

# Walking Your Dog in the Woods in Polynomial Time

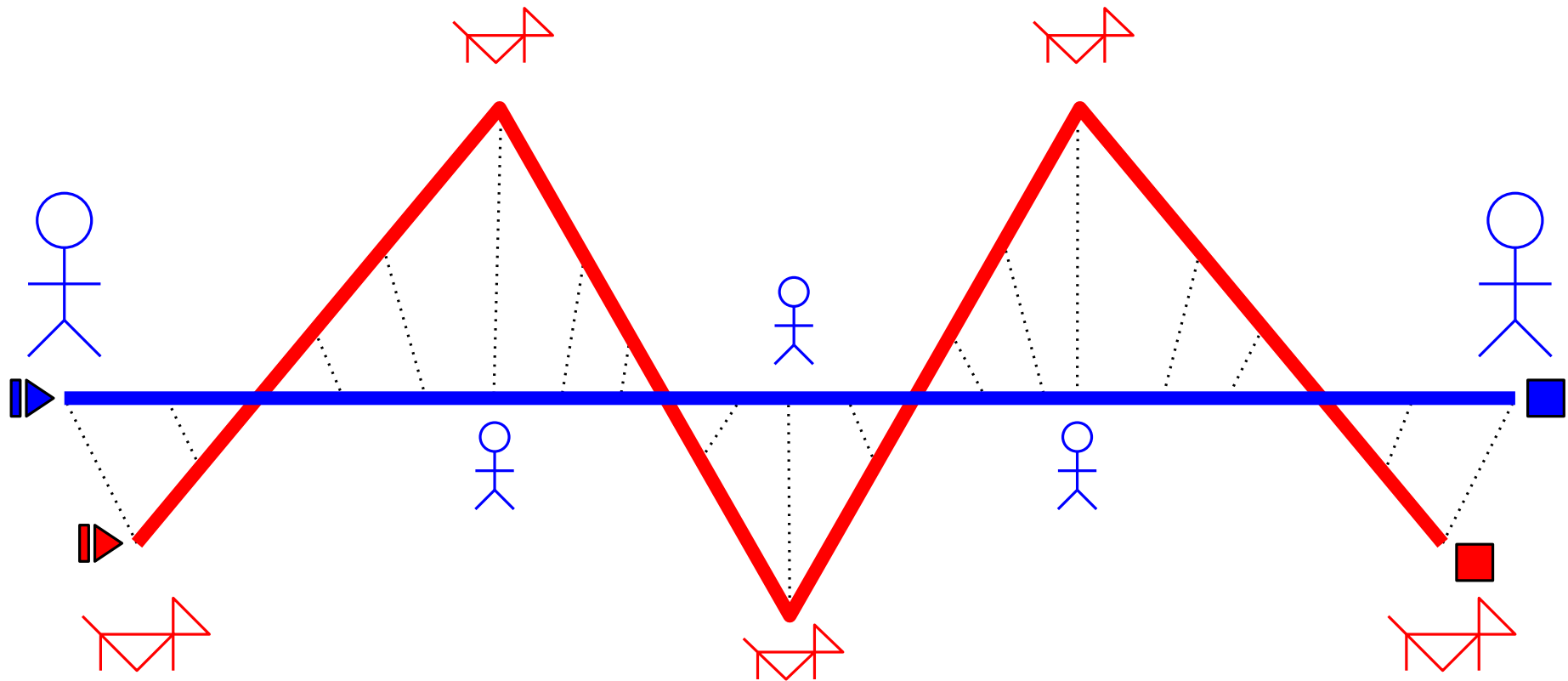


**Shripad Thite**

shripad@Caltech.edu

Joint work with Erin Wolf Chambers, Éric Colin de Verdière,  
Jeff Erickson, Sylvain Lazard, Francis Lazarus  
To appear at SoCG'08, invited to CGTA

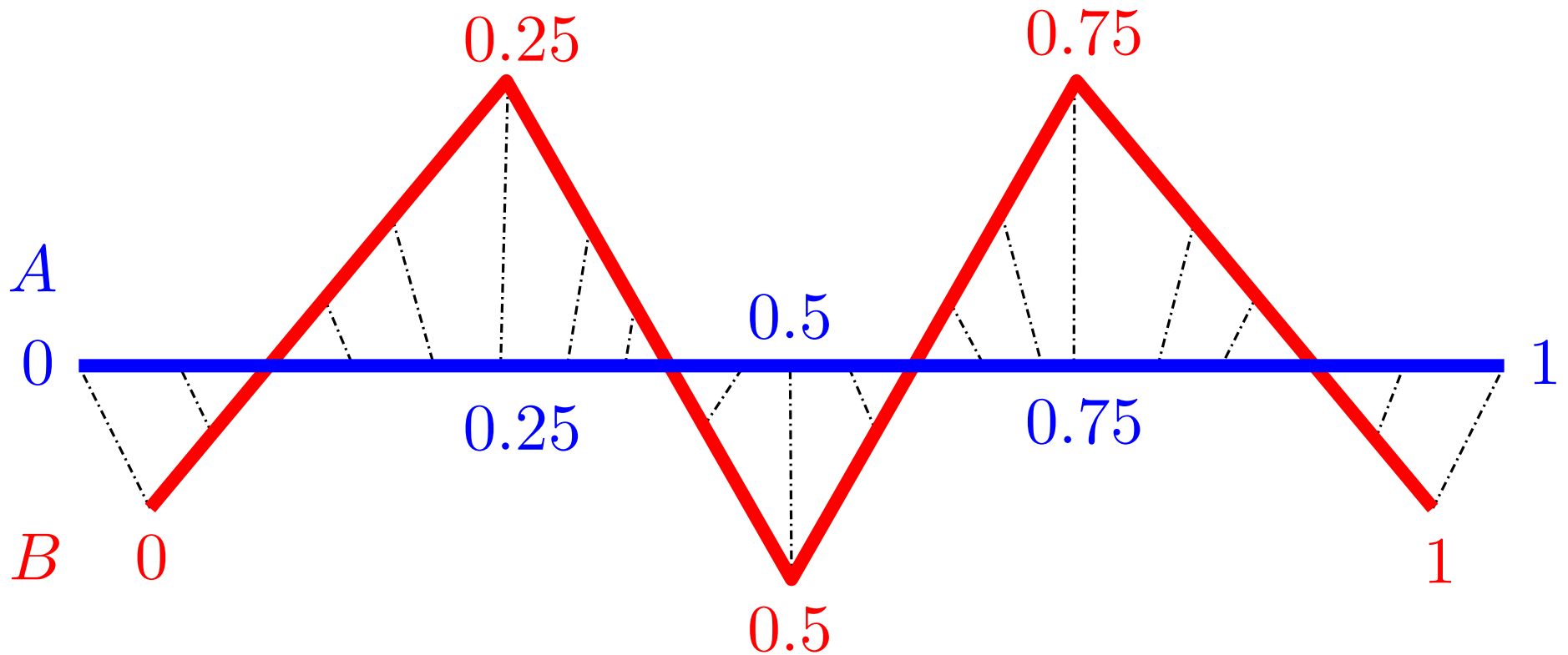
# Walking your dog



You and **your dog** walk along prescribed curves from beginning to end, continuously without backtracking, joined by a tight **leash**

Cost of a walk = maximum length of leash during the walk

# Fréchet distance = minimum cost of a walk



Curves  $A, B : [0, 1] \rightarrow \mathbb{E}^2$  continuous

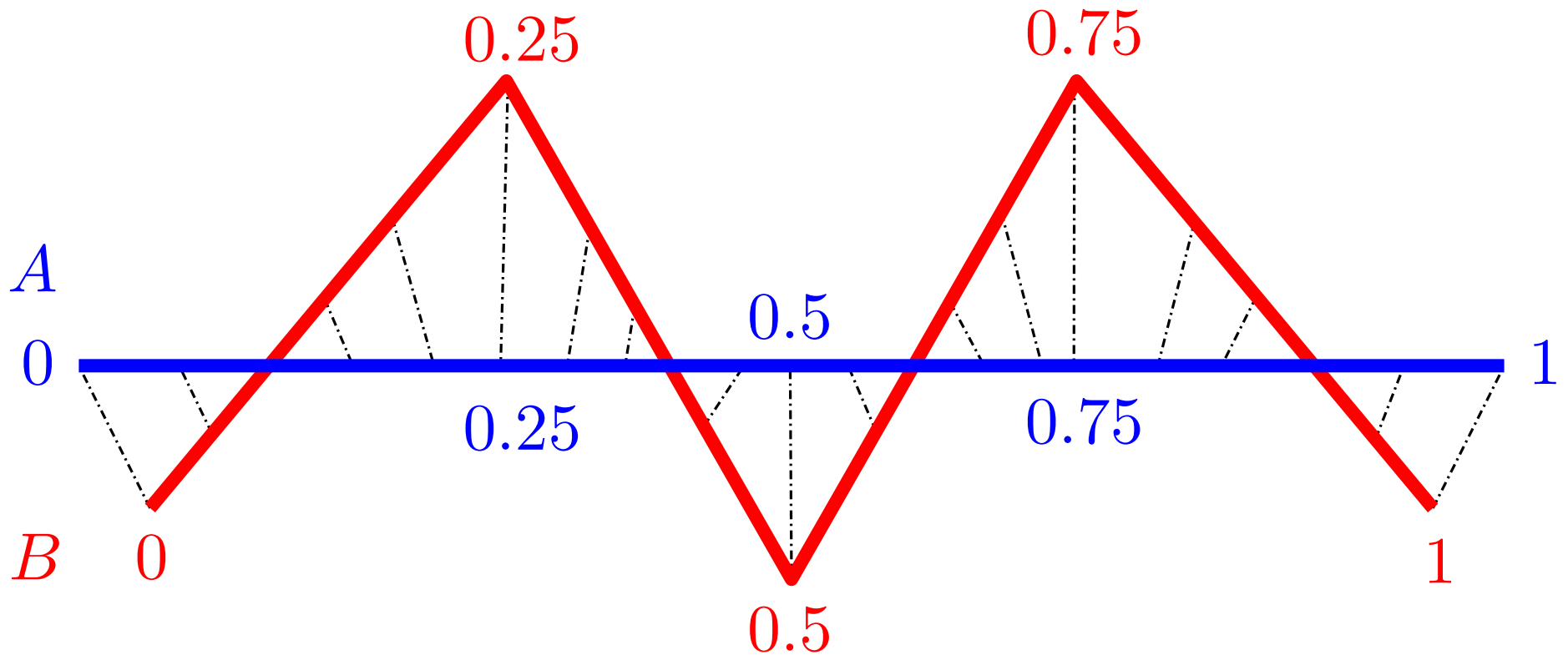
Re-parameterizations  $u, v : [0, 1] \rightarrow [0, 1]$  define a walk

At time  $t$ , **you** are at  $A(u(t))$  and **your dog** is at  $B(v(t))$

$u, v$  are continuous, nondecreasing, surjections

Fréchet distance =  $\inf_{u,v} \max_{t \in [0,1]} \|A(u(t)) - B(v(t))\|$

# Fréchet distance between curves



A metric defined by *Maurice Fréchet (1878-1973)*

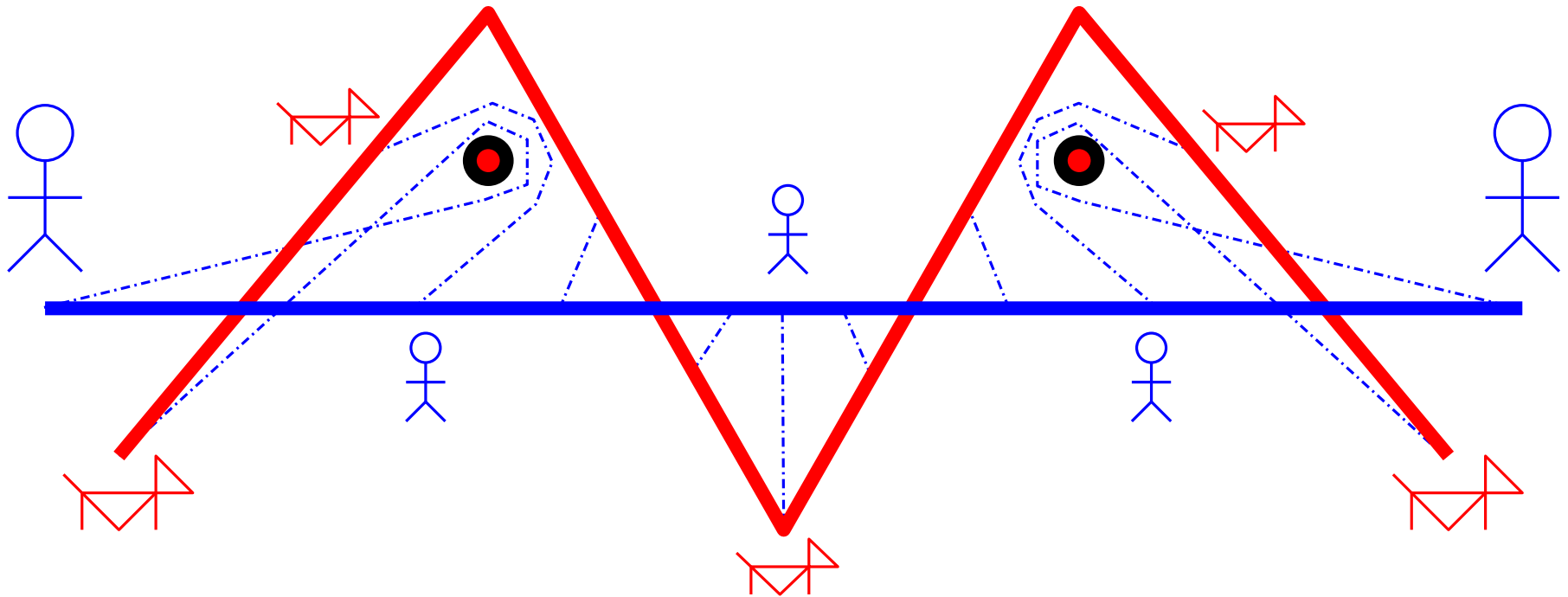
Better measure of similarity than Hausdorff distance since it accounts for the flow of the curves (handwriting recognition)

$O(N^2 \log N)$  algorithm by *Alt and Godau, 1995*  
where  $N = m + n + 2 =$  total input complexity

# Walking in the Woods

# Woods have trees ... and other obstacles

**New condition:** Leash must move **continuously**

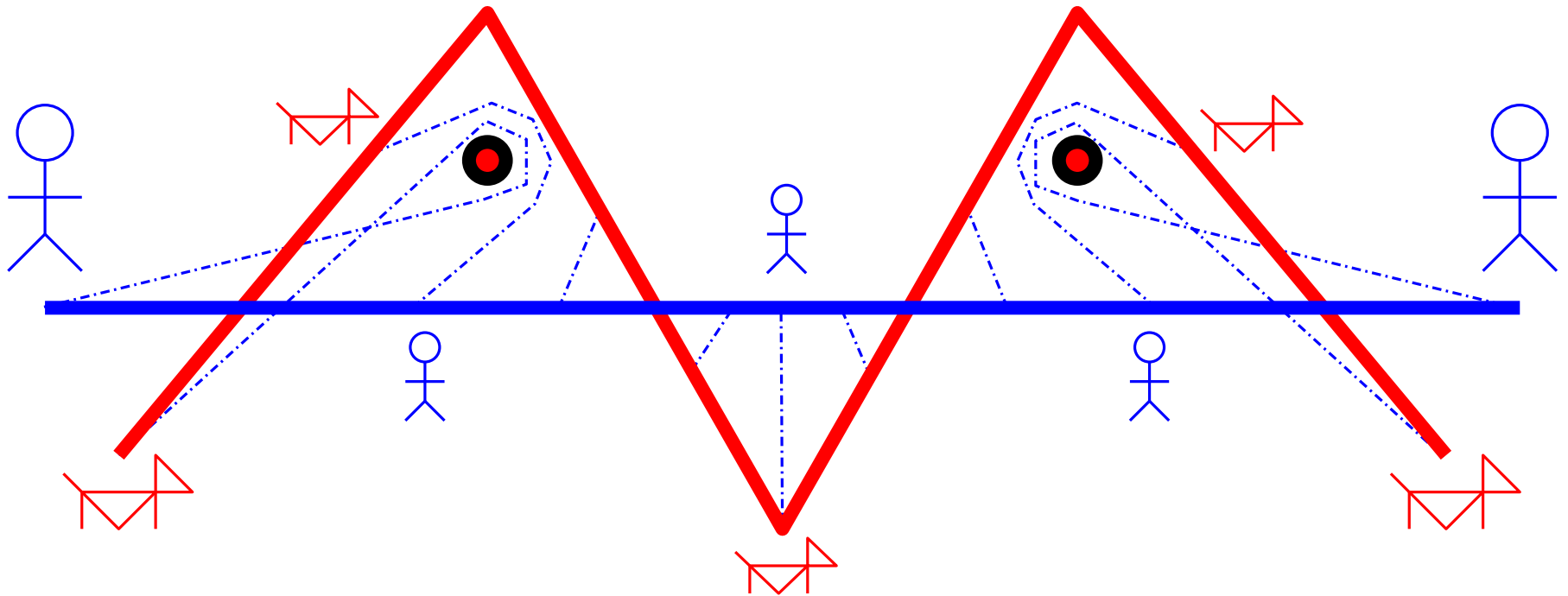


If there are obstacles, a longer leash may be required because the leash cannot jump over them

**Goal:** Walk the dog with the shortest leash possible

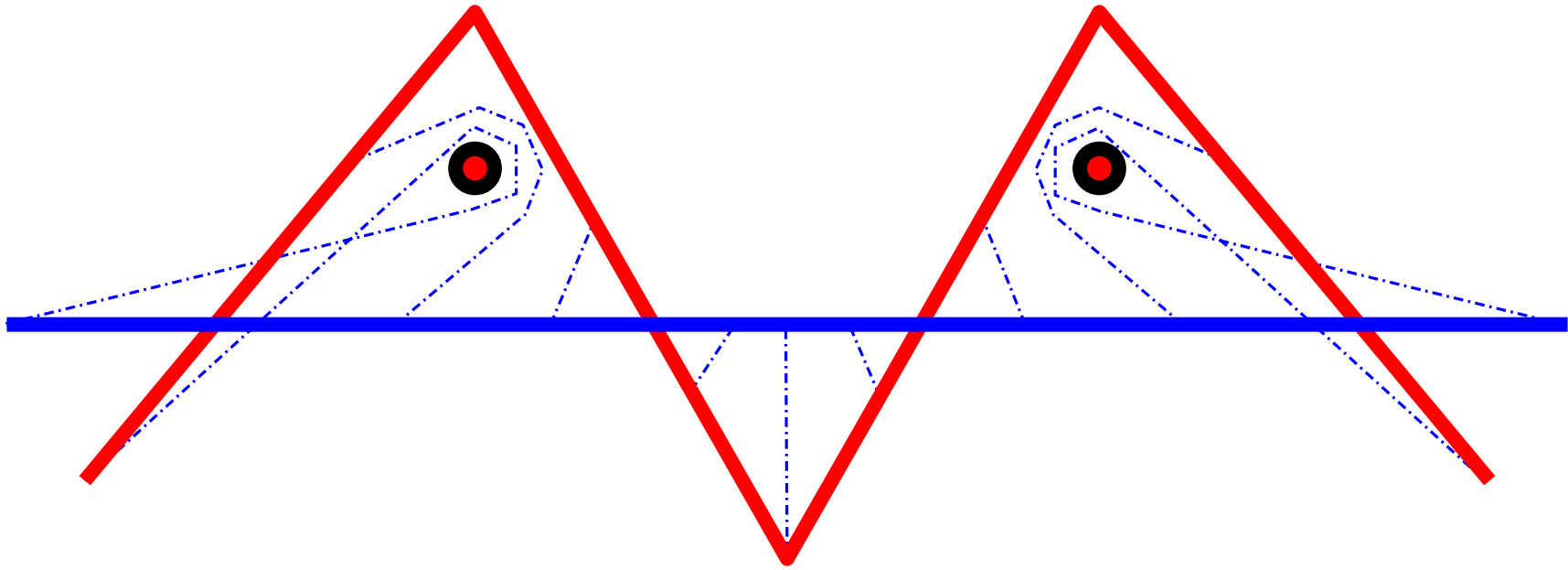
# Homotopic Fréchet distance

Dog-leash distance in a general metric space where the leash must move **continuously** in the metric space



We give a polynomial-time algorithm to compute the homotopic Fréchet distance between two polygonal curves in **the plane with obstacles** (= **punctured plane**)

# Application 1: Morphing

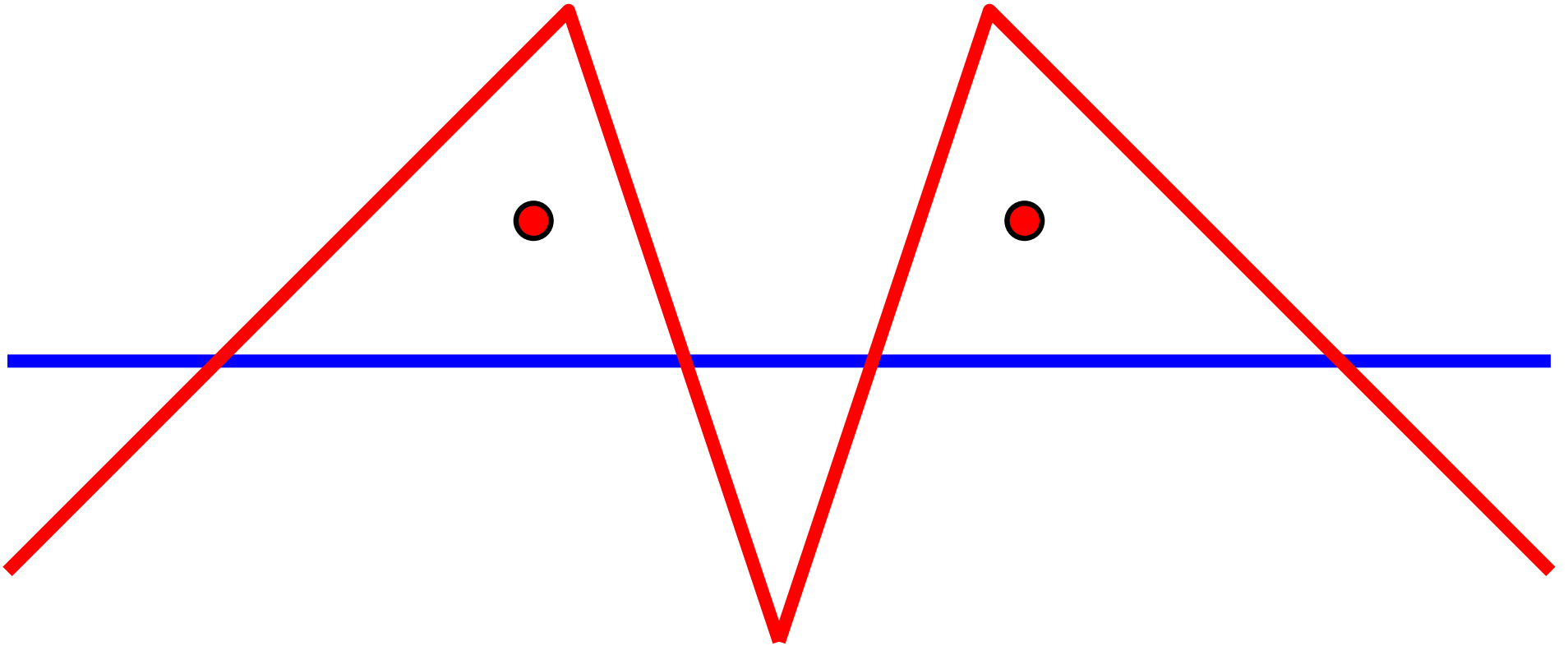


Leash motion encodes a continuous deformation between  $A$  and  $B$ , without penetrating obstacles

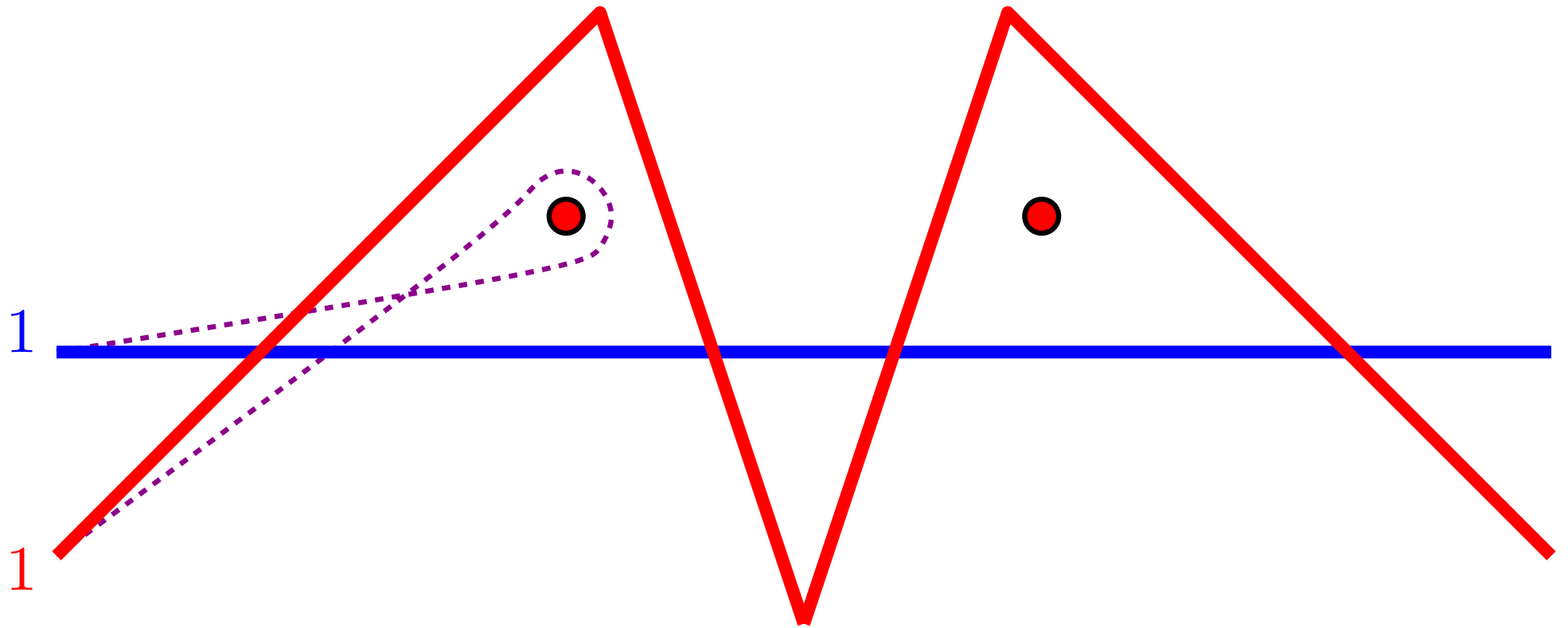
The “cost” of the deformation is the maximum distance any point has to travel, i.e., the Fréchet distance



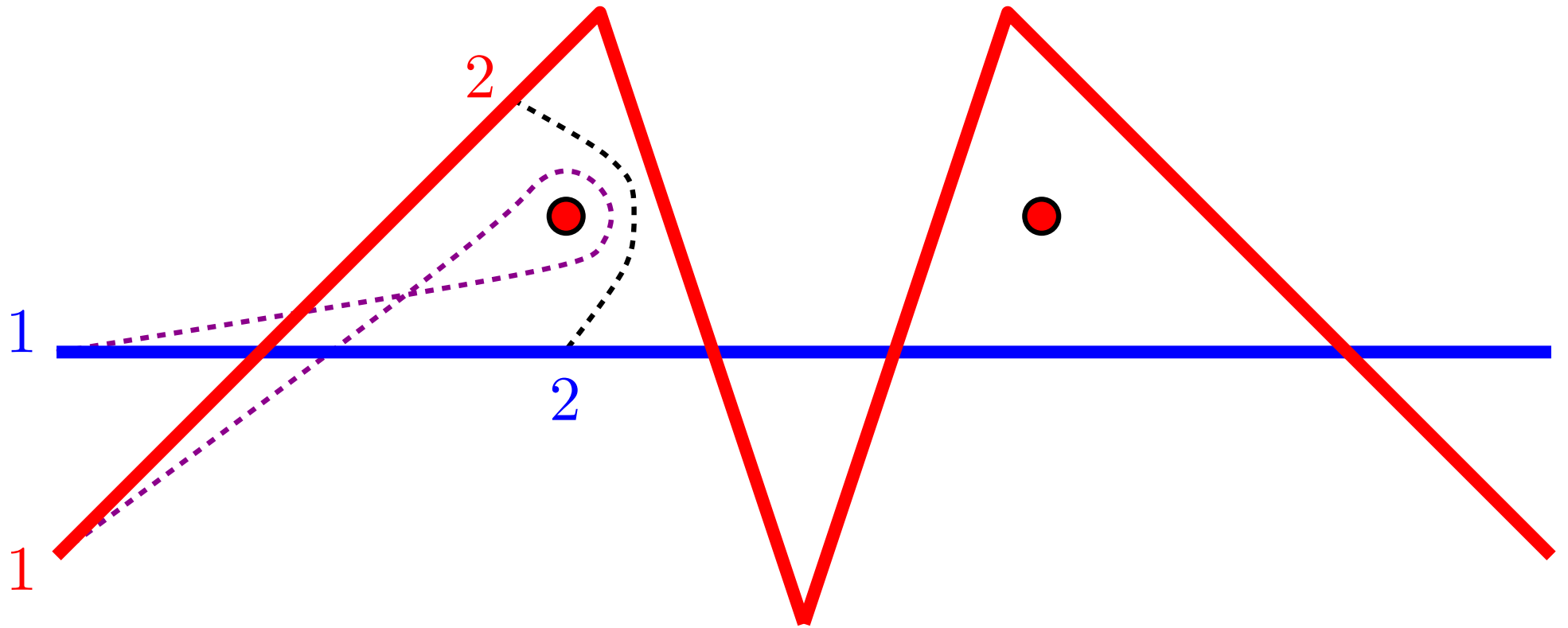
# Example 1



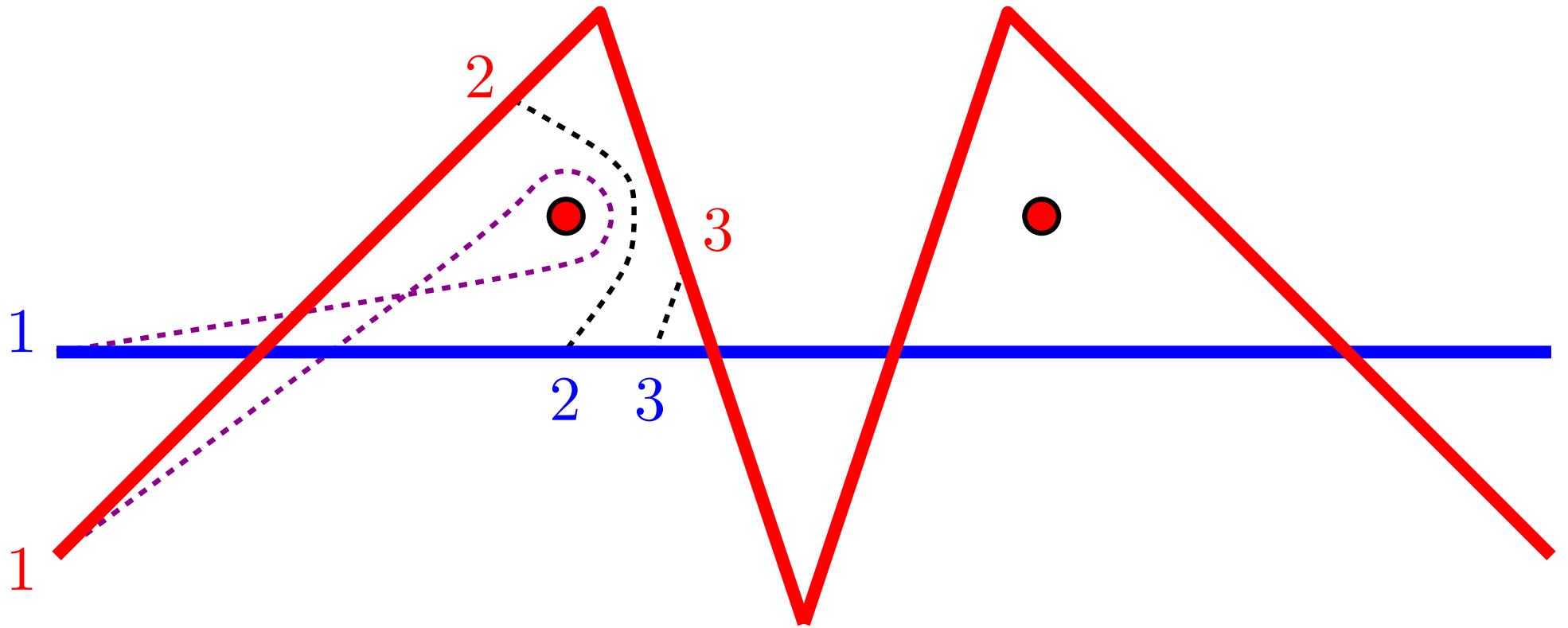
# Example 1



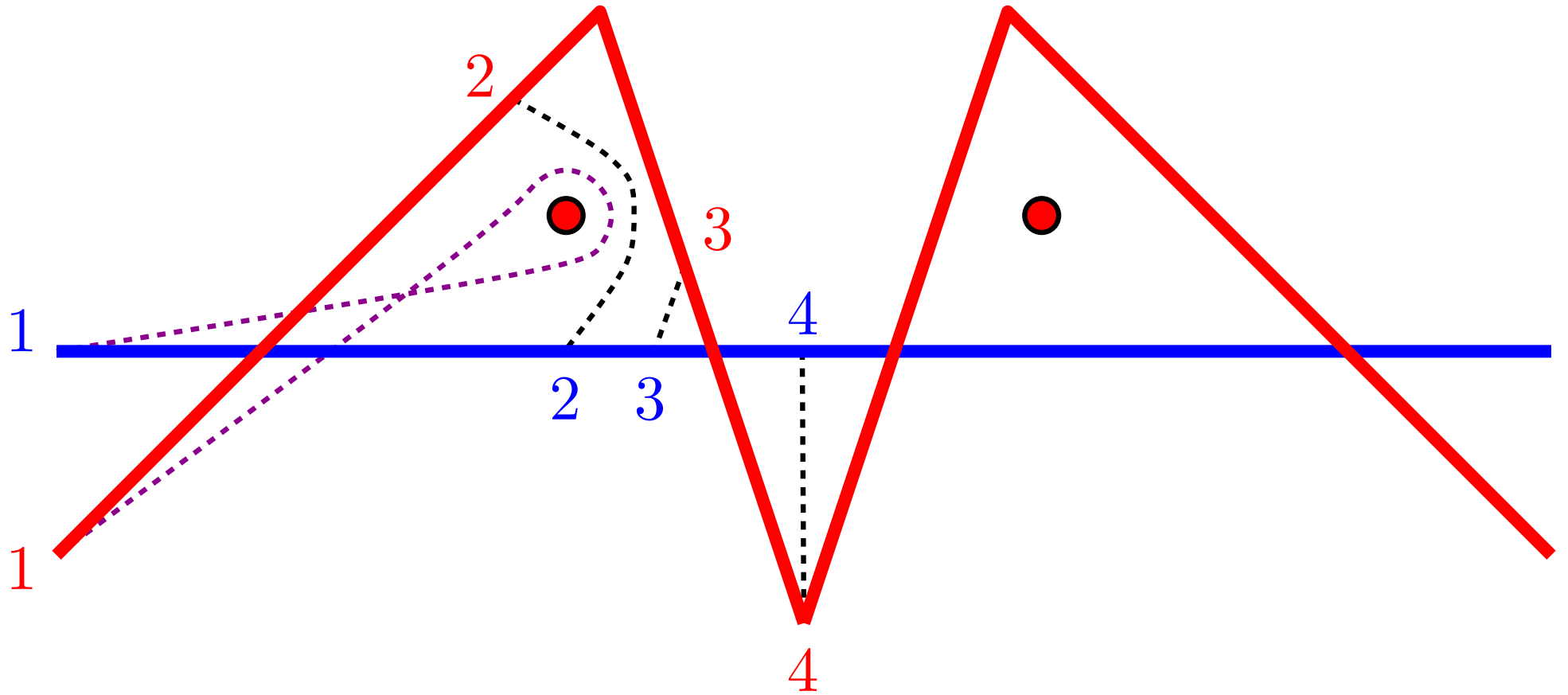
# Example 1



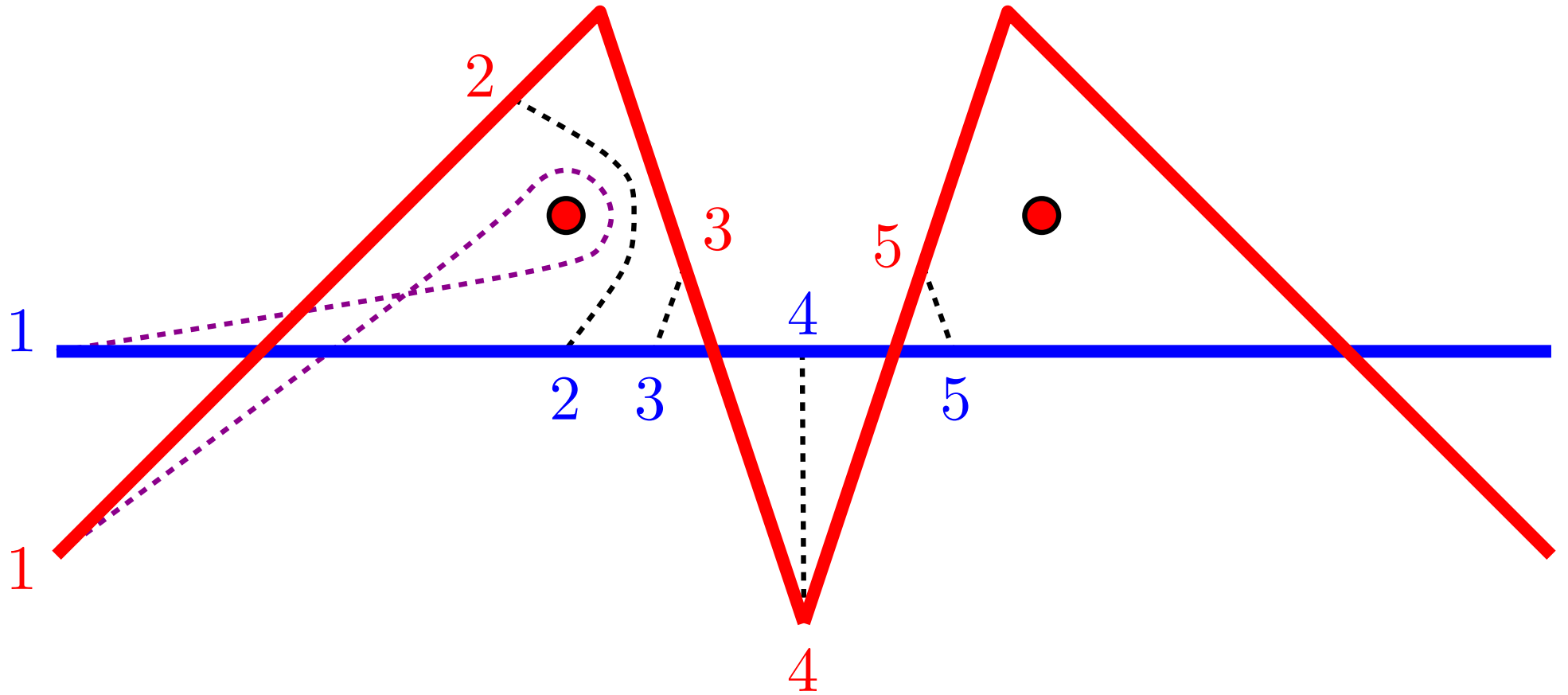
# Example 1



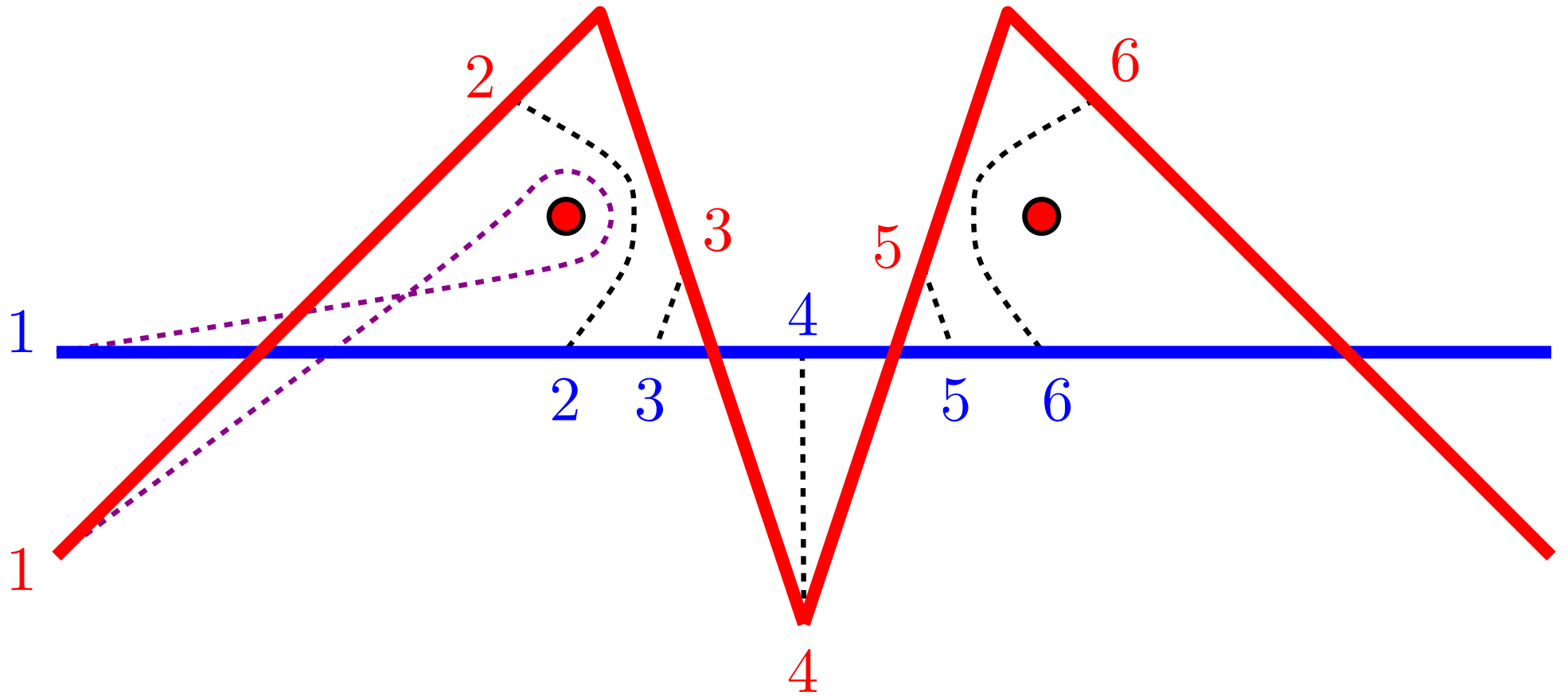
# Example 1



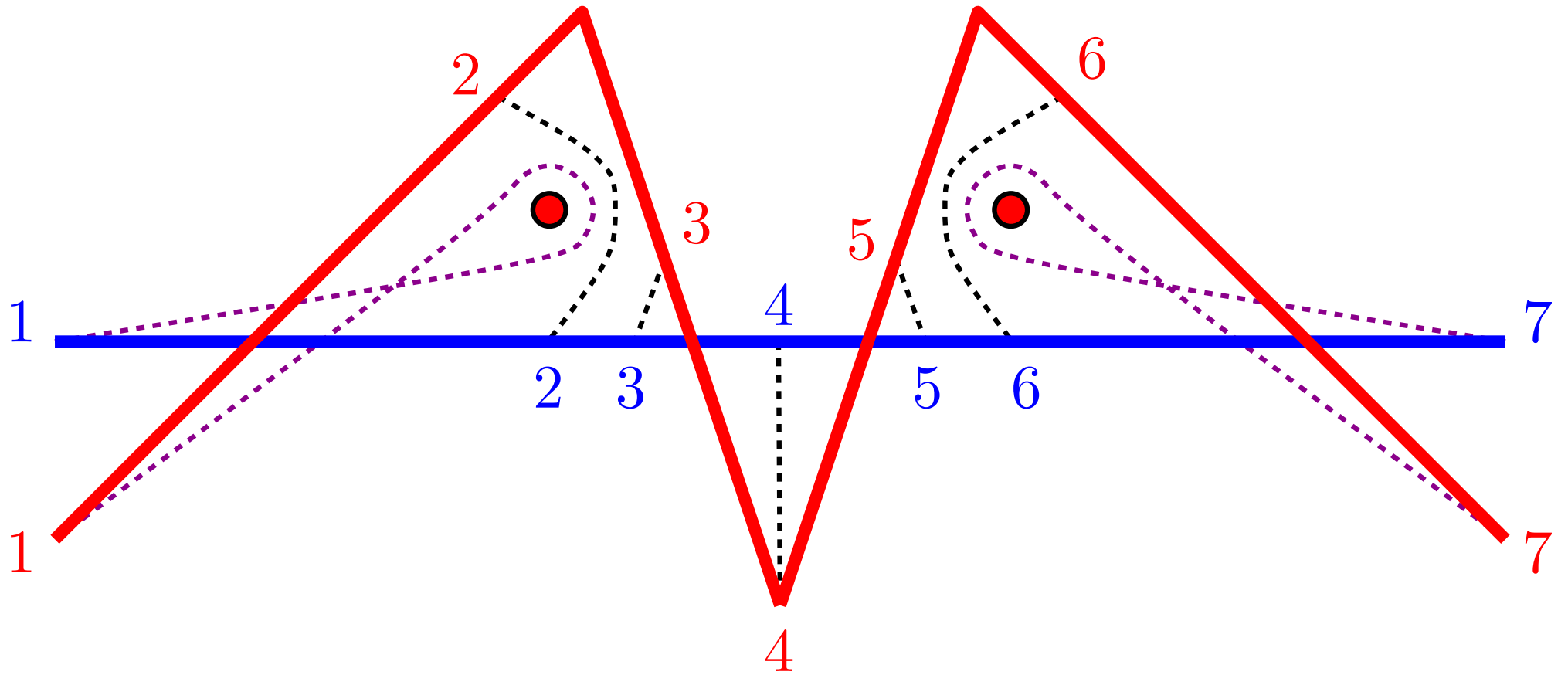
# Example 1



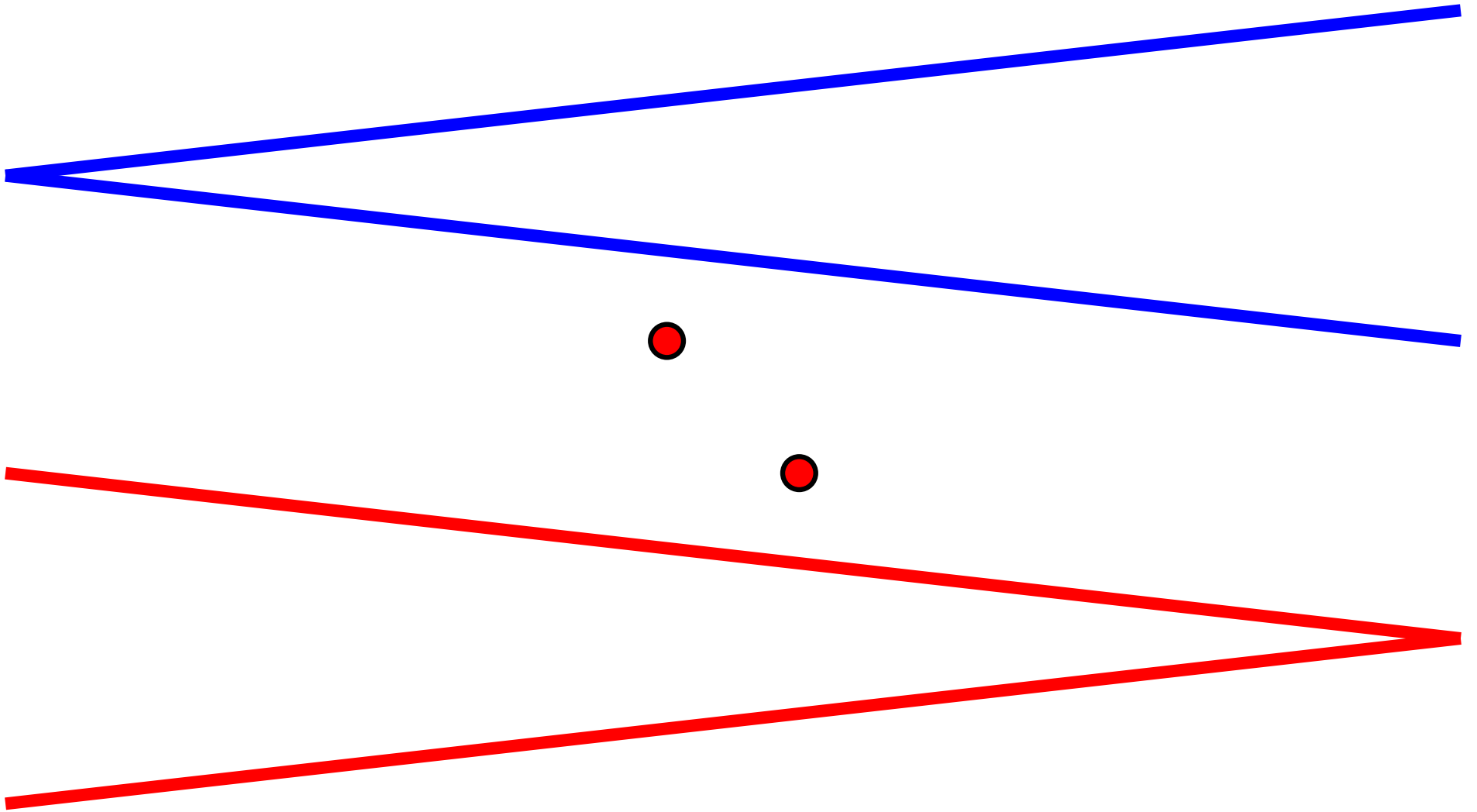
# Example 1



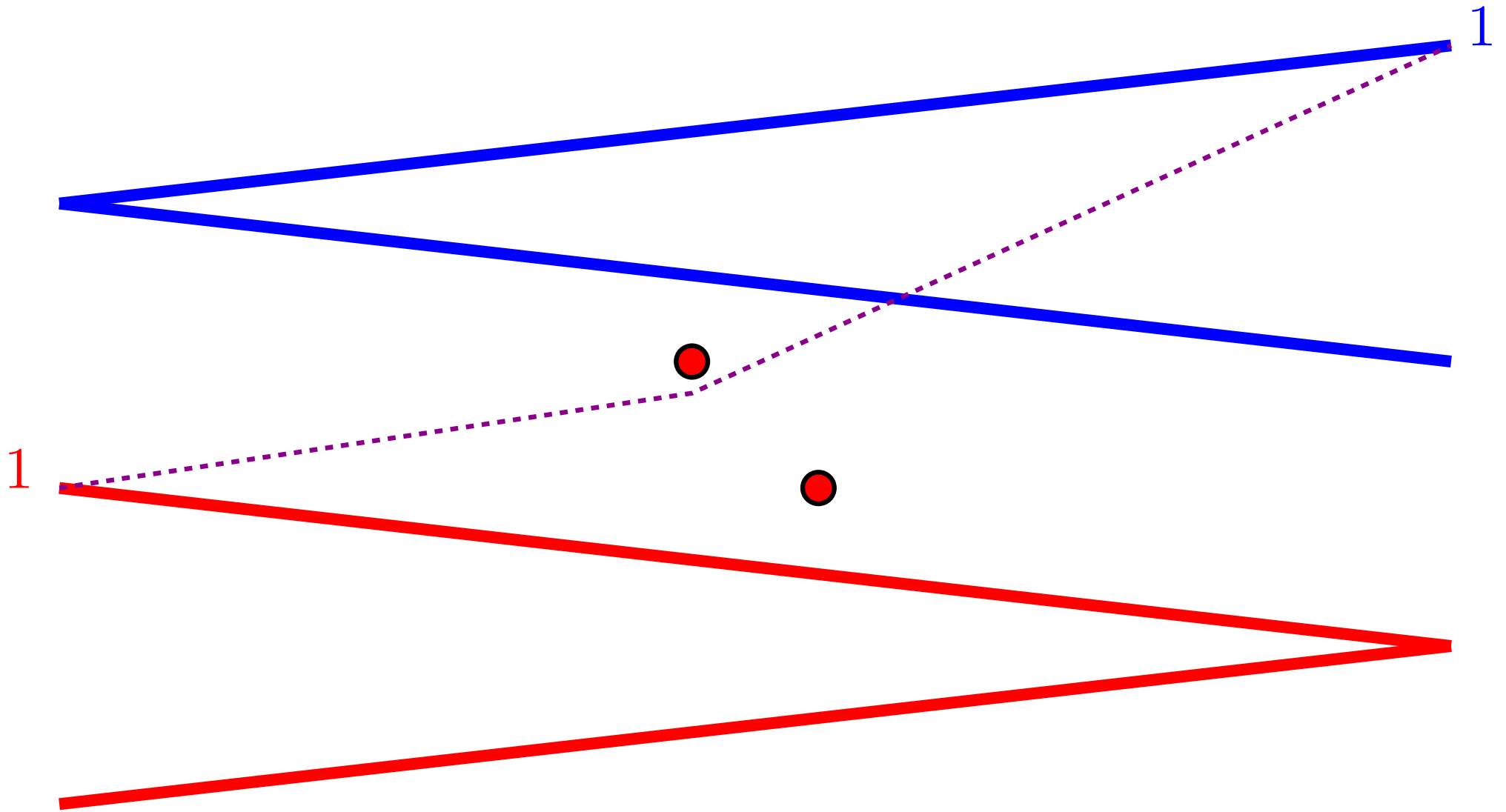
# Example 1



# Example 2



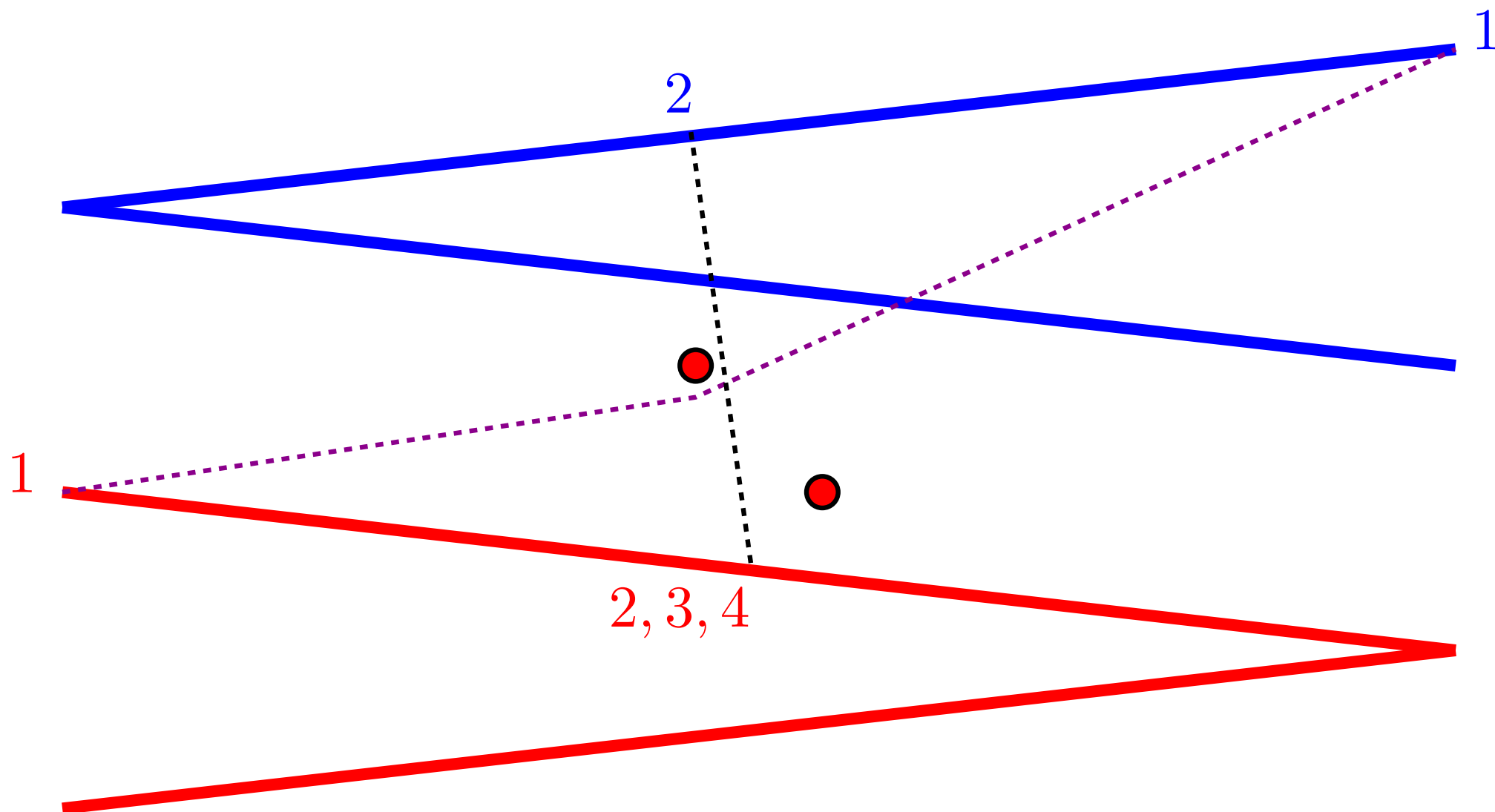
# Example 2



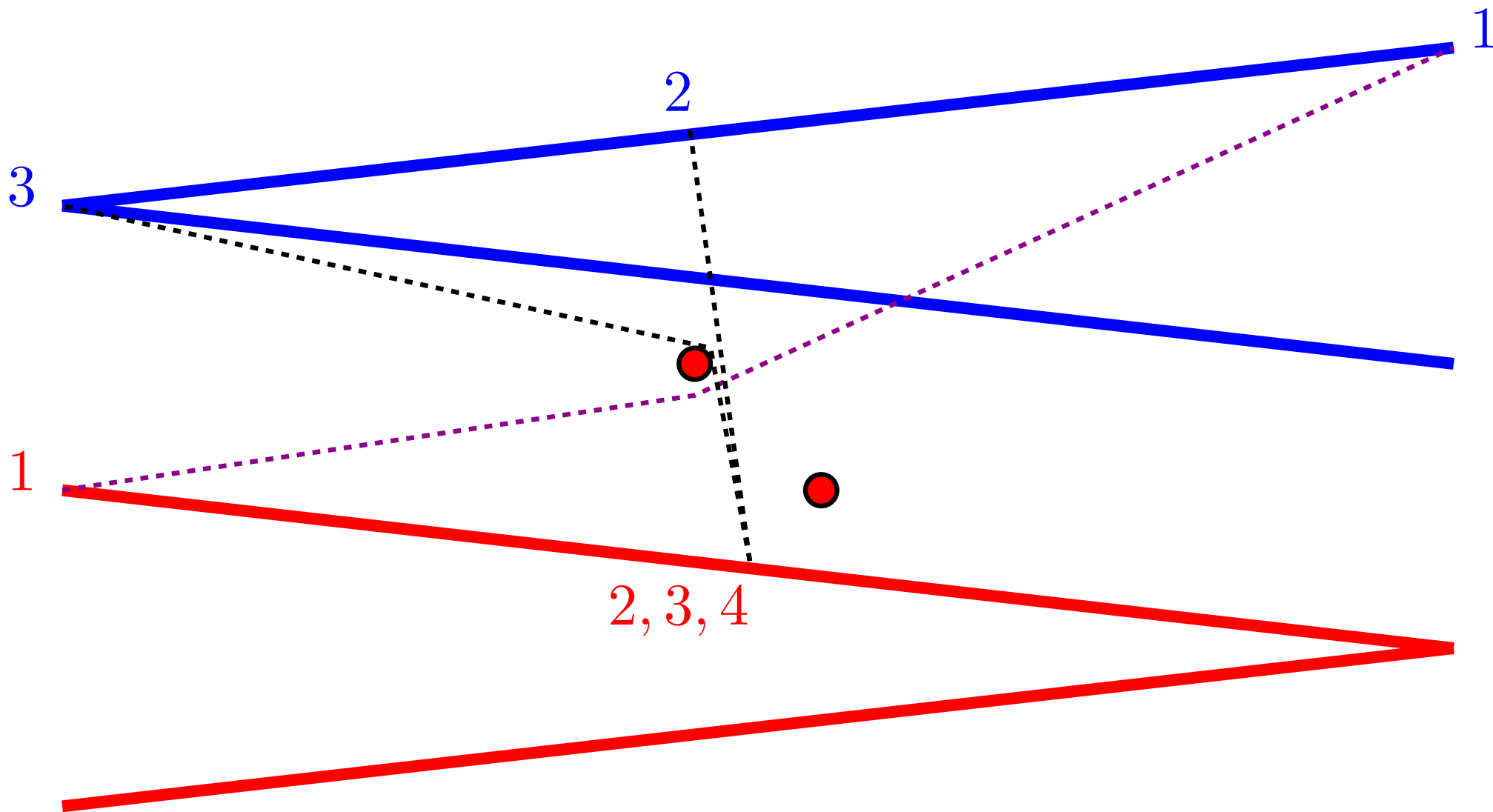
1

1

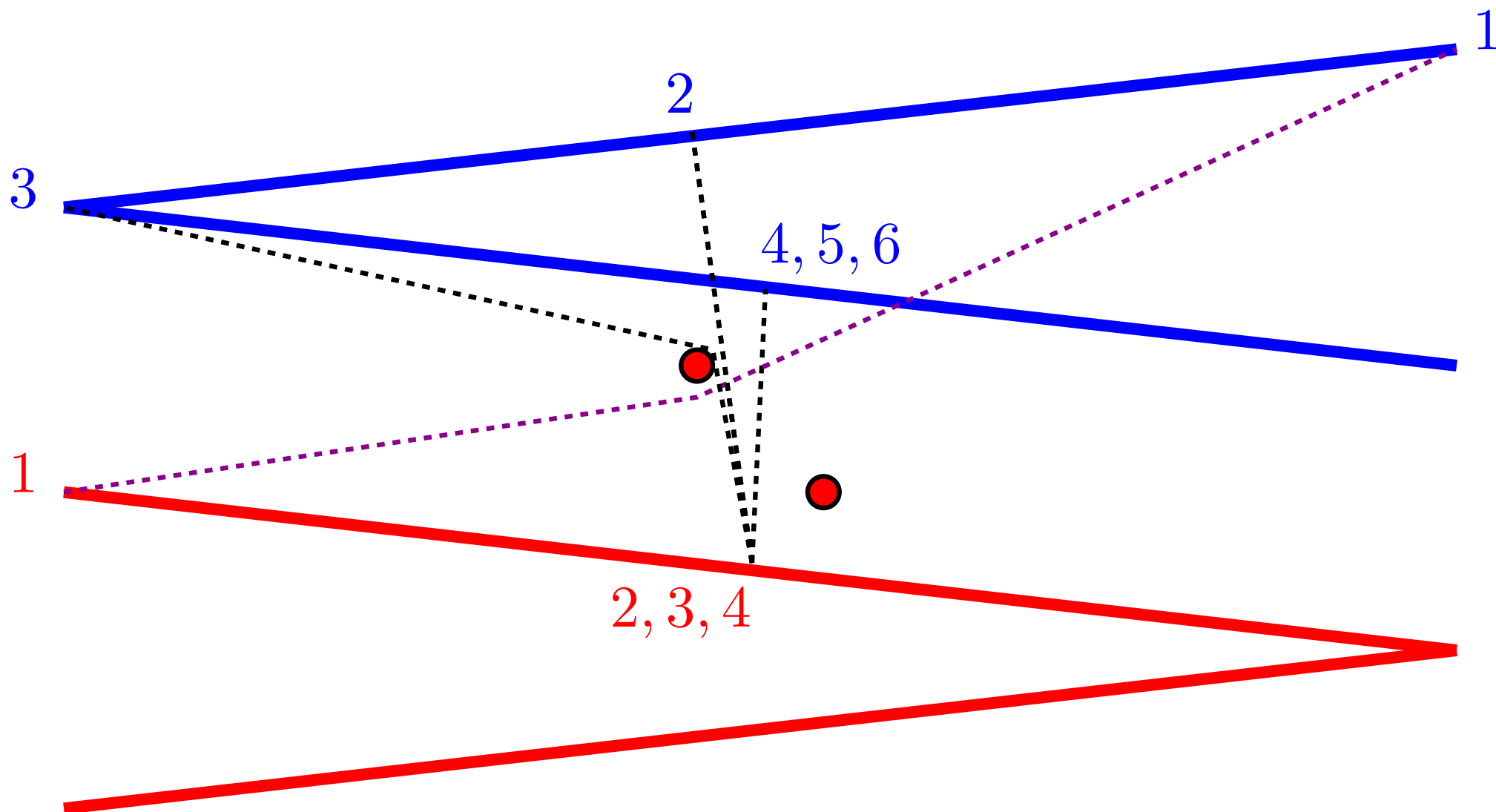
# Example 2



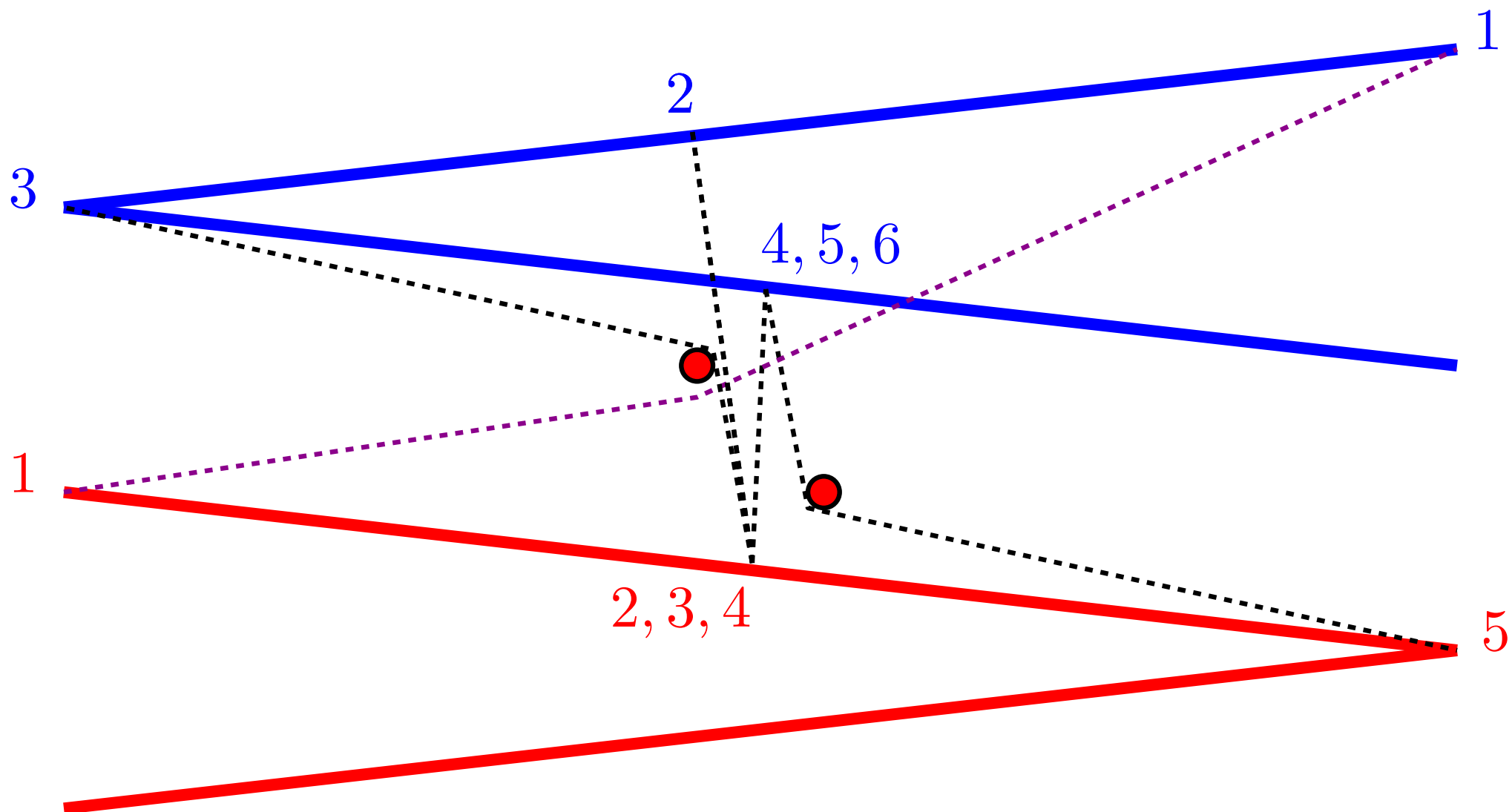
# Example 2



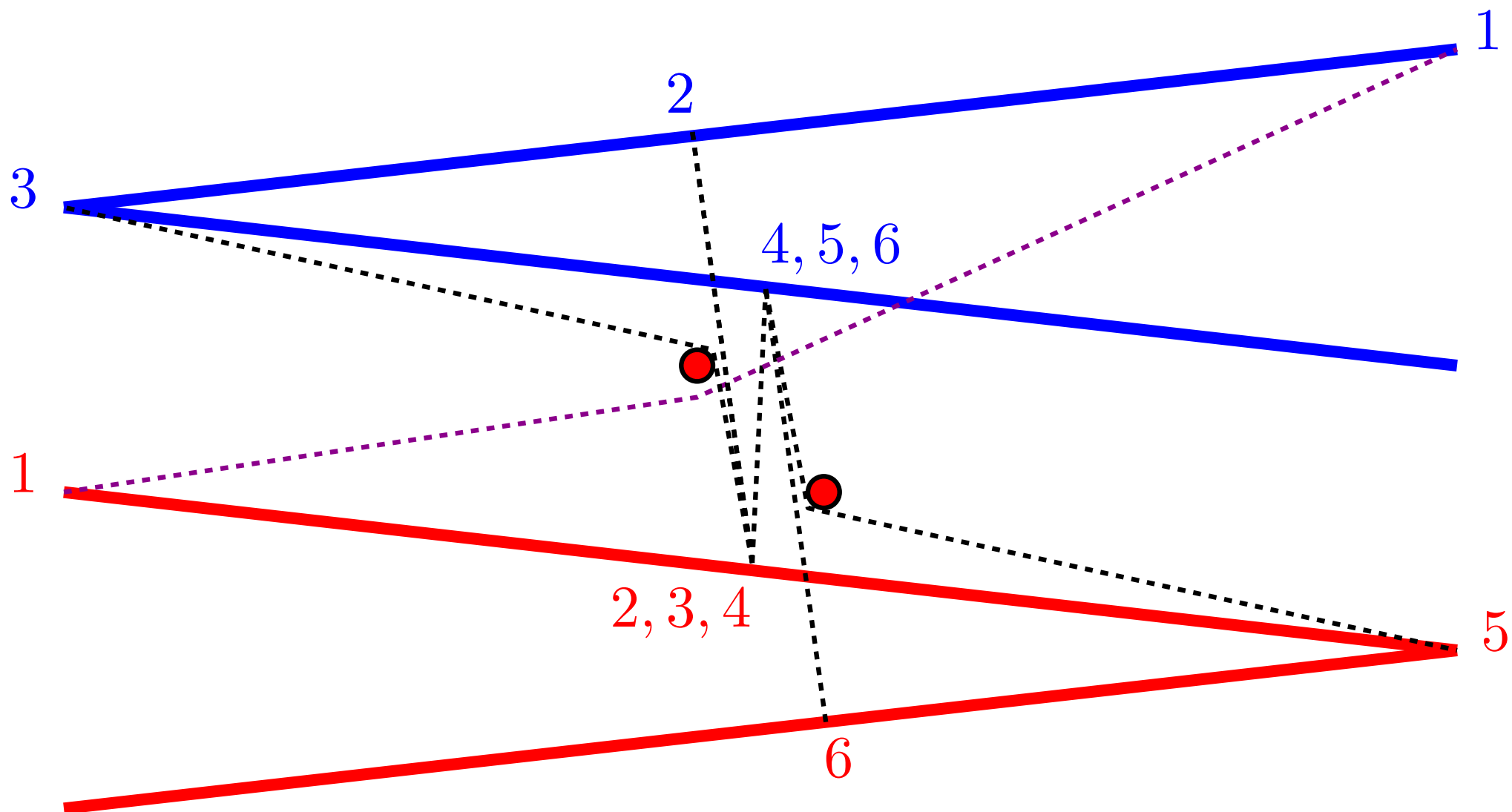
# Example 2



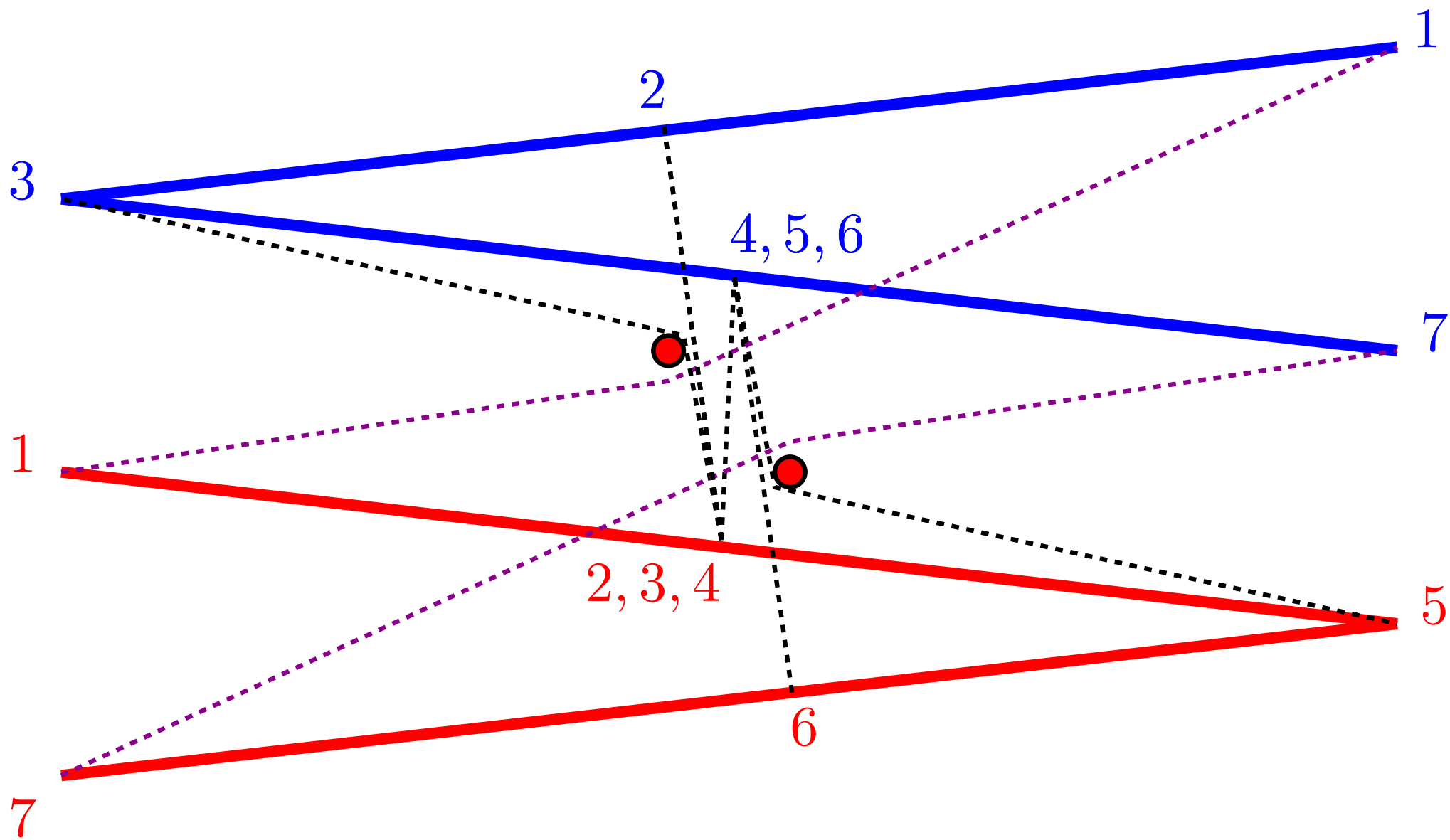
# Example 2



# Example 2

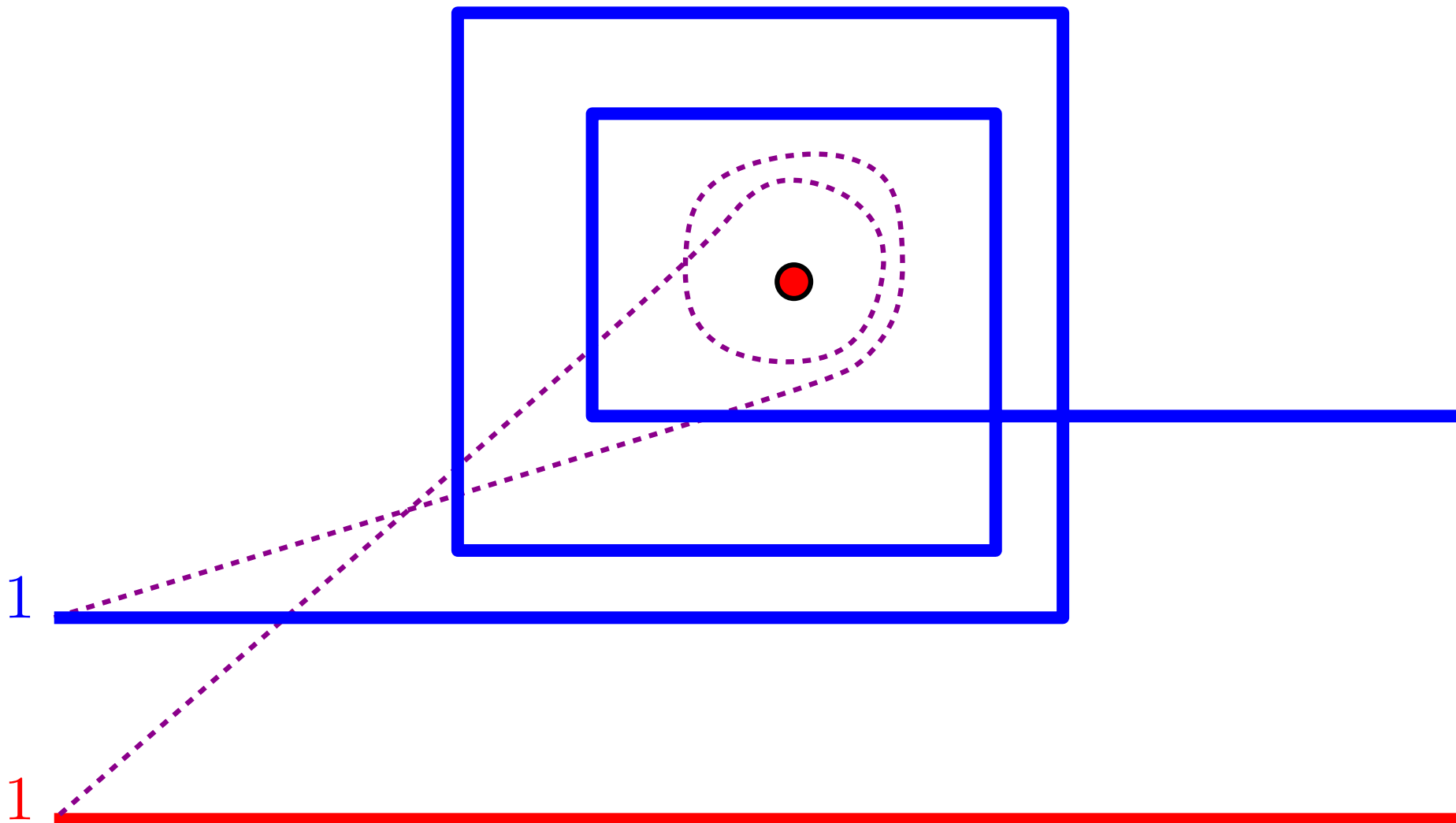


# Example 2

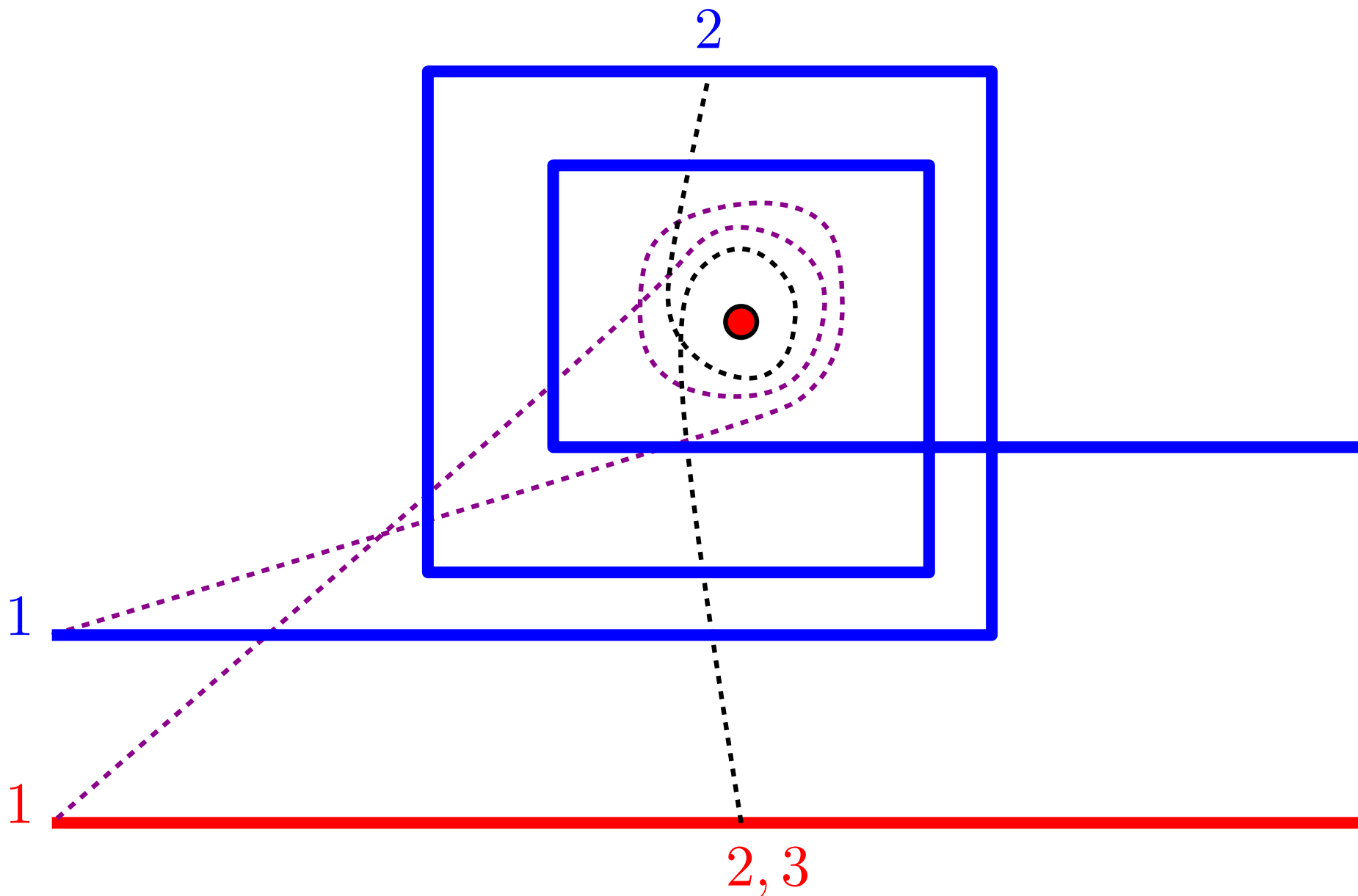




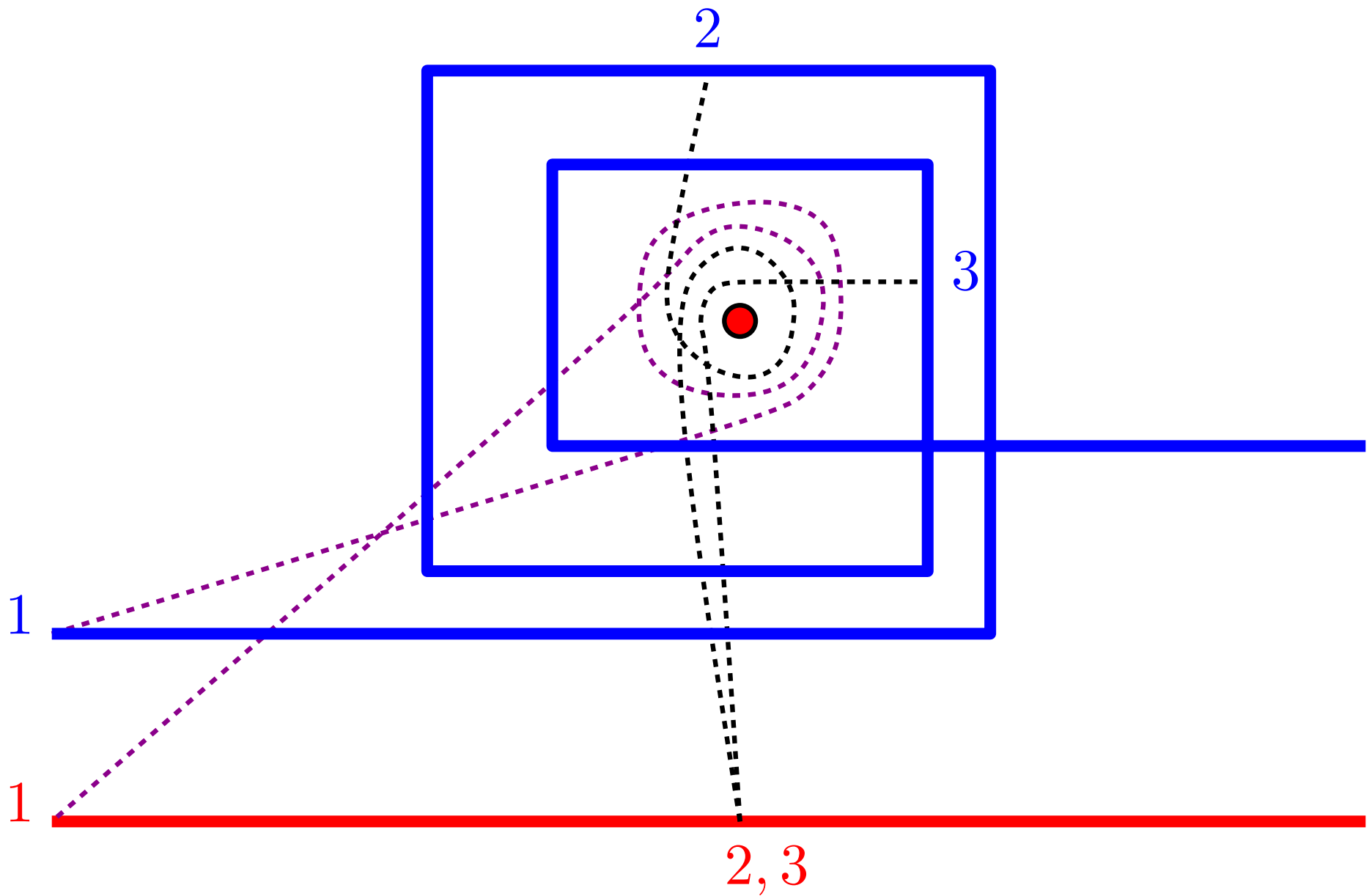
# Example 3



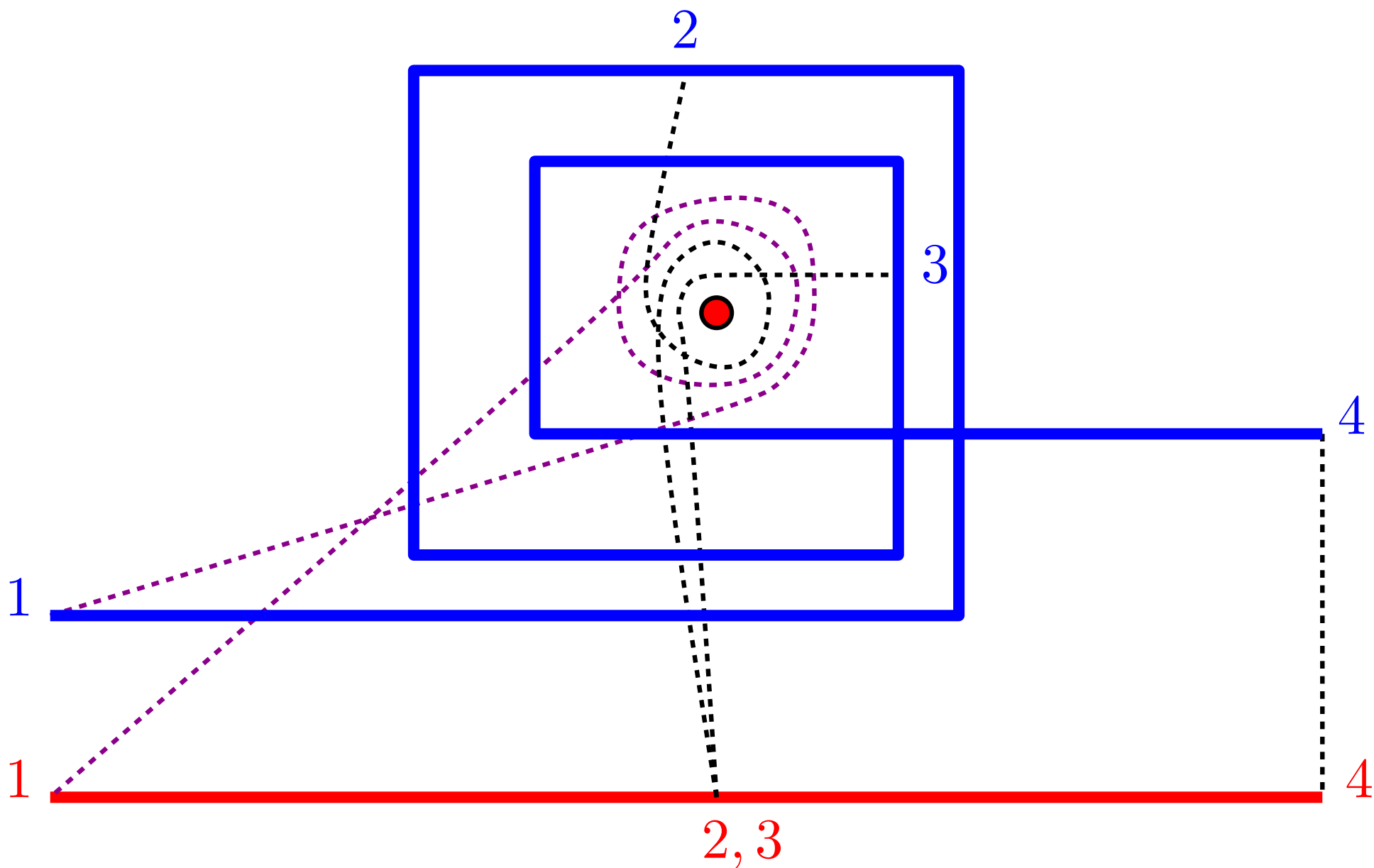
# Example 3



# Example 3



# Example 3



# Definitions



# Leash map

Continuous function

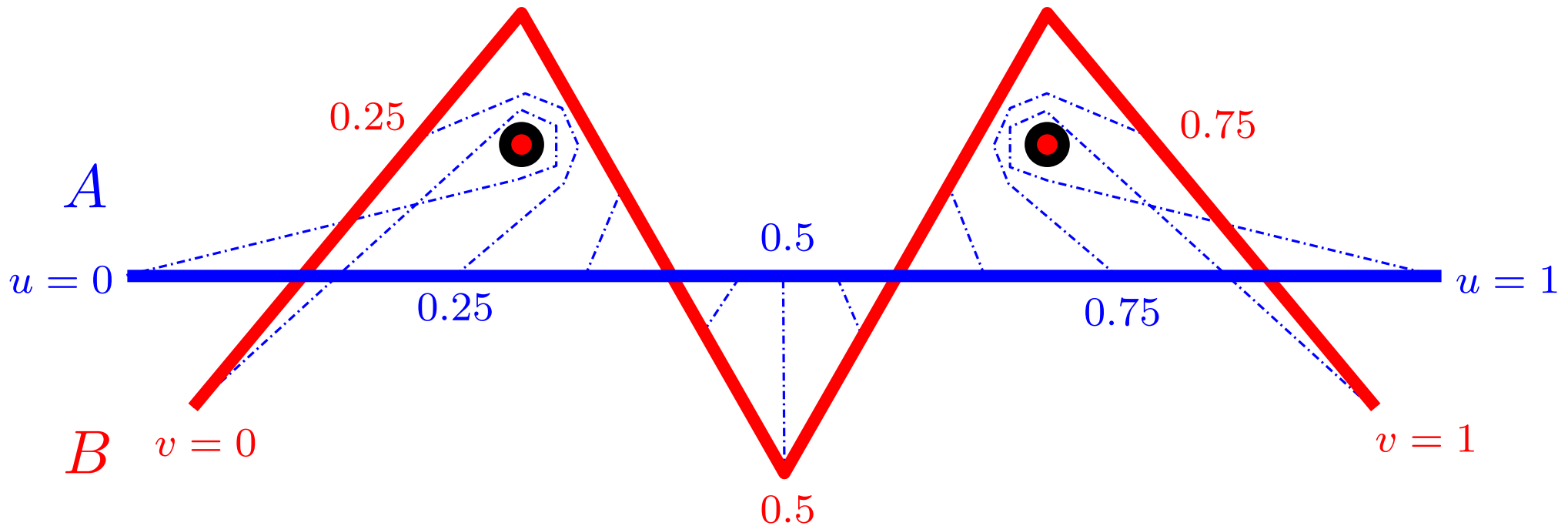
$$\ell : [0, 1] \times [0, 1] \rightarrow S$$

arc-length

time

metric space

s.t.  $u = \ell(0, \cdot)$ ,  $v = \ell(1, \cdot)$  are re-parameterizations of  $A$ ,  $B$



i.e.  $\ell(\cdot, t)$  is the leash at time  $t$  joining  $A(u(t))$  and  $B(v(t))$

# Homotopic Fréchet distance

Continuous function  $\ell : [0, 1] \times [0, 1] \rightarrow S$   
arc-length      time      metric space  
s.t.  $u = \ell(0, \cdot)$ ,  $v = \ell(1, \cdot)$  are re-parameterizations of  $A$ ,  $B$

The **cost** of a leash map  $\ell$  is the maximum length of the leash at any time during the leash motion:

$$\text{cost}(\ell) := \max_{t \in [0, 1]} \{ \text{Length of } \ell(\cdot, t) \}$$

# Homotopic Fréchet distance

Continuous function  $\ell : [0, 1] \times [0, 1] \rightarrow S$   
arc-length      time      metric space  
s.t.  $u = \ell(0, \cdot)$ ,  $v = \ell(1, \cdot)$  are re-parameterizations of  $A$ ,  $B$

The **cost** of a leash map  $\ell$  is the maximum length of the leash at any time during the leash motion:

$$\text{cost}(\ell) := \max_{t \in [0, 1]} \{ \text{Length of } \ell(\cdot, t) \}$$

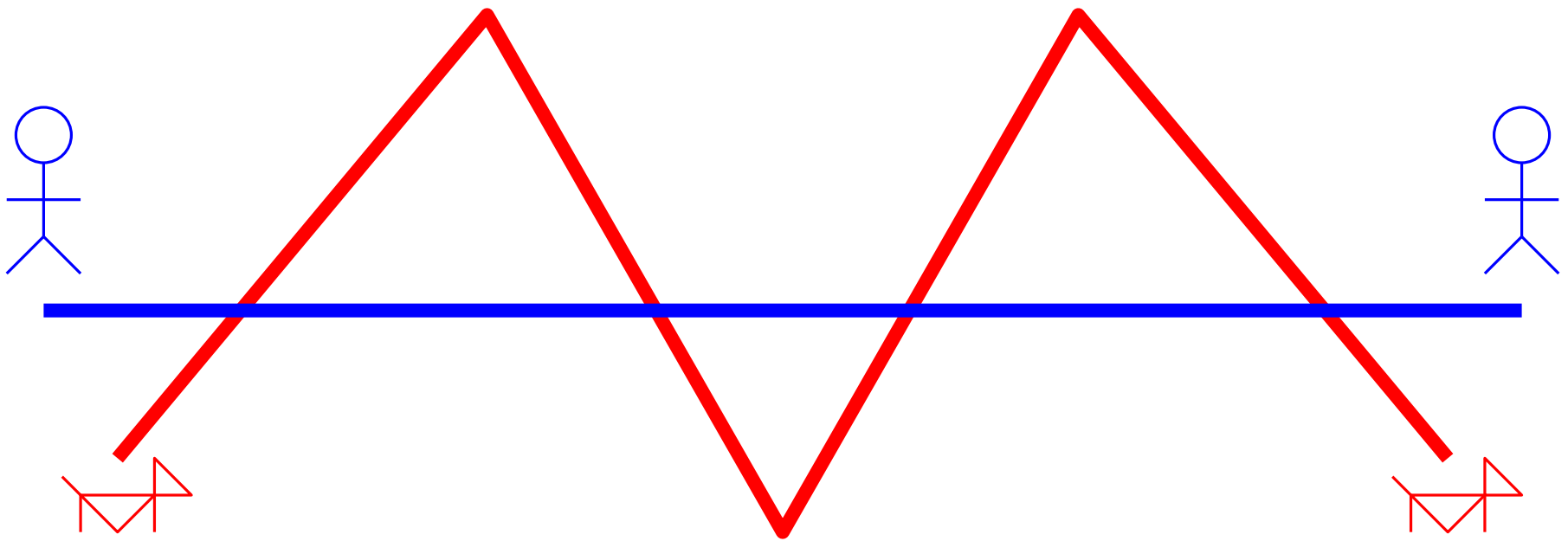
The **homotopic Fréchet distance** is the minimum cost of any leash map:

$$F(A, B) := \inf_{\text{leash map } \ell} \{ \text{cost}(\ell) \}$$

Meanwhile,  
back in the woods . . .

# Punctured plane

Let  $A$ ,  $B$  be two given curves in  $\mathbb{E}^2$

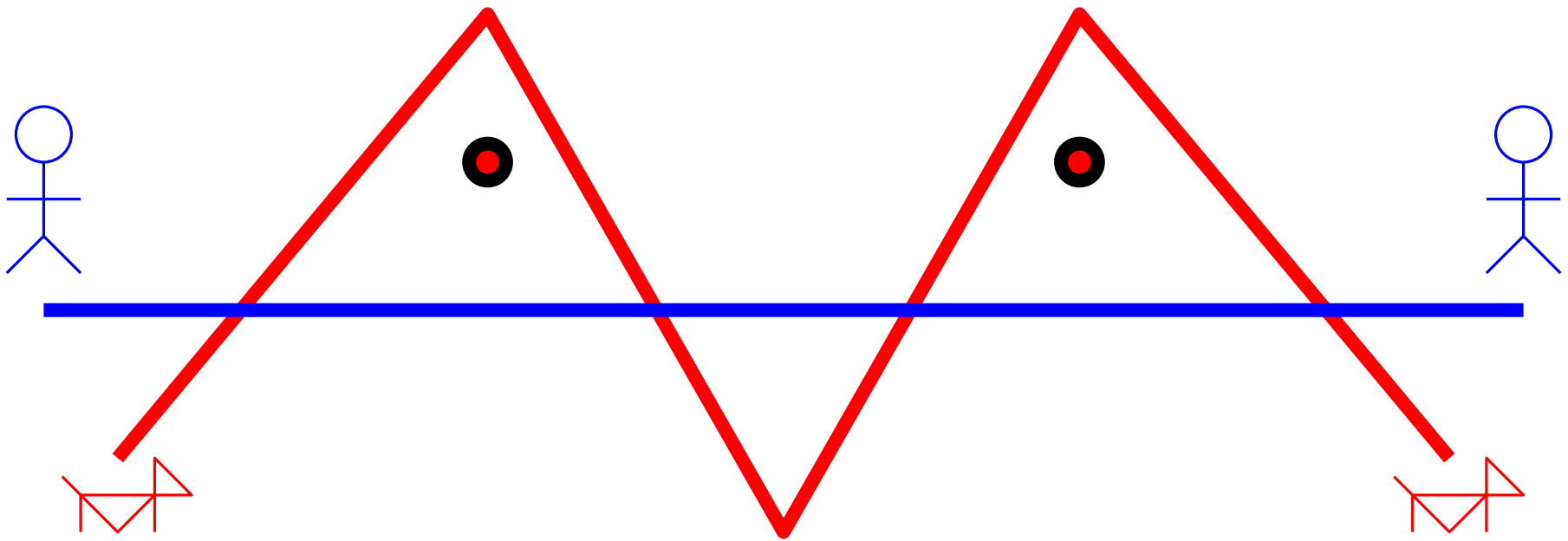


# Punctured plane

Let  $A$ ,  $B$  be two given curves in  $\mathbb{E}^2$

Let  $P$  be a set of obstacles, with total complexity  $|P|$

$$\text{Punctured Plane} = \mathbb{E}^2 \setminus P$$



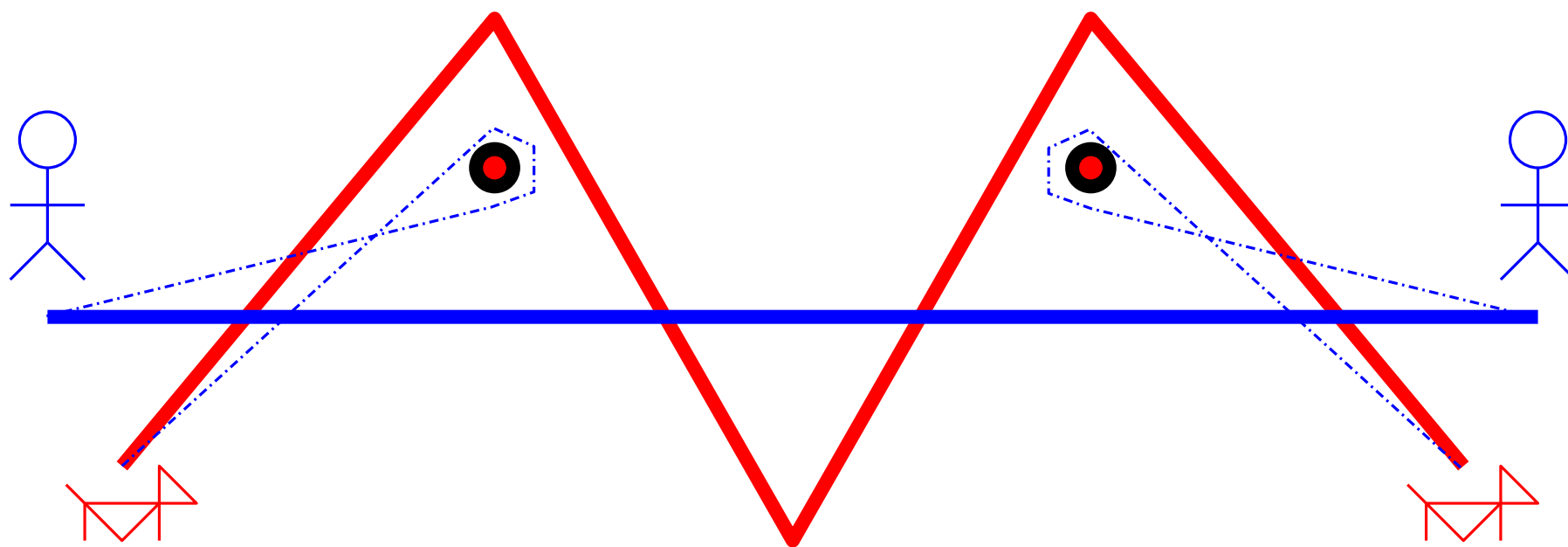
# Punctured plane

Let  $A$ ,  $B$  be two given curves in  $\mathbb{E}^2$

Let  $P$  be a set of obstacles, with total complexity  $|P|$

**Punctured Plane** =  $\mathbb{E}^2 \setminus P$

A **leash** is a curve in  $\mathbb{E}^2 \setminus P$  joining  $A$  and  $B$



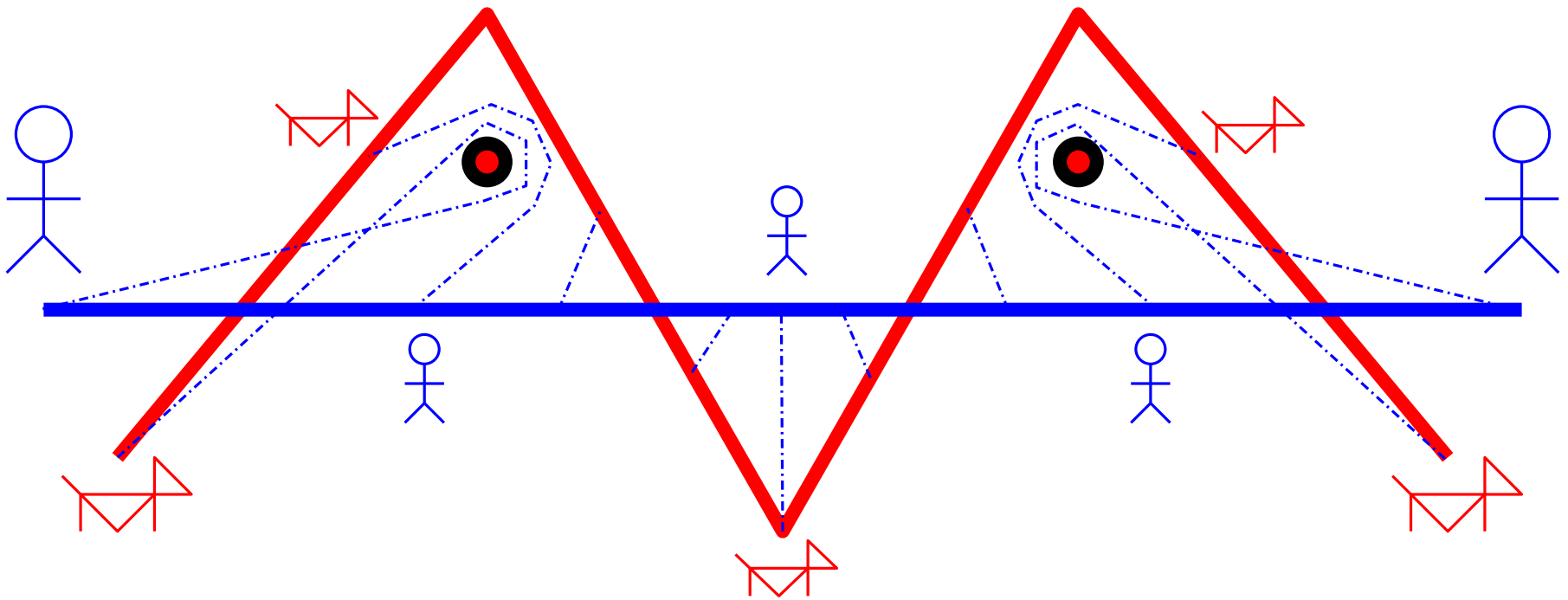
# Punctured plane

Let  $A, B$  be two given curves in  $\mathbb{E}^2$

Let  $P$  be a set of obstacles, with total complexity  $|P|$

**Punctured Plane**  $= \mathbb{E}^2 \setminus P$

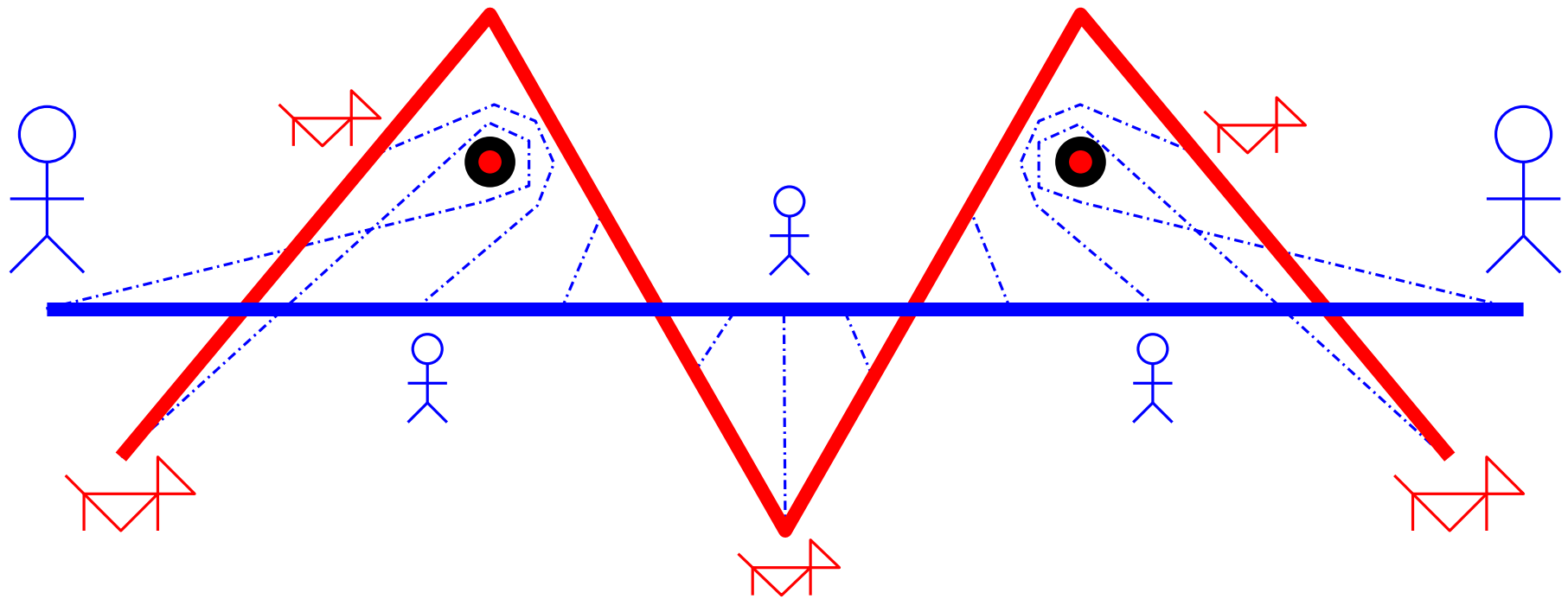
A **leash** is a curve in  $\mathbb{E}^2 \setminus P$  joining  $A$  and  $B$



Leash map  $\ell : [0, 1] \times [0, 1] \rightarrow \mathbb{E}^2 \setminus P$  must be continuous;  
so, the leash cannot jump over obstacles

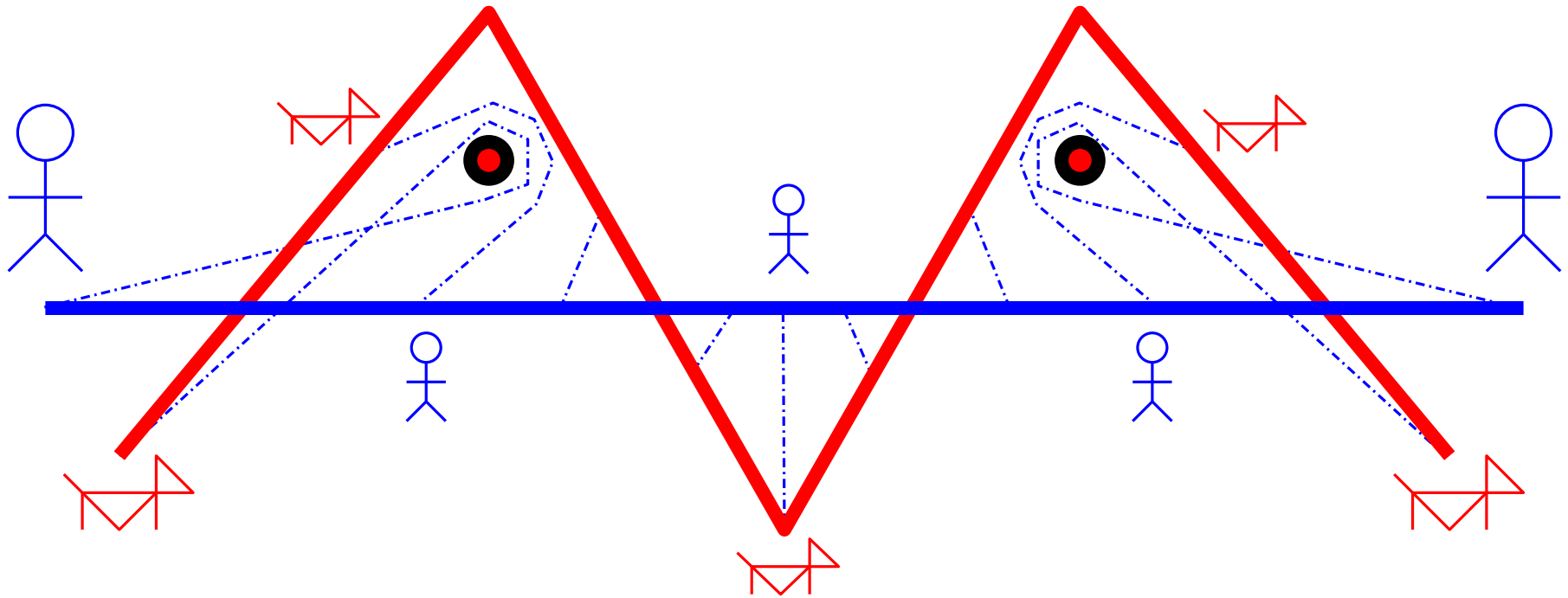
# Relative homotopy

Two leashes are **relatively homotopic** if one can be continuously transformed into the other in the punctured plane *while keeping their endpoints on the respective curves*



# Relative homotopy

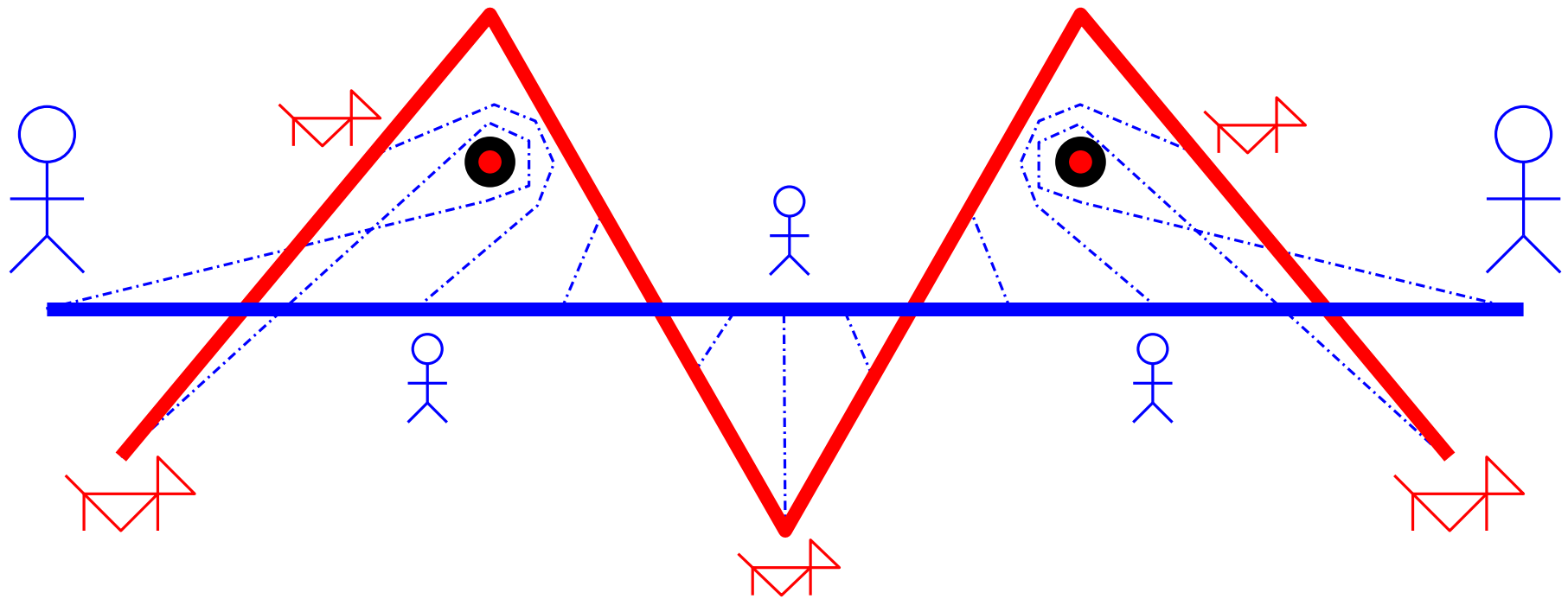
Two leashes are **relatively homotopic** if one can be continuously transformed into the other in the punctured plane while keeping their endpoints on the respective curves



Functions  $\alpha, \beta : X \rightarrow Y$  are (freely) **homotopic** if there is a continuous function  $h : X \times [0, 1] \rightarrow Y$  such that  $h(\cdot, 0) = \alpha(\cdot)$  and  $h(\cdot, 1) = \beta(\cdot)$

# Relative homotopy class

Every leash map  $\ell_h$  describes a set of leashes belonging to some relative homotopy class  $h$



# Homotopic Fréchet distance redux

Let  $h$  be a relative homotopy class

# Homotopic Fréchet distance redux

Let  $h$  be a relative homotopy class

Let  $\ell_h$  be a leash map in homotopy class  $h$

# Homotopic Fréchet distance redux

Let  $h$  be a relative homotopy class

Let  $\ell_h$  be a leash map in homotopy class  $h$

Let  $F_h(A, B) := \inf_{\ell_h} \{ \text{cost}(\ell_h) \}$

# Homotopic Fréchet distance redux

Let  $h$  be a relative homotopy class

Let  $\ell_h$  be a leash map in homotopy class  $h$

Let  $F_h(A, B) := \inf_{\ell_h} \{ \text{cost}(\ell_h) \}$

## Homotopic Fréchet distance

$$F(A, B) := \min_h \{ F_h(A, B) \}$$

# Key Insights

*a.k.a.*



‘Aha!’ moments

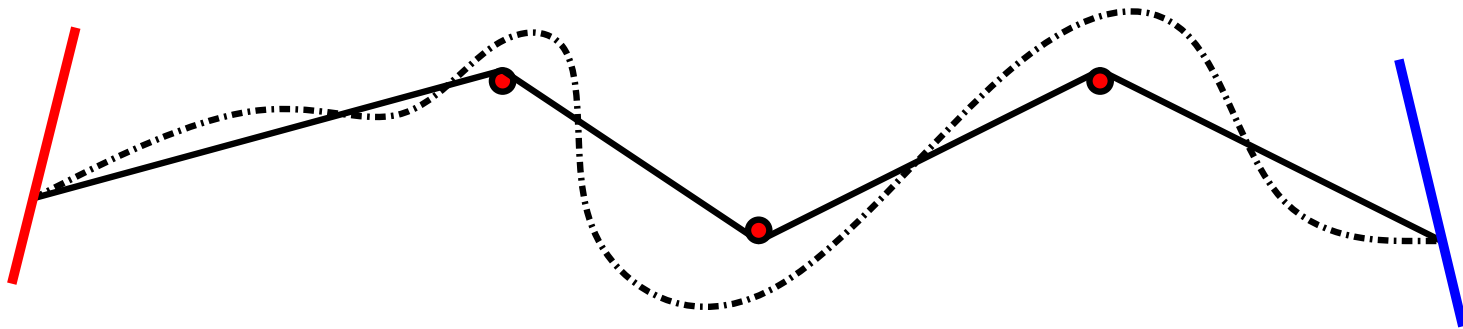
## Insight 1: Geodesic leashes

**Lemma:** There exists an optimum leash map such that the leash at every time is the shortest path in its homotopy class

# Insight 1: Geodesic leashes

**Lemma:** There exists an optimum leash map such that the leash at every time is the shortest path in its homotopy class

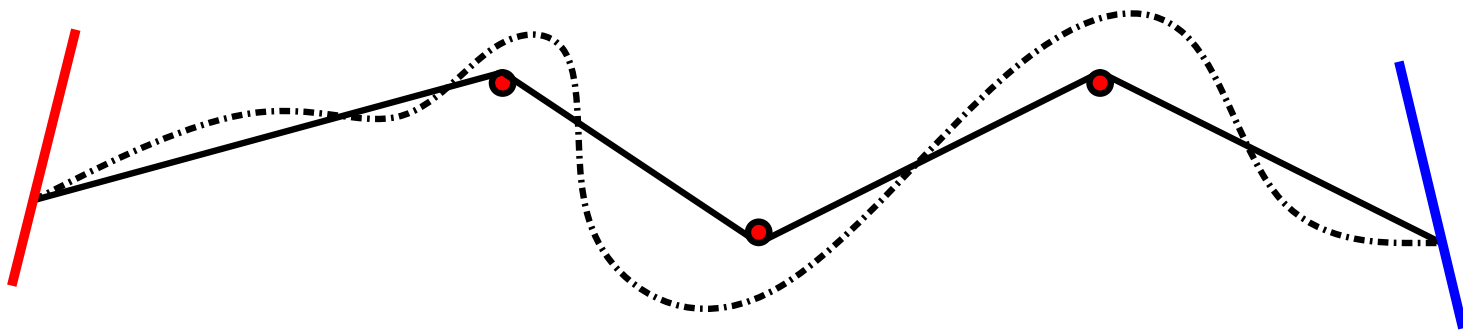
Hence, w.l.o.g.,  $\ell_h^*(\cdot, t)$  is the (unique) polygonal shortest path in homotopy class  $h$  between its endpoints



# Insight 1: Geodesic leashes

**Lemma:** There exists an optimum leash map such that the leash at every time is the shortest path in its homotopy class

Hence, w.l.o.g.,  $\ell_h^*(\cdot, t)$  is the (unique) polygonal shortest path in homotopy class  $h$  between its endpoints

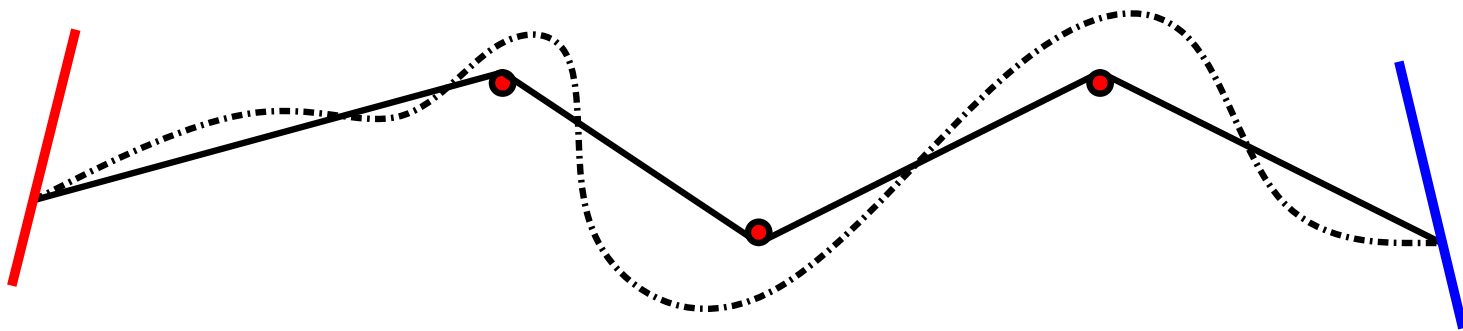


**Fact:** The geodesic between  $a \in A$  and  $b \in B$  moves continuously as  $a, b$  move continuously along their respective curves

# Insight 1: Geodesic leashes

**Lemma:** There exists an optimum leash map such that the leash at every time is the shortest path in its homotopy class

Hence, w.l.o.g.,  $\ell_h^*(\cdot, t)$  is the (unique) polygonal shortest path in homotopy class  $h$  between its endpoints



**Fact:** The geodesic between  $a \in A$  and  $b \in B$  moves continuously as  $a, b$  move continuously along their respective curves

*but* ... there are infinitely many geodesics with the same endpoints, one in each of infinitely many homotopy classes

## Insight 2: Minimal homotopy classes

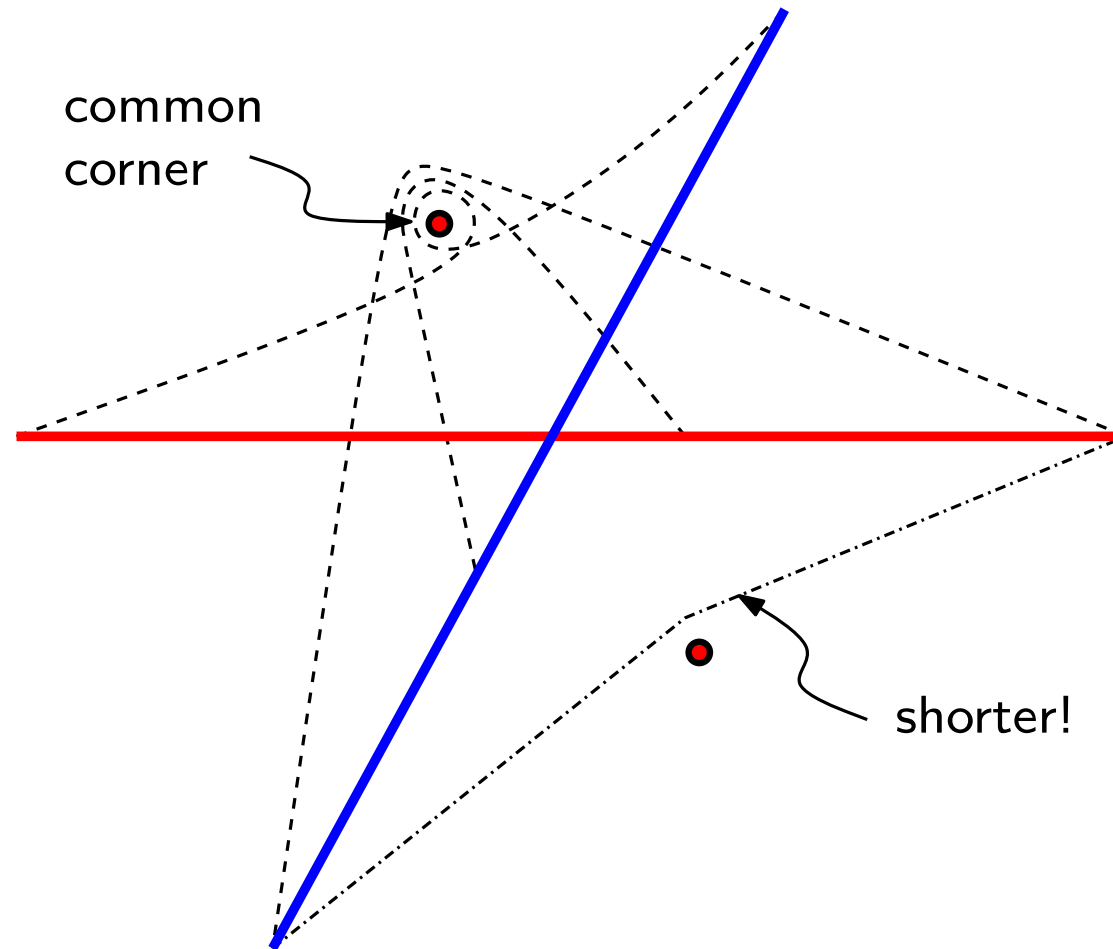
A homotopy class is **minimal** if it does not bend around obstacles unnecessarily

**Lemma:** There exists an optimum homotopy class that is minimal

Here onwards, we speak only of *minimal* homotopy classes

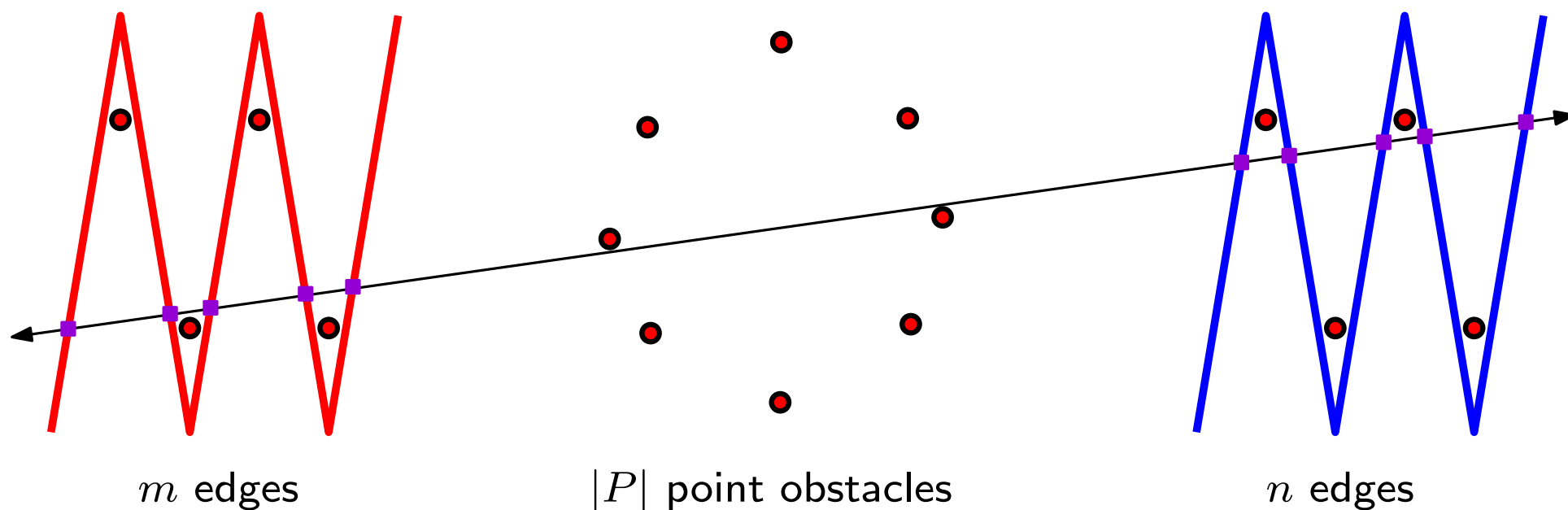
# Insight 3: No common corner

**Lemma:** There is no obstacle point  $p$  such that the leash always bends around  $p$  during any optimum leash map  $\ell^*$



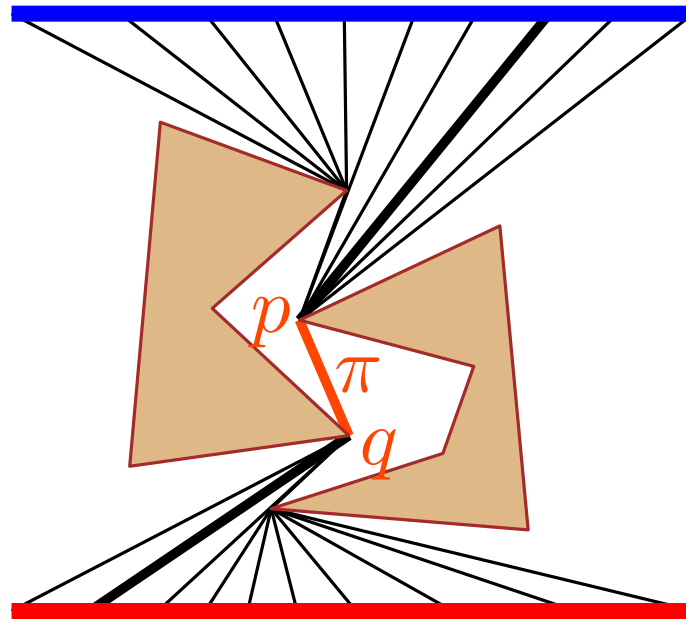
## Insight 4: Proper line segment

**Lemma:** The optimum homotopy class  $h^*$  must contain a **proper line segment**, i.e., a line segment joining  $A$  and  $B$  avoiding all obstacle points



## Insight 5: Pinned leash map

The optimum leash map  $\ell^*$  may be **pinned** at a common subpath  $\pi$ , i.e., a globally shortest  $p$ - $q$  path where  $p, q$  are obstacle boundary points



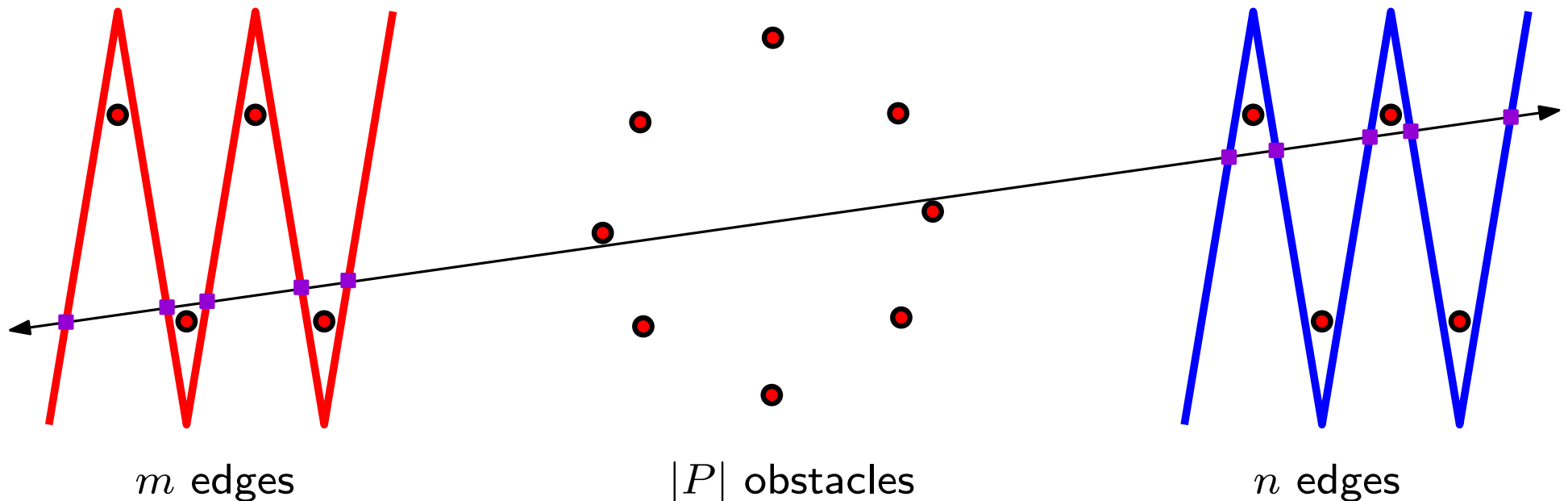
The optimum leash map  $\ell^*$  contains a **direct geodesic**

# The Algorithm

# Algorithm for point obstacles

List all candidate homotopy classes  $h$

There are  $O(mn|P|^2)$  *extremal* proper line segments, at least one in each homotopy class

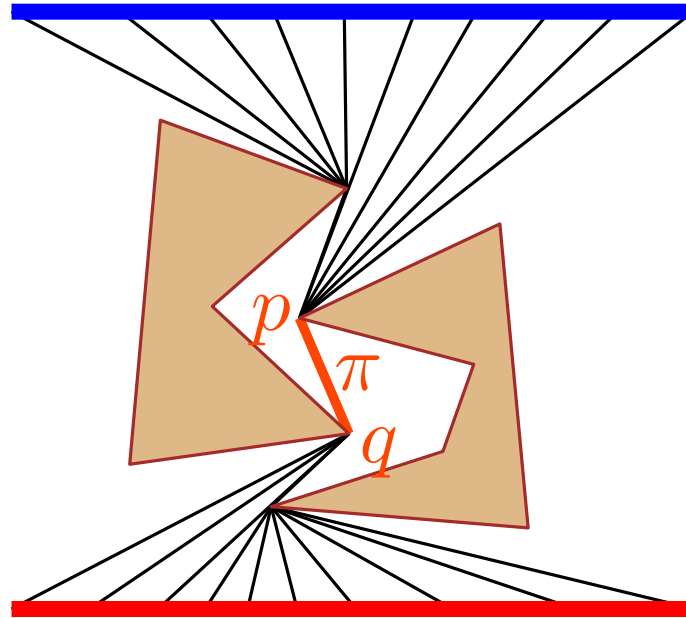


Compute  $F_h(A, B)$

In  $O(mn|P| \log mn|P|)$  time  
using **parametric search**

# Algorithm for polygonal obstacles

The optimum leash map  $\ell^*$  may be **pinned** at a common subpath  $\pi$ , i.e., a globally shortest  $p$ - $q$  path



Enumerate  $O(mn|P|^4)$  pinned homotopy classes  $h$

Compute  $F_h(A, B)$  in  $O(mn|P| \log mn|P|)$  time as before

# Summary

We defined the **homotopic Fréchet distance** between two curves in a general metric space

the correct generalization of Fréchet distance to arbitrary metric spaces

We gave a polynomial-time algorithm to compute the homotopic Fréchet distance between two polygonal curves  $A, B$  in the plane with point or polygonal obstacles

the punctured plane is the first metric space that we consider

# Open problem: On a convex polyhedron

Leash is not always a geodesic!

e.g., leash must have enough slack to cross over a vertex  
(a 'mountain')

**Challenge:** Characterize an optimum leash map

**Thank you!**

Gory details  
how to compute  
 $F_h(A, B)$

# Computing $F_h$

**Decision problem:** Given a real  $d \geq 0$ , is  $F_h(A, B) \leq d$ ?

# Computing $F_h$

**Decision problem:** Given a real  $d \geq 0$ , is  $F_h(A, B) \leq d$ ?

**Observation:** There are polynomially many **critical values** of  $d$  at which the answer may change from 'no' to 'yes'

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq d_{i+1} \leq d_{i+2} \leq \dots$$

# Computing $F_h$

**Decision problem:** Given a real  $d \geq 0$ , is  $F_h(A, B) \leq d$ ?

**Observation:** There are polynomially many **critical values** of  $d$  at which the answer may change from 'no' to 'yes'

$$\begin{array}{ccccccccccc} d_1 & \leq & d_2 & \leq & \dots & \leq & d_{i-1} & \leq & d^* & \leq & d_{i+1} & \leq & d_{i+2} & \leq & \dots \\ \times & & \times & & \times & & \times & & \checkmark & & \checkmark & & \checkmark & & \checkmark \end{array}$$

# Computing $F_h$

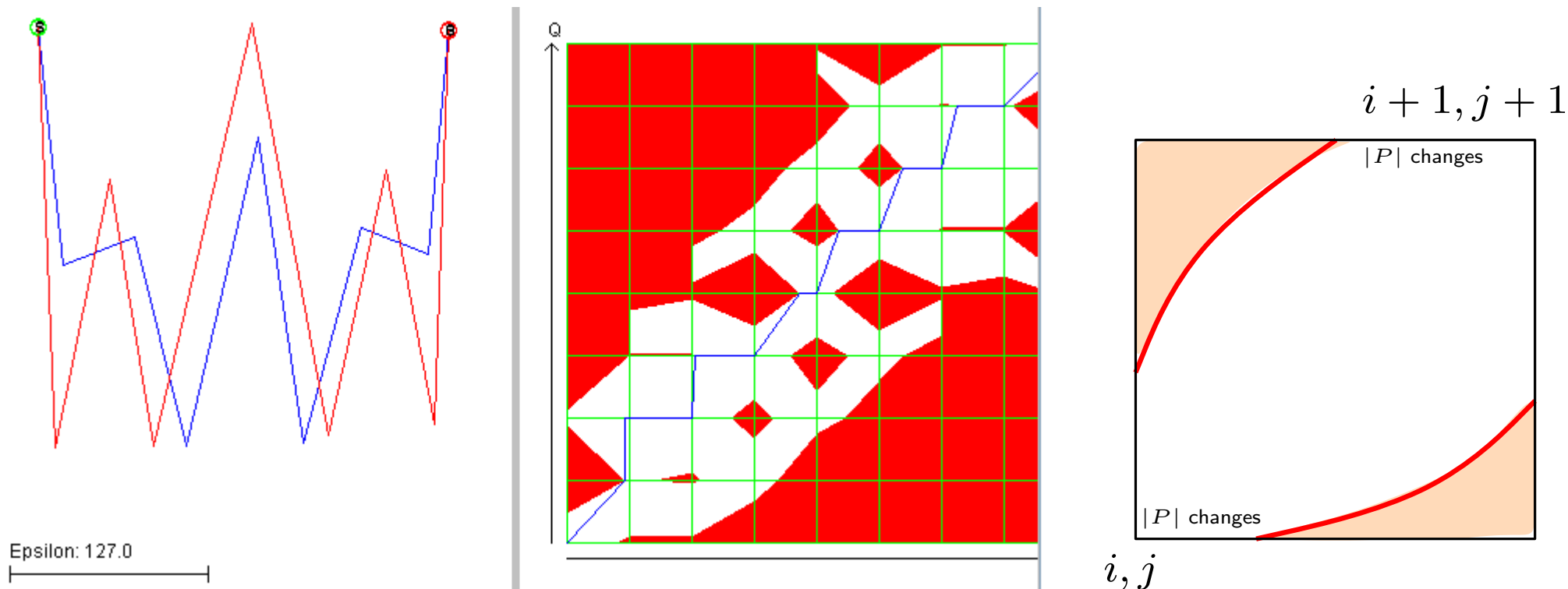
**Decision problem:** Given a real  $d \geq 0$ , is  $F_h(A, B) \leq d$ ?

**Observation:** There are polynomially many **critical values** of  $d$  at which the answer may change from 'no' to 'yes'

$$\begin{array}{ccccccccccc} d_1 & \leq & d_2 & \leq & \dots & \leq & d_{i-1} & \leq & d^* & \leq & d_{i+1} & \leq & d_{i+2} & \leq & \dots \\ \times & & \times & & \times & & \times & & \checkmark & & \checkmark & & \checkmark & & \checkmark \\ & & & & & & & & \downarrow & & & & & & \\ & & & & & & & & \boxed{F_h(A, B)} & & & & & & \end{array}$$

**Goal:** Find the smallest critical value  $d$  for which the answer above is 'yes'

Is  $F_h \leq d$ ?



<http://www.cim.mcgill.ca/~stephane/cs507/Project.html>, Stéphane Pelletier, 2002

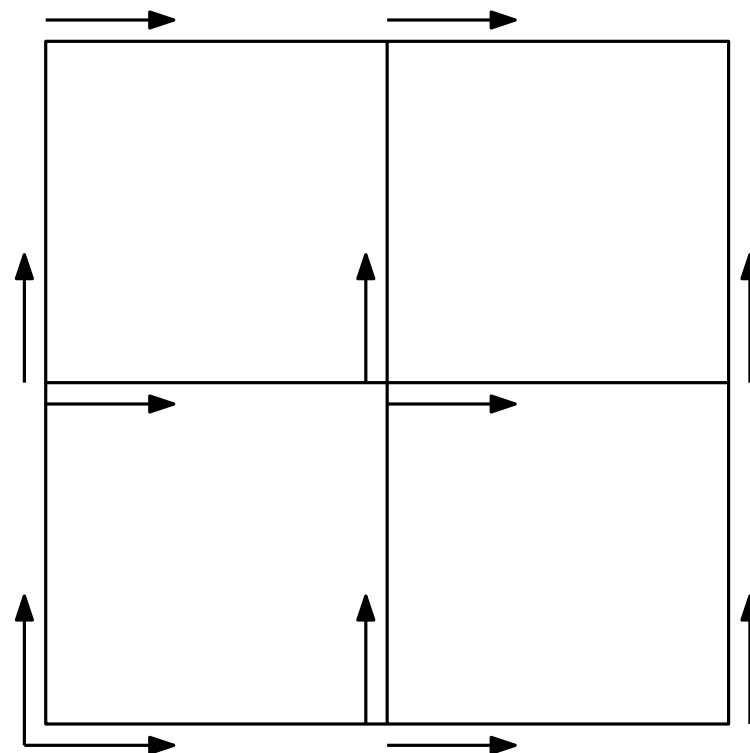
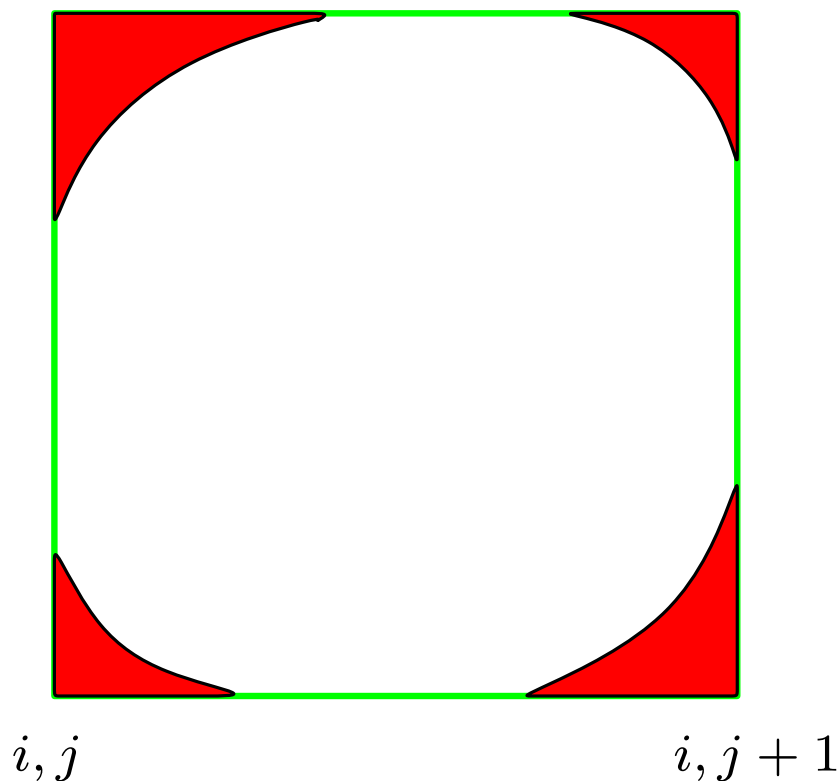
Is there a monotone path from  $(0, 0)$  to  $(m, n)$  in free space?

**Key lemma:** Free space within each cell  $C_{ij}$  is convex

i.e., least length is a convex function along any monotone path through  $C_{ij}$   
(think **hourglasses**)

# Computing free space

$i + 1, j$   $i + 1, j + 1$



Given the shortest path between  $a_i$  and  $b_j$ , compute:

- (1) the interval  $b \in [b_j, b_{j+1}]$  such that  $\text{dist}(a_i, b) \leq d$
- (2) the interval  $a \in [a_i, a_{i+1}]$  such that  $\text{dist}(b_j, a) \leq d$
- (3) shortest paths  $(a_i, b_{j+1})$ ,  $(a_{i+1}, b_j)$ , and  $(a_{i+1}, b_{j+1})$

# Funnels

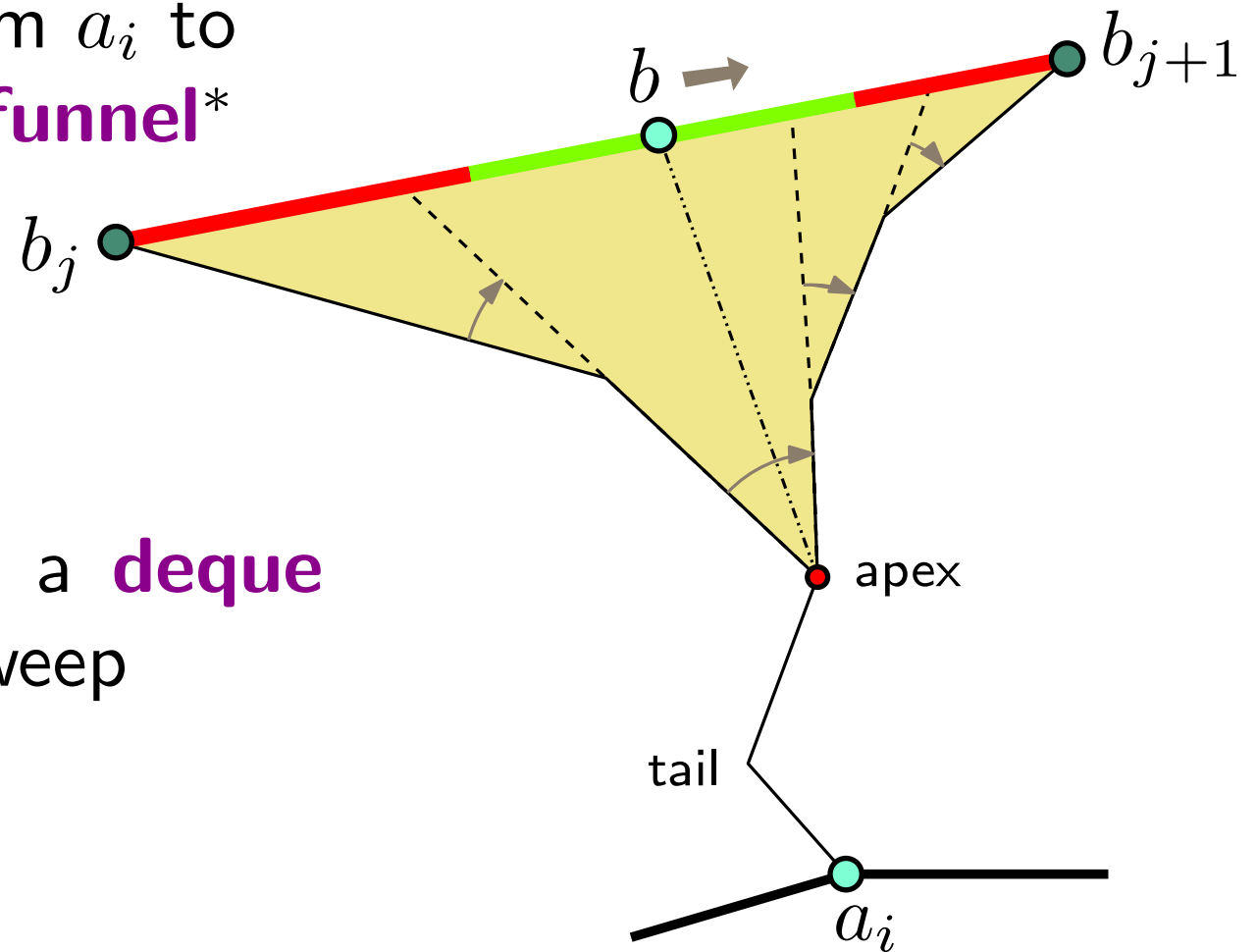
Shortest paths from  $a_i$  to  $[b_j, b_{j+1}]$  define a **funnel**\*

*\*in the universal cover*

# Funnels

Shortest paths from  $a_i$  to  $[b_j, b_{j+1}]$  define a **funnel**\*

*\*in the universal cover*

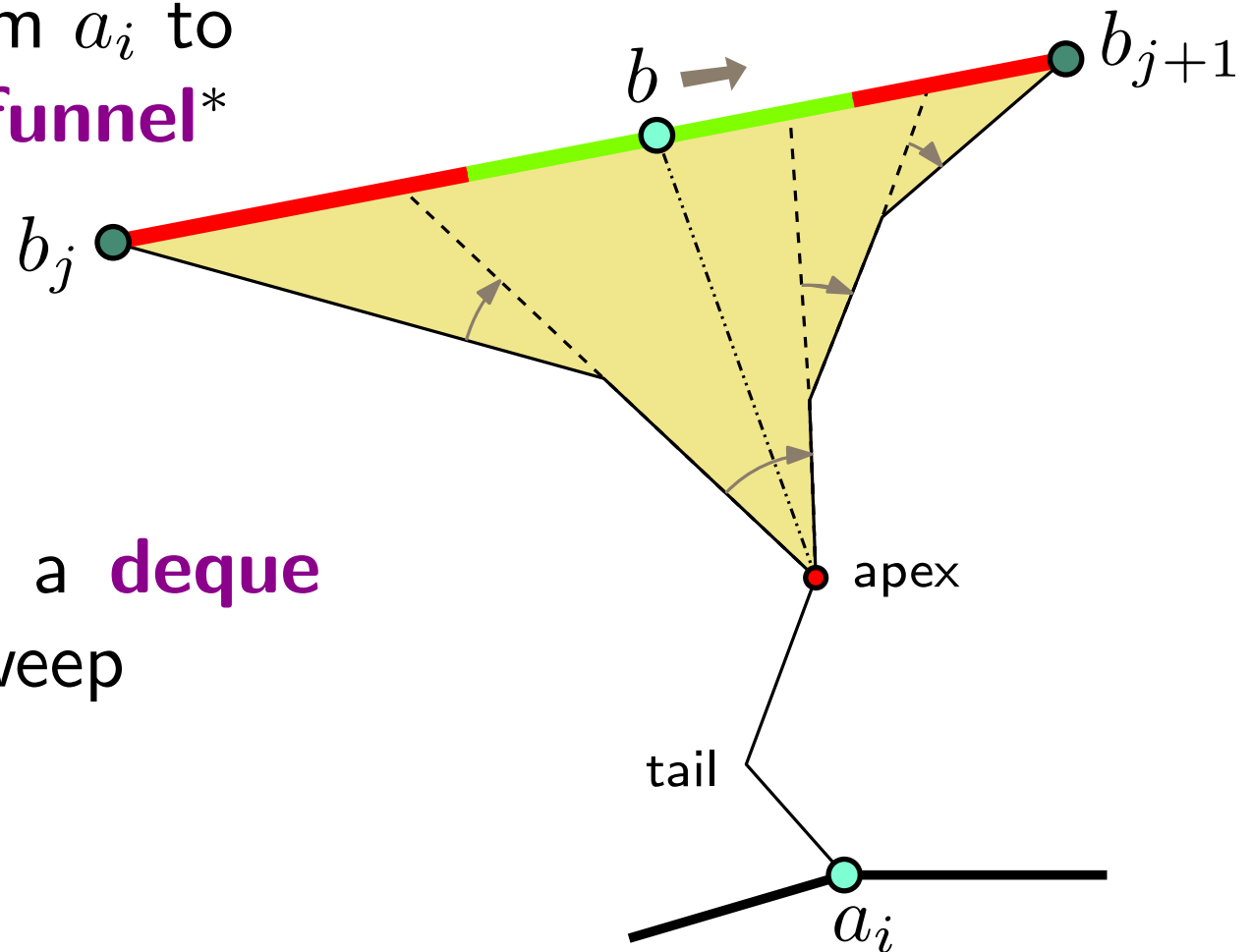


Leash evolves like a **deque**  
as its endpoints sweep

# Funnels

Shortest paths from  $a_i$  to  $[b_j, b_{j+1}]$  define a **funnel**\*

*\*in the universal cover*



Leash evolves like a **deque** as its endpoints sweep

In  $O(|P| \log |P|)$  time, build a **funnel** data structure of size  $O(|P|)$ , using a **deque**, such that each free interval can be computed in  $O(\log |P|)$  time

# Computing $F_h$

**Decision algorithm:** Given a real  $d \geq 0$ , is  $F_h(A, B) \leq d$ ?  
... in  $O(mn \log |P|)$  time

# Computing $F_h$

**Decision algorithm:** Given a real  $d \geq 0$ , is  $F_h(A, B) \leq d$ ?  
... in  $O(mn \log |P|)$  time

**Observation:** There are polynomially many **critical values** of  $d$  at which the answer may change from 'no' to 'yes'

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq d_{i+1} \leq d_{i+2} \leq \dots$$

# Computing $F_h$

**Decision algorithm:** Given a real  $d \geq 0$ , is  $F_h(A, B) \leq d$ ?  
... in  $O(mn \log |P|)$  time

**Observation:** There are polynomially many **critical values** of  $d$  at which the answer may change from 'no' to 'yes'

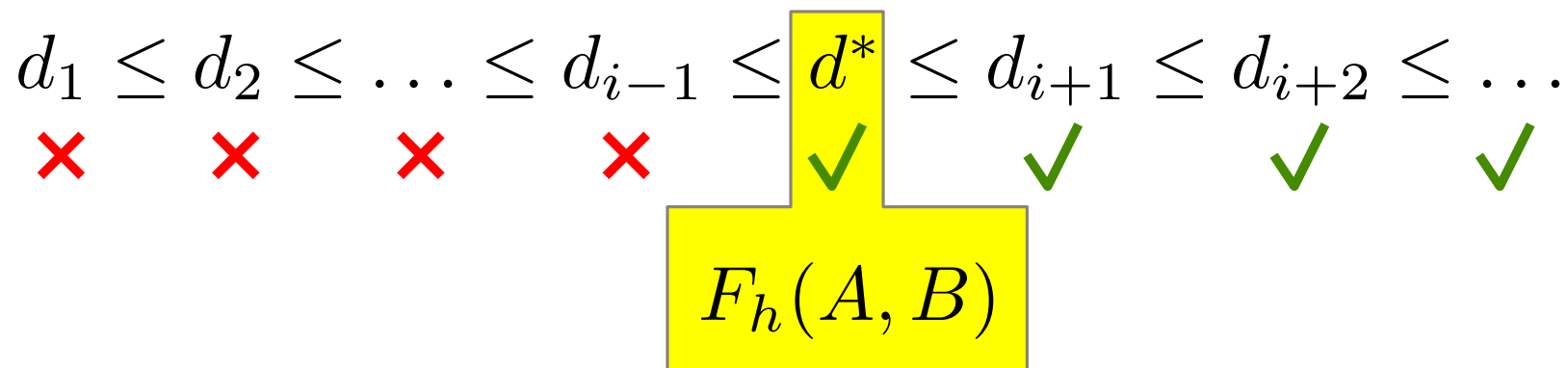
$$\begin{array}{cccccccc} d_1 & \leq & d_2 & \leq & \dots & \leq & d_{i-1} & \leq & d^* & \leq & d_{i+1} & \leq & d_{i+2} & \leq & \dots \\ \times & & \times & & \times & & \times & & \checkmark & & \checkmark & & \checkmark & & \checkmark \end{array}$$

# Computing $F_h$

**Decision algorithm:** Given a real  $d \geq 0$ , is  $F_h(A, B) \leq d$ ?  
... in  $O(mn \log |P|)$  time

**Observation:** There are polynomially many **critical values** of  $d$  at which the answer may change from 'no' to 'yes'

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq d_{i+1} \leq d_{i+2} \leq \dots$$



**Goal:** Find the smallest critical value  $d$  for which the answer above is 'yes'

Let  $A_s$  be an algorithm to decide, given a critical value  $d_i$ , whether  $F_h(A, B) \leq d_i$ , with running time  $O(T_s)$

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq \dots \leq d_j \leq \dots$$

✗   ✗   ✗   ✗   ✓   ✓   ✓   ✓

# Parametric search

[Megiddo '83]

Let  $A_s$  be an algorithm to decide, given a critical value  $d_i$ , whether  $F_h(A, B) \leq d_i$ , with running time  $O(T_s)$

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq \dots \leq d_j \leq \dots$$

✗   ✗   ✗   ✗   ✓   ✓   ✓   ✓

We simulate  $A_s$  on input  $d^*$ , with  $d^*$  as a symbolic variable

Let  $A_s$  be an algorithm to decide, given a critical value  $d_i$ , whether  $F_h(A, B) \leq d_i$ , with running time  $O(T_s)$

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq \dots \leq d_j \leq \dots$$

✗   ✗   ✗   ✗   ✓   ✓   ✓   ✓

We simulate  $A_s$  on input  $d^*$ , with  $d^*$  as a symbolic variable

The control flow of  $A_s$  depends on comparisons of the form  $d^* \leq d_j$  where  $d_j$  is a critical value

*Each  $d_j$  is a distance, i.e., a quadratic function of input coordinates.*

Let  $A_s$  be an algorithm to decide, given a critical value  $d_i$ , whether  $F_h(A, B) \leq d_i$ , with running time  $O(T_s)$

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq \dots \leq d_j \leq \dots$$

✗   ✗   ✗   ✗   ✓   ✓   ✓   ✓

We simulate  $A_s$  on input  $d^*$ , with  $d^*$  as a symbolic variable

The control flow of  $A_s$  depends on comparisons of the form  $d^* \leq d_j$  where  $d_j$  is a critical value

*Each  $d_j$  is a distance, i.e., a quadratic function of input coordinates.*

$d^* \leq d_j$ ? Run  $A_s$  on input  $d_j$ , in  $O(T_s)$  time

Let  $A_s$  be an algorithm to decide, given a critical value  $d_i$ , whether  $F_h(A, B) \leq d_i$ , with running time  $O(T_s)$

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq \dots \leq d_j \leq \dots$$

×   ×   ×   ×   ✓   ✓   ✓   ✓

We simulate  $A_s$  on input  $d^*$ , with  $d^*$  as a symbolic variable

The control flow of  $A_s$  depends on comparisons of the form  $d^* \leq d_j$  where  $d_j$  is a critical value

*Each  $d_j$  is a distance, i.e., a quadratic function of input coordinates.*

$d^* \leq d_j$ ? Run  $A_s$  on input  $d_j$ , in  $O(T_s)$  time

Total running time =  $O(T_s^2)$

# Parametric search on steroids

[Megiddo '83]

Let  $A_p$  be a **parallel** algorithm to decide, given  $d_i$ , whether  $F_h(A, B) \leq d_i$ , with parallel running time  $O(T_p)$  on  $k$  processors

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq \dots \leq d_j \leq \dots$$

×   ×   ×   ×   ✓   ✓   ✓   ✓

# Parametric search on steroids

[Megiddo '83]

Let  $A_p$  be a **parallel** algorithm to decide, given  $d_i$ , whether  $F_h(A, B) \leq d_i$ , with parallel running time  $O(T_p)$  on  $k$  processors

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq \dots \leq d_j \leq \dots$$

×   ×   ×   ×   ✓   ✓   ✓   ✓

We simulate  $A_p$  **sequentially** on input  $d^*$ , with  $d^*$  as a symbolic variable

# Parametric search on steroids

[Megiddo '83]

Let  $A_p$  be a **parallel** algorithm to decide, given  $d_i$ , whether  $F_h(A, B) \leq d_i$ , with parallel running time  $O(T_p)$  on  $k$  processors

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq \dots \leq d_j \leq \dots$$

✗   ✗   ✗   ✗   ✓   ✓   ✓   ✓

We simulate  $A_p$  **sequentially** on input  $d^*$ , with  $d^*$  as a symbolic variable

The control flow of  $A_p$  depends on comparisons of the form  $d^* \leq d_j$  where  $d_j$  is a critical value

# Parametric search on steroids

[Megiddo '83]

Let  $A_p$  be a **parallel** algorithm to decide, given  $d_i$ , whether  $F_h(A, B) \leq d_i$ , with parallel running time  $O(T_p)$  on  $k$  processors

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq \dots \leq d_j \leq \dots$$

×   ×   ×   ×   ✓   ✓   ✓   ✓

We simulate  $A_p$  **sequentially** on input  $d^*$ , with  $d^*$  as a symbolic variable

The control flow of  $A_p$  depends on comparisons of the form  $d^* \leq d_j$  where  $d_j$  is a critical value

In each parallel stage, do binary search on  $O(k)$  values  $d_j$

# Parametric search on steroids

[Megiddo '83]

Let  $A_p$  be a **parallel** algorithm to decide, given  $d_i$ , whether  $F_h(A, B) \leq d_i$ , with parallel running time  $O(T_p)$  on  $k$  processors

$$d_1 \leq d_2 \leq \dots \leq d_{i-1} \leq d^* \leq \dots \leq d_j \leq \dots$$

×   ×   ×   ×   ✓   ✓   ✓   ✓

We simulate  $A_p$  **sequentially** on input  $d^*$ , with  $d^*$  as a symbolic variable

The control flow of  $A_p$  depends on comparisons of the form  $d^* \leq d_j$  where  $d_j$  is a critical value

In each parallel stage, do binary search on  $O(k)$  values  $d_j$

Total running time =  $O(T_s T_p \log k + k T_p)$  *better than  $O(T_s^2)$*

## How we use parametric search

Let  $A_p$  be a **parallel sorting algorithm**  
(e.g., Cole's parallel merge sort)

## How we use parametric search

Let  $A_p$  be a **parallel sorting algorithm** (e.g., Cole's parallel merge sort)

We use  $A_p$  to sort  $O(mn)$  critical distances with parallel running time  $O(\log mn)$

We simulate  $A_p$  **sequentially** on input  $d^*$ , with  $d^*$  as a symbolic variable

## How we use parametric search

Let  $A_p$  be a **parallel sorting algorithm** (e.g., Cole's parallel merge sort)

We use  $A_p$  to sort  $O(mn)$  critical distances with parallel running time  $O(\log mn)$

We simulate  $A_p$  **sequentially** on input  $d^*$ , with  $d^*$  as a symbolic variable

The control flow of  $A_p$  depends on comparisons of the form  $d^* \leq d_j$  where  $d_j$  is a critical value

## How we use parametric search

Let  $A_p$  be a **parallel sorting algorithm** (e.g., Cole's parallel merge sort)

We use  $A_p$  to sort  $O(mn)$  critical distances with parallel running time  $O(\log mn)$

We simulate  $A_p$  **sequentially** on input  $d^*$ , with  $d^*$  as a symbolic variable

The control flow of  $A_p$  depends on comparisons of the form  $d^* \leq d_j$  where  $d_j$  is a critical value

To simulate each parallel stage, we resolve all  $O(mn)$  comparisons  $d^* \leq d_j$  by computing  $O(mn|P|)$  critical intervals and locating each  $d_j$  in some interval

## How we use parametric search

Let  $A_p$  be a **parallel sorting algorithm** (e.g., Cole's parallel merge sort)

We use  $A_p$  to sort  $O(mn)$  critical distances with parallel running time  $O(\log mn)$

We simulate  $A_p$  **sequentially** on input  $d^*$ , with  $d^*$  as a symbolic variable

The control flow of  $A_p$  depends on comparisons of the form  $d^* \leq d_j$  where  $d_j$  is a critical value

To simulate each parallel stage, we resolve all  $O(mn)$  comparisons  $d^* \leq d_j$  by computing  $O(mn|P|)$  critical intervals and locating each  $d_j$  in some interval

Total running time =  $O(N^3 \log N)$

# Extra slides

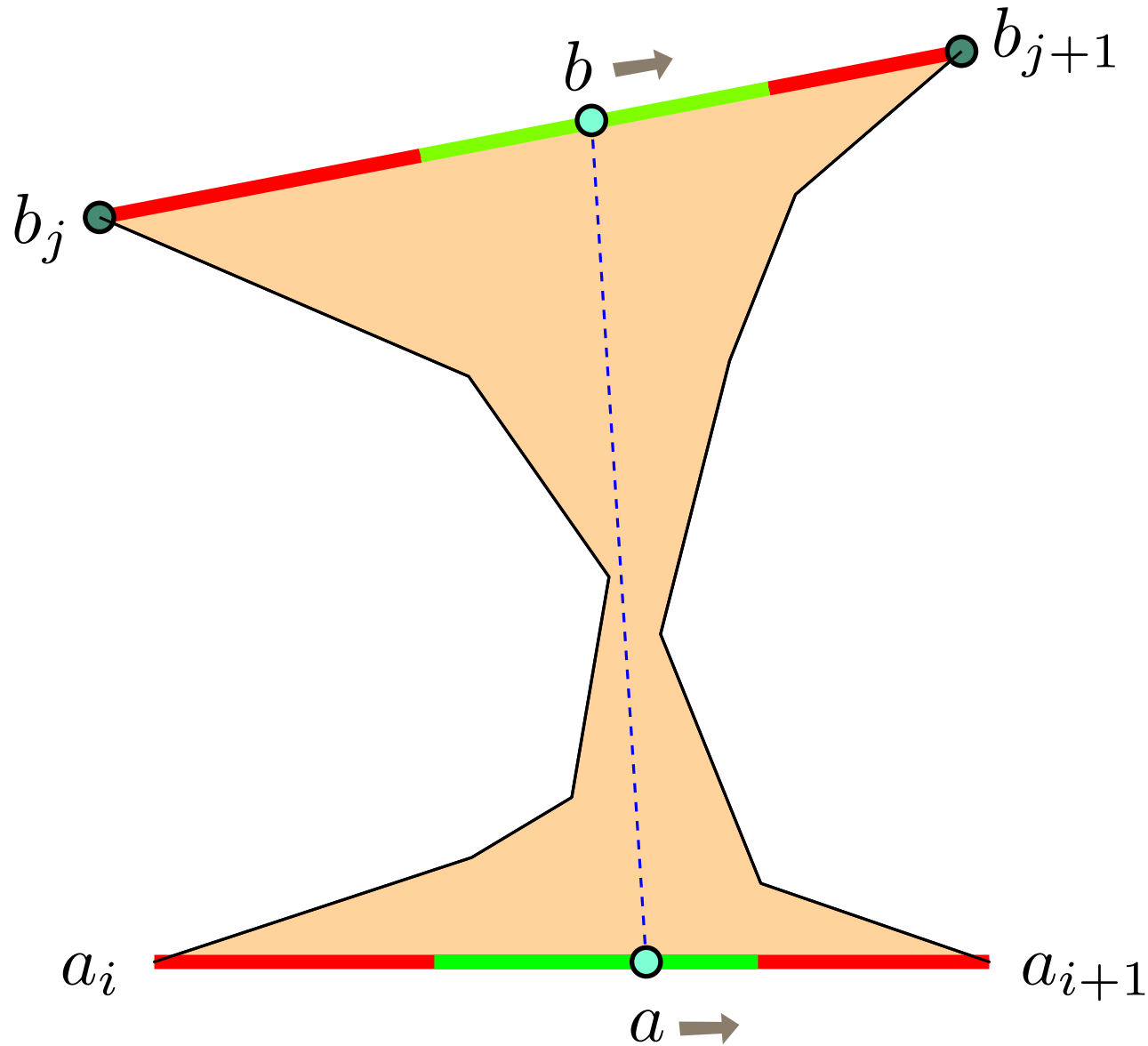
## Universal cover

We compute a constrained Delaunay triangulation  $T$  of the obstacles and curves  $A, B$

The universal cover  $\hat{E}$  of  $\mathbb{E}^2 \setminus P$  is an infinite tree of Euclidean triangles, where copies of adjacent triangle of  $T$  are glued to each other *ad infinitum*

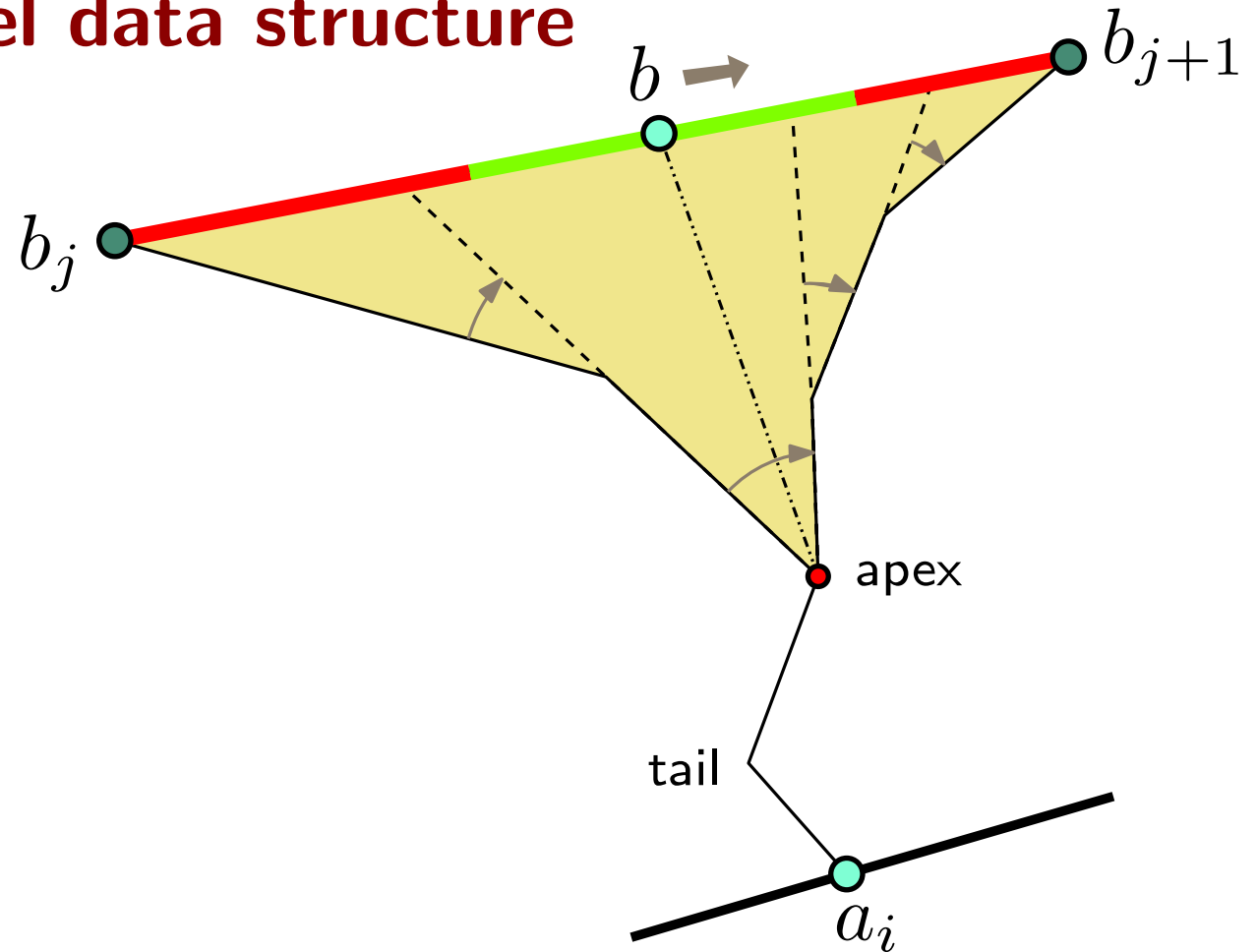
To compute a funnel, we construct a portion of  $\hat{E}$  by tracing all the triangles intersected by a geodesic between apex and base

# Intuition why geodesic distance is convex



Suppose  $A = [a_i, a_{i+1}]$  and  $B = [b_j, b_{j+1}]$ ; then, the leash between  $a \in A$  and  $b \in B$  sweeps an **hourglass**\* in the

# Building the funnel data structure



Each obstacle vertex is pushed and popped at most once from the leash

Thus, funnel has  $O(|P|)$  complexity, computed in  $O(|P| \log |P|)$  time

# Minimal homotopy classes

Informally, a homotopy class is **minimal** if it does not wind around obstacles *unnecessarily*

# Minimal homotopy classes

Informally, a homotopy class is **minimal** if it does not wind around obstacles *unnecessarily*

**Lemma:** There exists an optimum homotopy class that is minimal

# Minimal homotopy classes

Informally, a homotopy class is **minimal** if it does not wind around obstacles *unnecessarily*

**Lemma:** There exists an optimum homotopy class that is minimal

**For point obstacles:**

**Lemma:** A homotopy class is minimal if and only if it contains a proper line segment

# Minimal homotopy classes

Informally, a homotopy class is **minimal** if it does not wind around obstacles *unnecessarily*

**Lemma:** There exists an optimum homotopy class that is minimal

**For point obstacles:**

**Lemma:** A homotopy class is minimal if and only if it contains a proper line segment

**For general obstacles:**

**Lemma:** If a homotopy class is minimal, then it contains either a proper line segment or a pinned subpath

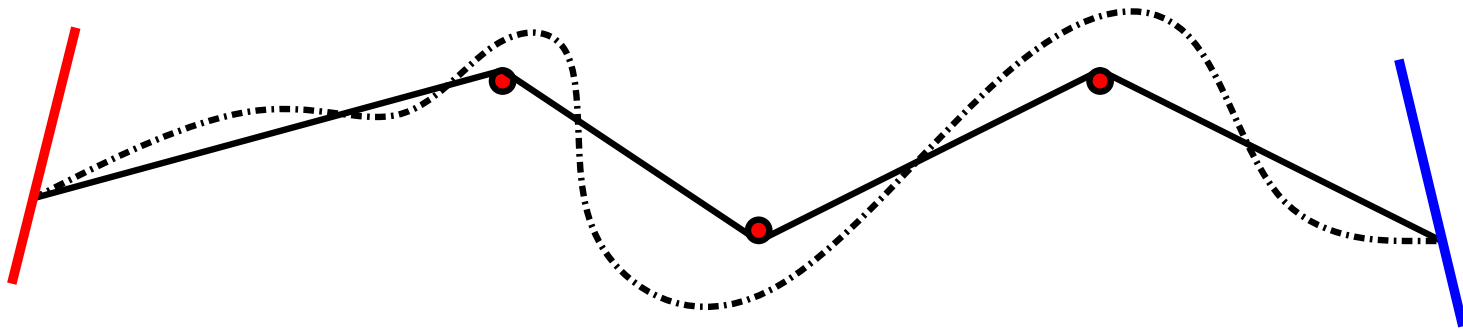
# Geodesic leashes

**Lemma:** There exists an optimum leash map such that the leash at every time is the shortest path in its homotopy class

# Geodesic leashes

**Lemma:** There exists an optimum leash map such that the leash at every time is the shortest path in its homotopy class

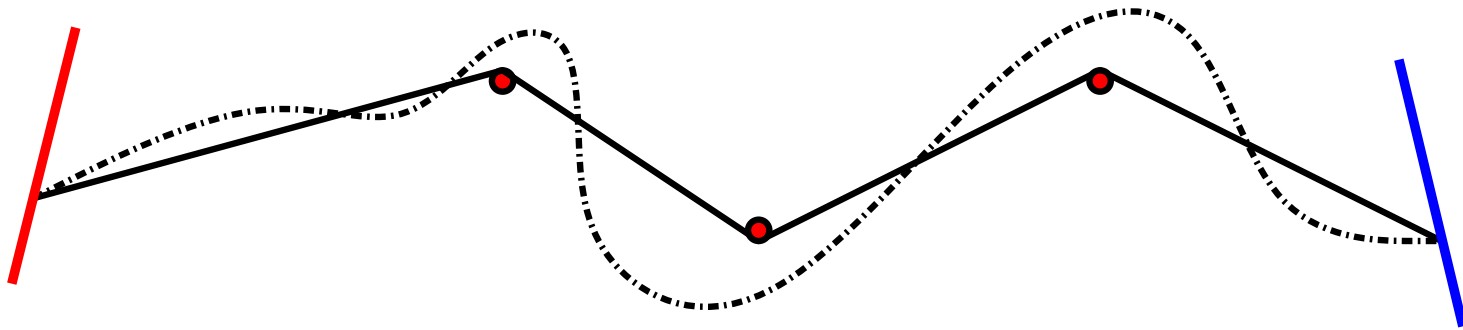
Hence, w.l.o.g.,  $\ell_h^*(\cdot, t)$  is the (unique<sup>\*</sup>) shortest path in homotopy class  $h$  between its endpoints



# Geodesic leashes

**Lemma:** There exists an optimum leash map such that the leash at every time is the shortest path in its homotopy class

Hence, w.l.o.g.,  $\ell_h^*(\cdot, t)$  is the (unique\*) shortest path in homotopy class  $h$  between its endpoints



*\* We allow the leash to pass through obstacle points. A turning angle at every obstacle point uniquely identifies the homotopy class of the leash. Now, unique shortest paths exist in every homotopy class.*