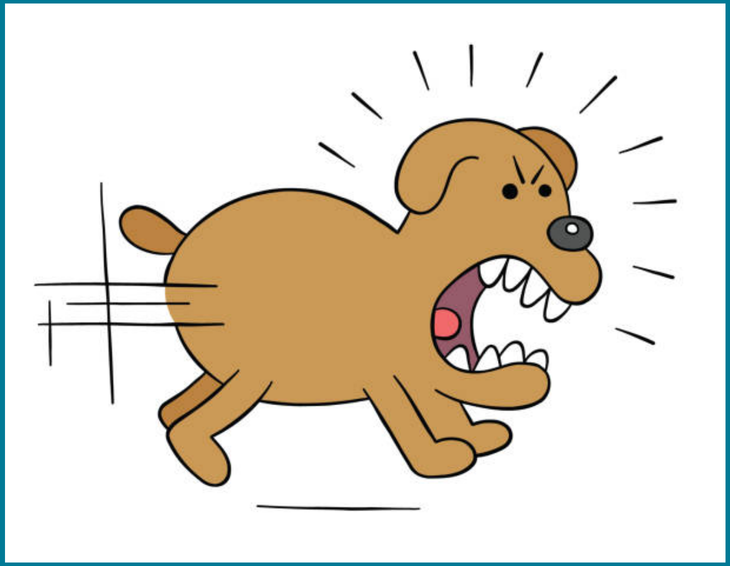
Barking Dogs:

A Fréchet distance variant for detour detection

Ivor van der Hoog, Fabian Klute,
Irene Parada, Patrick Schnider

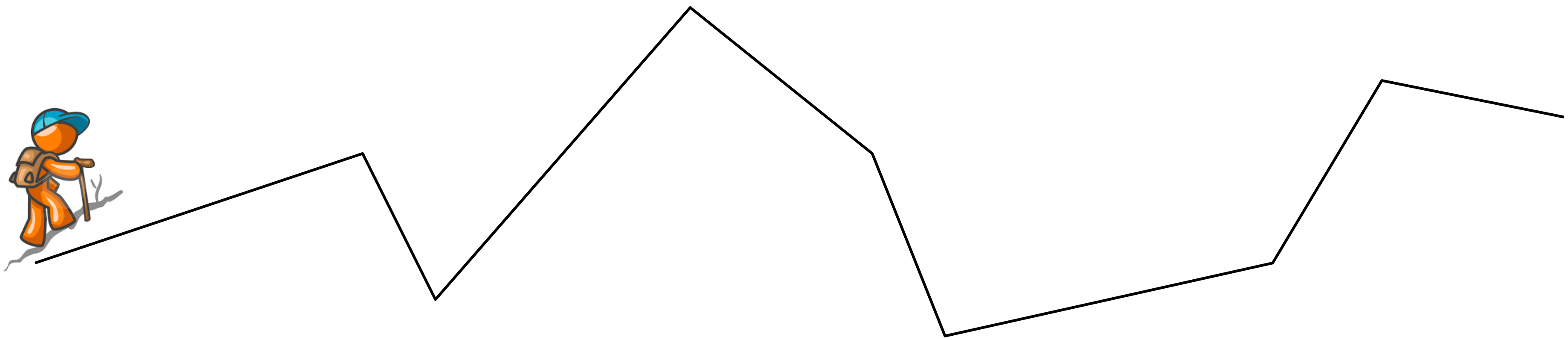
EuroCG 2024



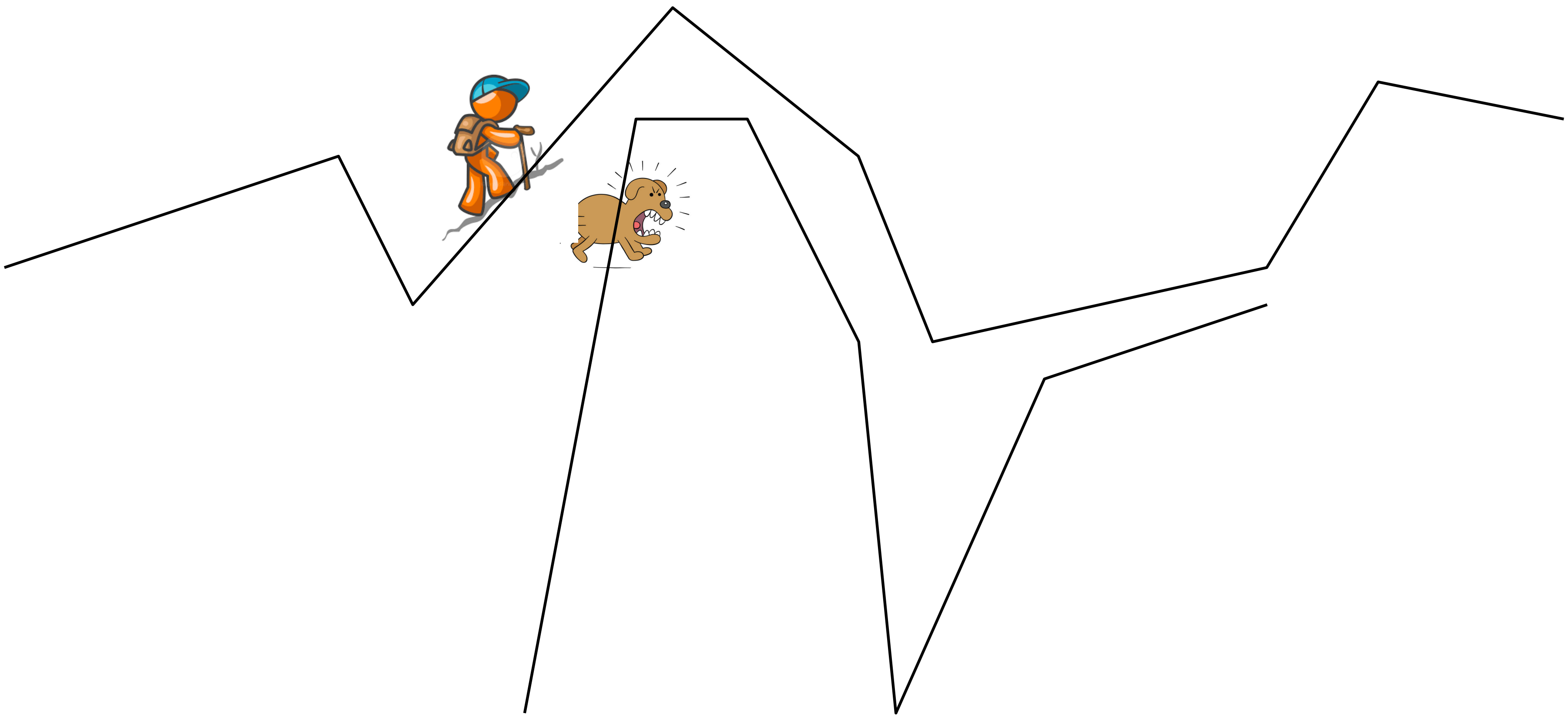
A hike in Perugia

A hike in Perugia

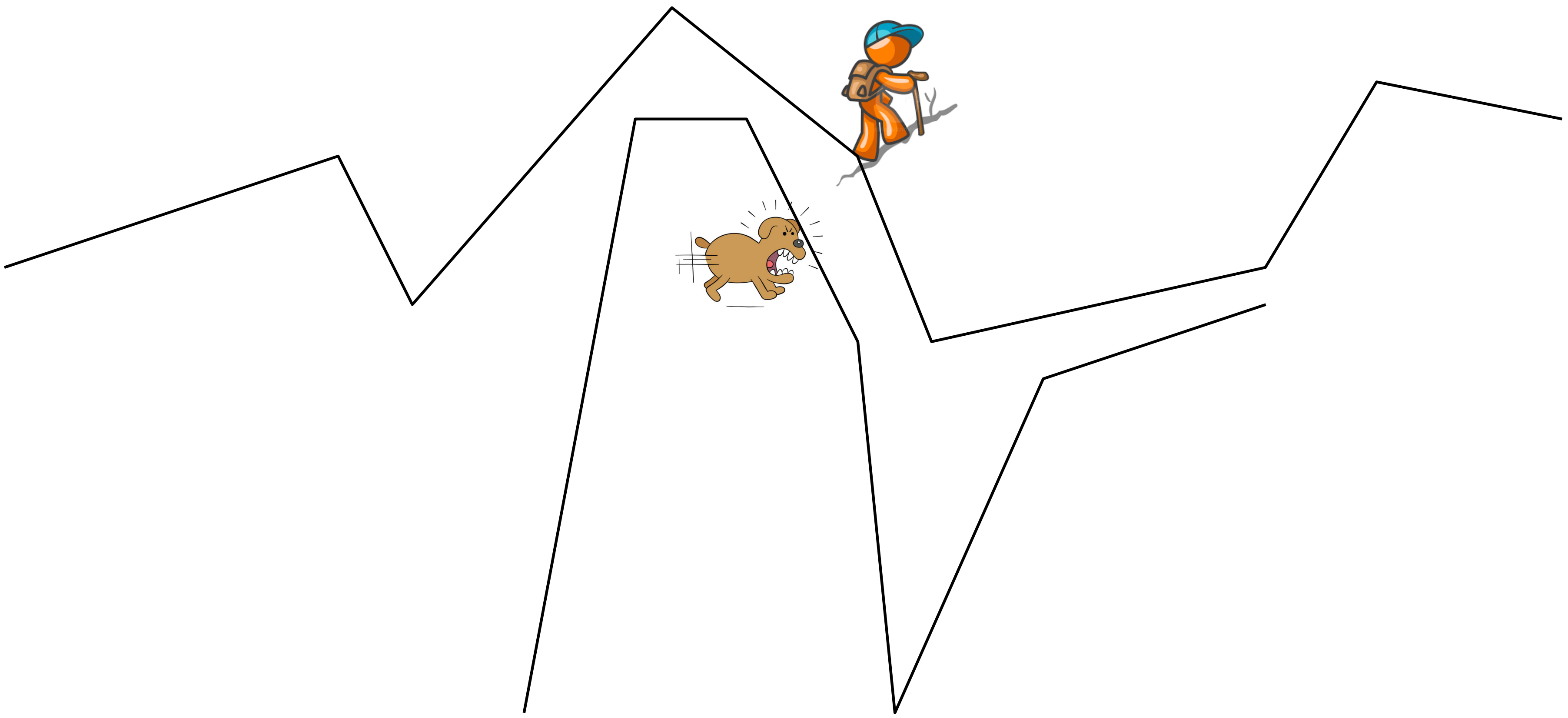
A hike in Perugia



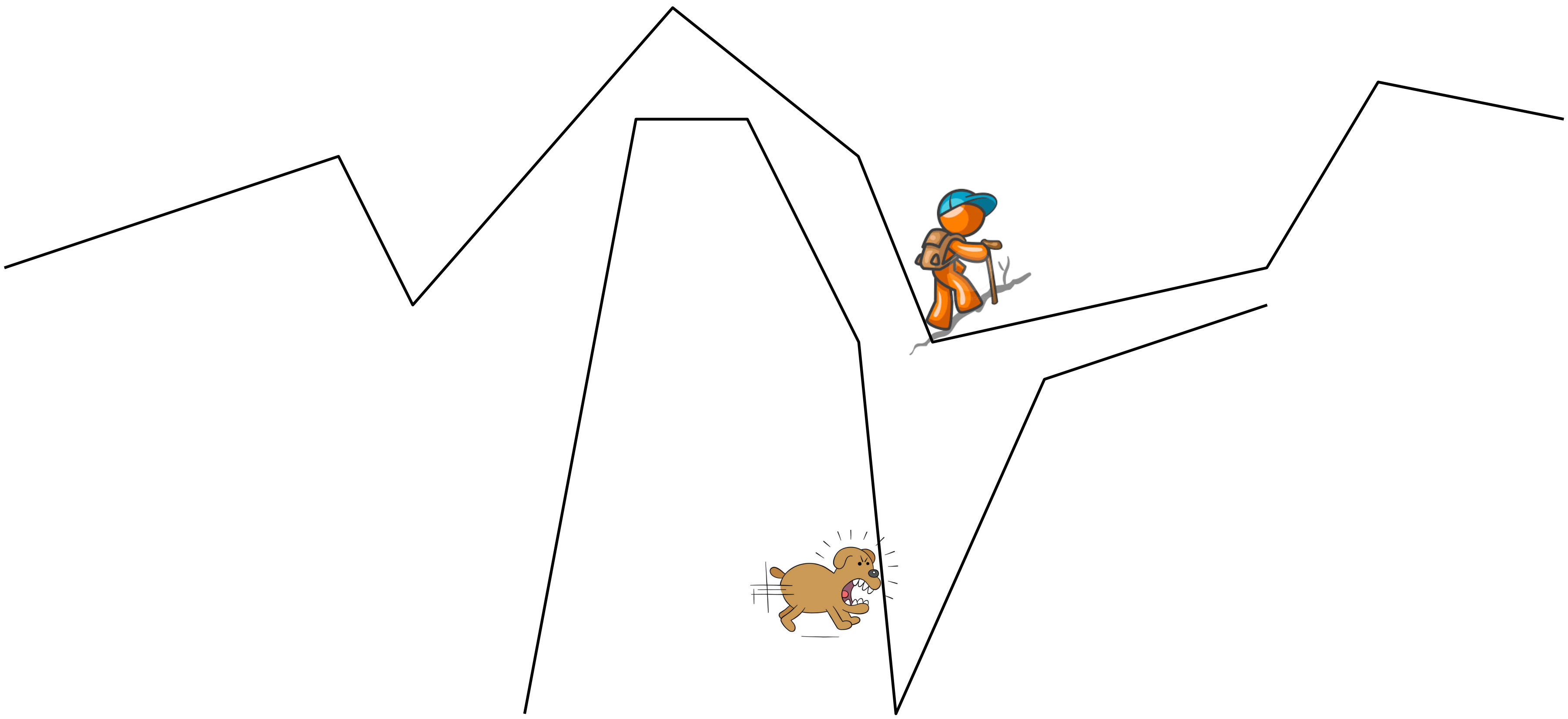
A hike in Perugia



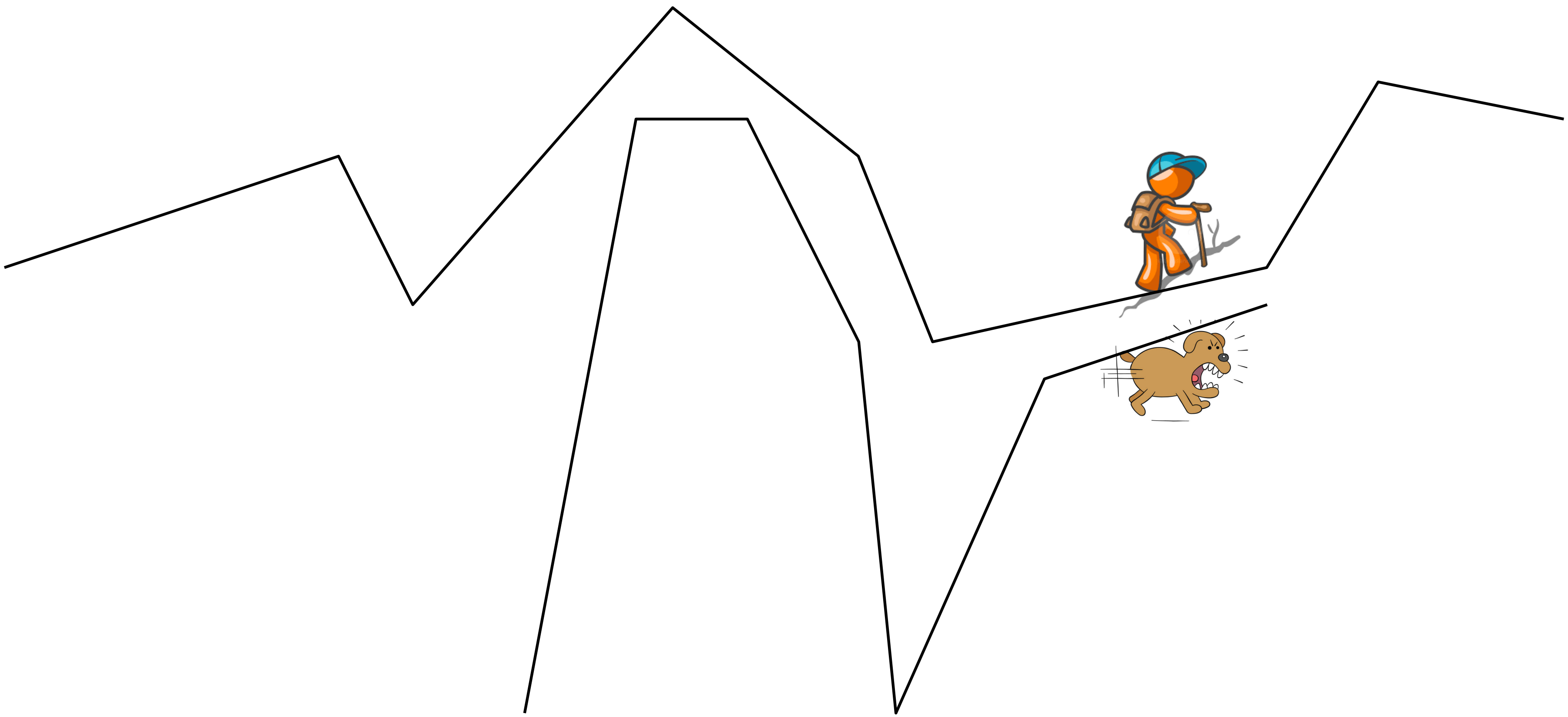
A hike in Perugia



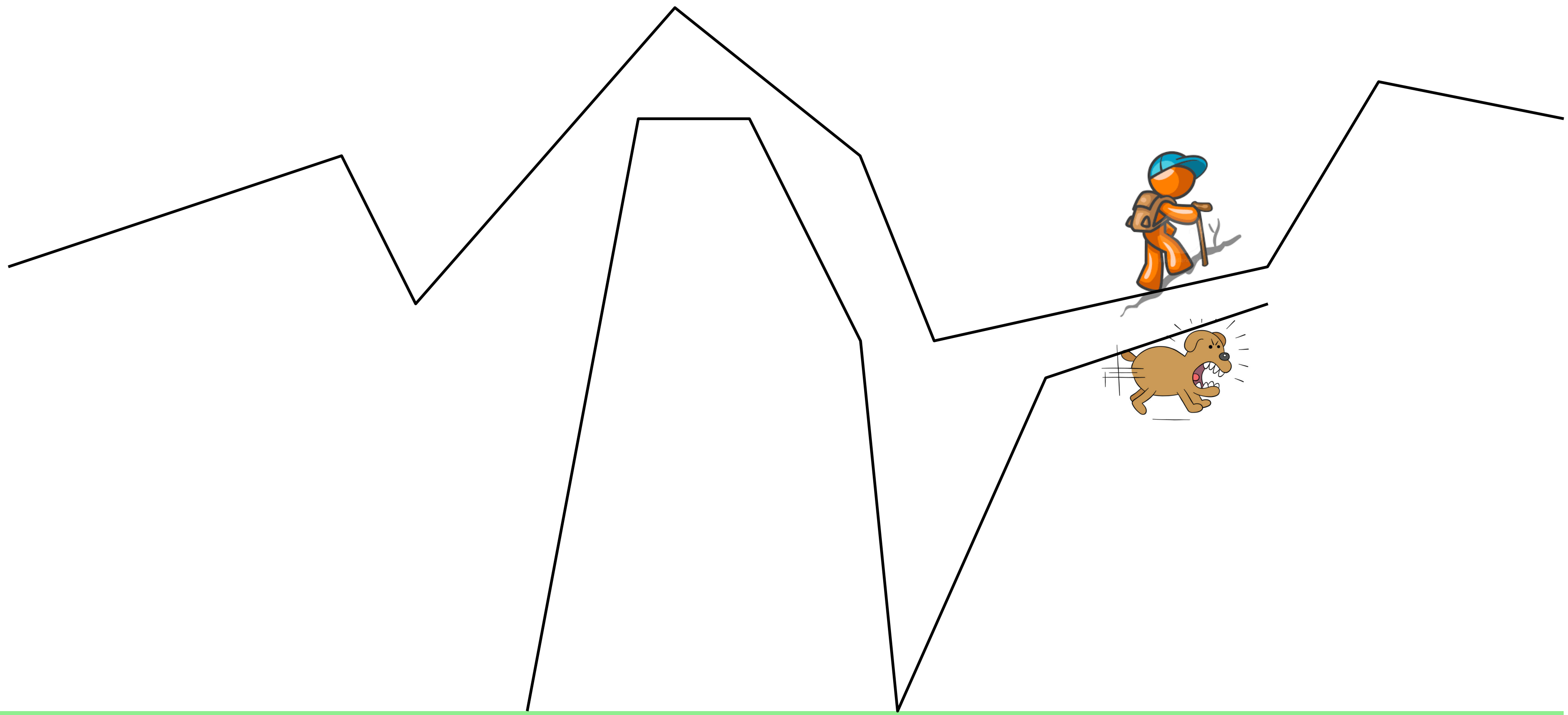
A hike in Perugia



A hike in Perugia



A hike in Perugia

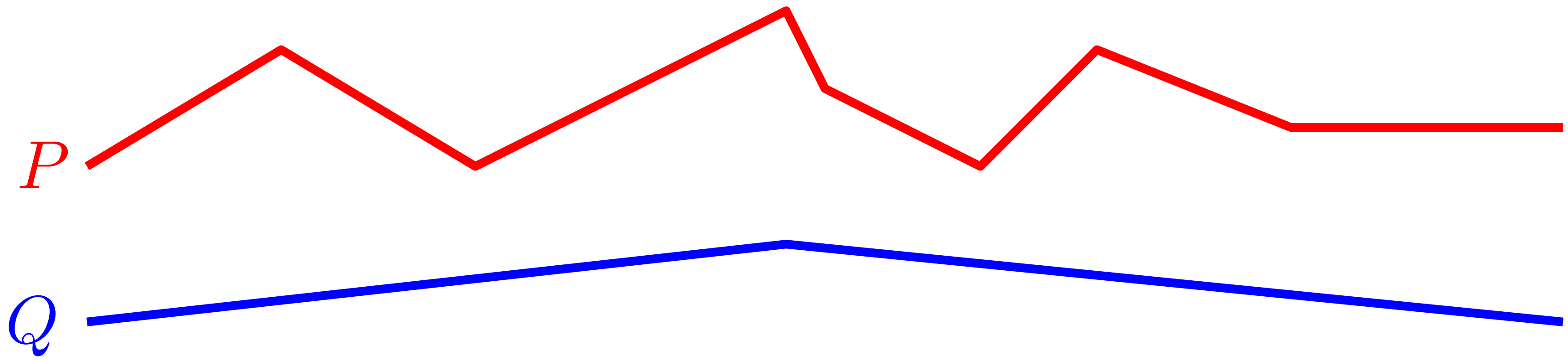


How can the angry dog maximize the time it barks at us?

Formal definition

Formal definition

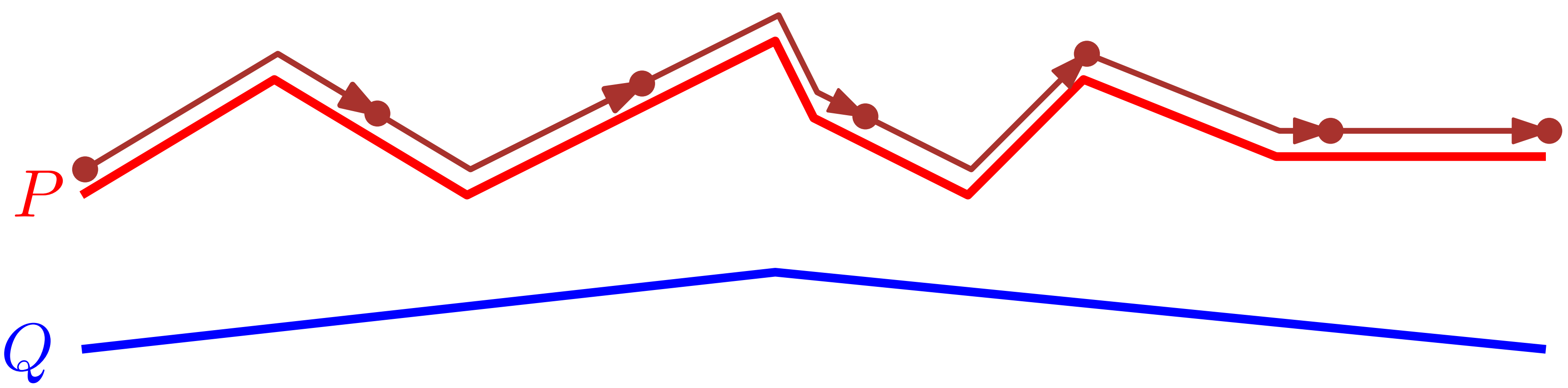
P, Q polygonal curves



Formal definition

P, Q polygonal curves

$f : [0, 1] \to P$ fixed (uniform) traversal of P

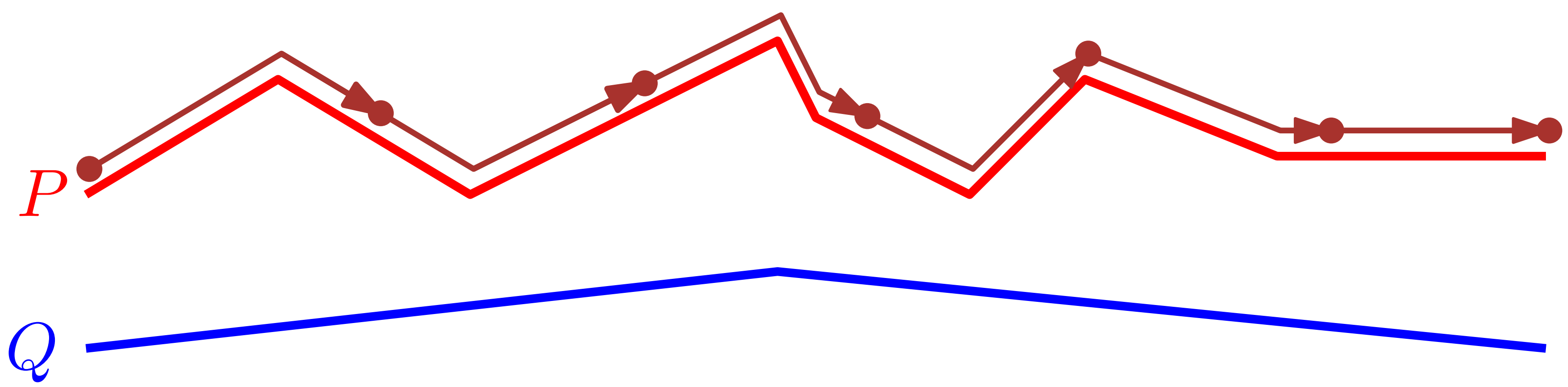


Formal definition

P, Q polygonal curves

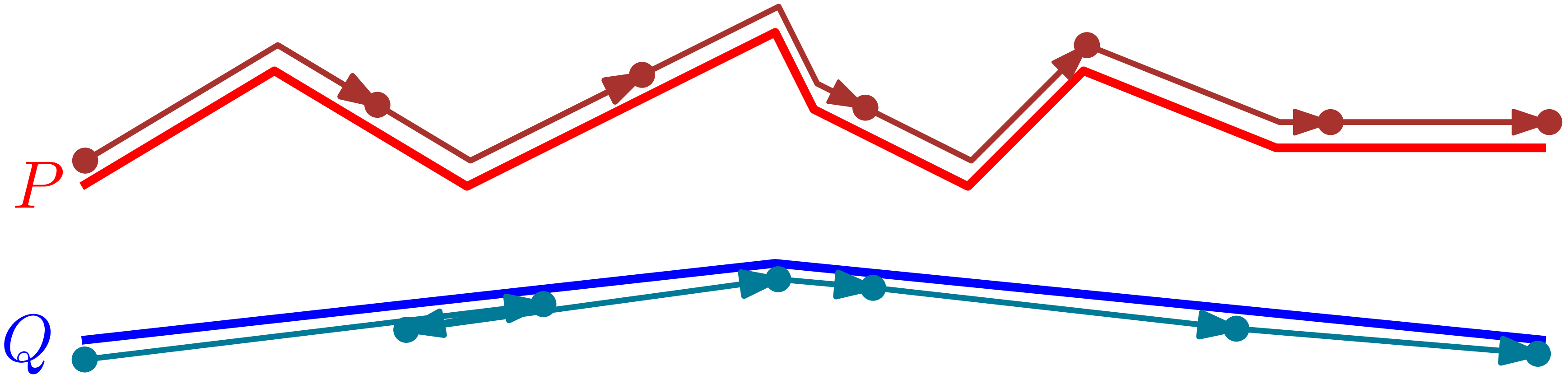
$f : [0,1] \to P$ fixed (uniform) traversal of P

$g : [0,1] \to Q$ (non-monotone) traversal of Q



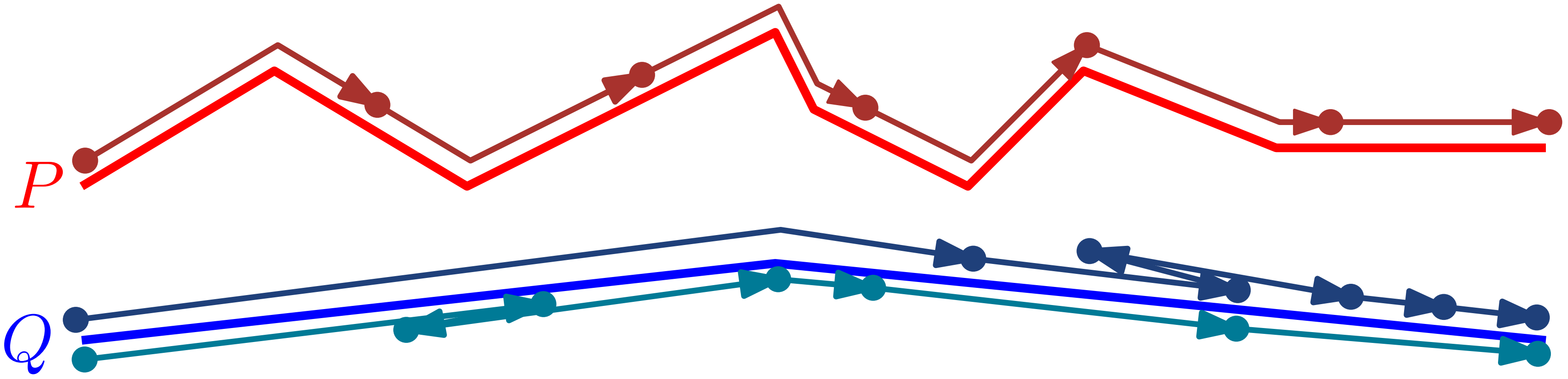
Formal definition

P, Q polygonal curves
$f : [0,1] \to P$ fixed (uniform) traversal of P
$g : [0,1] \to Q$ (non-monotone) traversal of Q



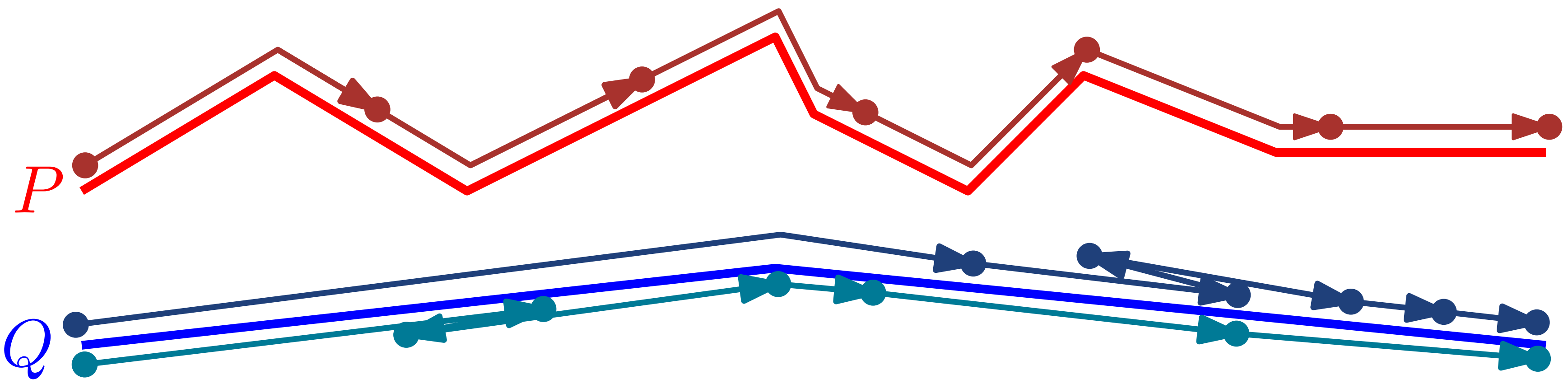
Formal definition

P, Q polygonal curves
$f : [0,1] \to P$ fixed (uniform) traversal of P
$g : [0,1] \to Q$ (non-monotone) traversal of Q



Formal definition

P, Q polygonal curves
$f : [0, 1] \to P$ fixed (uniform) traversal of P
$g : [0, 1] \to Q$ (non-monotone) traversal of Q
s: speed bound for g



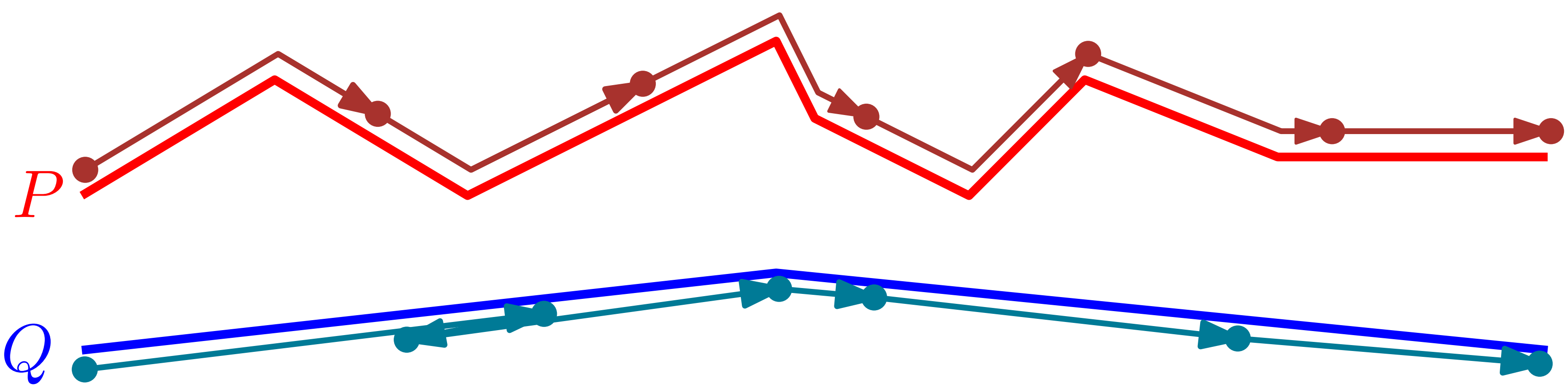
Formal definition

P, Q polygonal curves

$f : [0,1] \to P$ fixed (uniform) traversal of P

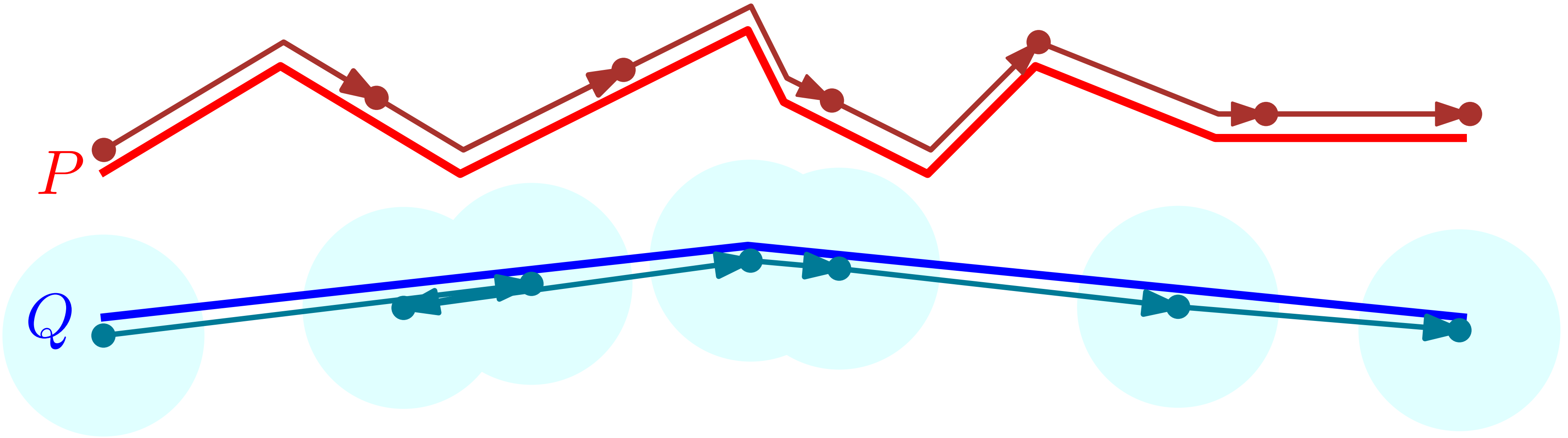
$g : [0,1] \to Q$ (non-monotone) traversal of Q

s: speed bound for g



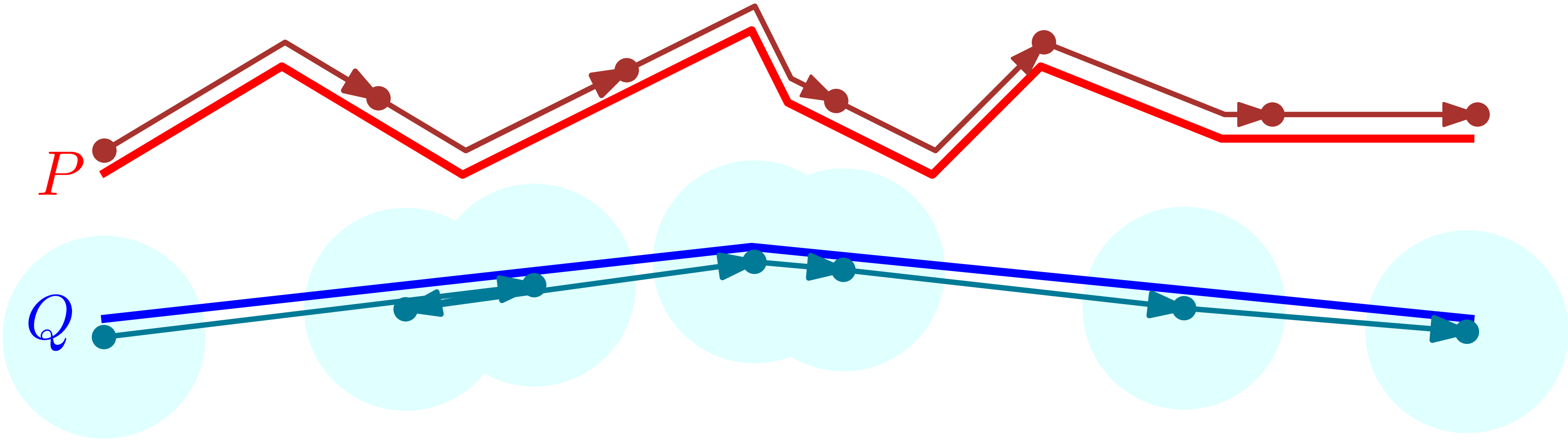
Formal definition

P, Q polygonal curves
$f : [0, 1] \to P$ fixed (uniform) traversal of P
$g : [0, 1] \to Q$ (non-monotone) traversal of Q
s: speed bound for g
θ: threshold function (barking radius)



Formal definition

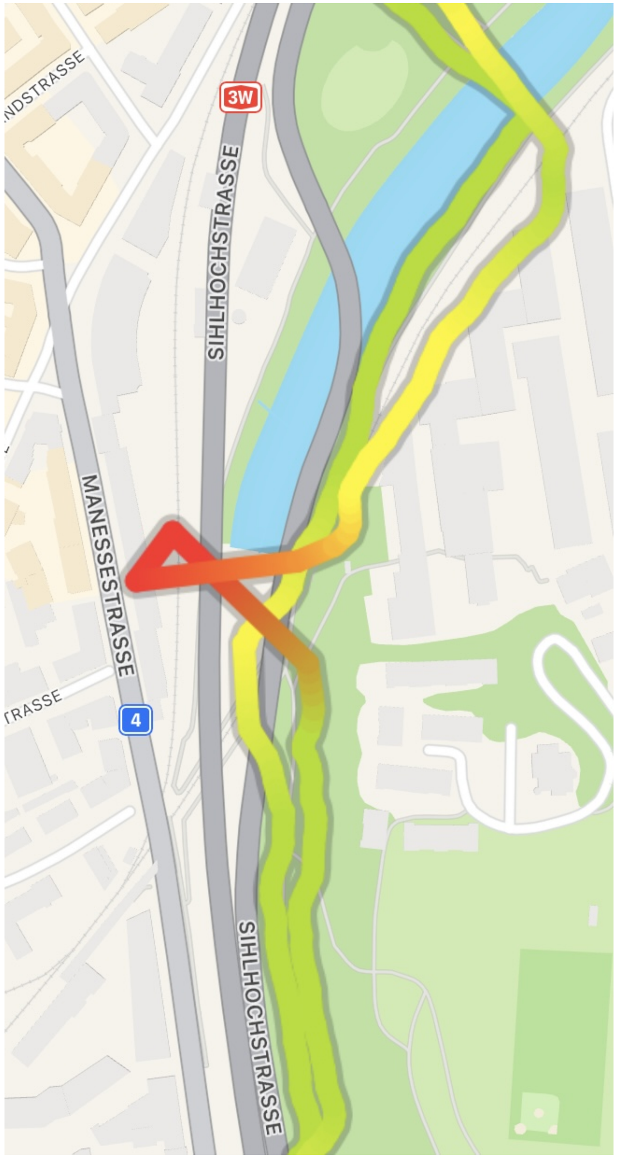
P, Q polygonal curves
$f : [0, 1] \to P$ fixed (uniform) traversal of P
$g : [0, 1] \to Q$ (non-monotone) traversal of Q
s: speed bound for g
θ: threshold function (barking radius)



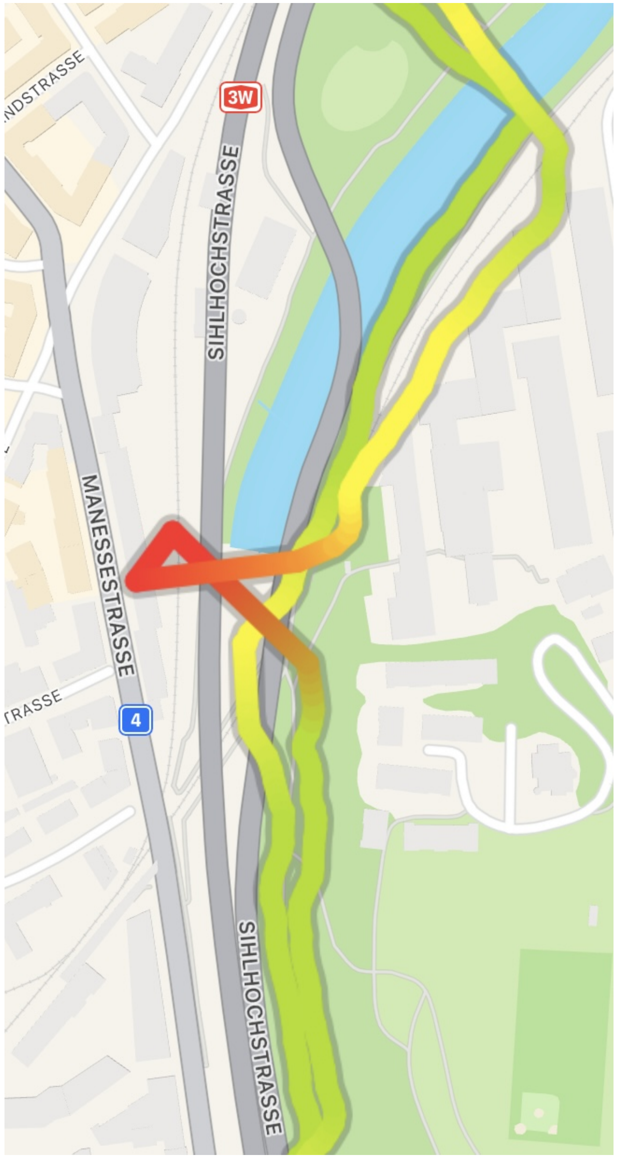
$$D_B(P, Q) = \int_{g' \leq s} \theta(f(t), g(t))dt$$

Detour detection

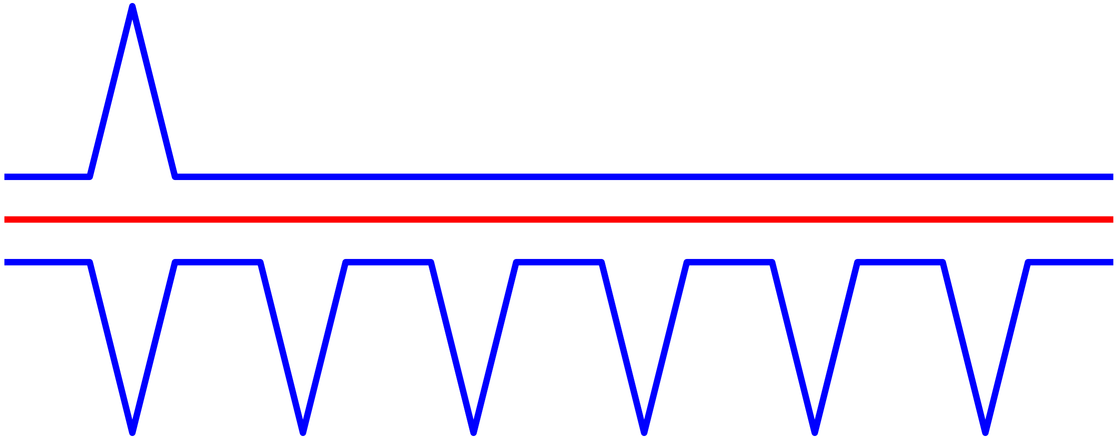
Detour detection



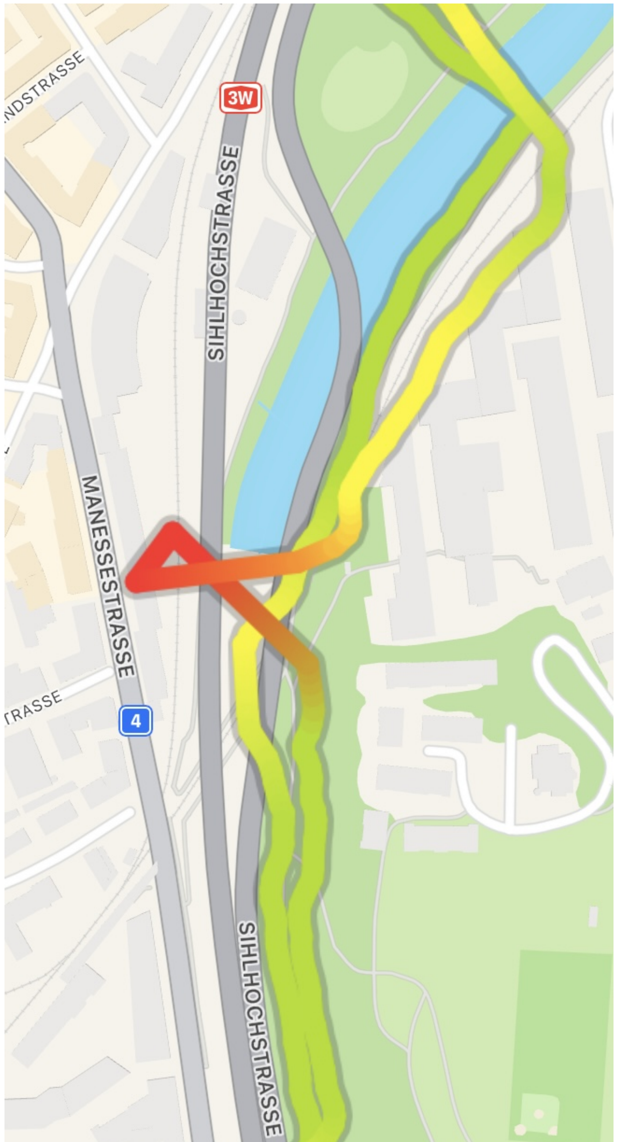
Detour detection



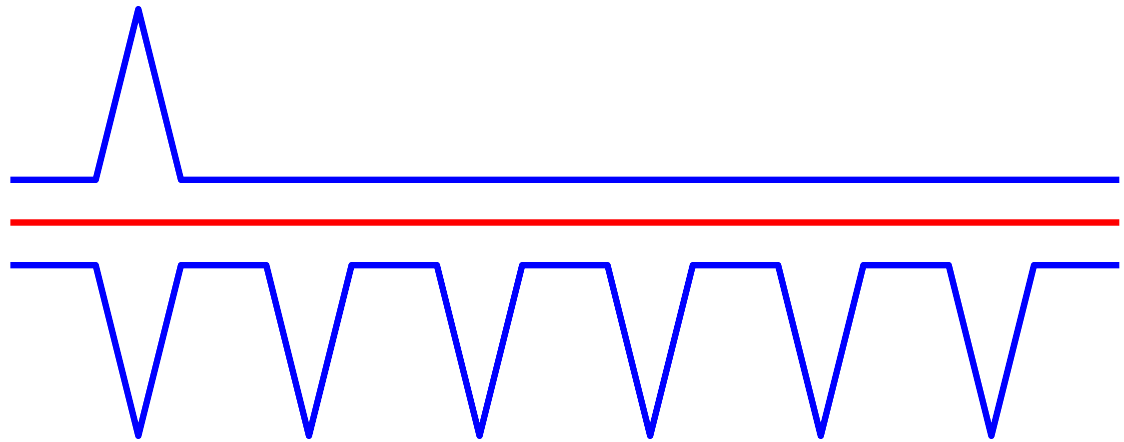
Fréchet can’t distinguish



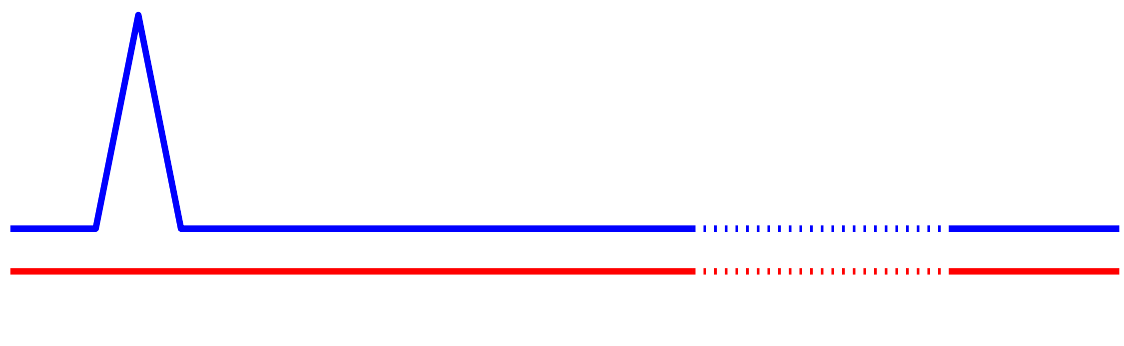
Detour detection



Fréchet can't distinguish



DTW can't distinguish



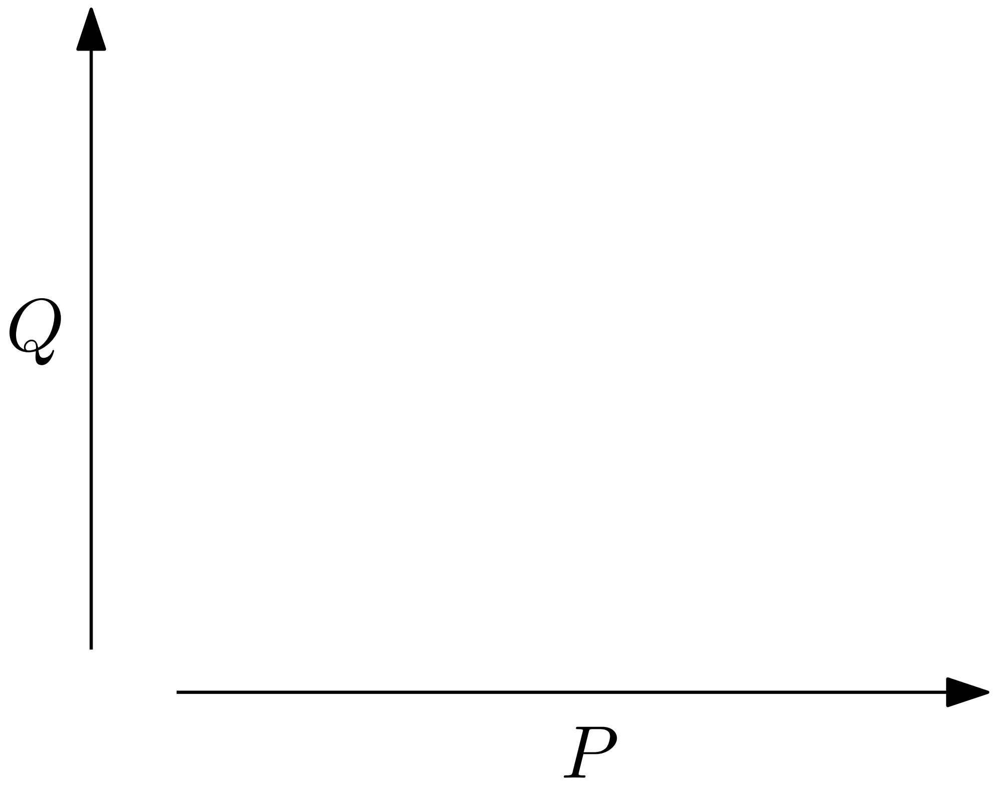
The discrete setting

The discrete setting

Traversals are discrete (both hiker and dog are frogs)

The discrete setting

Traversals are discrete (both hiker and dog are frogs)



The discrete setting

Traversals are discrete (both hiker and dog are frogs)

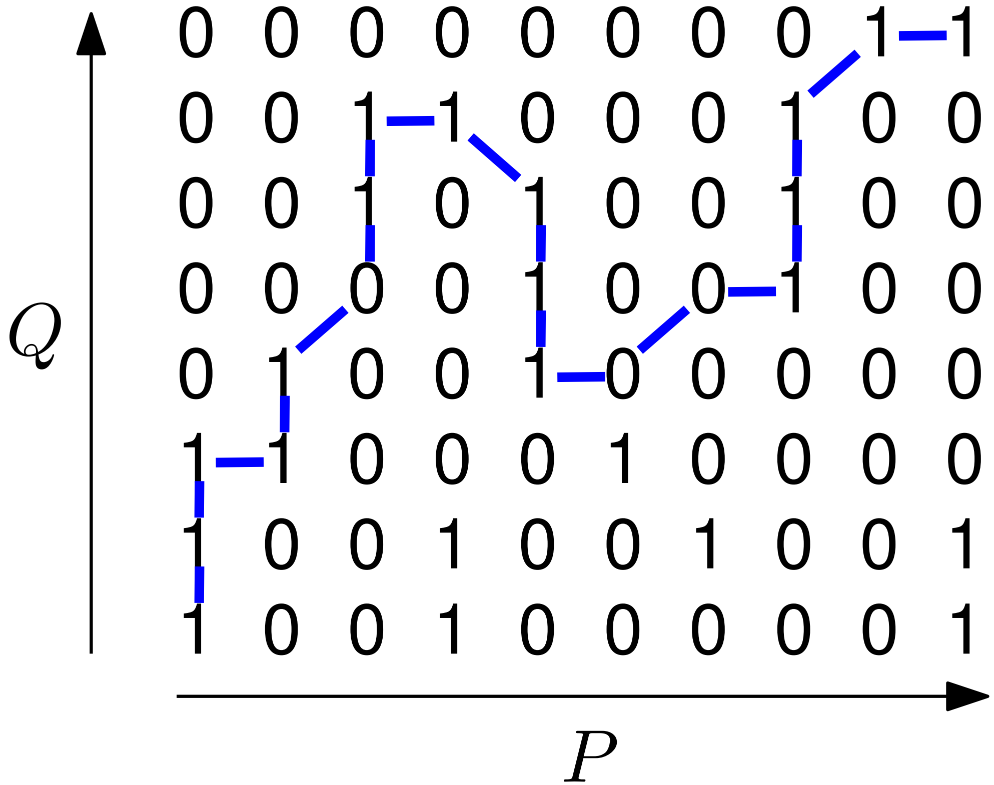
Q

0	0	0	0	0	0	0	0	1	1
0	0	1	1	0	0	0	1	0	0
0	0	1	0	1	0	0	1	0	0
0	0	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0
1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	0	0	0	1

P

The discrete setting

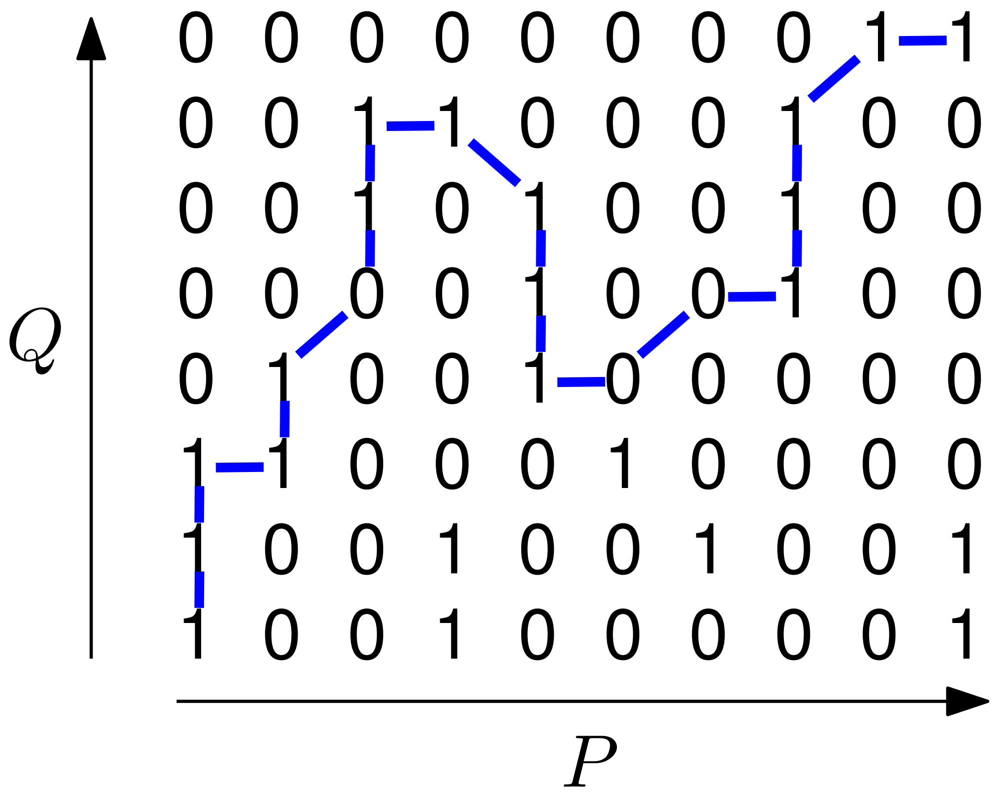
Traversals are discrete (both hiker and dog are frogs)



optimal traversal: lattice path minimizing number of zeros

The discrete setting

Traversals are discrete (both hiker and dog are frogs)

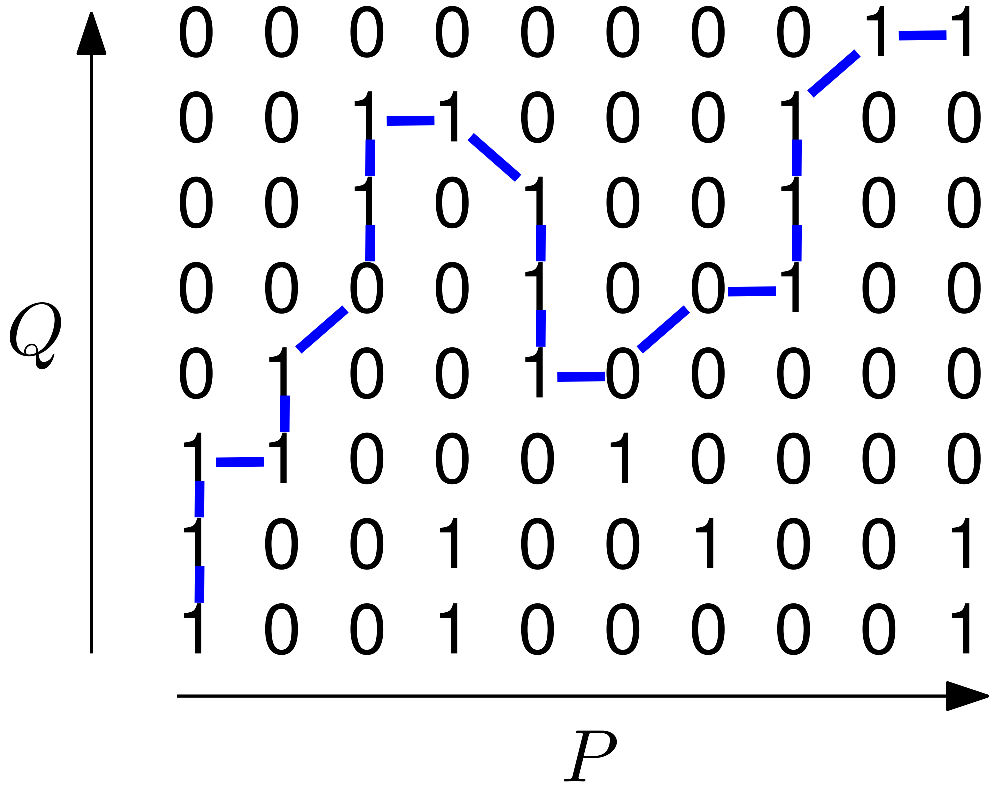


optimal traversal: lattice path minimizing number of zeros

speed bound: max. number of vertical steps

The discrete setting

Traversals are discrete (both hiker and dog are frogs)



optimal traversal: lattice path minimizing number of zeros

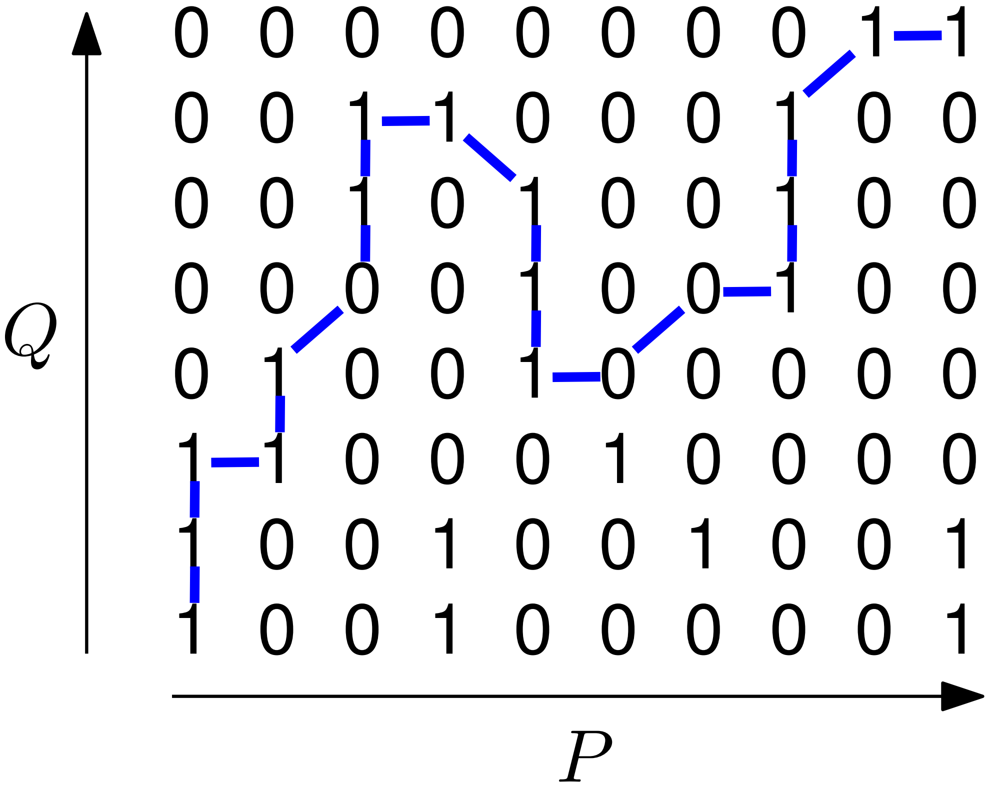
speed bound: max. number of vertical steps

$R_i(j_1, j_2)$: cost of horizontal path

$C_j(i_1, i_2)$: cost of vertical path

The discrete setting

Traversals are discrete (both hiker and dog are frogs)



optimal traversal: lattice path minimizing number of zeros

speed bound: max. number of vertical steps

$R_i(j_1, j_2)$: cost of horizontal path
$C_j(i_1, i_2)$: cost of vertical path
$F_\delta(i, j)$: min. cost to (i, j) if last step was $\delta \in \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow\}$

The discrete setting

Traversals are discrete (both hiker and dog are frogs)

$$F_d(i,j) = \begin{cases} \min\{C_j(i-k,i) + F_\delta(i-k,j) \mid \delta \in \{\nearrow, \rightarrow, \searrow\} \wedge k \in [1,s]\} & \text{if } d = \uparrow \\ F(i-1,j-1) + w(v_{i-1,j-1}) & \text{if } d = \nearrow \\ \min\{R_i(j-k,j) + F_\delta(i,j-k) \mid \delta \in \{\uparrow, \nearrow, \searrow, \downarrow\} \wedge k \in [1,s]\} & \text{if } d = \rightarrow \\ F(i+1,j+1) + w(v_{i+1,j+1}) & \text{if } d = \searrow \\ \min\{C_j(i+k,i) + F_\delta(i+k,j) \mid \delta \in \{\nearrow, \rightarrow, \searrow\} \wedge k \in [1,s]\} & \text{if } d = \downarrow \end{cases}$$

Q

$$\begin{matrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

P

$R_i(j_1, j_2)$: cost of horizontal path

$C_j(i_1, i_2)$: cost of vertical path

$F_\delta(i,j)$: min. cost to (i,j) if last step was $\delta \in \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow\}$

The discrete setting

Traversals are discrete (both hiker and dog are frogs)

$$F_d(i,j) = \begin{cases} \min\{C_j(i-k,i) + F_\delta(i-k,j) \mid \delta \in \{\nearrow, \rightarrow, \searrow\} \wedge k \in [1,s]\} & \text{if } d = \uparrow \\ F(i-1,j-1) + w(v_{i-1,j-1}) & \text{if } d = \nearrow \\ \min\{R_i(j-k,j) + F_\delta(i,j-k) \mid \delta \in \{\uparrow, \nearrow, \searrow, \downarrow\} \wedge k \in [1,s]\} & \text{if } d = \rightarrow \\ F(i+1,j+1) + w(v_{i+1,j+1}) & \text{if } d = \searrow \\ \min\{C_j(i+k,i) + F_\delta(i+k,j) \mid \delta \in \{\nearrow, \rightarrow, \searrow\} \wedge k \in [1,s]\} & \text{if } d = \downarrow \end{cases}$$

$$\begin{array}{c|cccccccccc} & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ Q & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline & & & & & P & & & & & \end{array}$$

$R_i(j_1, j_2)$: cost of horizontal path

$C_j(i_1, i_2)$: cost of vertical path

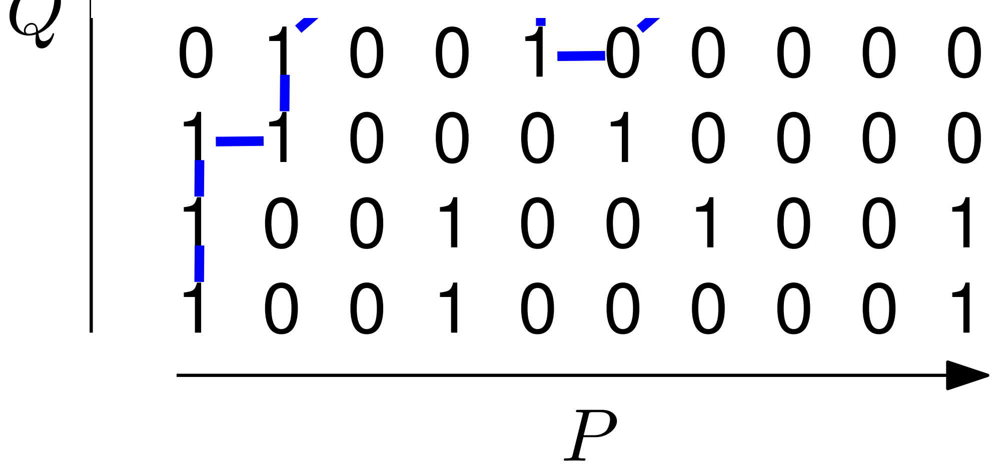
$F_\delta(i,j)$: min. cost to (i,j) if last step was $\delta \in \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow\}$

Dynamic programming + right data structures

The discrete setting

Traversals are discrete (both hiker and dog are frogs)

$$F_d(i,j) = \begin{cases} \min\{C_j(i-k,i) + F_\delta(i-k,j) \mid \delta \in \{\nearrow, \rightarrow, \searrow\} \wedge k \in [1,s]\} & \text{if } d = \uparrow \\ F(i-1,j-1) + w(v_{i-1,j-1}) & \text{if } d = \nearrow \\ \min\{R_i(j-k,j) + F_\delta(i,j-k) \mid \delta \in \{\uparrow, \nearrow, \searrow, \downarrow\} \wedge k \in [1,s]\} & \text{if } d = \rightarrow \\ F(i+1,j+1) + w(v_{i+1,j+1}) & \text{if } d = \searrow \\ \min\{C_j(i+k,i) + F_\delta(i+k,j) \mid \delta \in \{\nearrow, \rightarrow, \searrow\} \wedge k \in [1,s]\} & \text{if } d = \downarrow \end{cases}$$



$R_i(j_1, j_2)$: cost of horizontal path
$C_j(i_1, i_2)$: cost of vertical path
$F_\delta(i,j)$: min. cost to (i,j) if last step was $\delta \in \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow\}$

Dynamic programming + right data structures

Runtime $O(nm \log s)$

Other settings

Other settings

Semi-discrete setting:
Traversal for hiker is discrete, traversal for dog is continuous.

Other settings

Semi-discrete setting:
Traversal for hiker is discrete, traversal for dog is continuous.

$O(nm\log(nm))$ time algorithm

Other settings

Semi-discrete setting:
Traversal for hiker is discrete, traversal for dog is continuous.

$O(nm \log(nm))$ time algorithm

Continuous setting:
Both traversals continuous.

Other settings

Semi-discrete setting:
Traversal for hiker is discrete, traversal for dog is continuous.

$O(nm\log(nm))$ time algorithm

Continuous setting:
Both traversals continuous.

$O(n^4m^3\log(nm))$ time algorithm

Other settings

Semi-discrete setting:
Traversal for hiker is discrete, traversal for dog is continuous.

$O(nm \log(nm))$ time algorithm

Continuous setting:
Both traversals continuous.

$O(n^4 m^3 \log(nm))$ time algorithm

For $n = m$ no $O(n^{2-\varepsilon})$ algorithm exists, assuming SETH.

Conclusion

Conclusion

- Barking distance as a new measure for detour detection

Conclusion

- Barking distance as a new measure for detour detection
- Efficient algorithms for many settings

Conclusion

- Barking distance as a new measure for detour detection
- Efficient algorithms for many settings
- Matching lower bound (up to logarithmic factors) for two settings

Conclusion

- Barking distance as a new measure for detour detection
- Efficient algorithms for many settings
- Matching lower bound (up to logarithmic factors) for two settings
    - More efficient algorithm for the continuous setting?

Conclusion

- Barking distance as a new measure for detour detection
- Efficient algorithms for many settings
- Matching lower bound (up to logarithmic factors) for two settings
    - More efficient algorithm for the continuous setting?
    - Algorithms for optimizing speed or barking radius?

Conclusion

- Barking distance as a new measure for detour detection
- Efficient algorithms for many settings
- Matching lower bound (up to logarithmic factors) for two settings
    - More efficient algorithm for the continuous setting?
    - Algorithms for optimizing speed or barking radius?

Thank you!