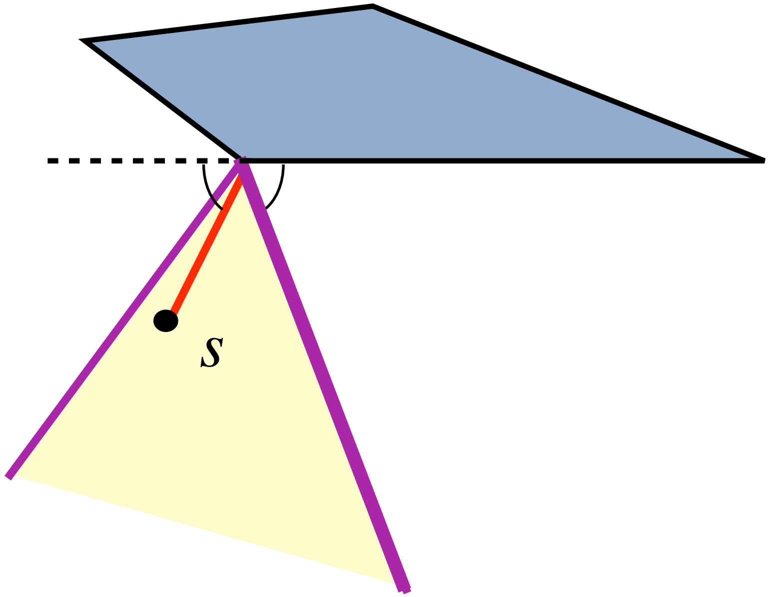


# Ideas of Algorithm:  (1) Local Optimality

a path is locally optimal if moving any one bend of the path does not improve it



# Ideas of Algorithm: (1) Local Optimality

a path is locally optimal if moving any one bend of the path does not improve it

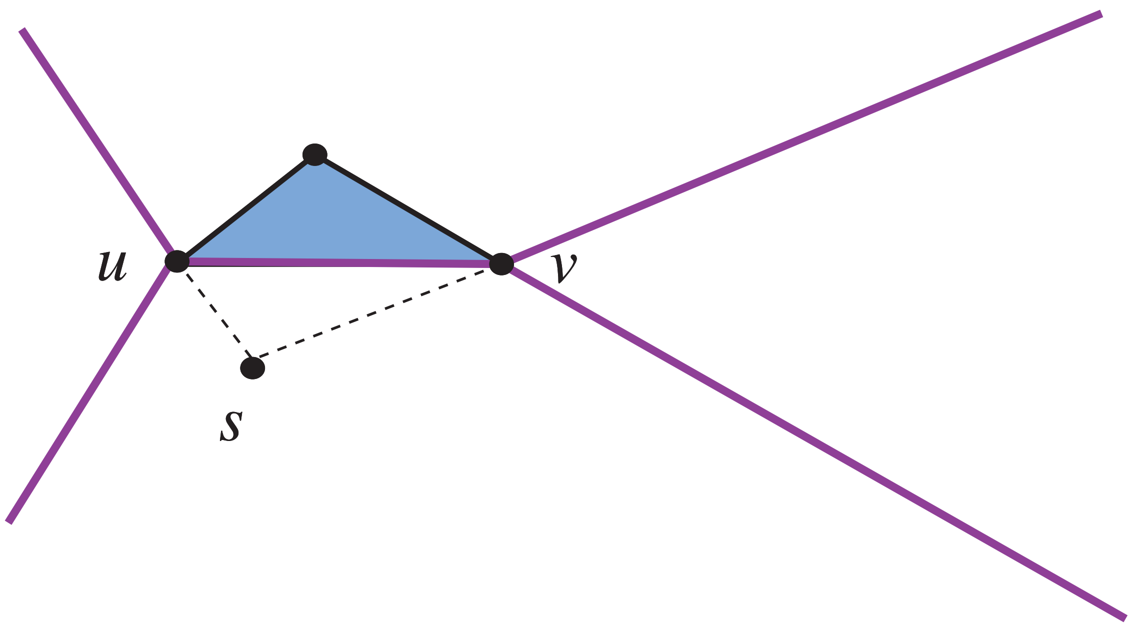
Theorem. Locally optimal = globally optimal for TPP.

Proof:

Theorem. A locally optimal path is unique.

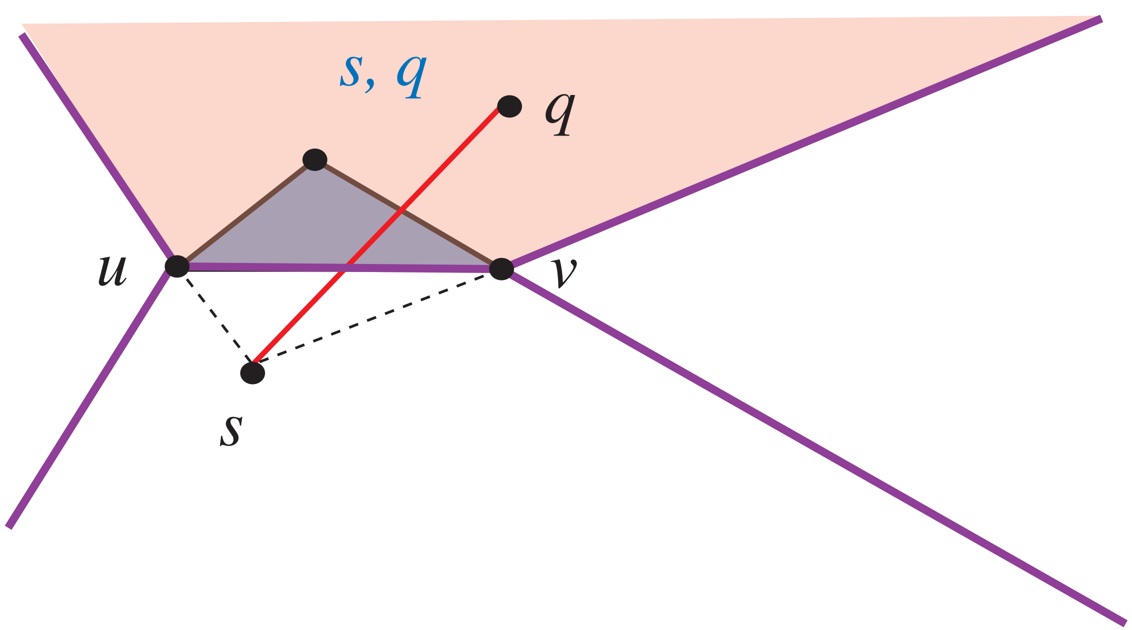
# Ideas of Algorithm: (2) Shortest Path Maps

shortest path map:
divide plane into regions by combinatorics of shortest path



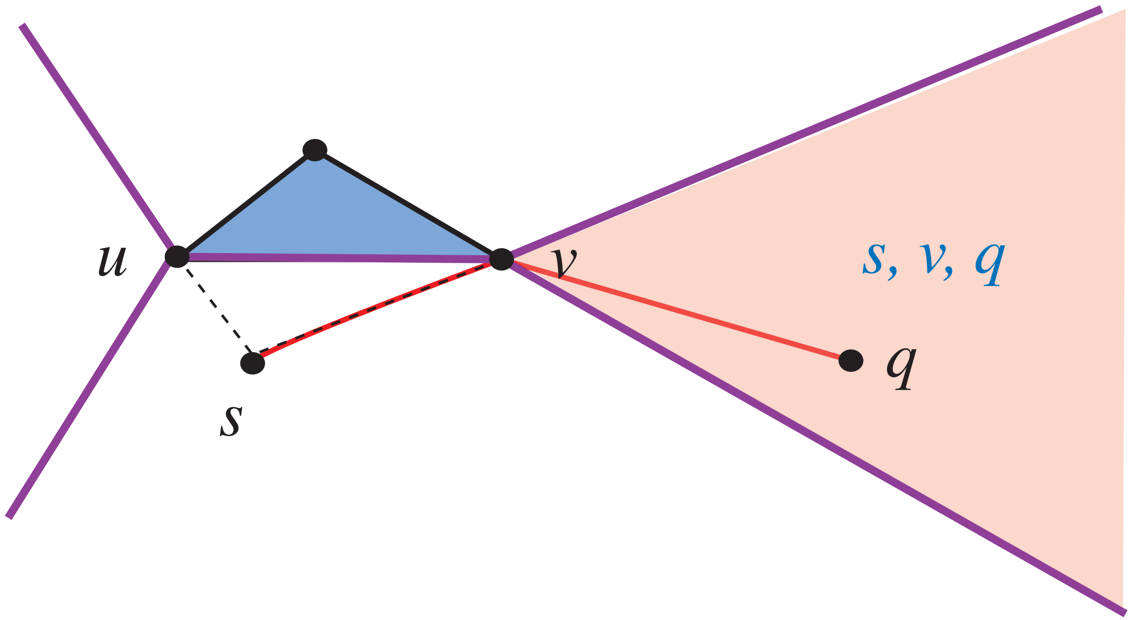
# Ideas of Algorithm: (2) Shortest Path Maps

shortest path map:
divide plane into regions by combinatorics of shortest path



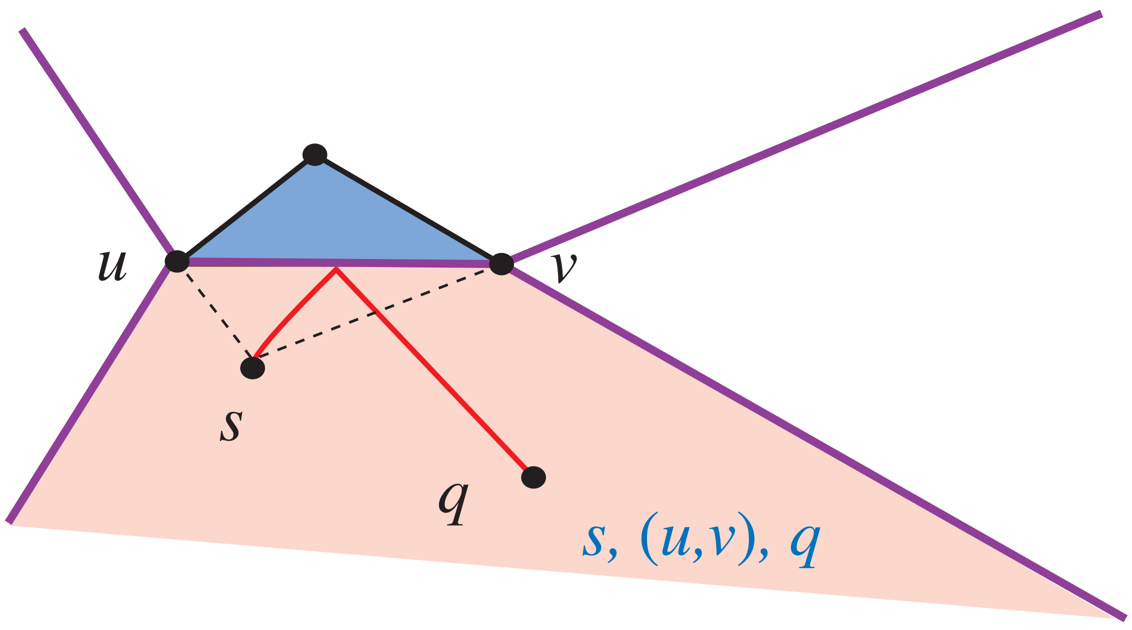
# Ideas of Algorithm: (2) Shortest Path Maps

shortest path map:
divide plane into regions by combinatorics of shortest path



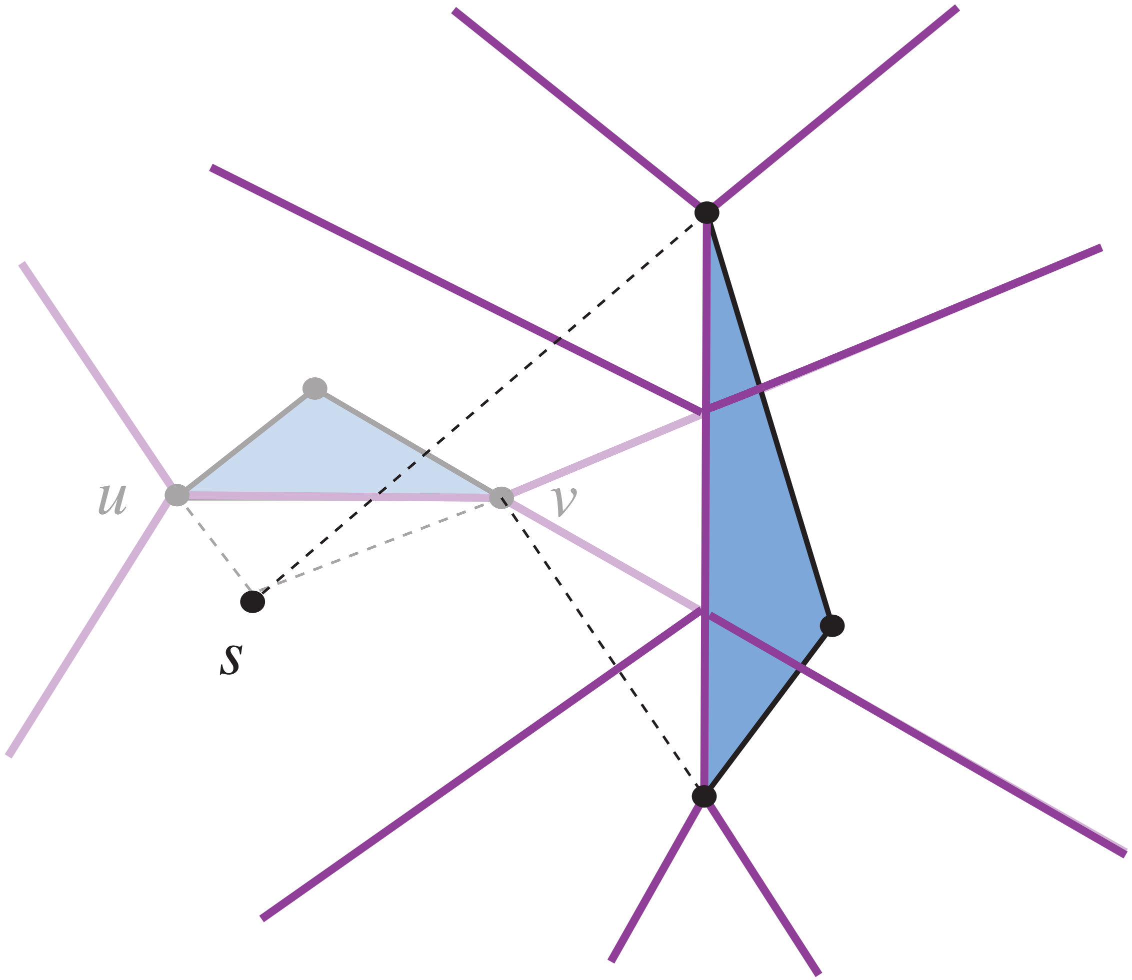
# Ideas of Algorithm: (2) Shortest Path Maps

shortest path map:
divide plane into regions by combinatorics of shortest path



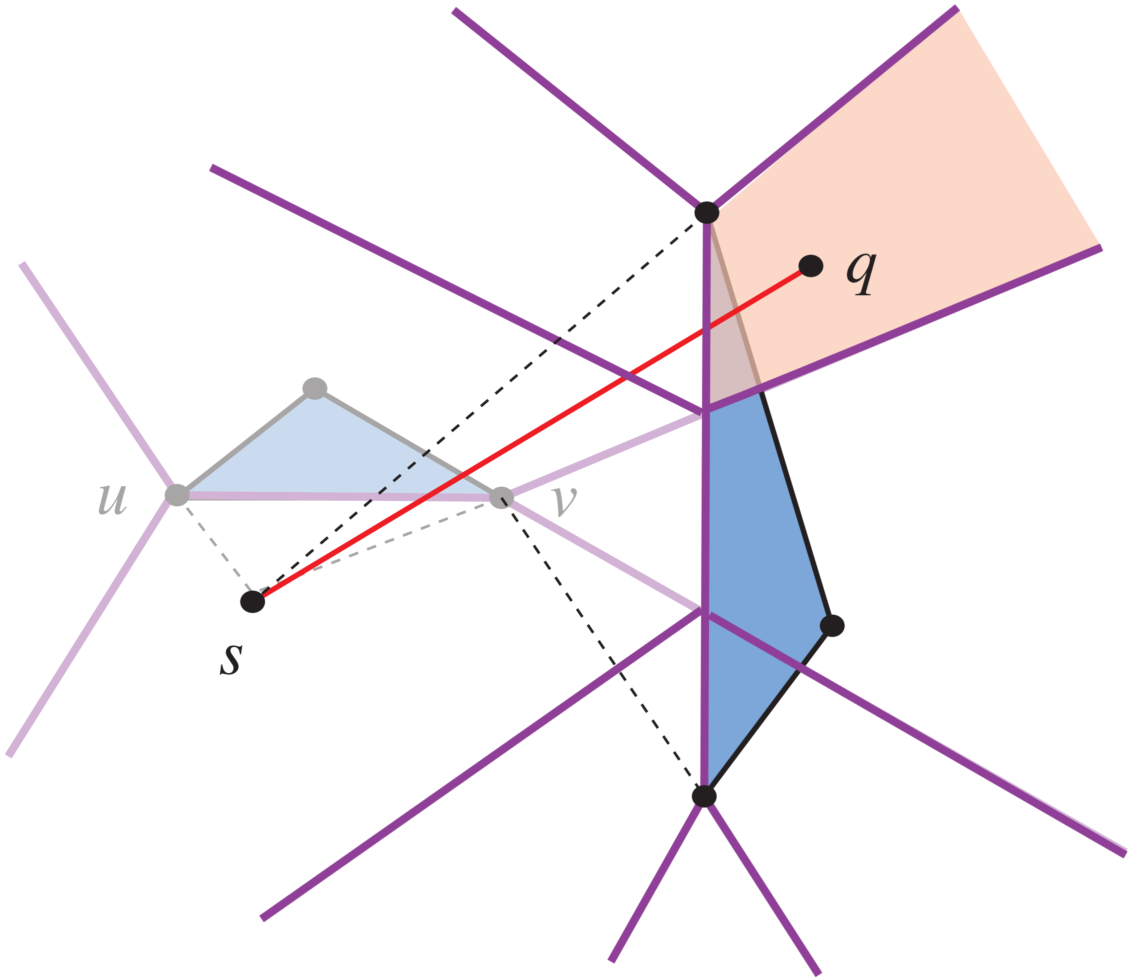
# Ideas of Algorithm: (2) Shortest Path Maps

shortest path map:
divide plane into regions by combinatorics of shortest path



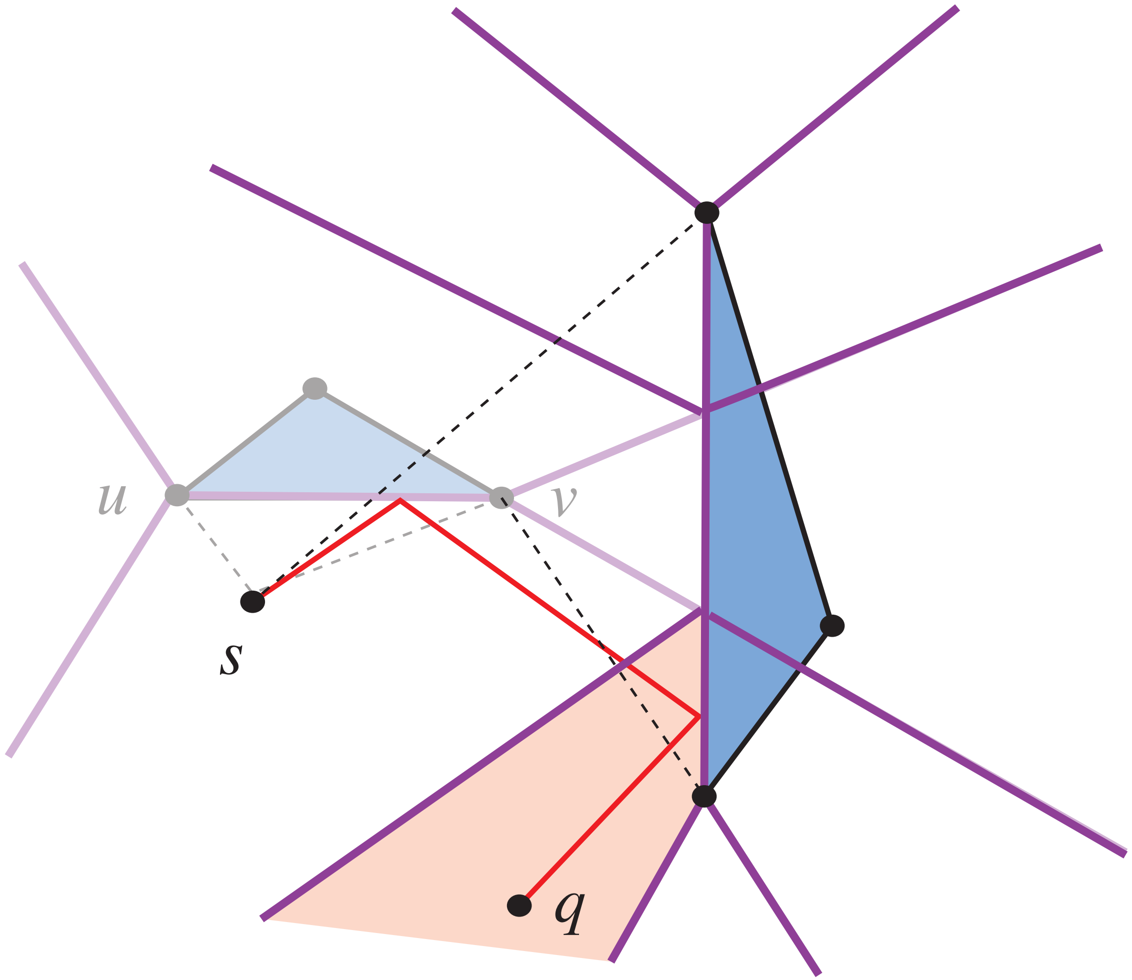
# Ideas of Algorithm: (2) Shortest Path Maps

shortest path map:
divide plane into regions by combinatorics of shortest path



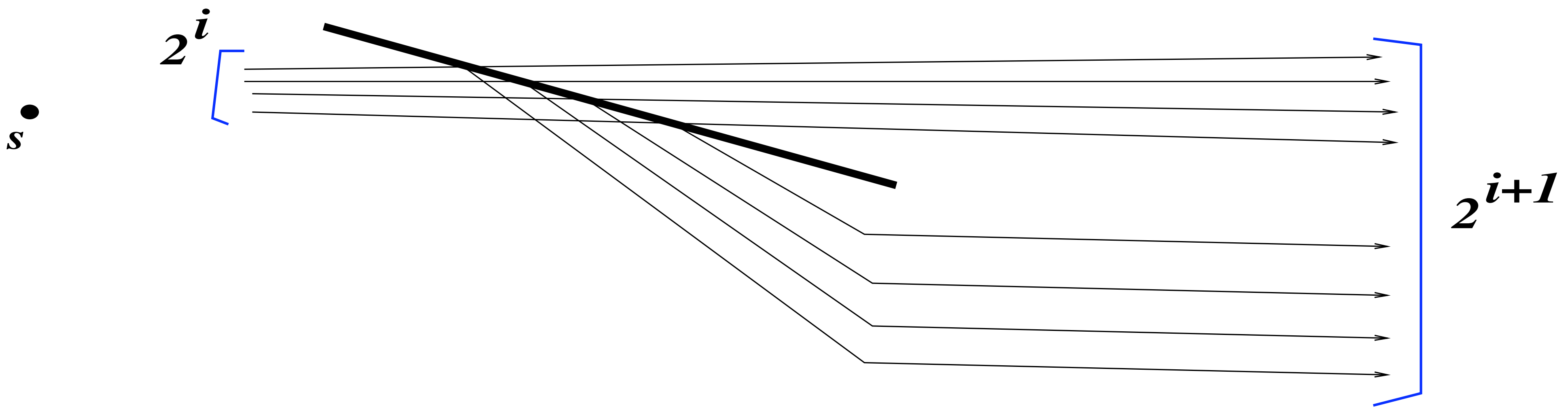
# Ideas of Algorithm: (2) Shortest Path Maps

shortest path map:
divide plane into regions by combinatorics of shortest path



# Shortest Path Maps

Theorem: The worst-case complexity of the shortest path map is $\Omega((n-k)\,2^k)$.



# Shortest Path Maps

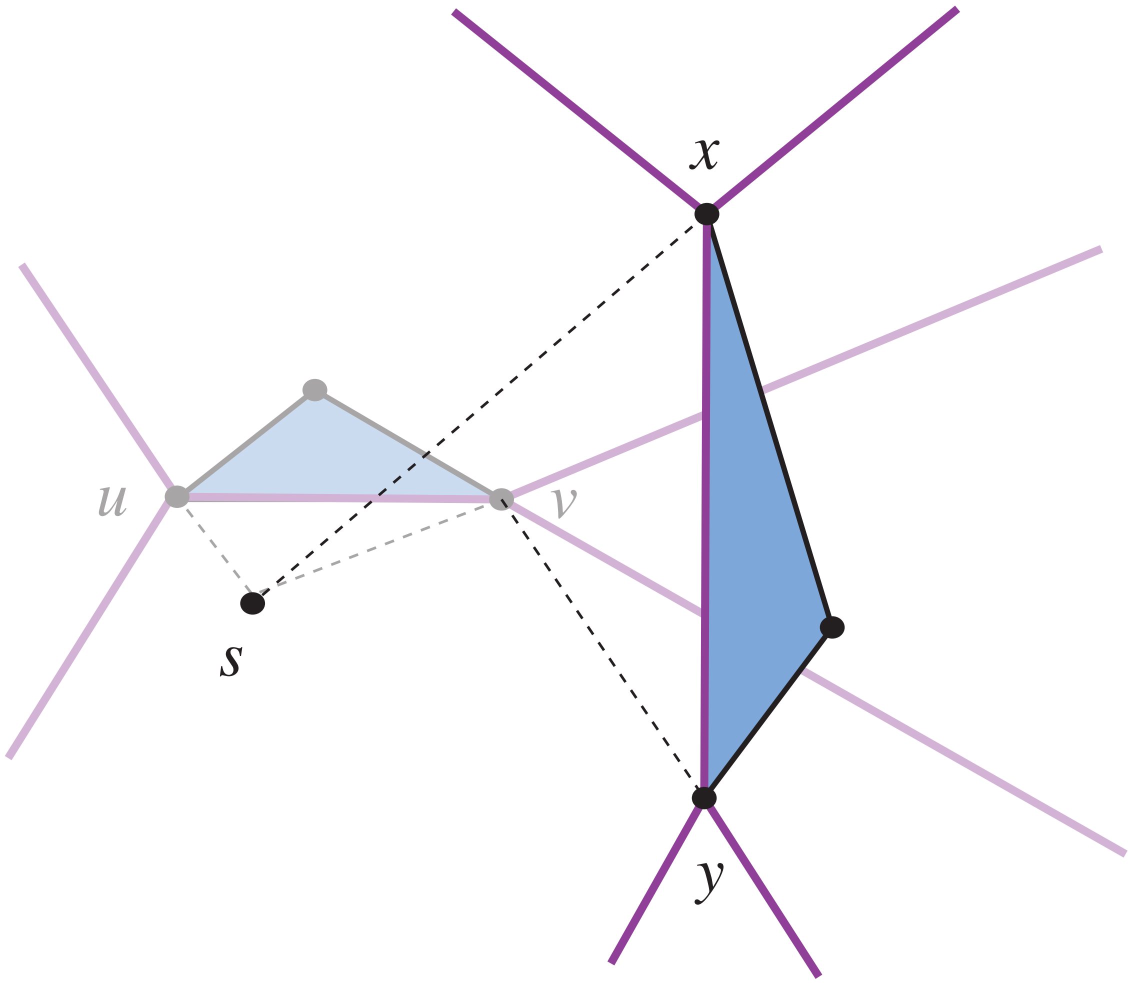
Theorem: The worst-case complexity of the shortest path map is $O((n-k)\,2^k)$.

We can compute it in output sensitive time, then do efficient queries.

(For the zoo-keeper problem, the shortest path map has polynomial size.)

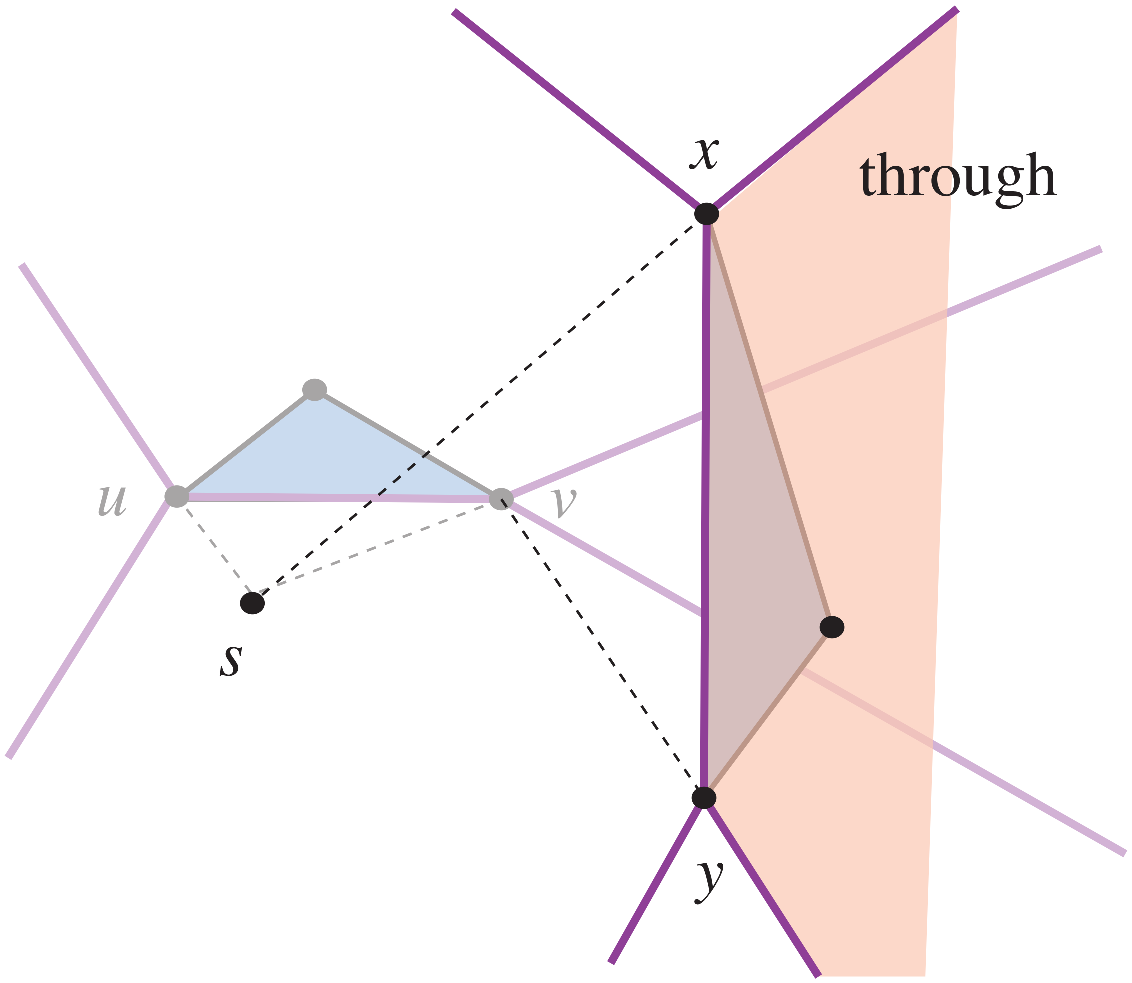
# Shortest Path Maps

last step shortest path map:
divide plane into regions by combinatorics of last step of shortest path



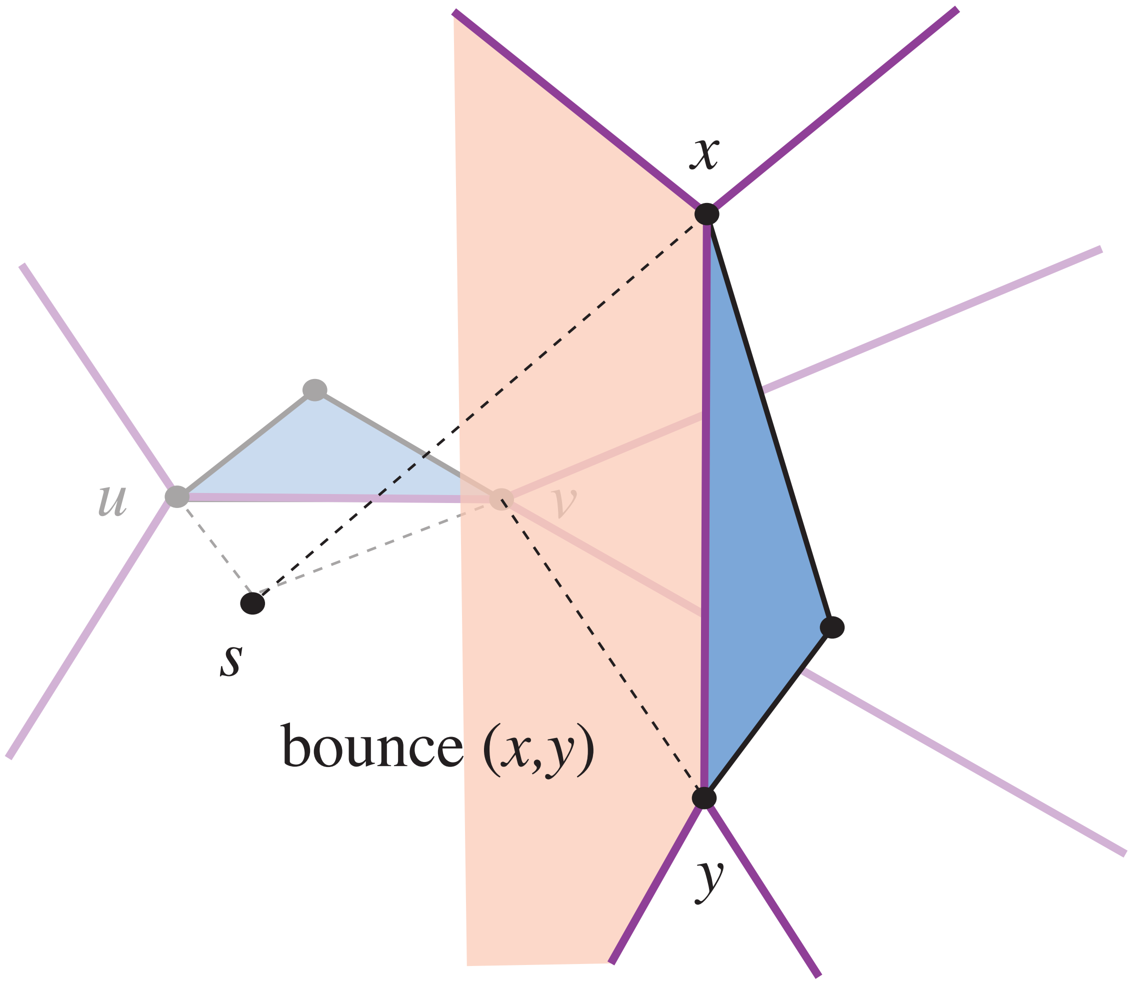
# Shortest Path Maps

last step shortest path map:
divide plane into regions by combinatorics of last step of shortest path



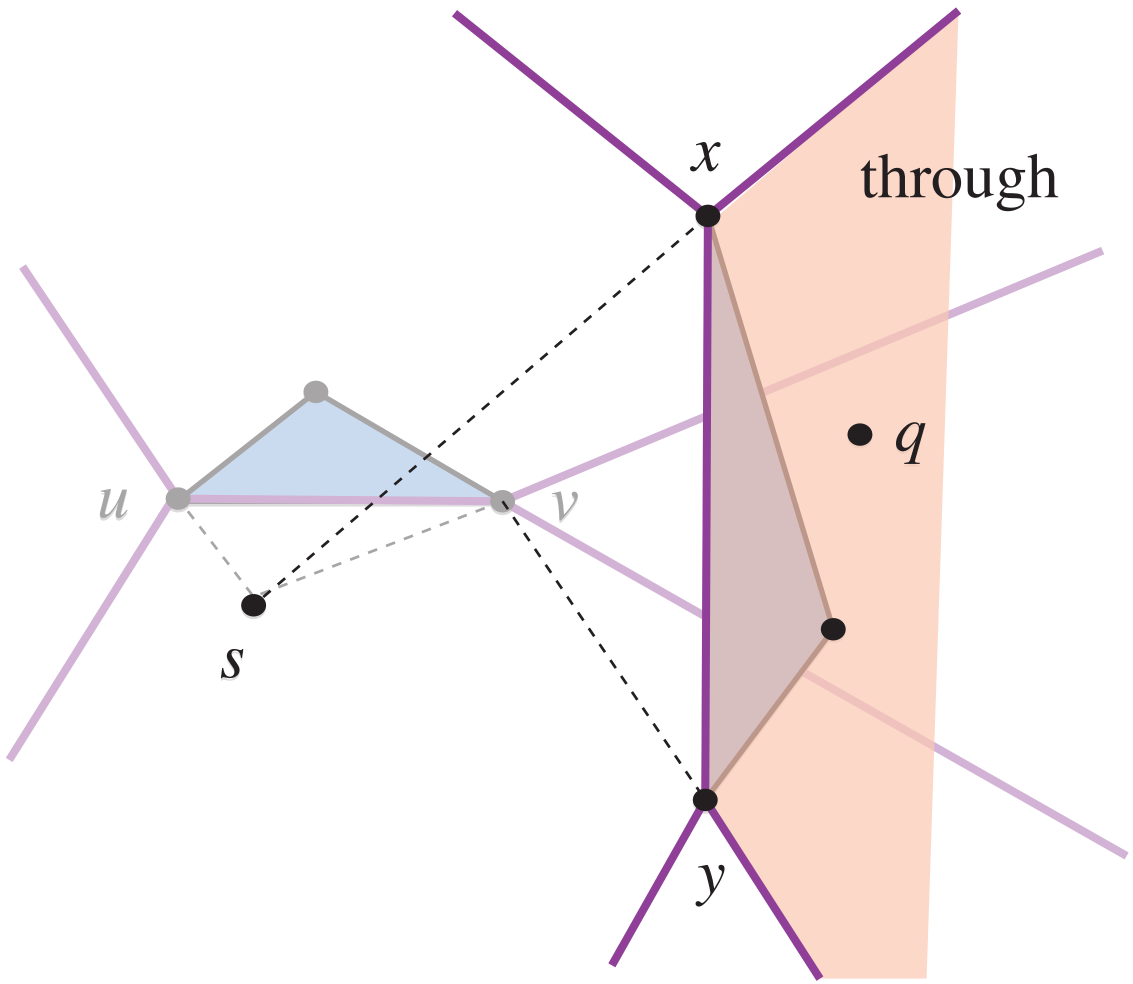
# Shortest Path Maps

last step shortest path map:
divide plane into regions by combinatorics of last step of shortest path



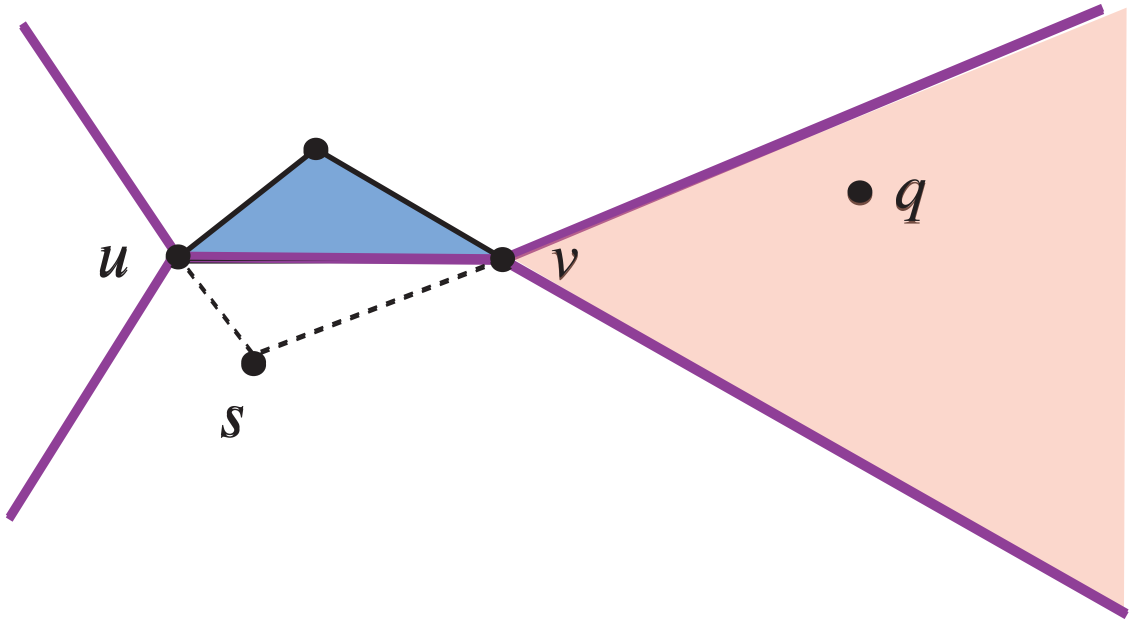
# Shortest Path Maps

answering queries using the last step shortest path map



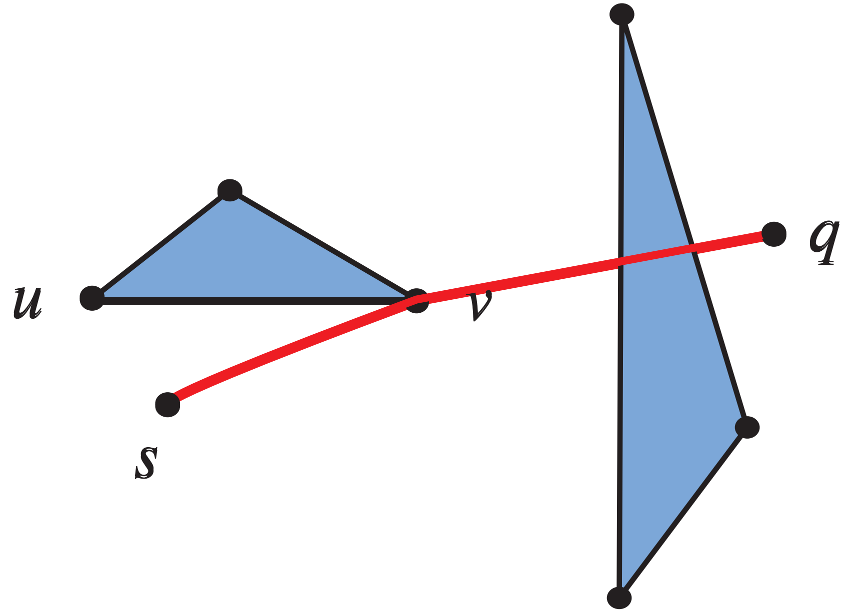
# Shortest Path Maps

answering queries using the last step shortest path map



# Shortest Path Maps

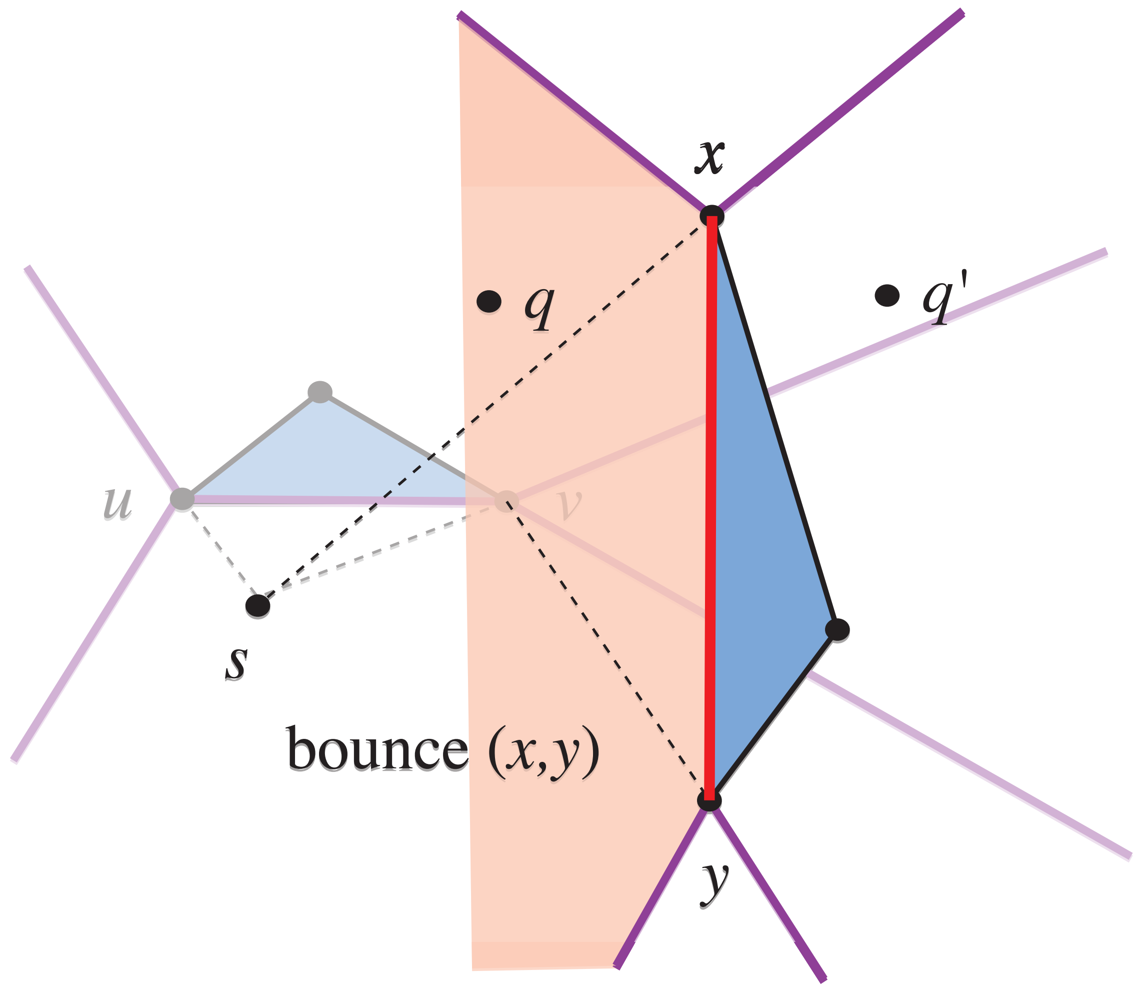
answering queries using the last step shortest path map



# Shortest Path Maps

answering queries using the last step shortest path map

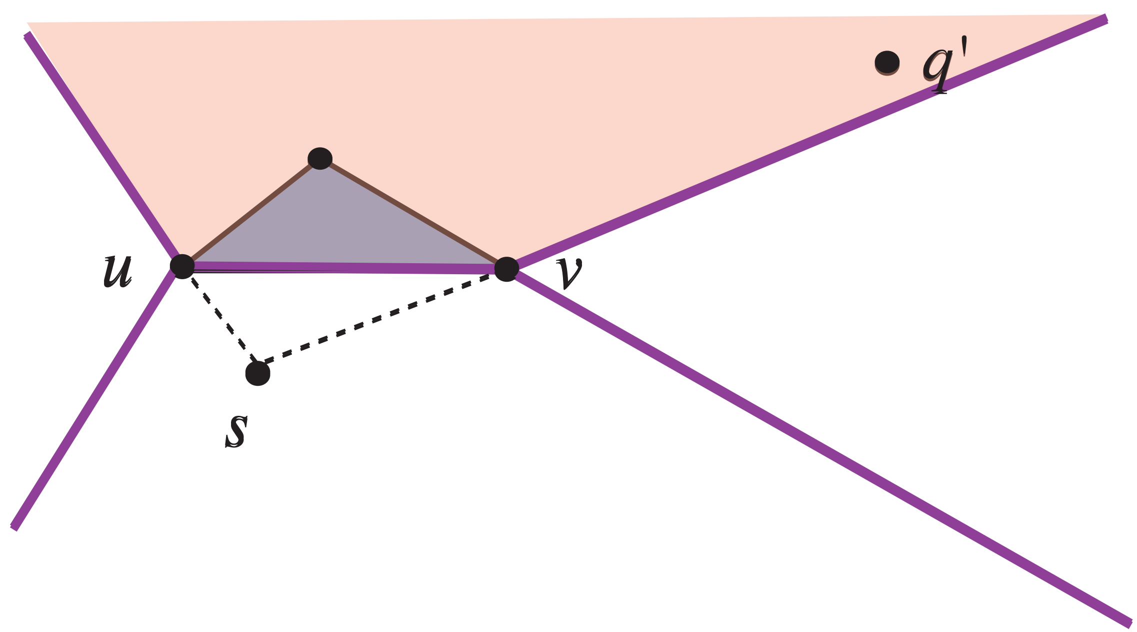
Example 2



# Shortest Path Maps

answering queries using the last step shortest path map

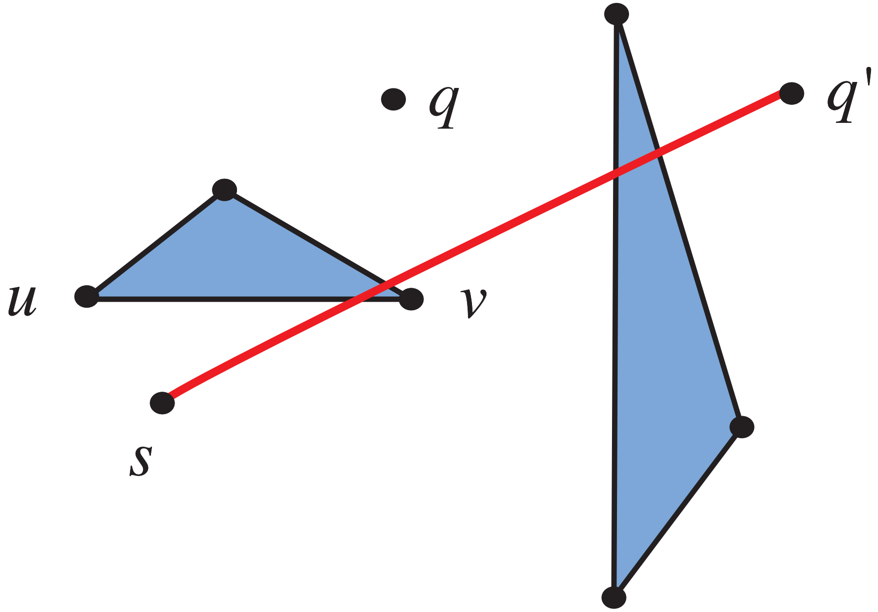
Example 2



# Shortest Path Maps

answering queries using the last step shortest path map

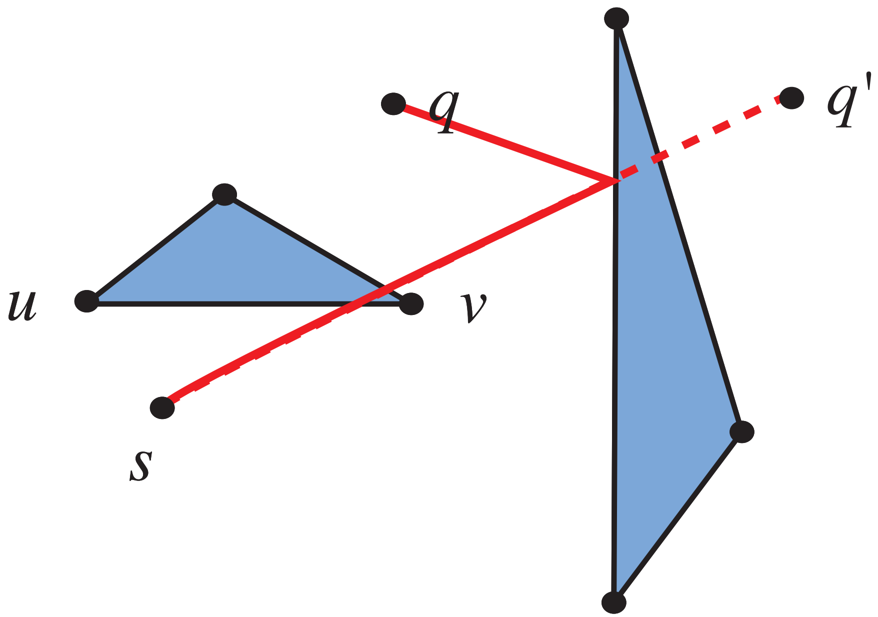
Example 2



# Shortest Path Maps

answering queries using the last step shortest path map

Example 2



# Shortest Path Maps

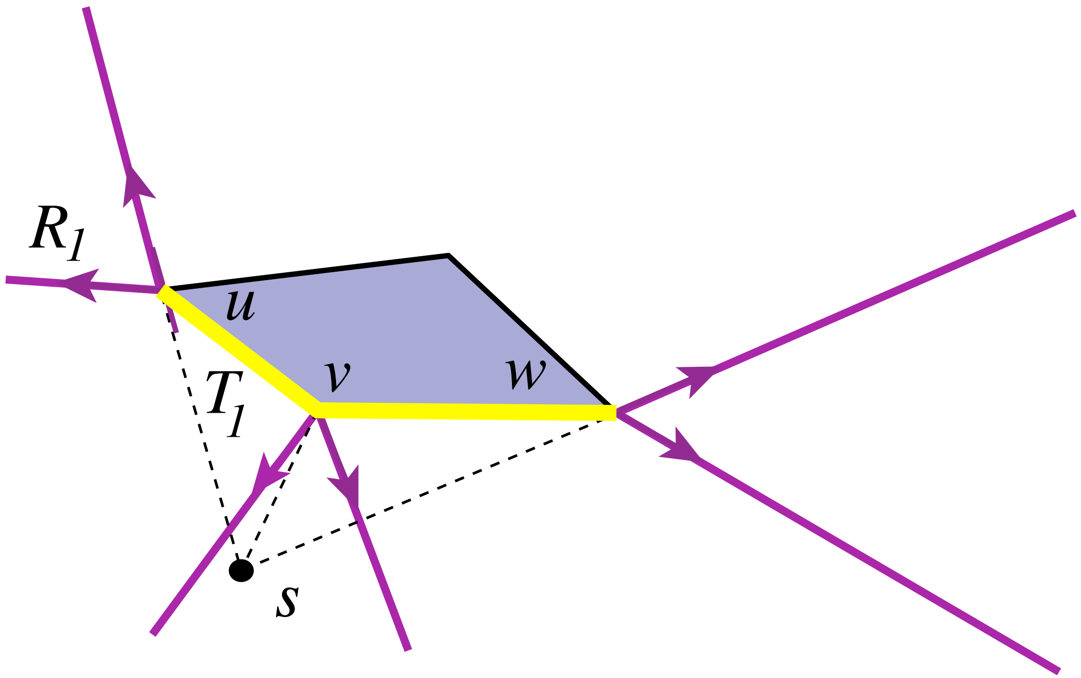
Lemma: Using last step shortest path maps, we can answer shortest path queries in $O(k \log n)$.

Ideas of Algorithm:

(1) Local Optimality

(2) Last Step Shortest Path Maps

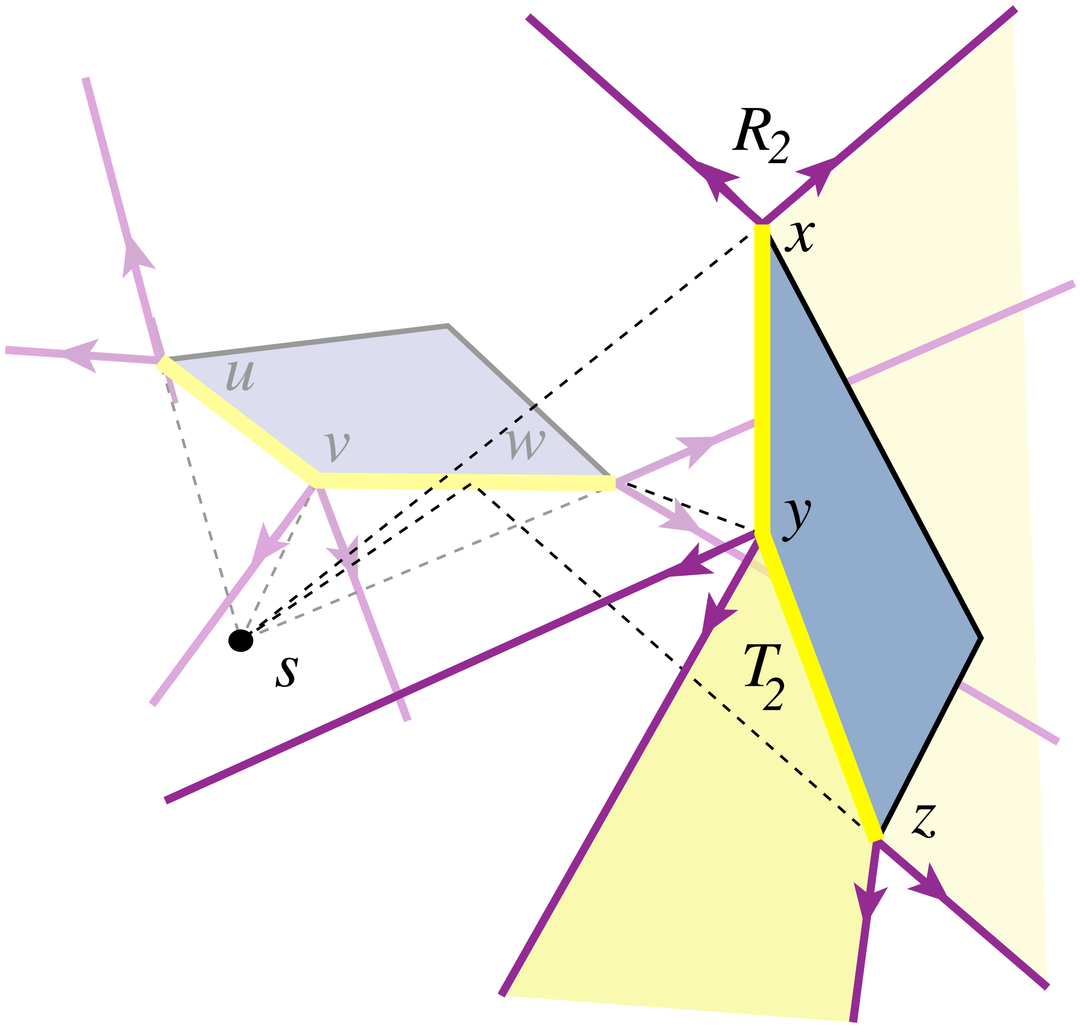
# Unconstrained TPP for disjoint convex polygons




$T_i$ — first contact set of shortest paths $s, P_1, \ldots, P_{i-1}$ with $P_i$

$R_i$ — shortest path rays leaving $P_i$

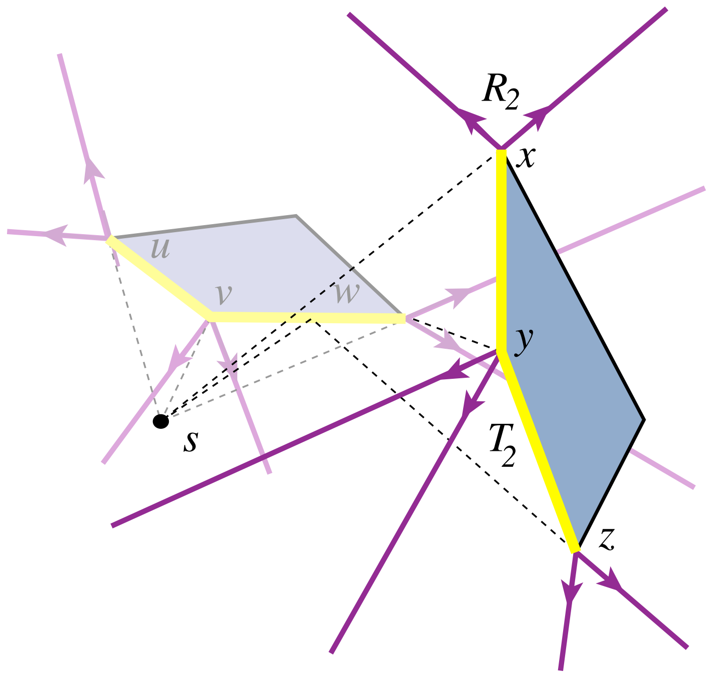
# Unconstrained TPP for disjoint convex polygons




$T_i$ — first contact set of shortest paths $s, P_1, \ldots, P_{i-1}$ with $P_i$

$R_i$ — shortest path rays leaving $P_i$

# Unconstrained TPP for disjoint convex polygons




## Structural Results

Lemma: $T_i$ is a chain on the boundary of $P_i$ .

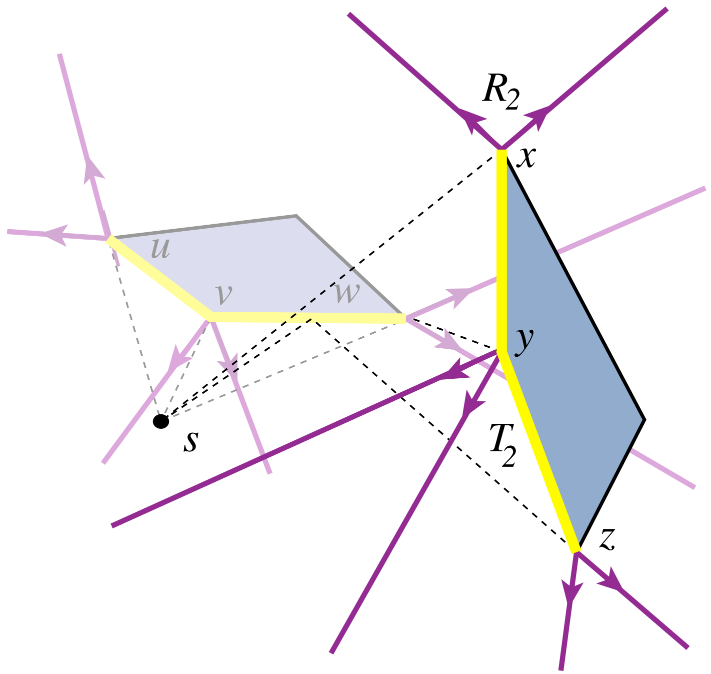
Lemma: $R_i$ is a *starburst* — i.e. there is a unique ray to every point of the plane.

Corollary: Locally shortest paths are unique.

$T_i$ — first contact set of shortest paths $s, P_1, \ldots, P_{i-1}$ with $P_i$

$R_i$ — shortest path rays leaving $P_i$

# Unconstrained TPP for disjoint convex polygons




## Algorithm

$T_0 = s$

for $i = 1 \ldots k$

compute $T_i$ and $R_i$

for each vertex $v$ of $P_i$

find $d_{i-1}(v)$

if it arrives at $v$ from outside $P_i$ then

$v$ is a vertex of $T_i$

use $d_{i-1}(v)$ to compute rays of $R_i$ at v

$T_i$ — first contact set with $P_i$

$R_i$ — rays leaving $P_i$

$d_i(v)$ = shortest path $s, P_1, \ldots, P_i, v$

# Unconstrained TPP for disjoint convex polygons

## Analysis

- shortest path query:

  $O(k \log n)$

- algorithm total:

  $O(n\, k \log n)$

## Algorithm

$T_0 = s$

for $i = 1 \,.\,.\, k$

  compute $T_i$ and $R_i$

  for each vertex $v$ of $P_i$

    find $d_{i-1}(v)$

    if it arrives at $v$ from outside $P_i$ then

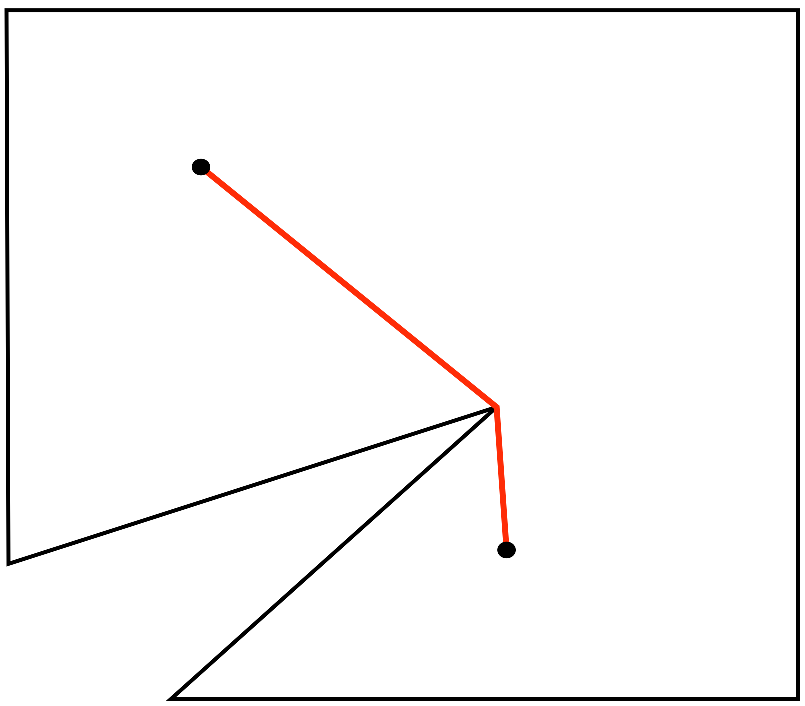
      $v$ is a vertex of $T$

      use $d_{i-1}(v)$ to compute rays of $R_i$ at v

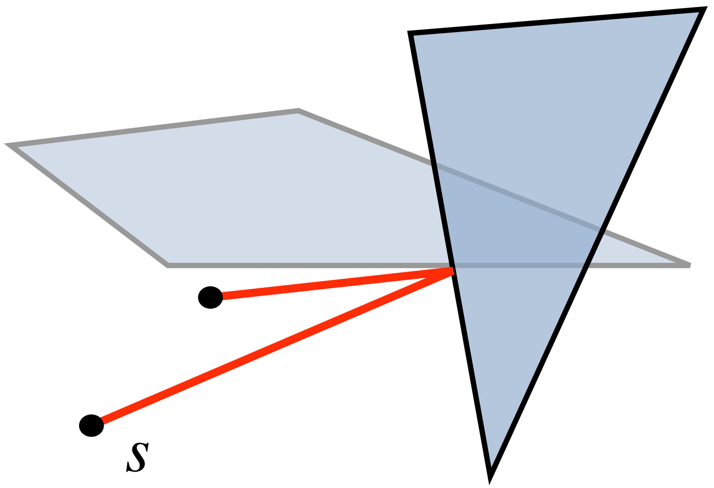
$d_i(v)$ = shortest path $s, P_1, \ldots, P_i, v$

# General TPP : local optimality

fences

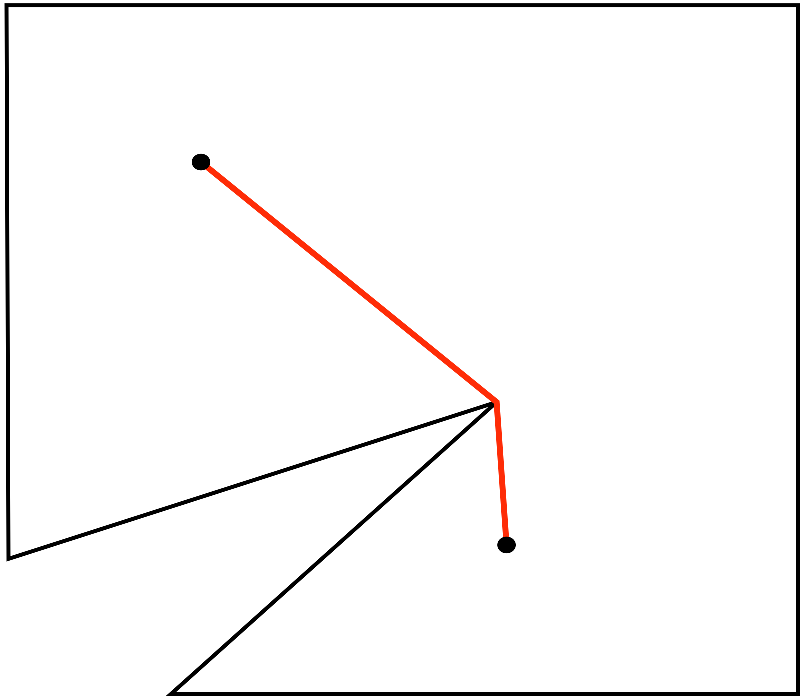


intersecting polygons

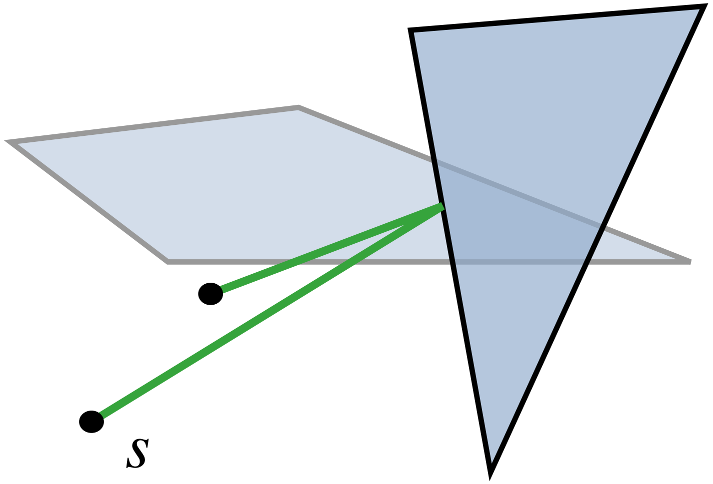


# General TPP: local optimality

fences

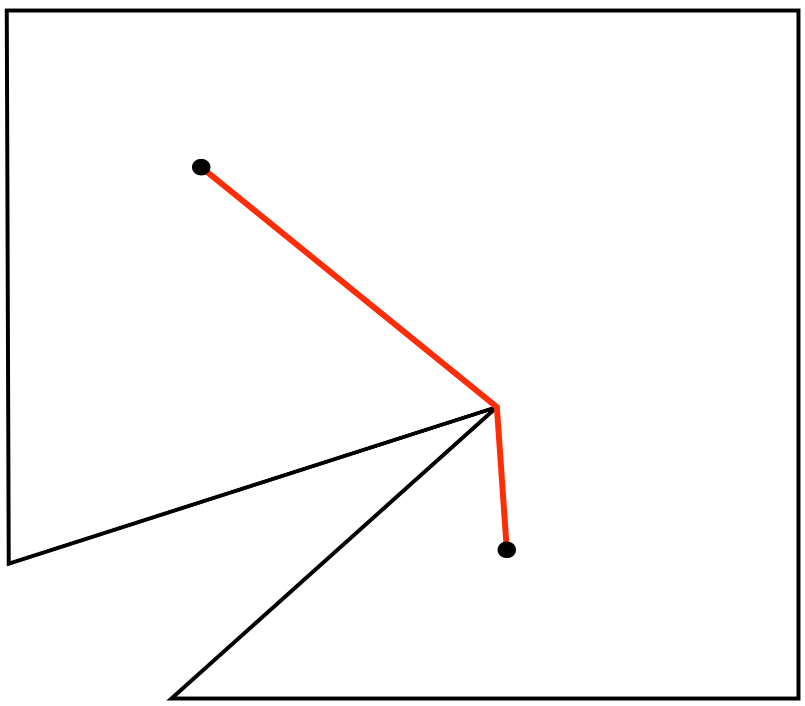


intersecting polygons

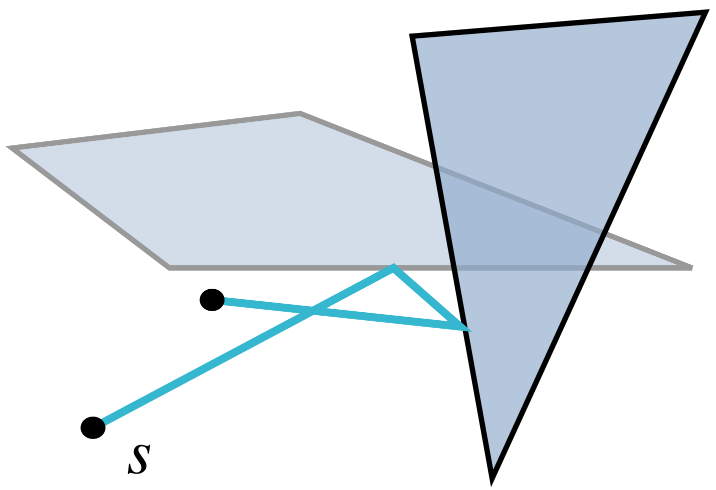


# General TPP : local optimality

fences

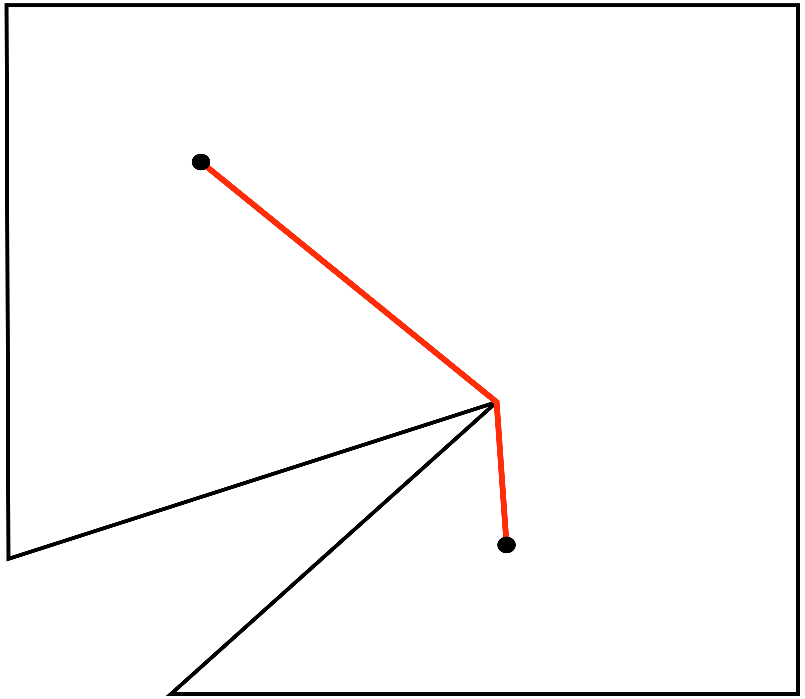


intersecting polygons

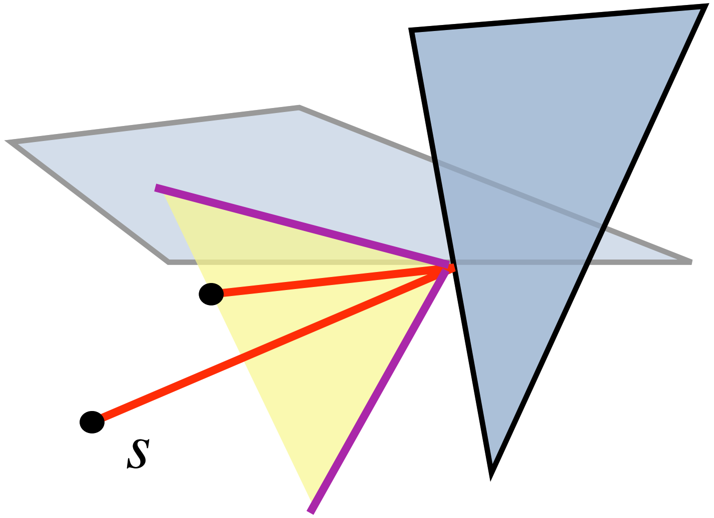


# General TPP : local optimality

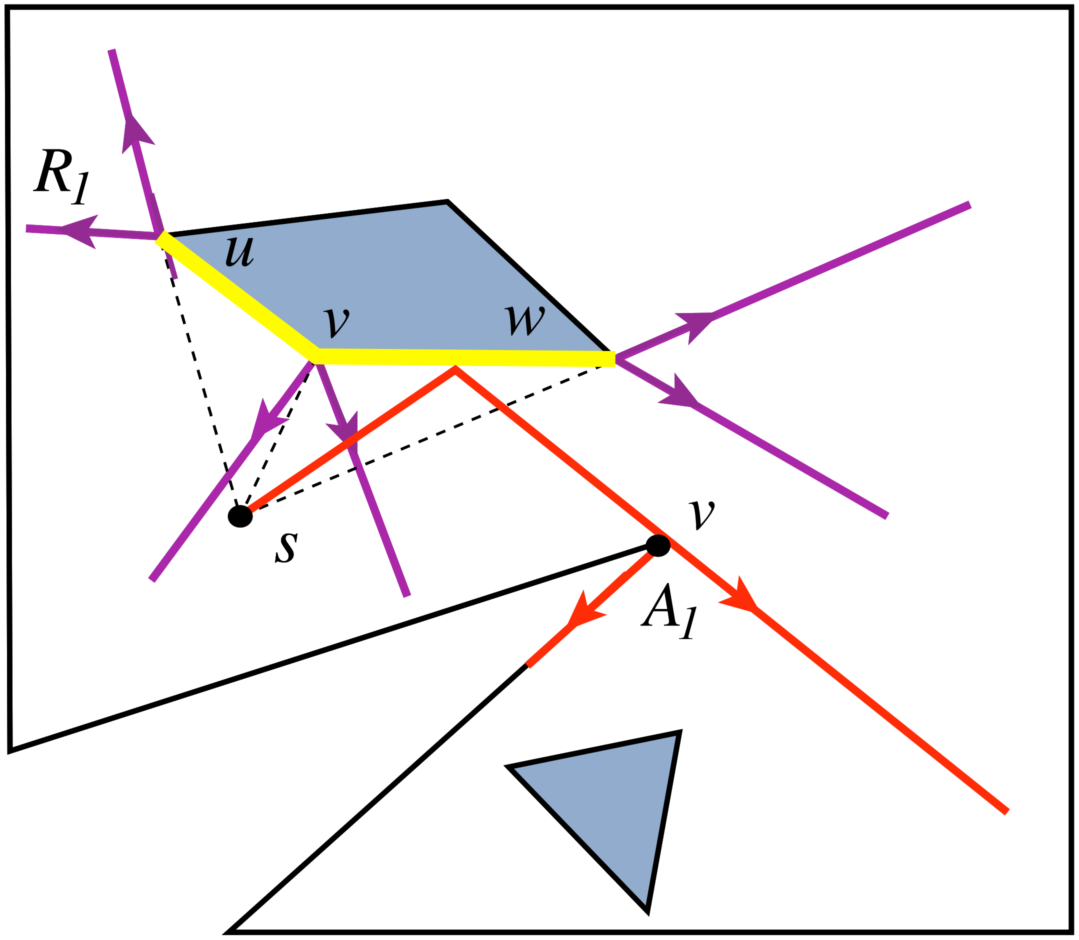
fences



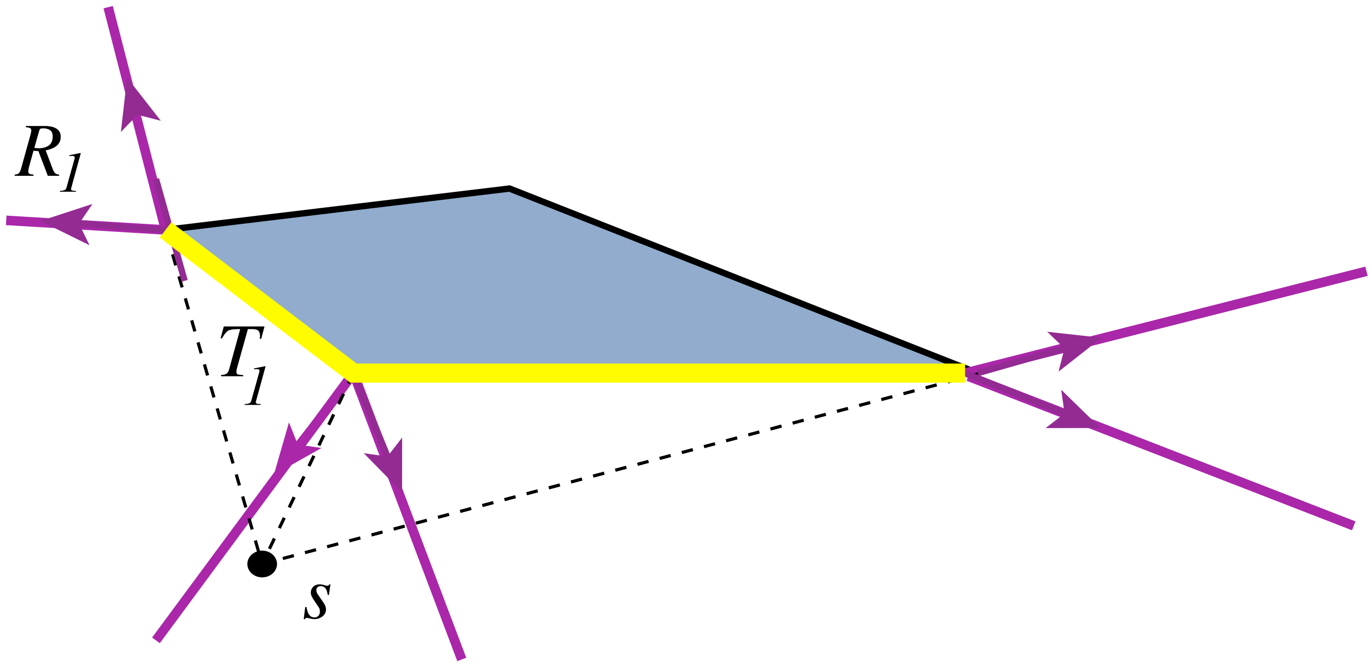
intersecting polygons



# General TPP : fences



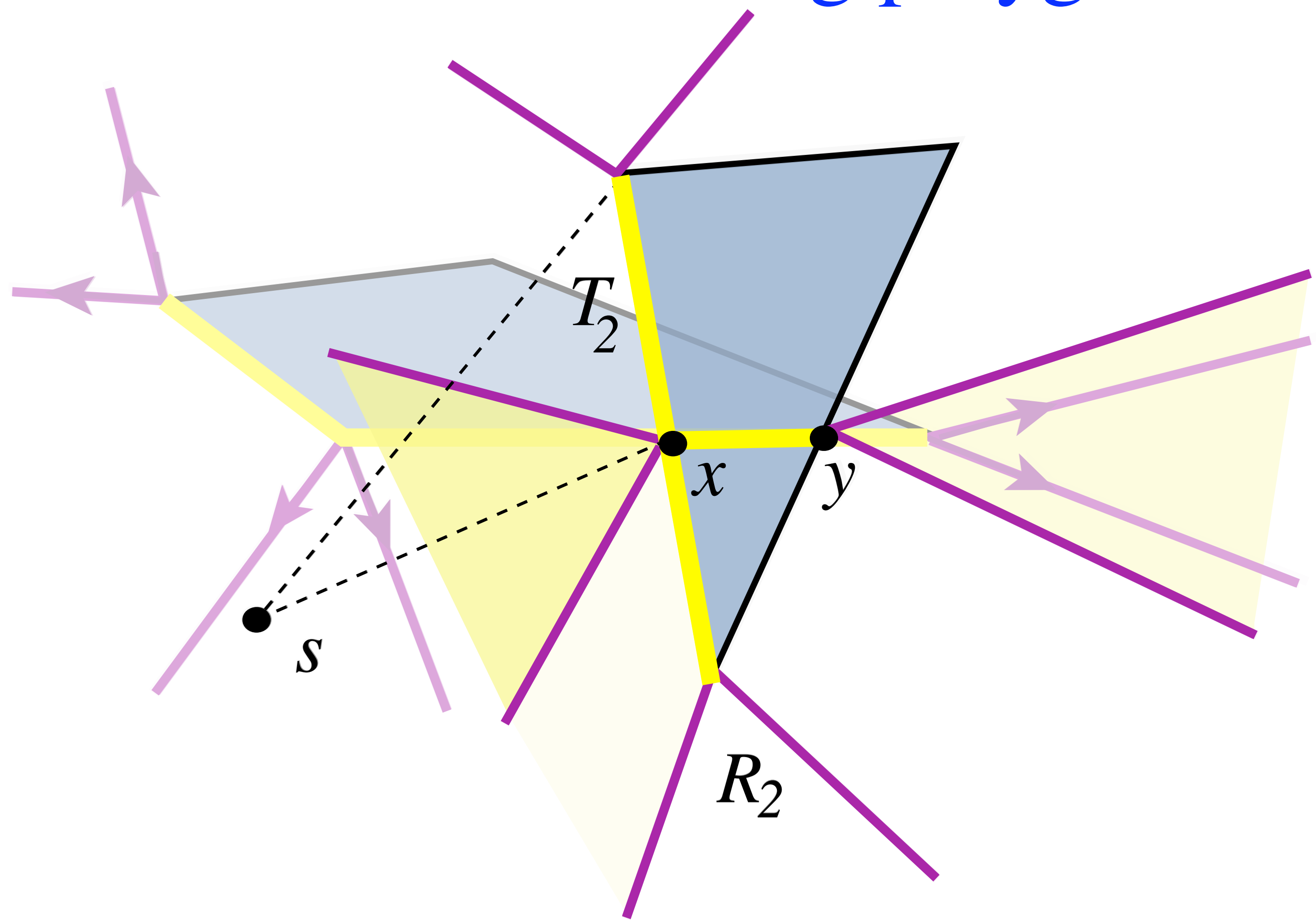
# General TPP : intersecting polygons




$T_i$ — first contact set of shortest paths $s, P_1, \ldots, P_{i-1}$ with $P_i$

$R_i$ — shortest path rays leaving $P_i$

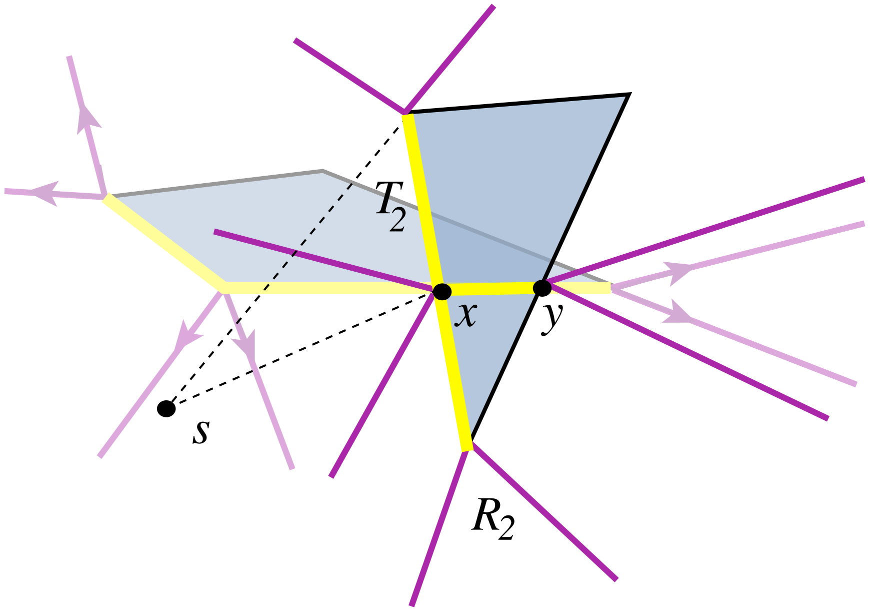
# General TPP: intersecting polygons




$T_i$ — first contact set of shortest paths $s, P_1, \ldots, P_{i-1}$ with $P_i$

$R_i$ — shortest path rays leaving $P_i$

# General TPP




## Structural Results

Lemma: $T_i$ is a tree.

Lemma: $R_i$ is a *starburst* — i.e. there is a unique ray to every point of the plane.

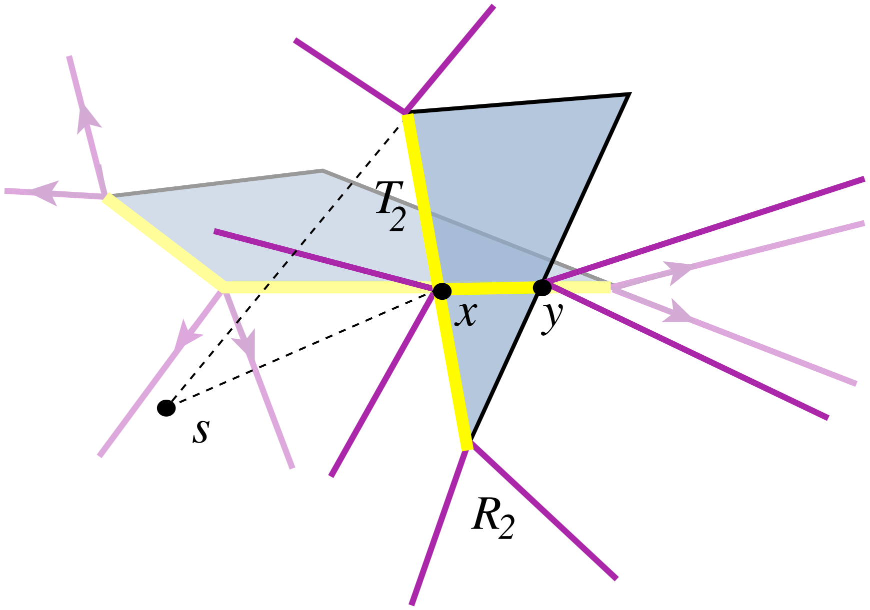
Cor. Locally shortest paths are unique.

$T_i$ — first contact set of shortest paths $s, P_1, \ldots, P_{i-1}$ with $P_i$

$R_i$ — shortest path rays leaving $P_i$

$A_i$ — rays arriving at $P_{i+1}$ after travelling through fence $F_i$

# General TPP




## Algorithm

$T_0 = s$, $R_0$ = all rays from $s$

$A_0$ = rays inside $F_0$

for $i = 1 \ldots k$

compute $T_i$, $R_i$, and $A_i$

$O(nk^2 \log n)$

$T_i$ — first contact set with $P_i$

$R_i$ — rays leaving $P_i$

$A_i$ — rays arriving at $P_{i+1}$ after travelling through fence $F_i$

$d_i(v)$ = shortest path $s, P_1, \ldots, P_i, v$

# Extensions

reminder of TPP:

Given: a sequence of possibly intersecting, convex [facade] polygons, a start point $s$ and a target point $t$

Find: a shortest path that starts at $s$, visits the polygons in sequence, respecting the fences, and ends at $t$

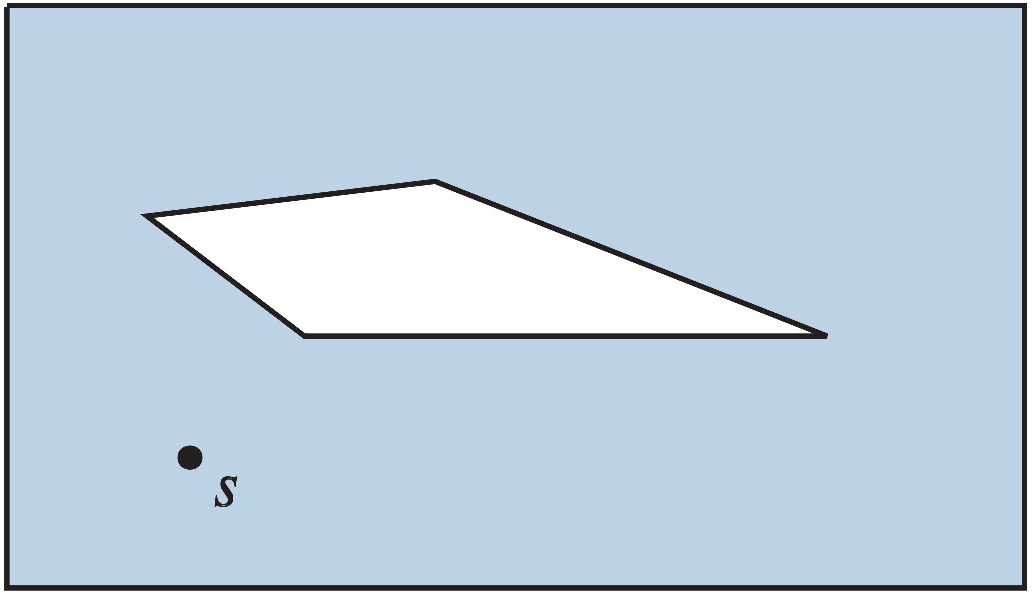
non-convex polygons

Theorem. TPP is NP-hard for non-convex polygons (even without fences).

Proof. From 3-SAT, based on a careful adaptation of the Canny-Reif proof.

# Extensions

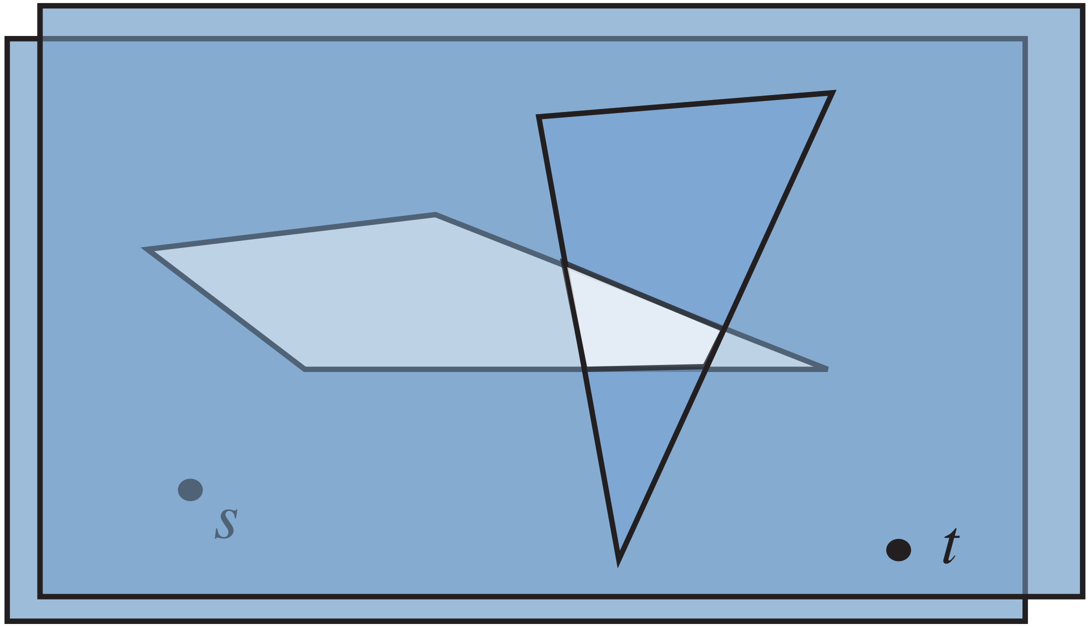
TPP as a 3-D shortest path problem.




Thus there is a fully polynomial time approximation scheme
(even for non-convex polygons).

# Extensions

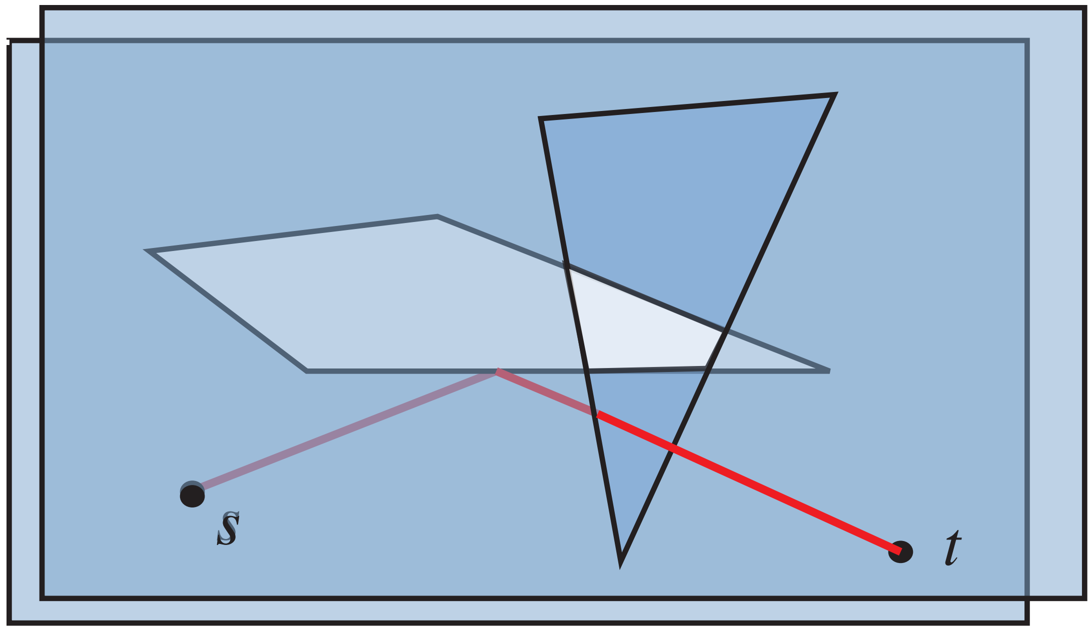
TPP as a 3-D shortest path problem.




Thus there is a fully polynomial time approximation scheme
(even for non-convex polygons).

# Extensions

TPP as a 3-D shortest path problem.




Thus there is a fully polynomial time approximation scheme
(even for non-convex polygons).

# Extensions

**Open.** What is the complexity of TPP for disjoint non-convex polygons.

# The End