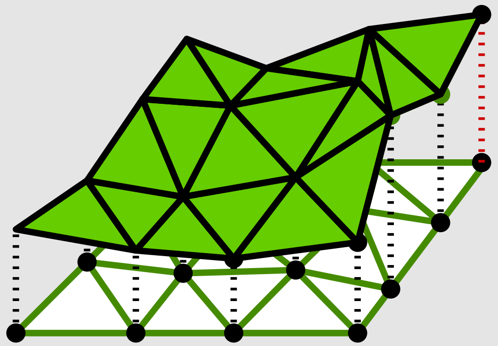
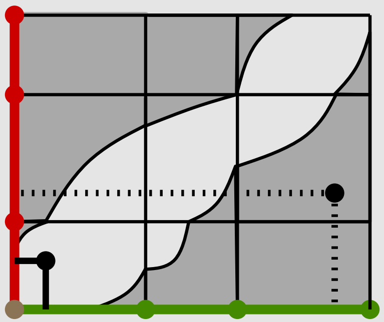
Computational Geometry
(Two) Selected Topics

Mark de Berg

TU Eindhoven

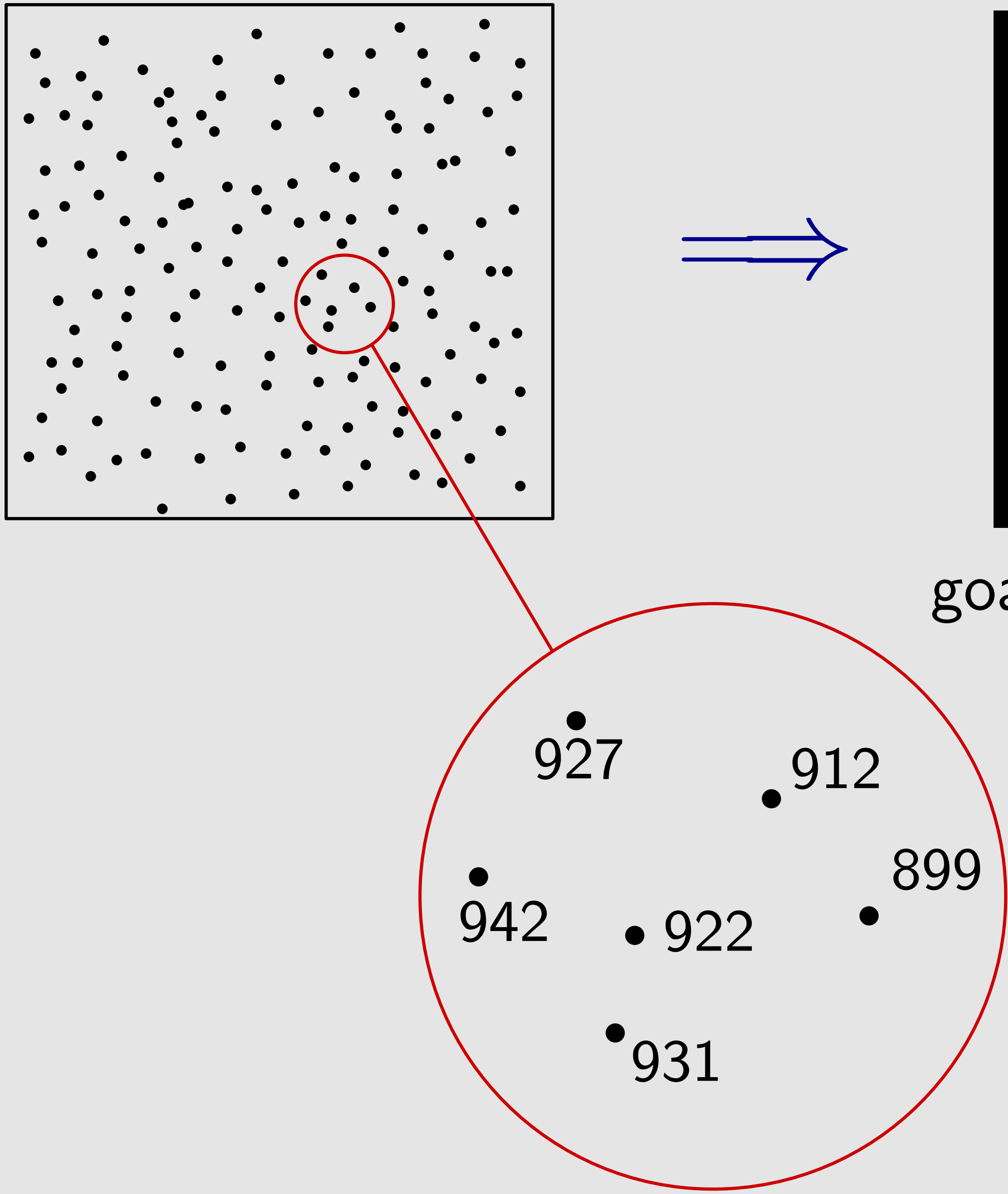


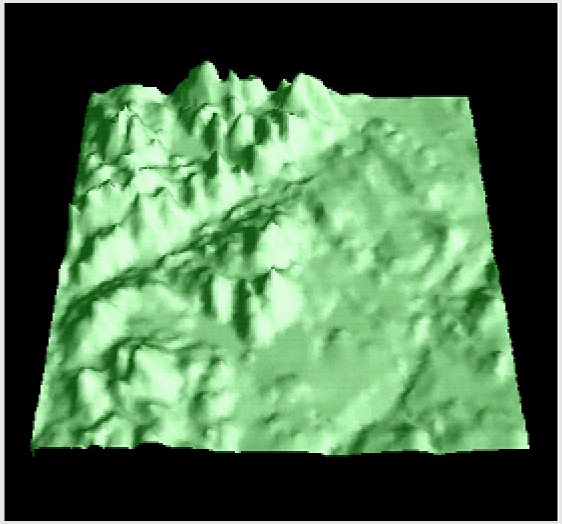


TU/e

Reconstructing Terrains from Elevantion Data

given: terrain elevation at sample points

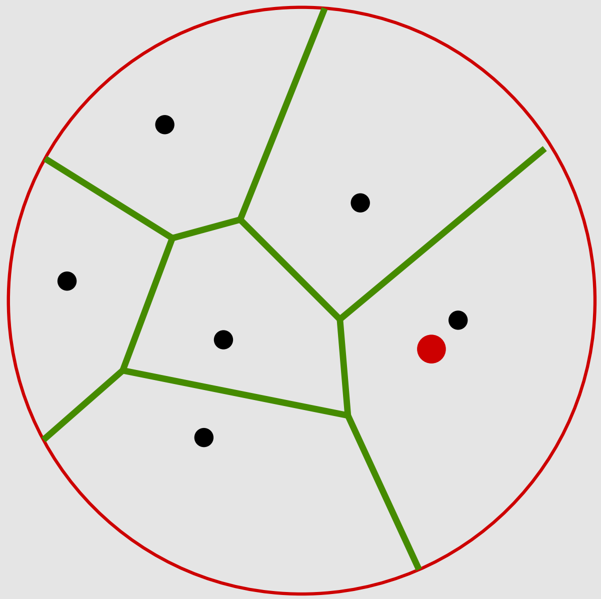




goal: construct terrain surface

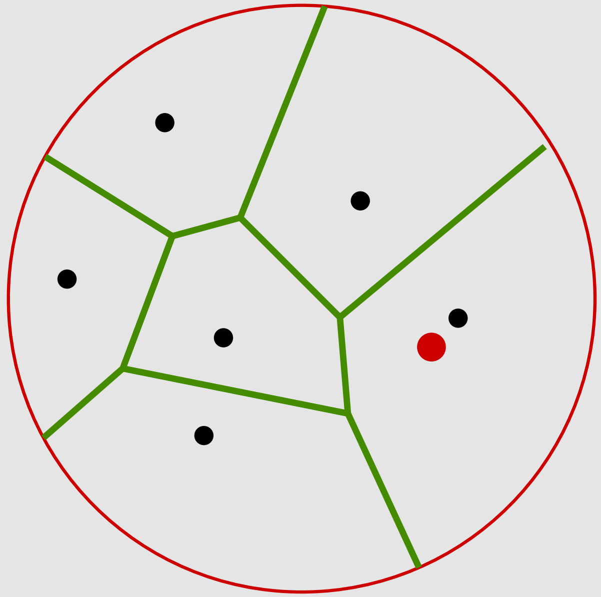

927
912
899
942
922
elevation?
931

Idea: use elevation of nearest sample point



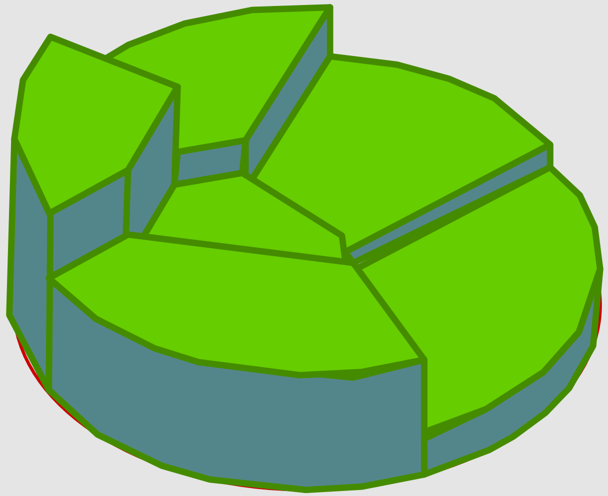
Voronoi diagram

Idea: use elevation of nearest sample point

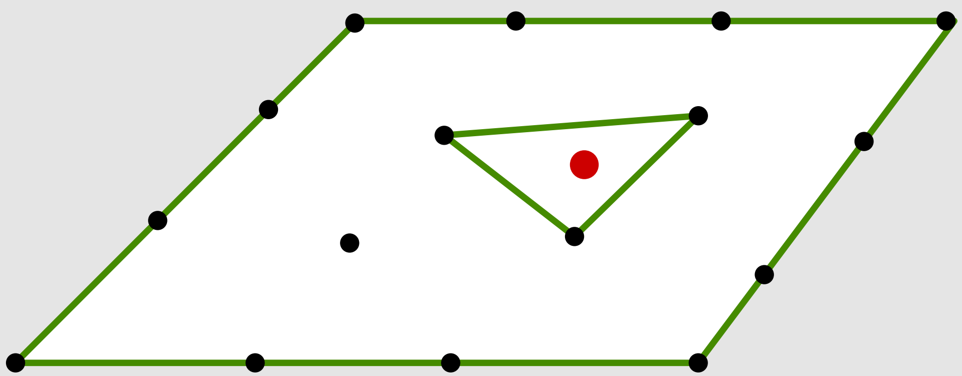


Voronoi diagram

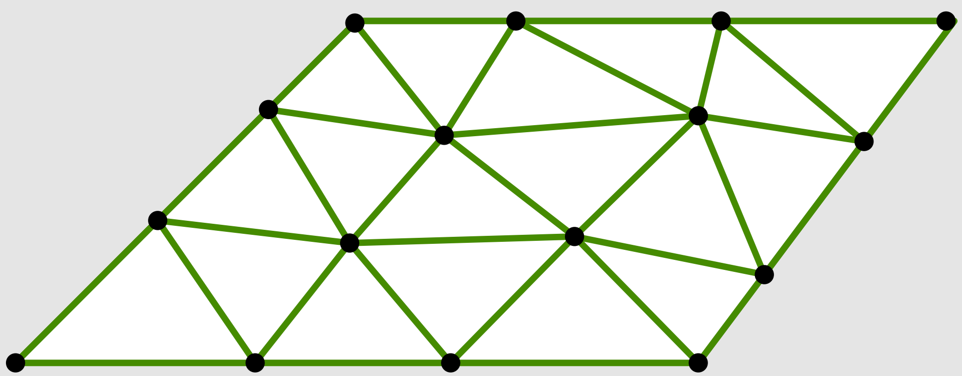
Does not give good result: surface not continuous



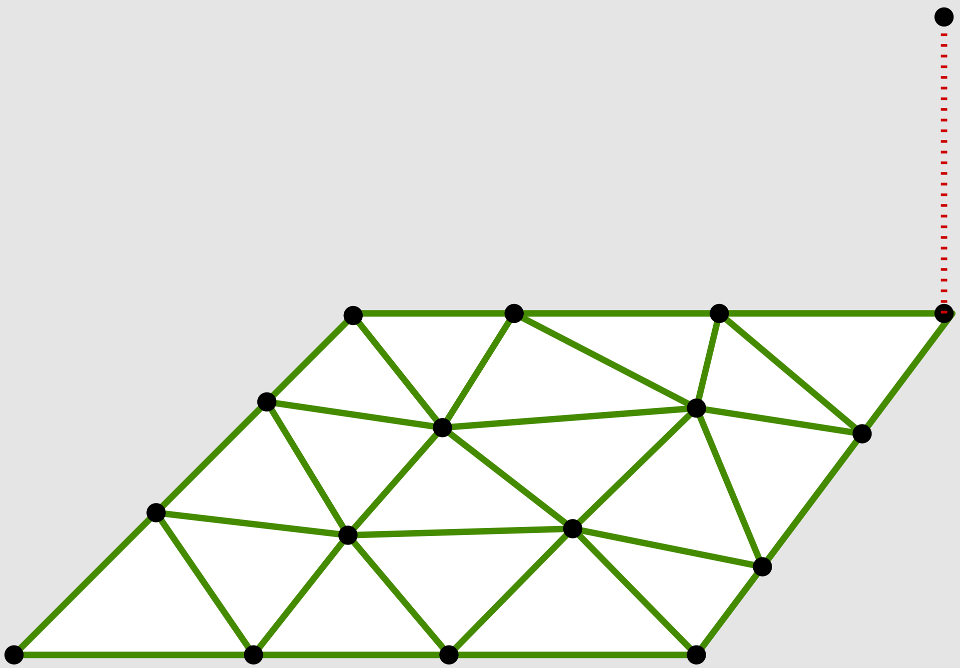
Better: determine elevation using interpolation



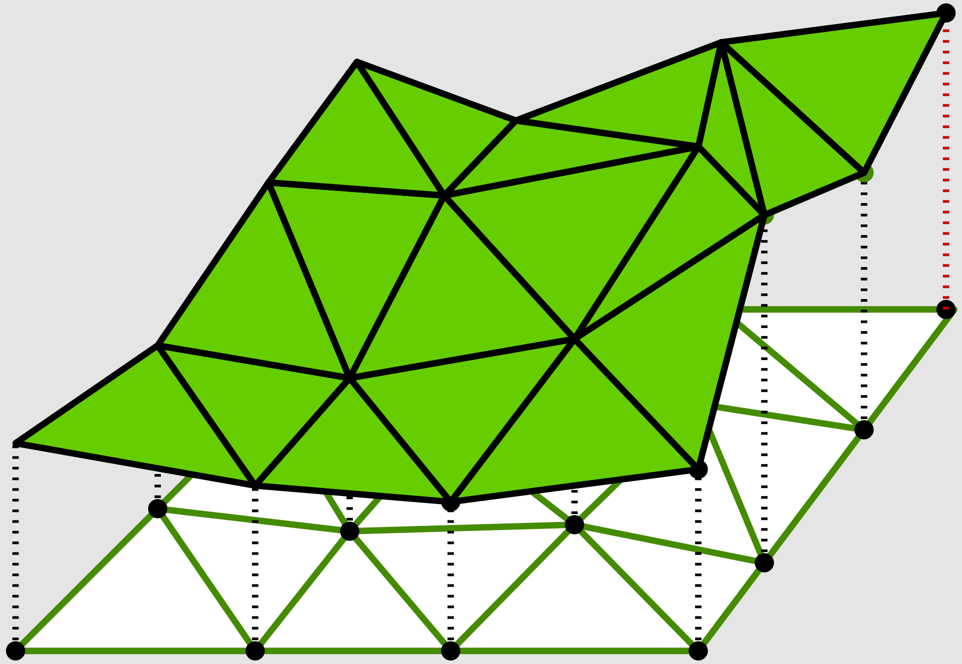
Better: determine elevation using interpolation



Better: determine elevation using interpolation

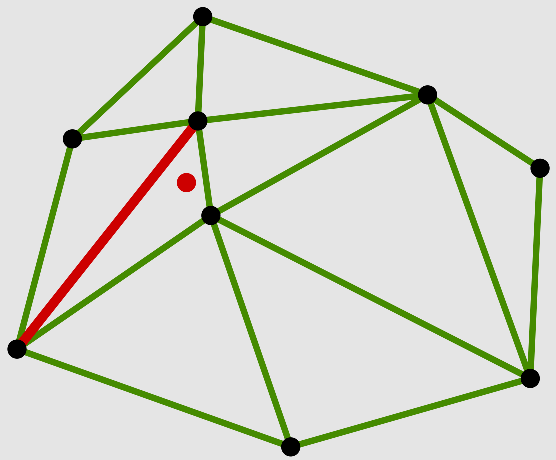


Better: determine elevation using interpolation

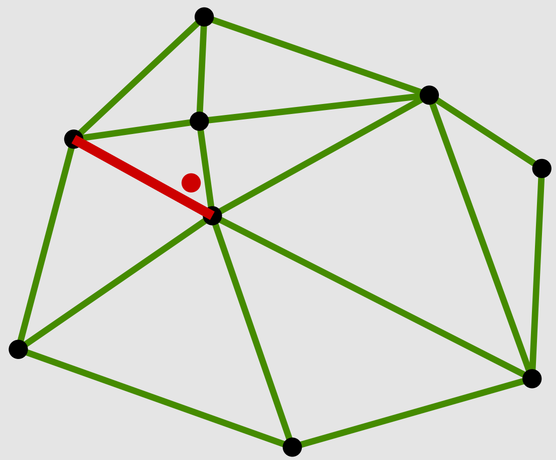


gives continuous surface

What is a good triangulation?

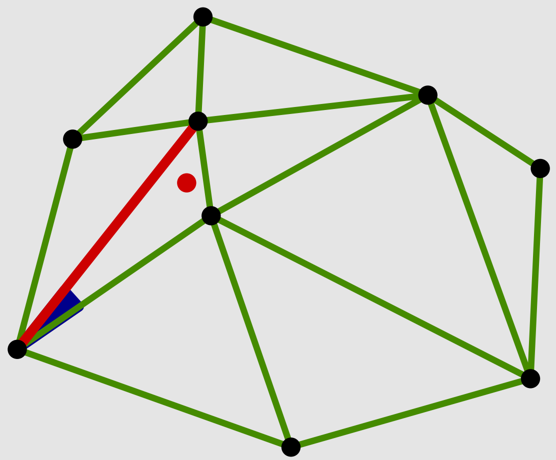


or

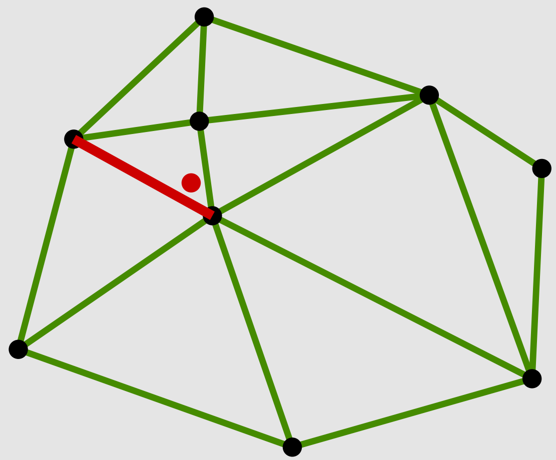


or . . .

What is a good triangulation?



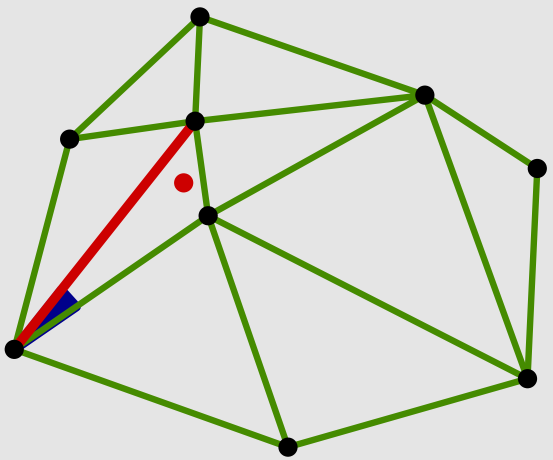
or



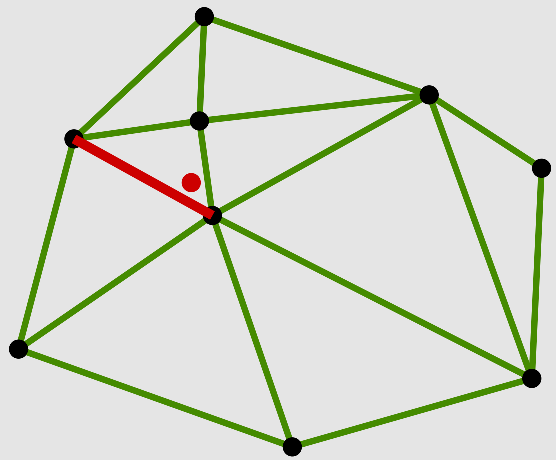
or . . .

long and thin triangles are bad $\Longrightarrow$ try to avoid small angles

What is a good triangulation?



or

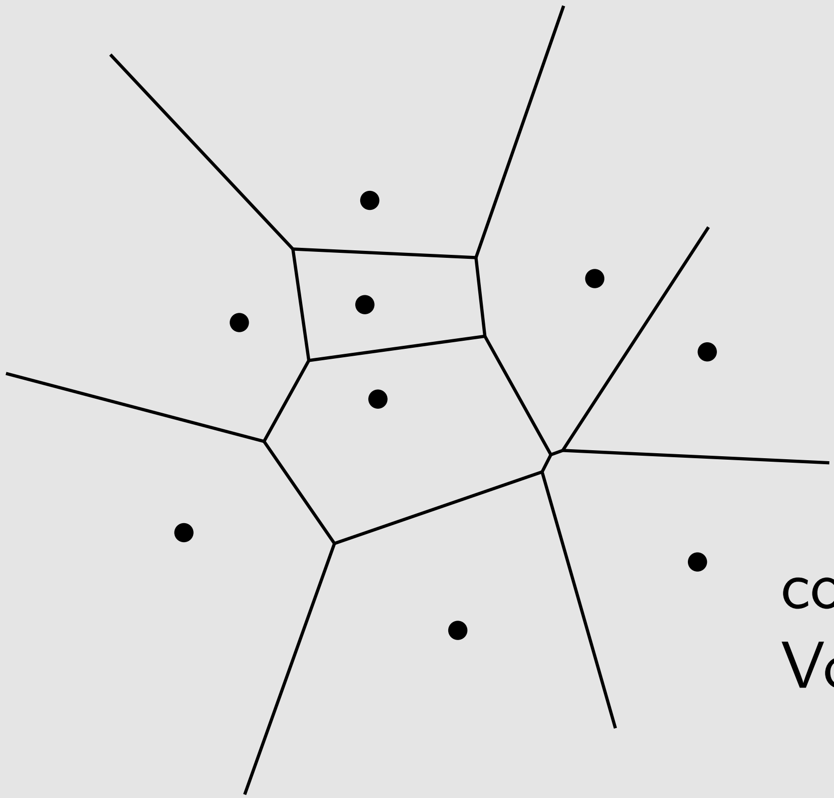


or . . .

long and thin triangles are bad $\Longrightarrow$ try to avoid small angles

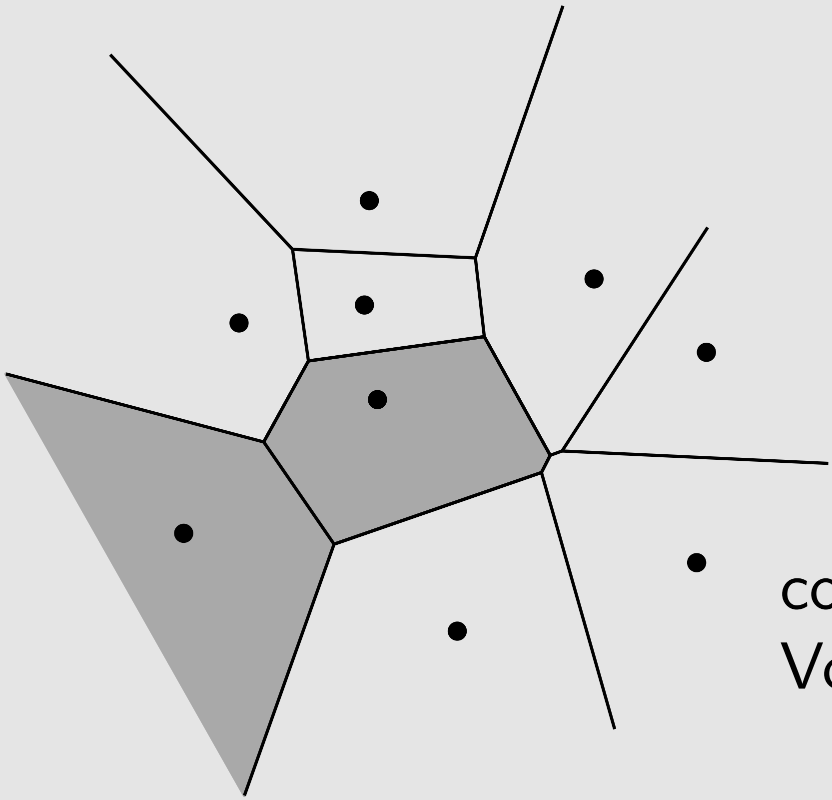
Algorithmic problem: how can we quickly compute a triangulation that maximizes the angles?

Voronoi diagram



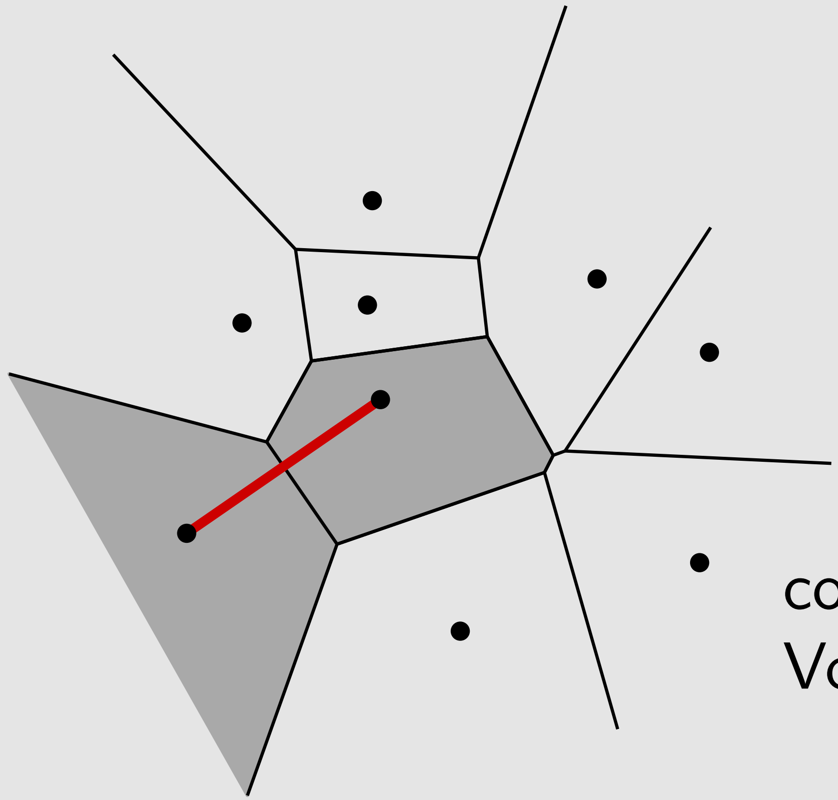
connect points whose
Voronoi cells are neighbors

Voronoi diagram



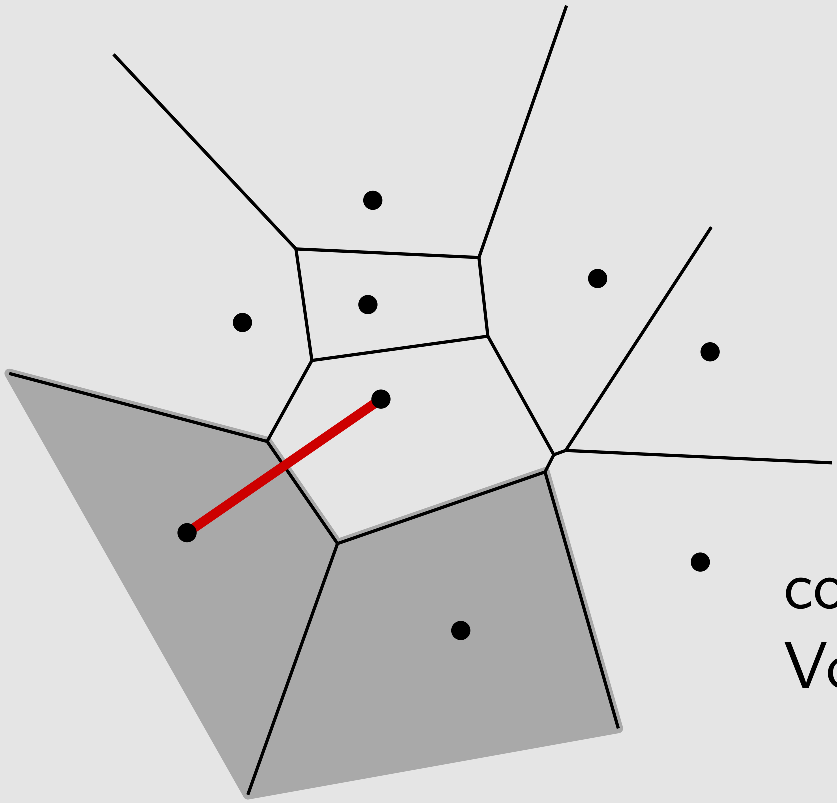
connect points whose
Voronoi cells are neighbors

Voronoi diagram



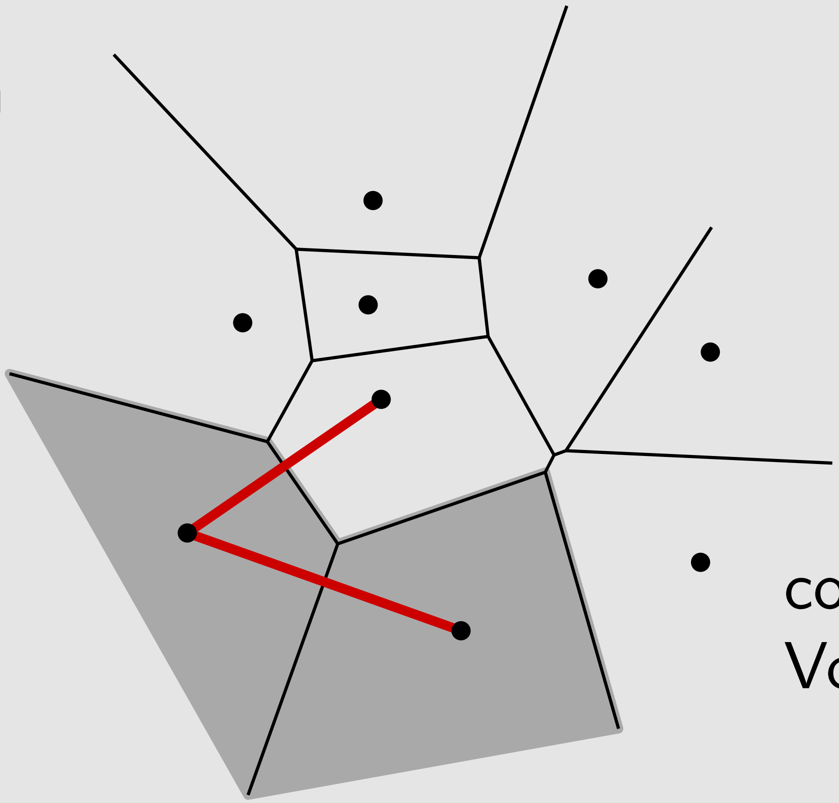
connect points whose
Voronoi cells are neighbors

Voronoi diagram



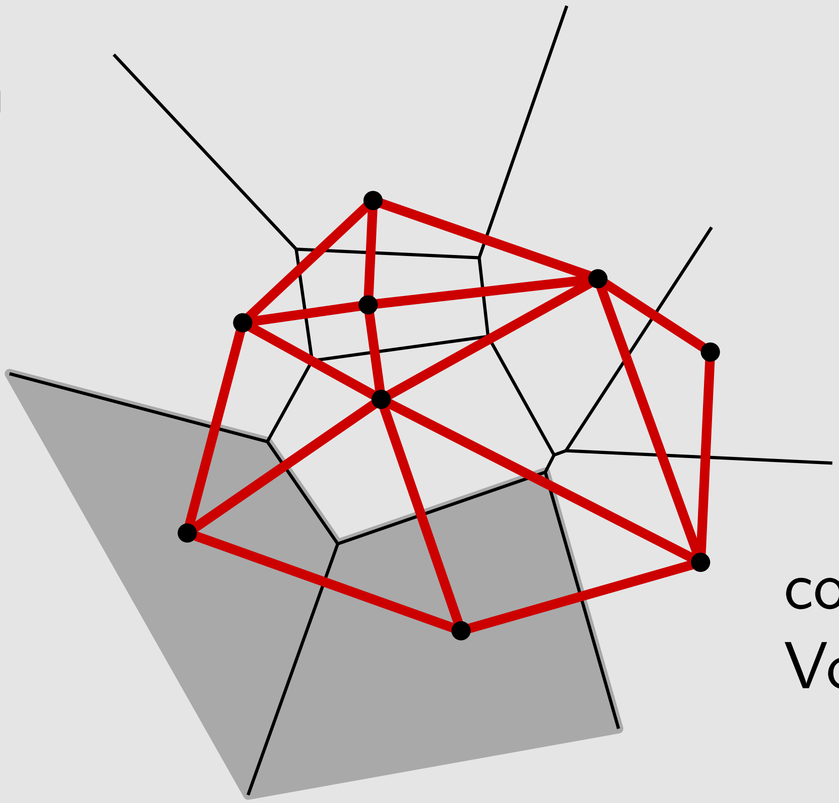
connect points whose
Voronoi cells are neighbors

Voronoi diagram



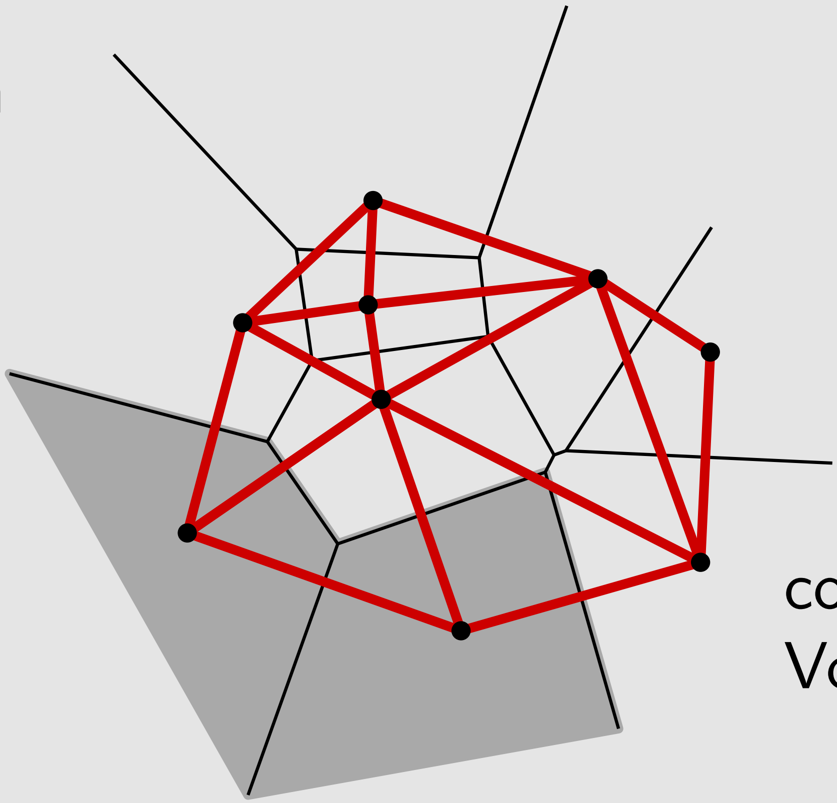
connect points whose
Voronoi cells are neighbors

Voronoi diagram



connect points whose
Voronoi cells are neighbors

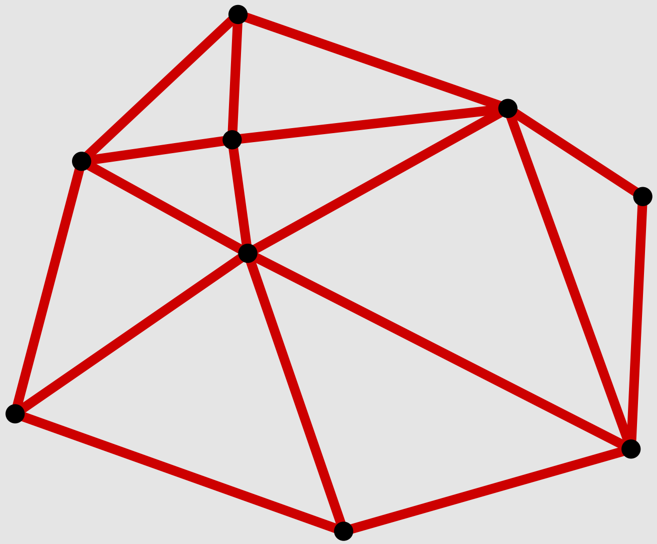
Voronoi diagram



connect points whose
Voronoi cells are neighbors

Delaunay triangulation:

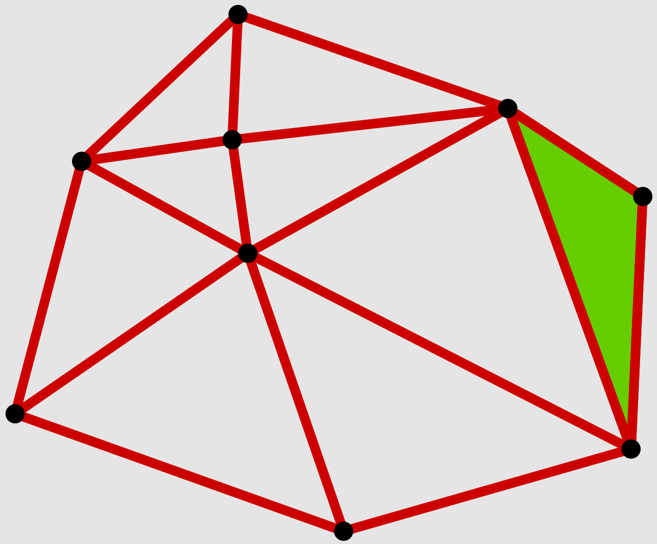
triangulation that maximizes
the minimum angle!



$\Delta(p, q, r)$ is triangle in Delaunay triangulation

$\Longleftrightarrow$

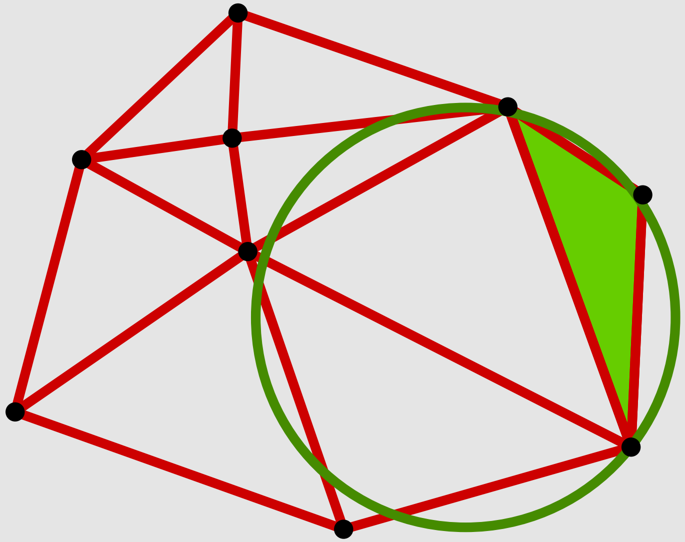
Circle(p, q, r) does not contain any other point



$\Delta(p, q, r)$ is triangle in Delaunay triangulation

$\Longleftrightarrow$

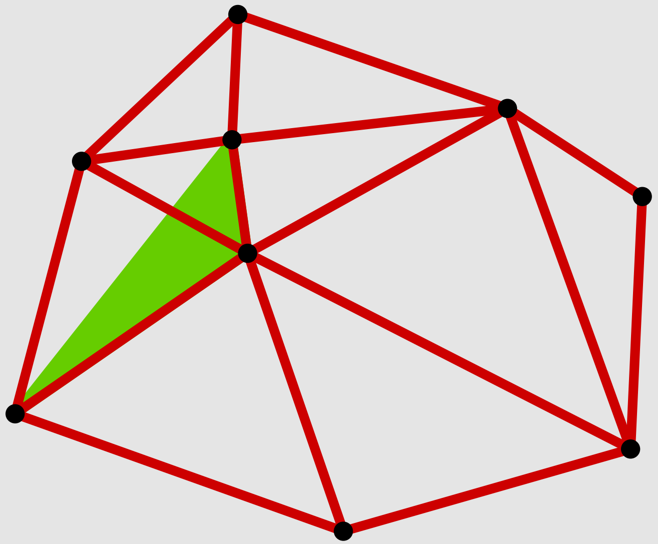
Circle(p, q, r) does not contain any other point



$\Delta(p, q, r)$ is triangle in Delaunay triangulation

$\Longleftrightarrow$

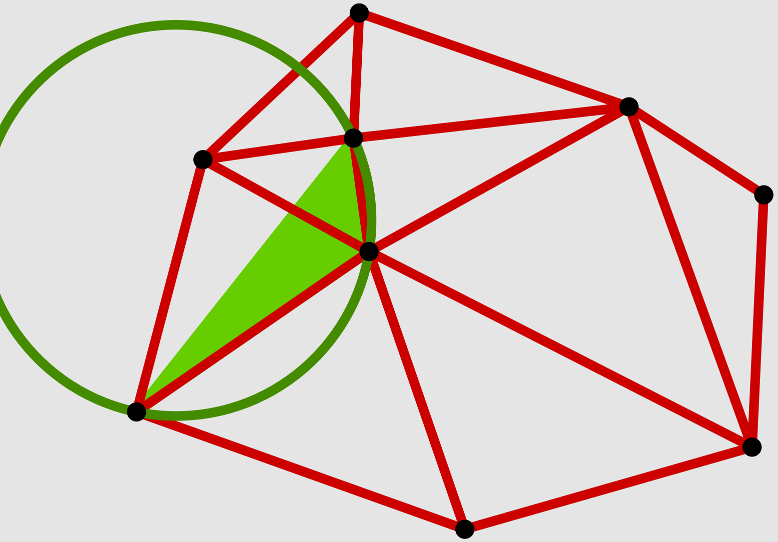
Circle(p, q, r) does not contain any other point



$\Delta(p, q, r)$ is triangle in Delaunay triangulation

$\Longleftrightarrow$

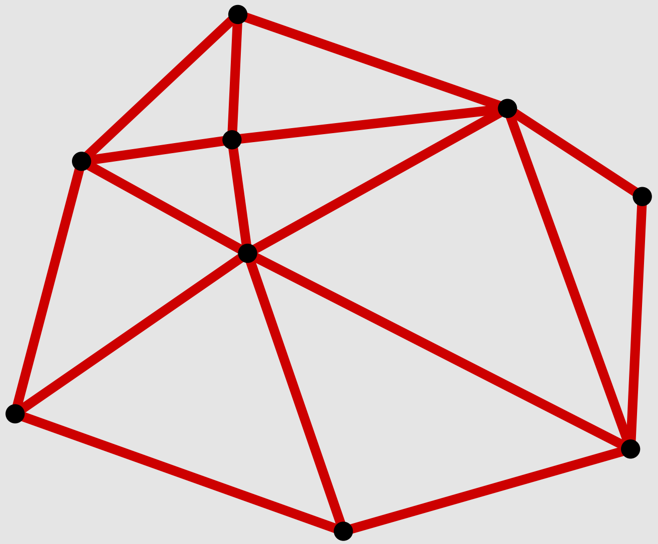
Circle(p, q, r) does not contain any other point



$\Delta(p,q,r)$ is triangle in Delaunay triangulation

$\Longleftrightarrow$

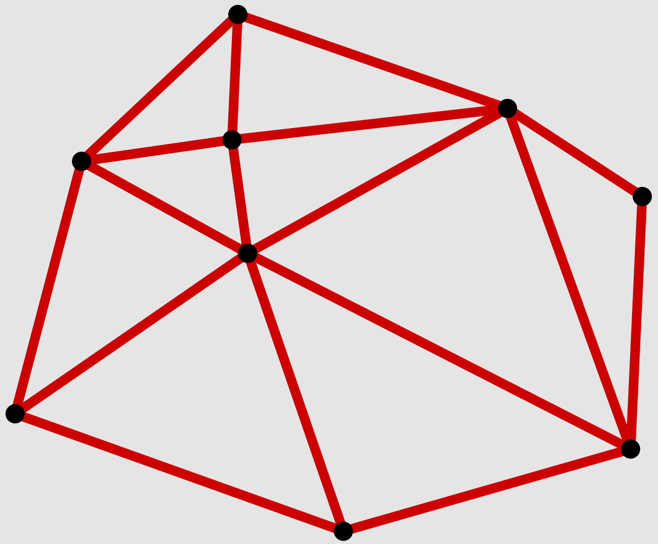
Circle(p,q,r) does not contain any other point



$\Delta(p, q, r)$ is triangle in Delaunay triangulation

$\Longleftrightarrow$

Circle(p, q, r) does not contain any other point



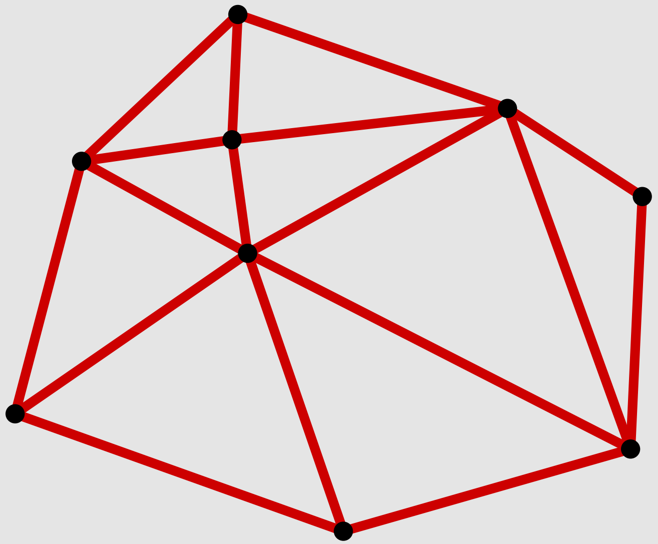
$\Delta(p, q, r)$ is triangle in Delaunay triangulation

$\Longleftrightarrow$

Circle(p, q, r) does not contain any other point

Algorithm for computing Delaunay triangulations:

1. **for** every triple of points
2. **do if** all other points lie outside Circle(p, q, r)
3. **then** Add triangle $\Delta(p, q, r)$ to DT



$\Delta(p,q,r)$ is triangle in Delaunay triangulation

$\Longleftrightarrow$

Circle(p,q,r) does not contain any other point

Algorithm for computing Delaunay triangulations:

1. **for** every triple of points
2. **do if** all other points lie outside Circle(p,q,r)
3. **then** Add triangle $\Delta(p,q,r)$ to DT

- running time $O(n^4)$: very slow!
- can we do better? yes: there are algorithms running in $O(n \log n)$ time

TU/e

Analyzing Trajectories of Moving Objects

Moving objects (spatio-temporal data): now widely available

- GPS data, RFID tags, surveillance cameras, simulations

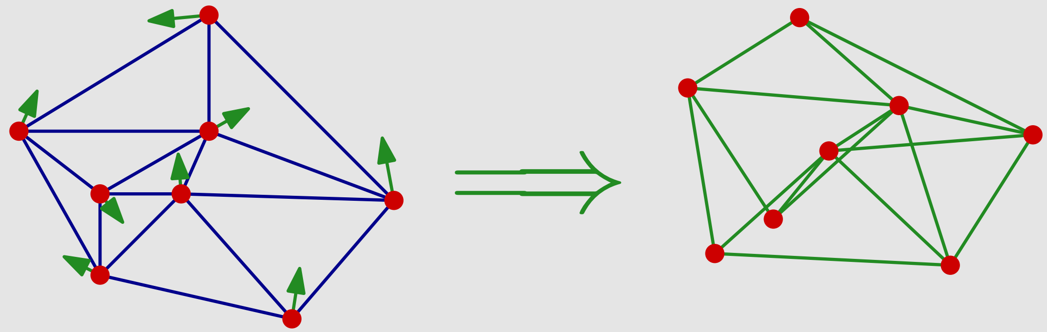
Movement data can be considered in two ways:

Moving objects (spatio-temporal data): now widely available

- GPS data, RFID tags, surveillance cameras, simulations

Movement data can be considered in two ways:

- Individual locations, which change over time, are relevant

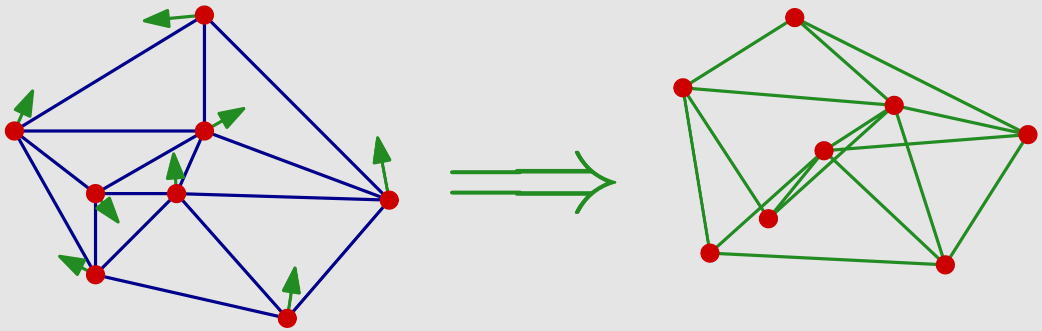


Moving objects (spatio-temporal data): now widely available

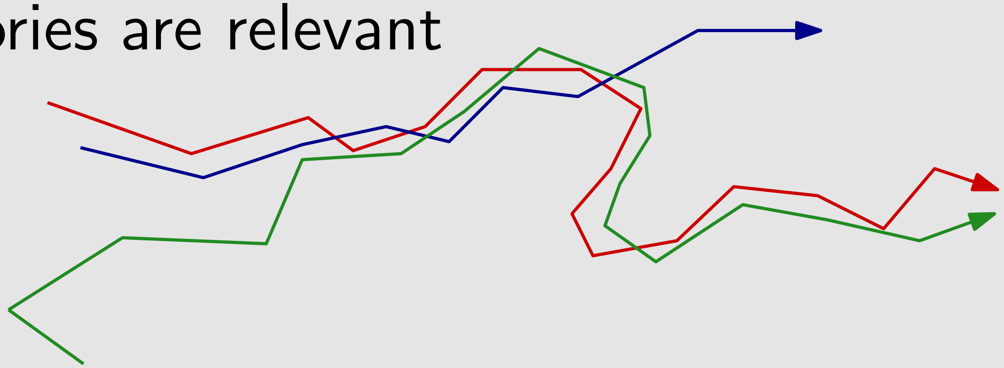
- GPS data, RFID tags, surveillance cameras, simulations

Movement data can be considered in two ways:

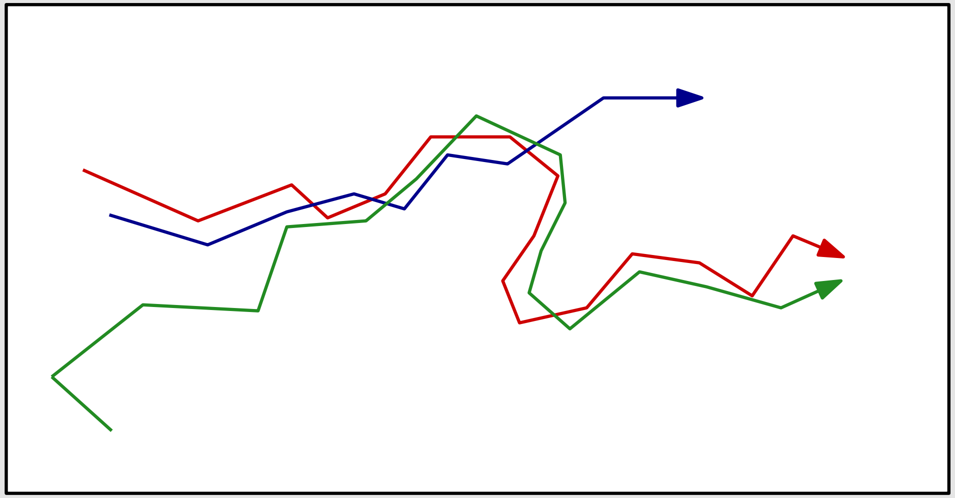
- Individual locations, which change over time, are relevant



- Trajectories are relevant



Analyzing trajectories: applications



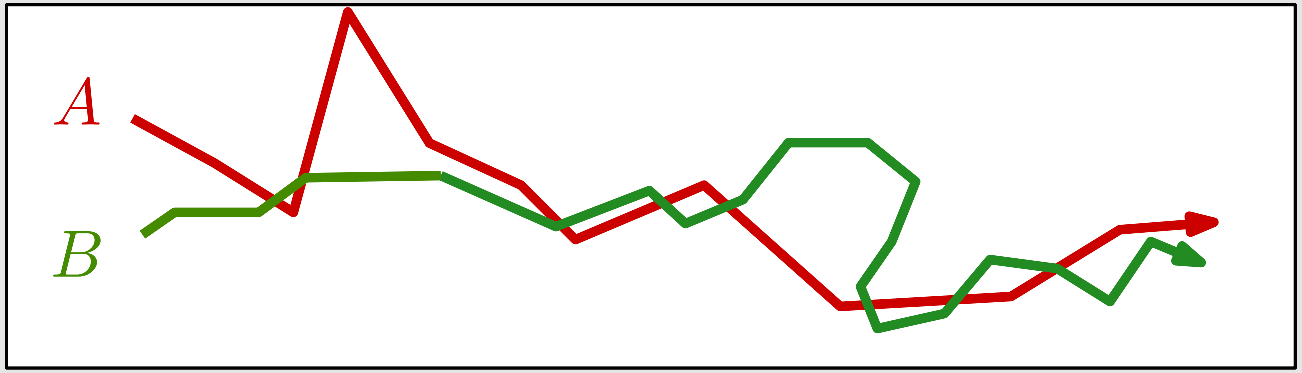
- behavorial studies on animals
- analysis of sports games
- analyzing traffic

Analyzing trajectories: examples of interesting questions

- given two trajectories, how similar are they?
- given a collection of trajectories, report all subtrajectories similar to a query trajectory
- is there a certain motion pattern? (leader following, flocking, . . .)

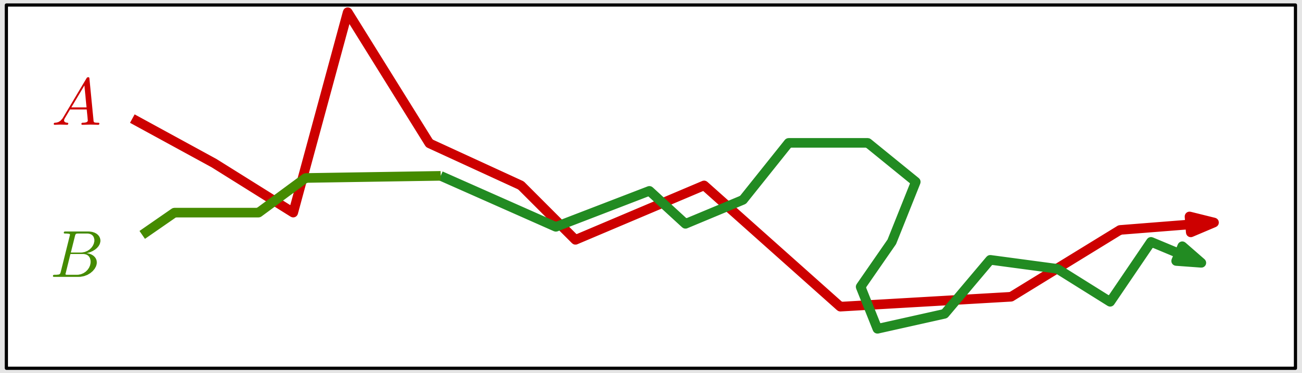
Basic question: how to measure similarity of two trajectories

We need a distance function *dist*$(\cdot,\cdot)$ on the trajectories:
the smaller *dist*(A, B), the more similar A and B are



Basic question: how to measure similarity of two trajectories

We need a distance function $\mathit{dist}(\cdot,\cdot)$ on the trajectories:
the smaller $\mathit{dist}(A,B)$, the more similar A and B are

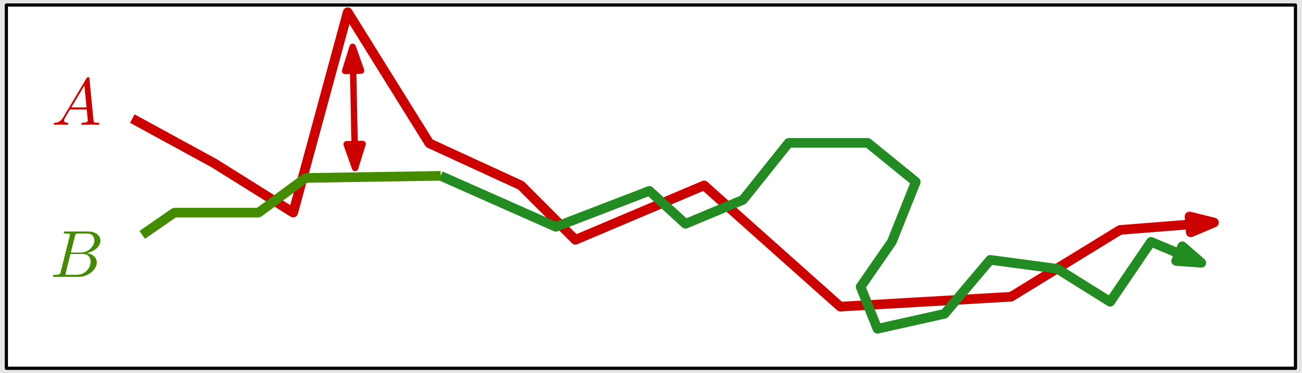


First attempt: Hausdorff distance

$$\mathit{dist}(A,B) \;=\; \max\left(\max_{a\in A}\min_{b\in B} d(a,b),\; \max_{b\in B}\min_{a\in a} d(b,a)\right)$$

Basic question: how to measure similarity of two trajectories

We need a distance function $\mathit{dist}(\cdot,\cdot)$ on the trajectories: the smaller $\mathit{dist}(A,B)$, the more similar A and B are

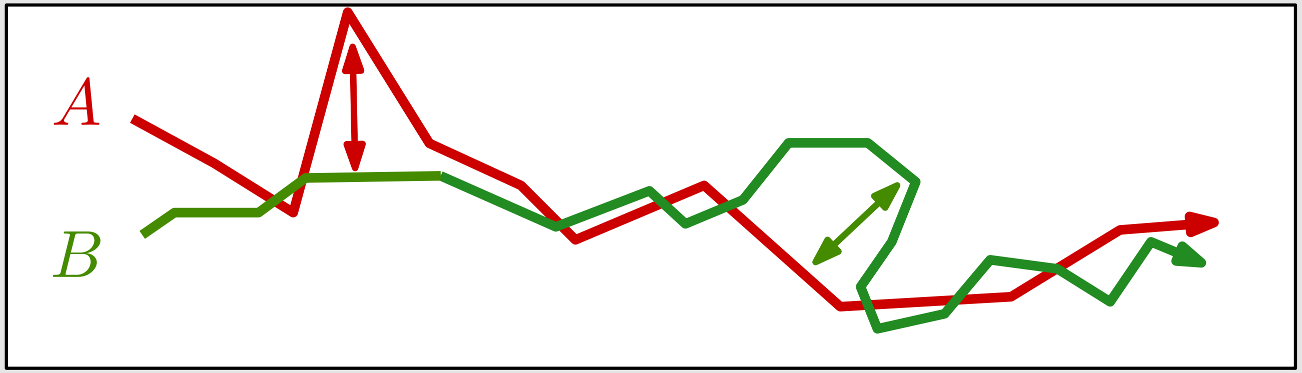


First attempt: Hausdorff distance

$$\mathit{dist}(A,B) \;=\; \max\left(\max_{a\in A}\min_{b\in B} d(a,b),\; \max_{b\in B}\min_{a\in a} d(b,a)\right)$$

Basic question: how to measure similarity of two trajectories

We need a distance function *dist*$(\cdot,\cdot)$ on the trajectories:
the smaller *dist*(A, B), the more similar A and B are



First attempt: Hausdorff distance

$$\mathit{dist}(A, B) = \max\left(\max_{a \in A} \min_{b \in B} d(a, b), \max_{b \in B} \min_{a \in a} d(b, a) \right)$$

Is the Hausdorff distance a good measure for trajectory similarity?

Hausdorff distance:

Is the Hausdorff distance a good measure for trajectory similarity?

Hausdorff distance:

Is the Hausdorff distance a good measure for trajectory similarity?

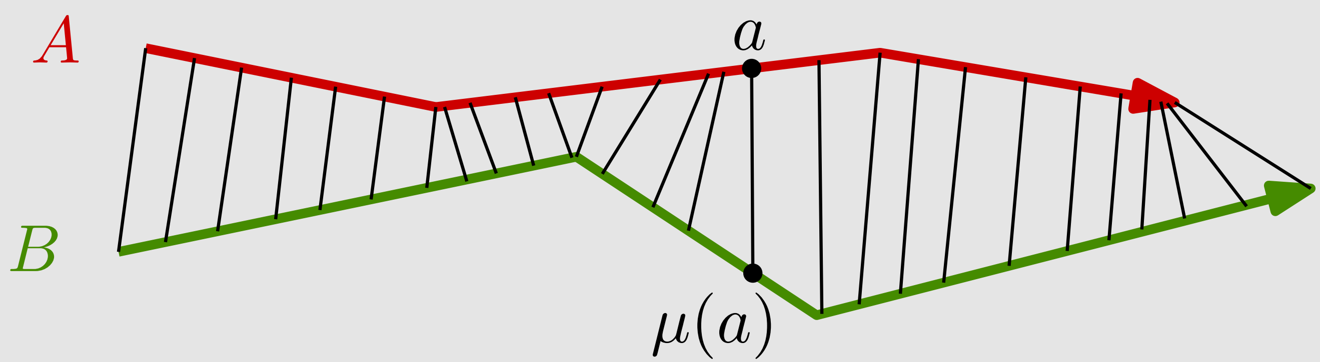
Hausdorff distance:

Hausdorff distance can be small, even when trajectories are very different

Hausdorff distance does not take continuity of trajectories into account

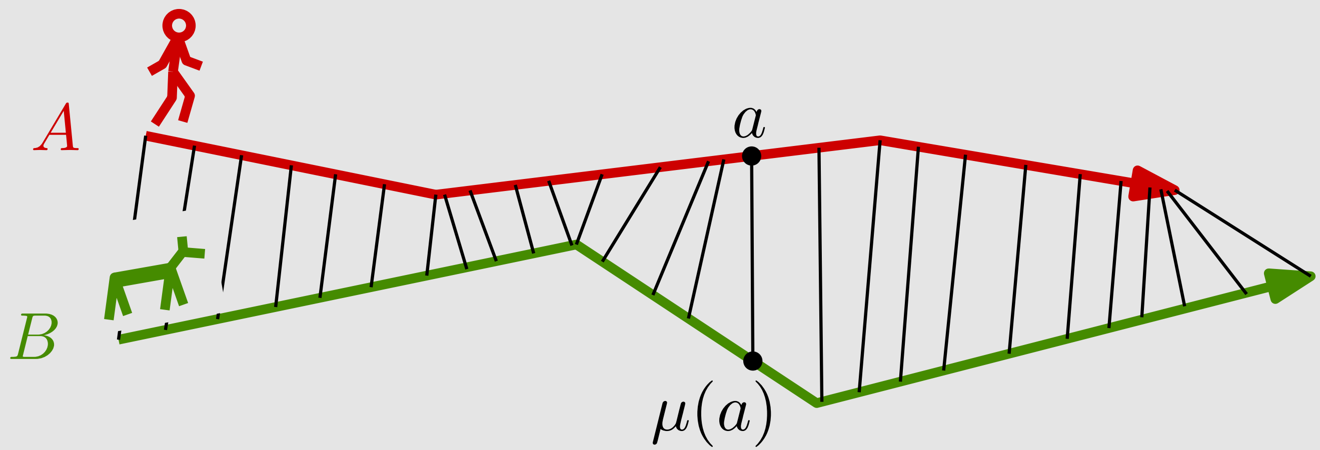
A better measure: Fréchet distance

- take continuous mapping $\mu : A \rightarrow B$
- take as distance $\max_{a \in A} d(a, \mu(a))$



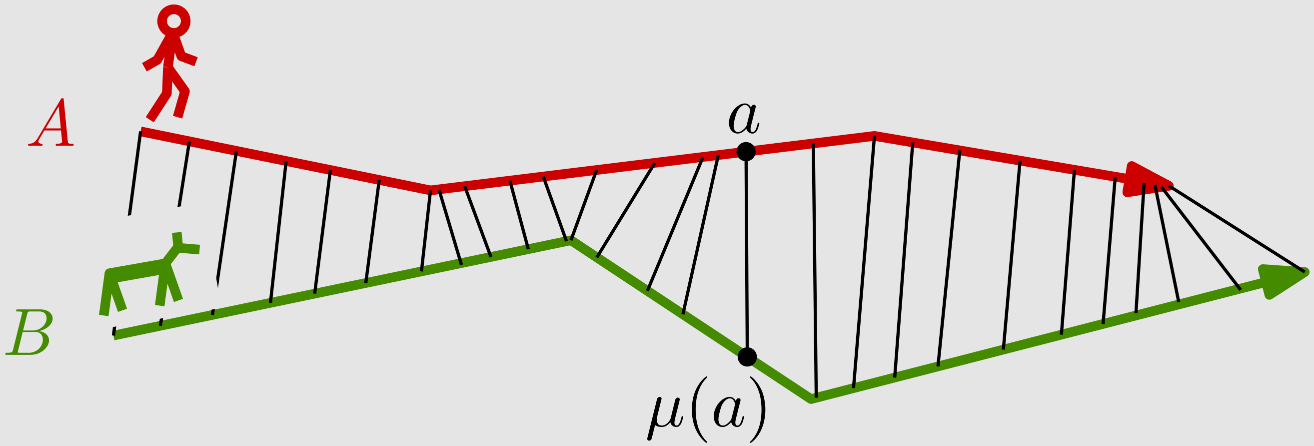
A better measure: Fréchet distance

- take continuous mapping $\mu : A \rightarrow B$
- take as distance $\max_{a \in A} d(a, \mu(a))$



A better measure: Fréchet distance

- take continuous mapping $\mu : A \rightarrow B$
- take as distance $\max_{a \in A} d(a, \mu(a))$

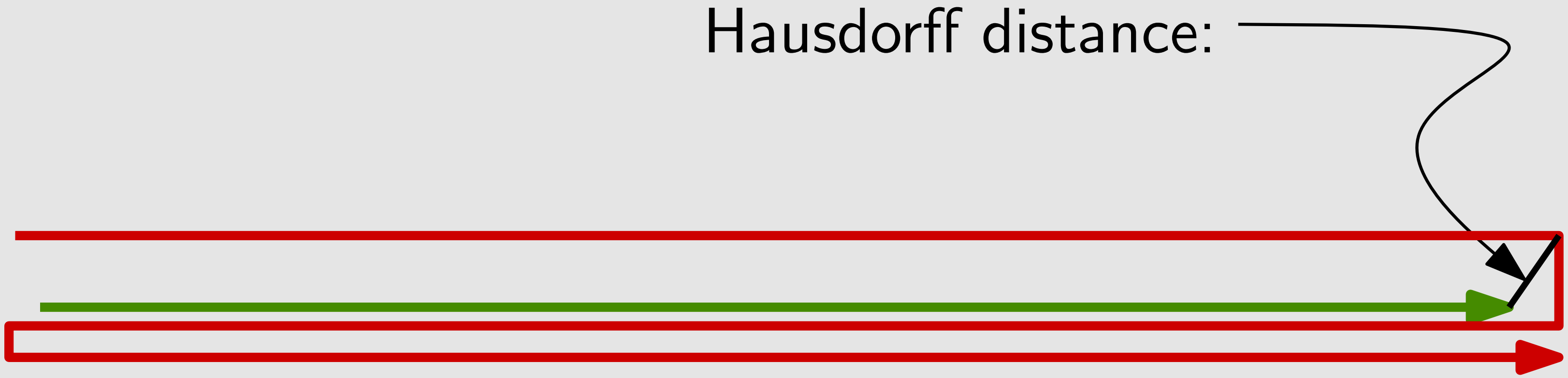


Which mapping? Take the best one:

$$\text{Fréchet-dist}(A, B) = \min_{\mu} \max_{a \in A} d(a, \mu(a)),$$

where min is taken over all continuous 1-to-1 mappings

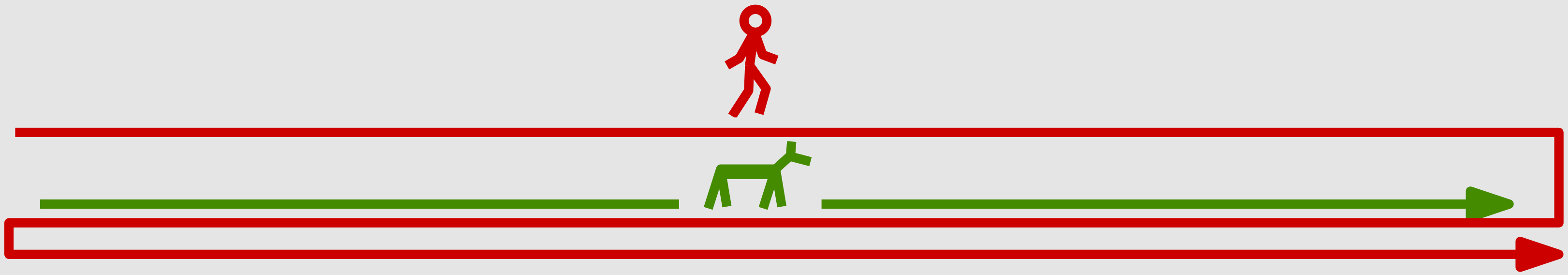
Hausdorff distance:


Hausdorff distance:

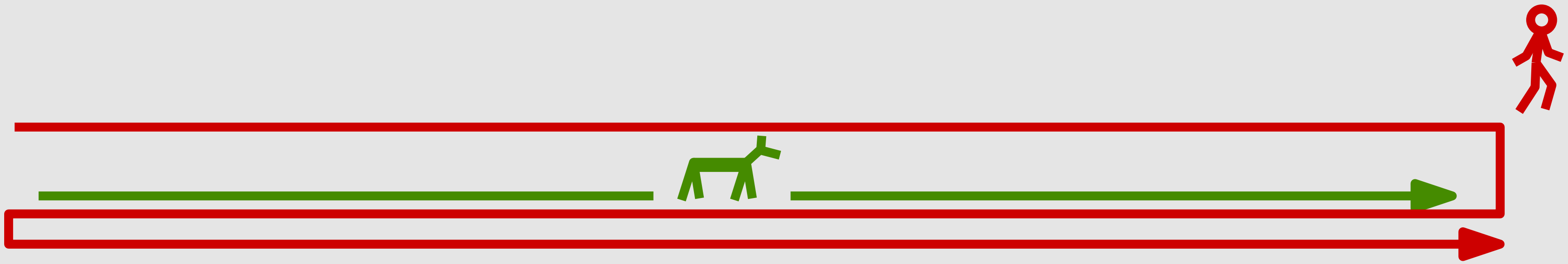
Fréchet distance:

Fréchet distance:

TU/e

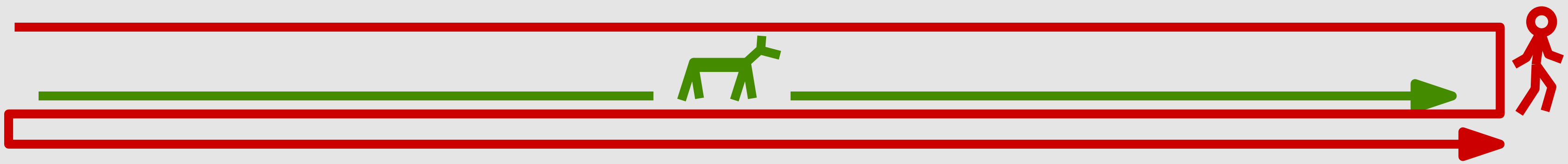


Fréchet distance:

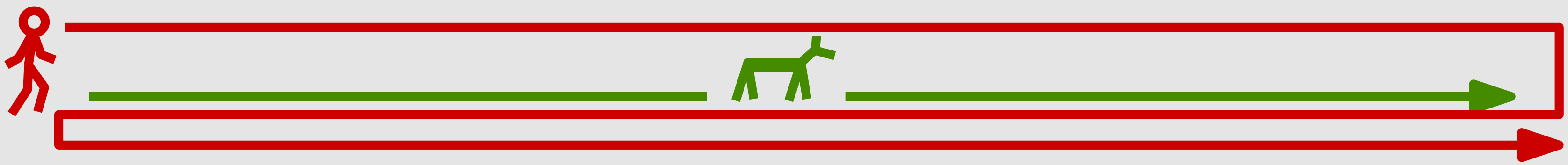


Fréchet distance:

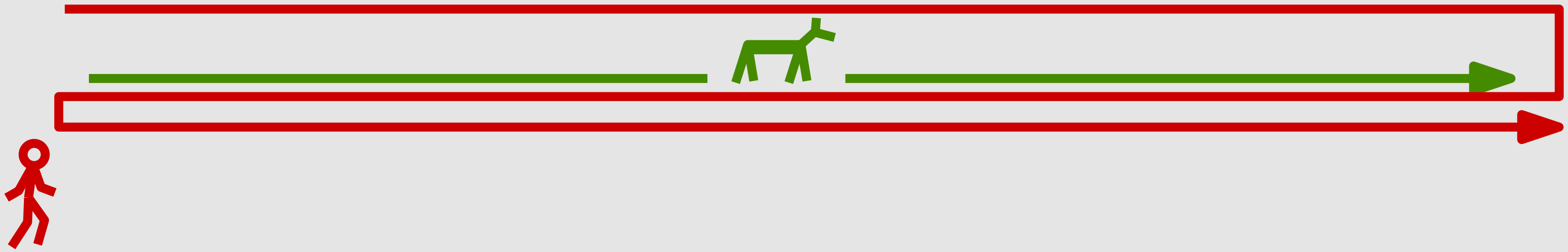
TU/e



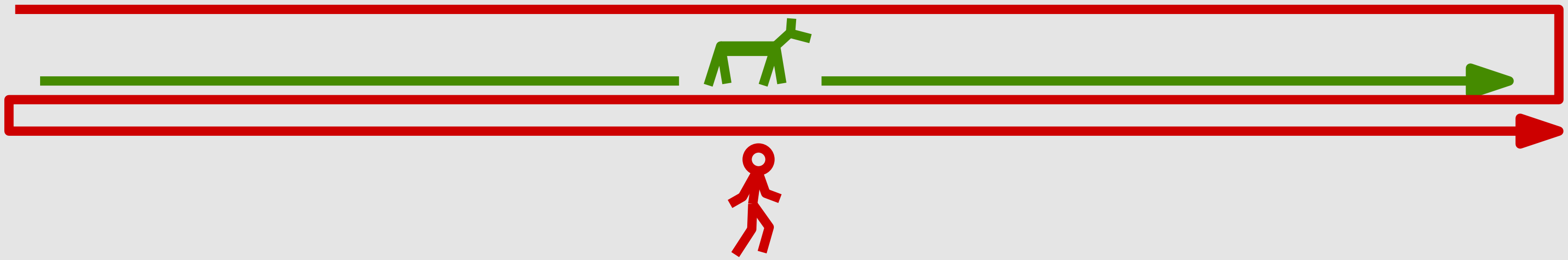
Fréchet distance:



Fréchet distance:

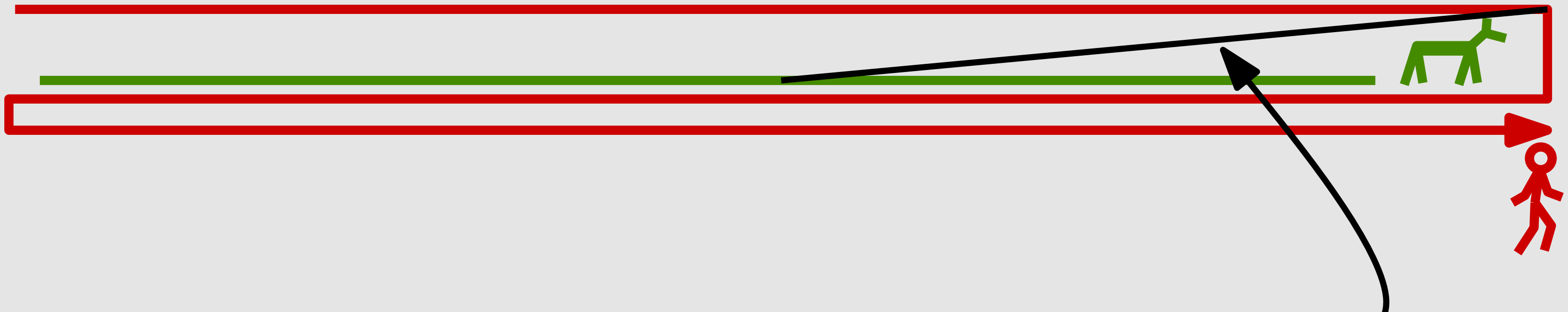


Fréchet distance:



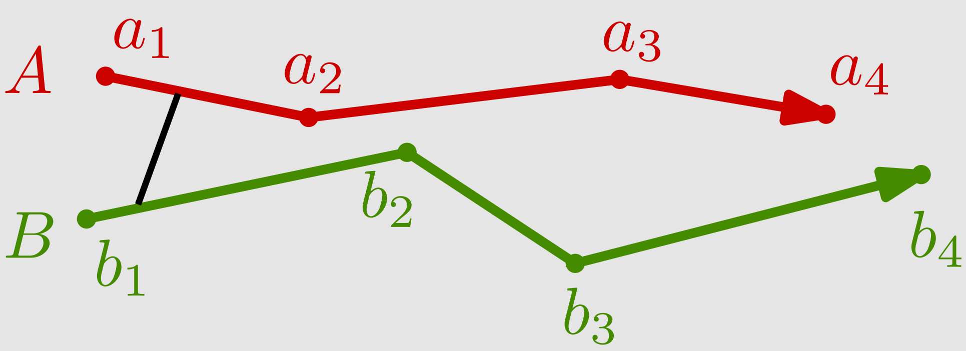
Fréchet distance:

Fréchet distance:

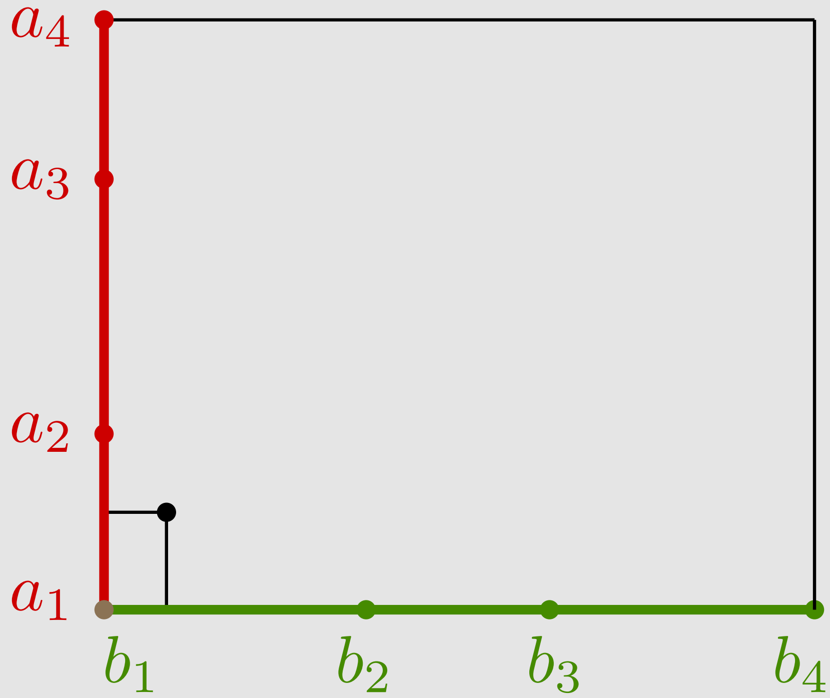


Fréchet distance:

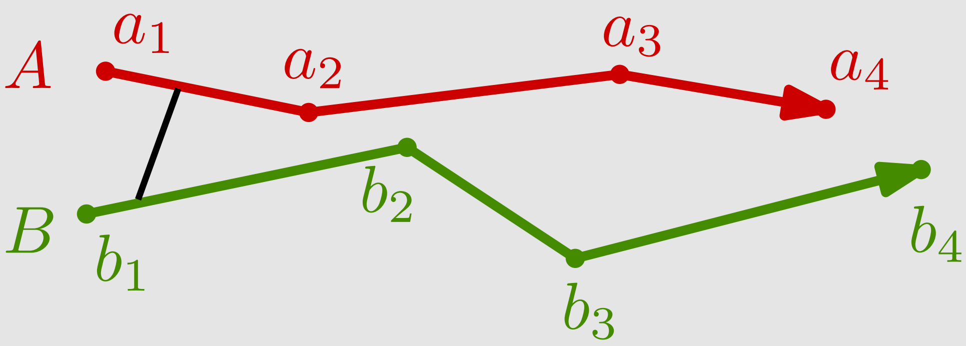
Computing similarity: deciding if Fréchet-dist$(A, B) \leq \varepsilon$



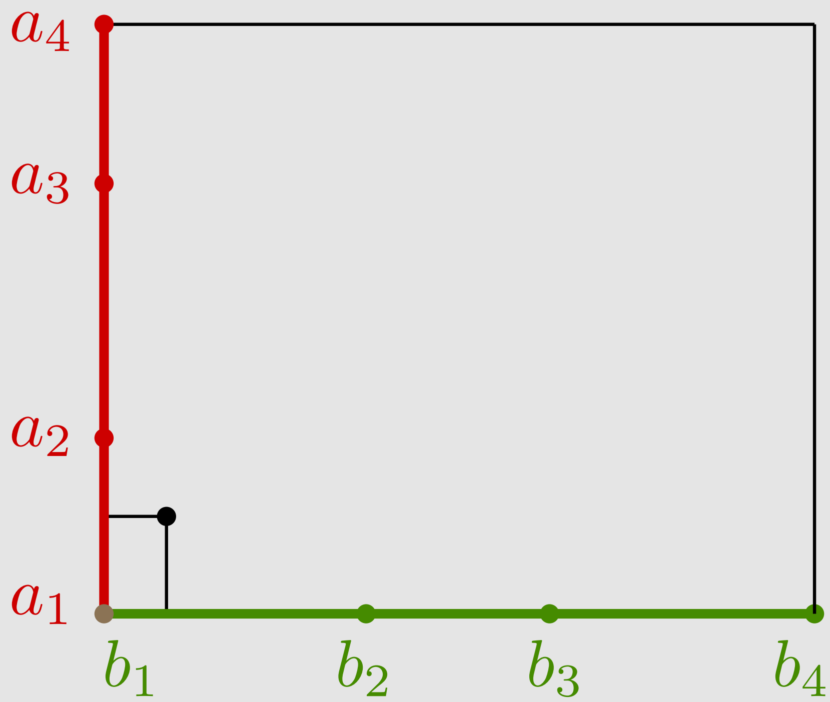
free-space diagram (for given ε)



Computing similarity: deciding if Fréchet-dist$(A, B) \leq \varepsilon$

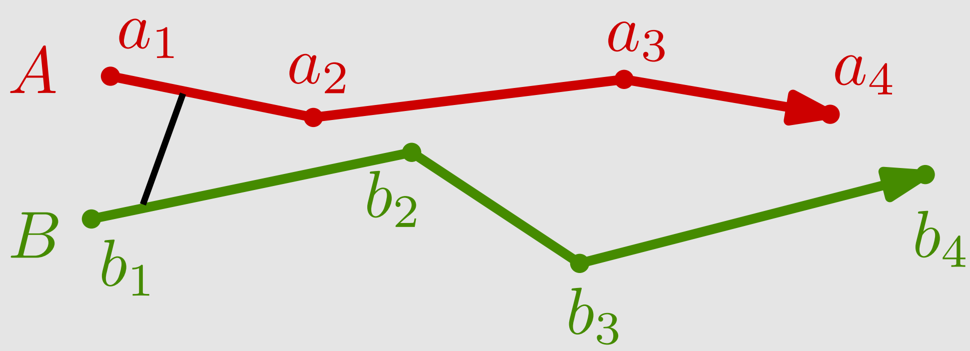


free-space diagram (for given ε)

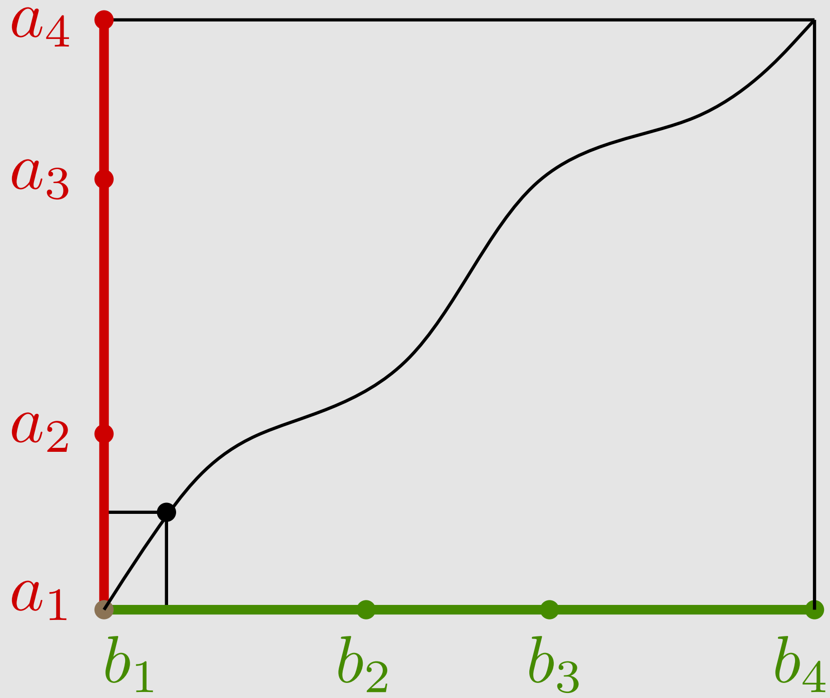


what does continuous 1-to-1 mapping correspond to in diagram?

Computing similarity: deciding if Fréchet-dist$(A, B) \leq \varepsilon$



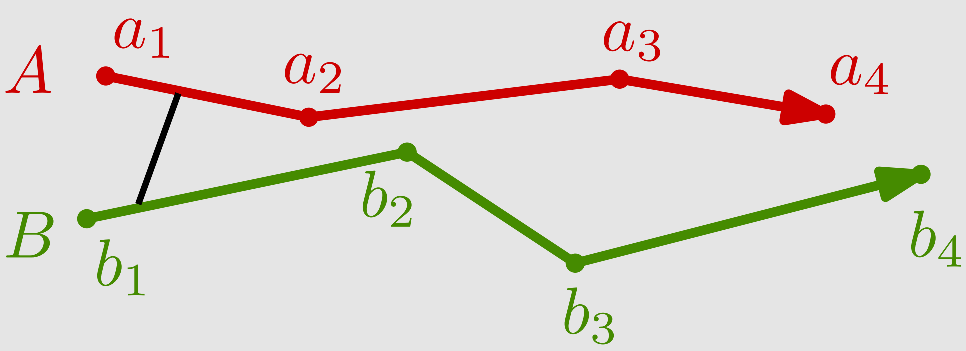
free-space diagram (for given ε)



what does continuous 1-to-1 mapping correspond to in diagram?

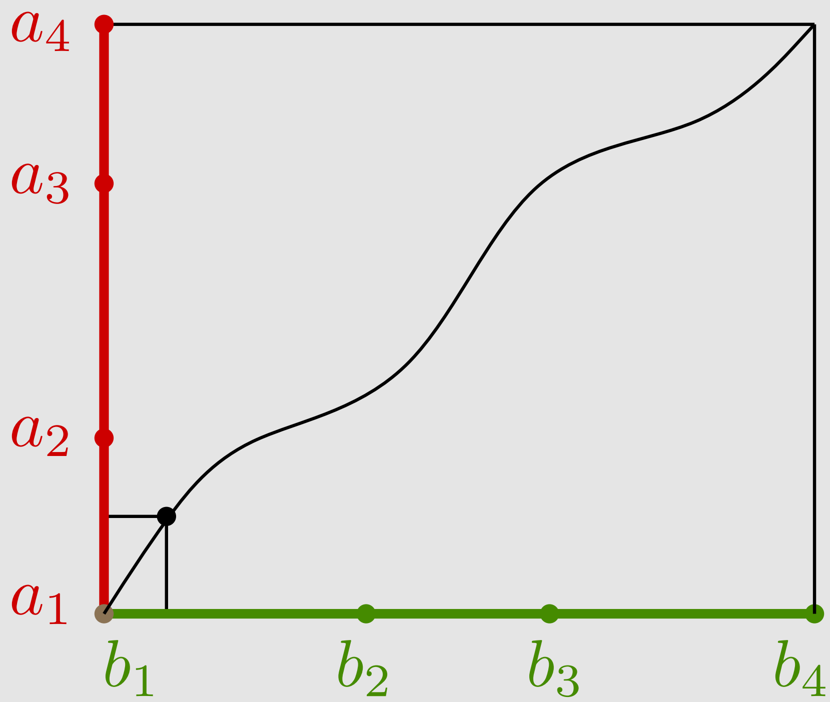
xy-monotone path from lower left to upper right

Computing similarity: deciding if Fréchet-dist$(A, B) \leq \varepsilon$



must decide if monotone path exists such that distance is always $\leq \varepsilon$

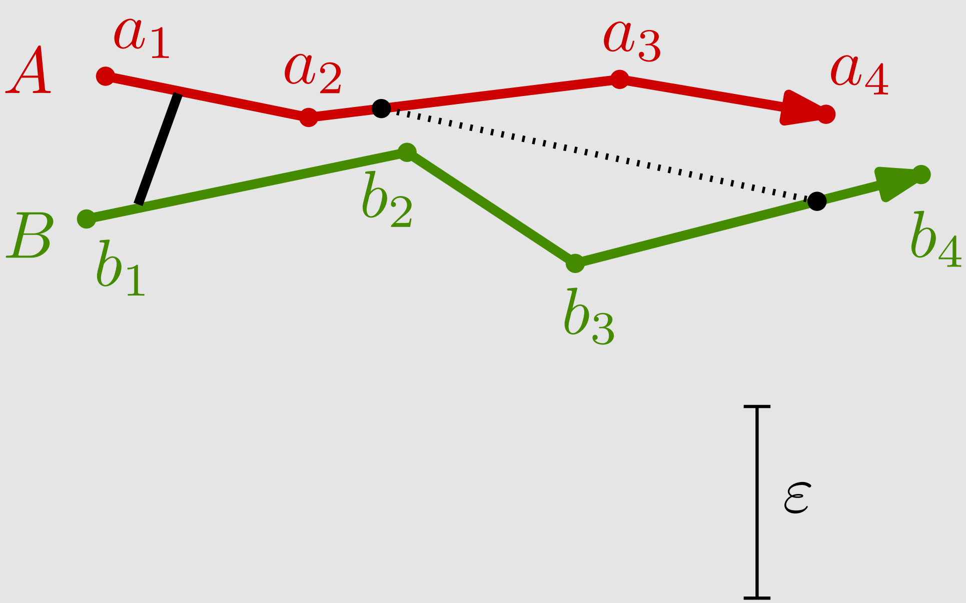
free-space diagram (for given ε)



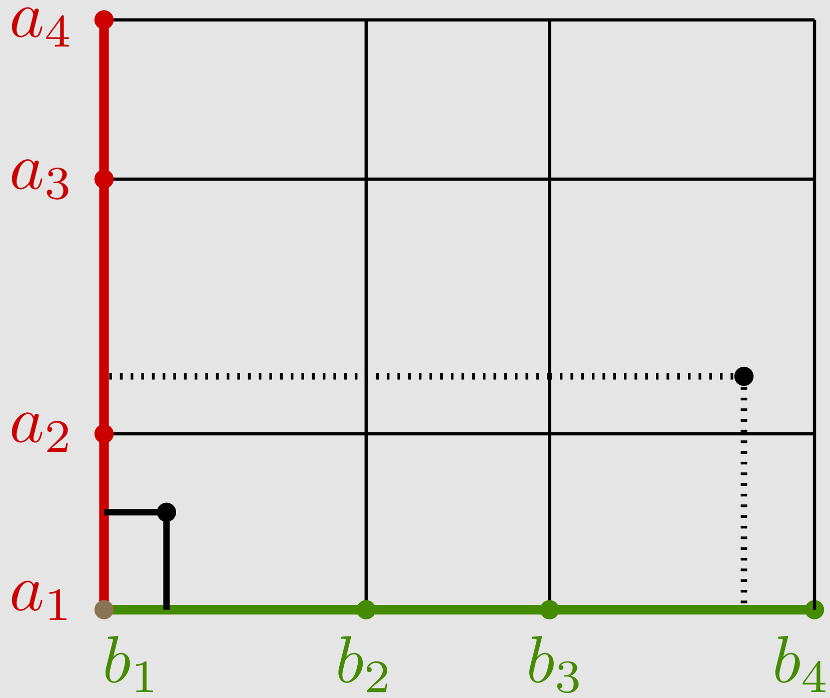
what does continuous 1-to-1 mapping correspond to in diagram?

xy-monotone path from lower left to upper right

Computing similarity: deciding if Fréchet-dist$(A, B) \leq \varepsilon$

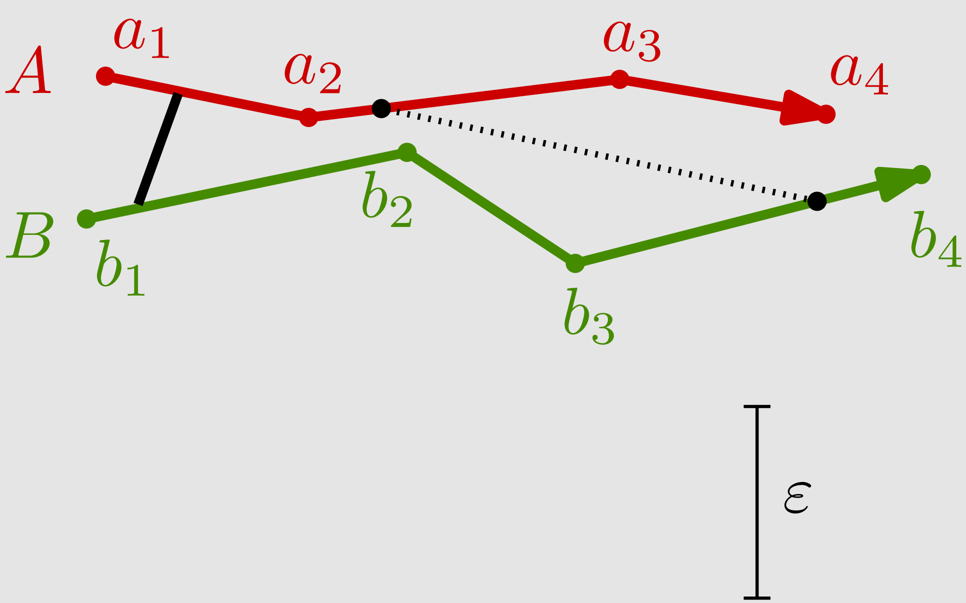


free-space diagram (for given ε)

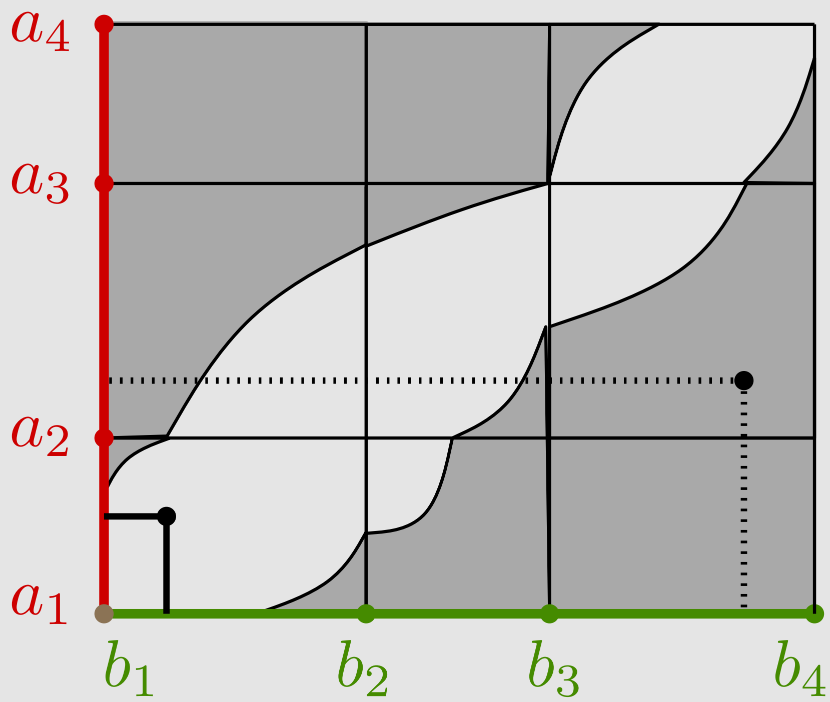


forbidden region in diagram:
(a, b) such that $d(a, b) > \varepsilon$

Computing similarity: deciding if Fréchet-dist$(A, B) \leq \varepsilon$

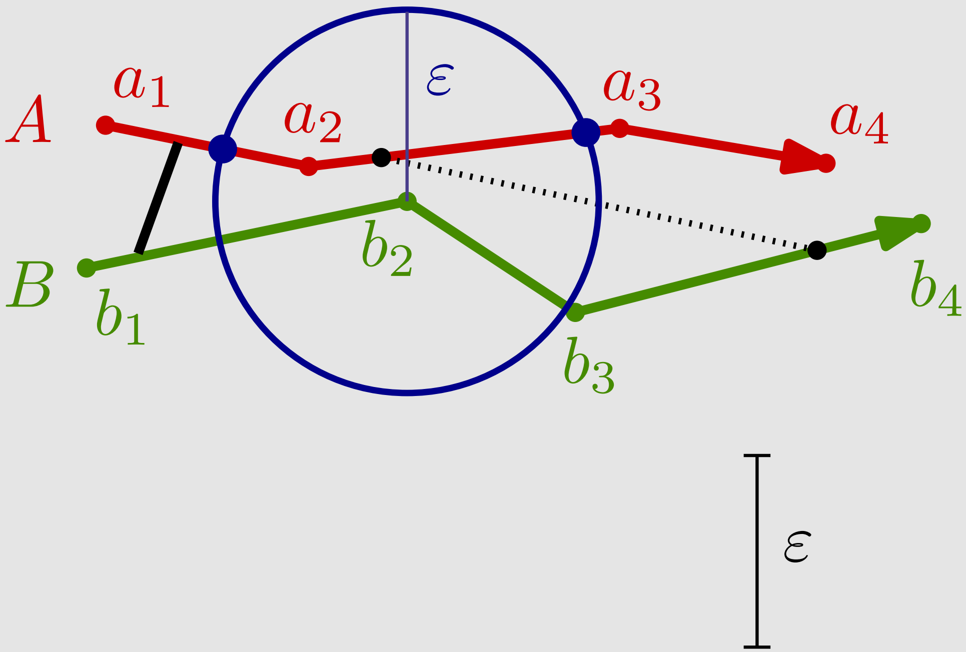


free-space diagram (for given ε)

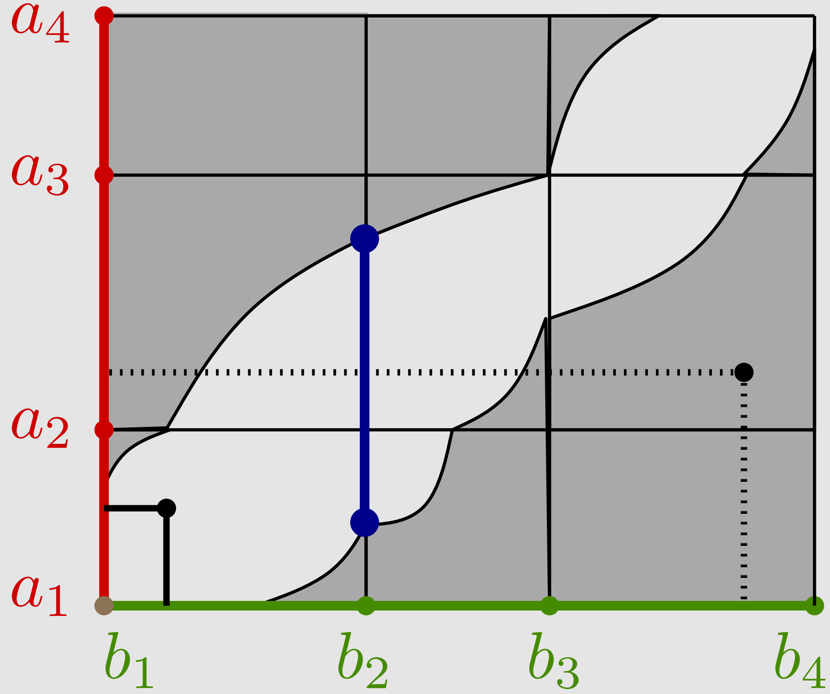


forbidden region in diagram:
(a, b) such that $d(a, b) > \varepsilon$

Computing similarity: deciding if Fréchet-dist$(A, B) \leq \varepsilon$

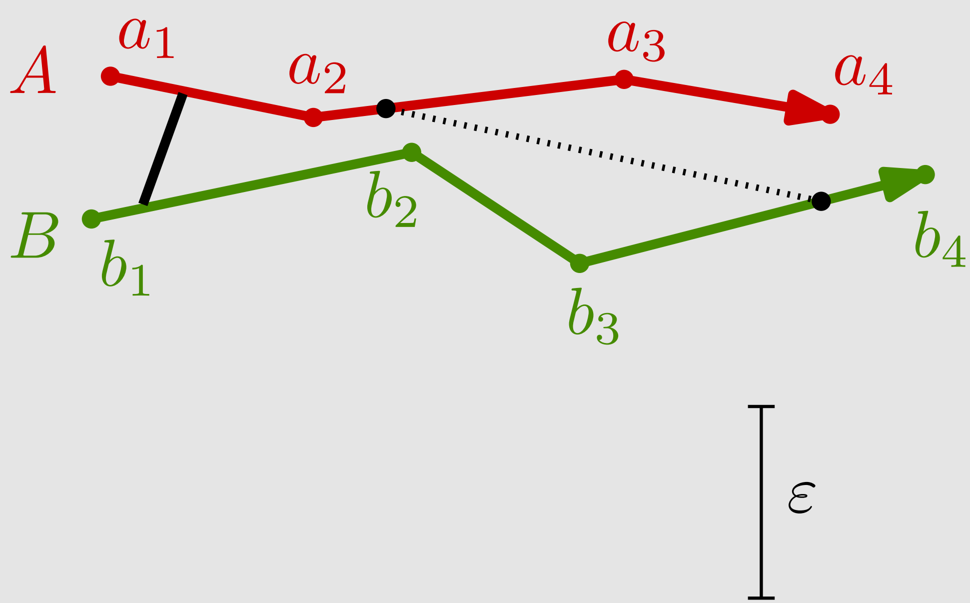


free-space diagram (for given ε)

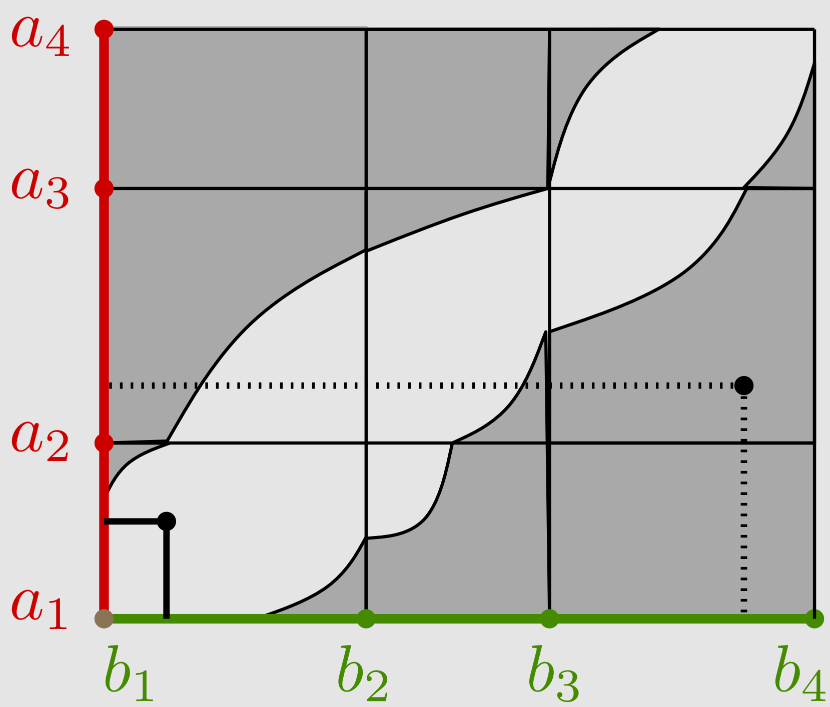


forbidden region in diagram:
(a, b) such that $d(a, b) > \varepsilon$

Computing similarity: deciding if Fréchet-dist$(A, B) \leq \varepsilon$



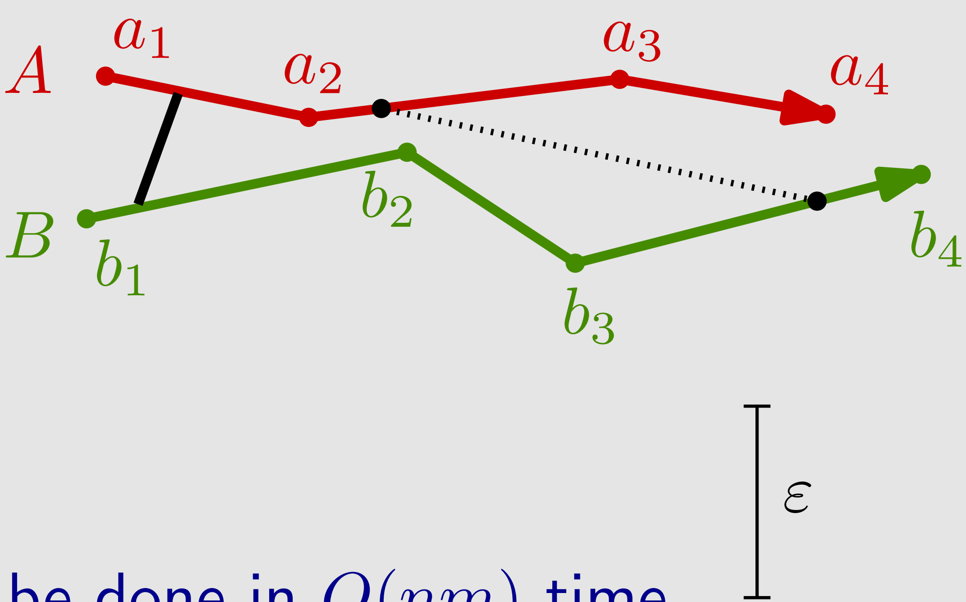
free-space diagram (for given ε)



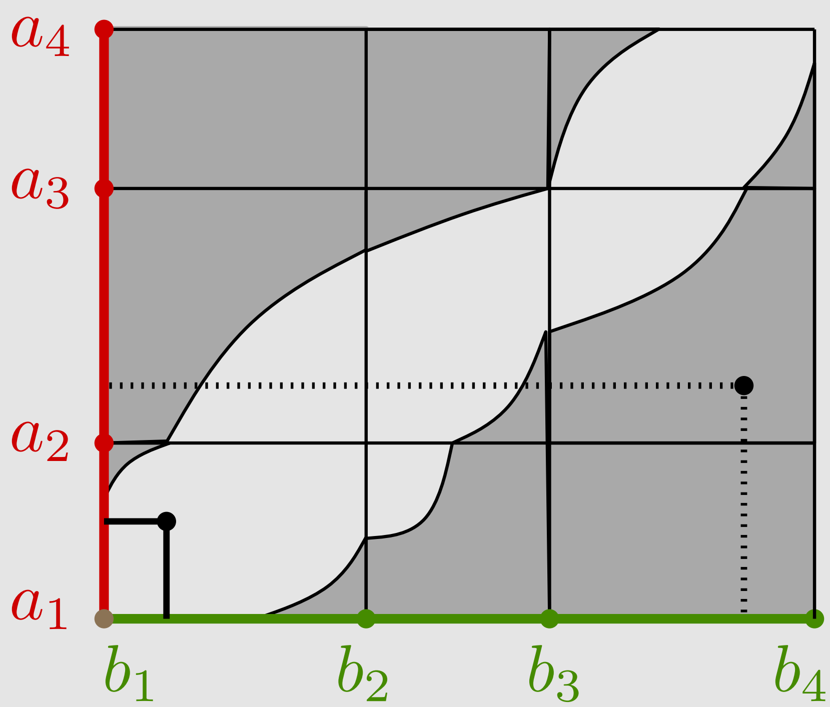
Algorithm:
Compute decomposition into free and forbidden regions, and check if monotone path exists in free region.

forbidden region in diagram:
(a, b) such that $d(a, b) > \varepsilon$

Computing similarity: deciding if Fréchet-dist$(A, B) \leq \varepsilon$



free-space diagram (for given ε)

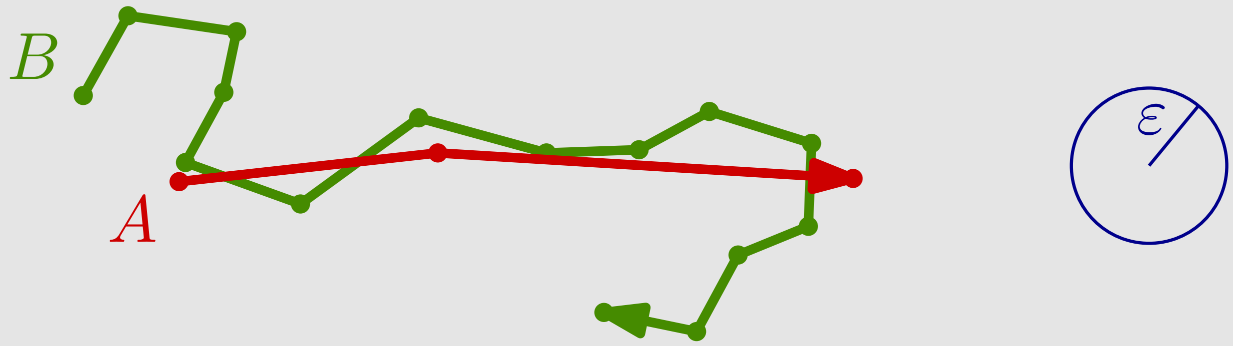


Can be done in $O(nm)$ time

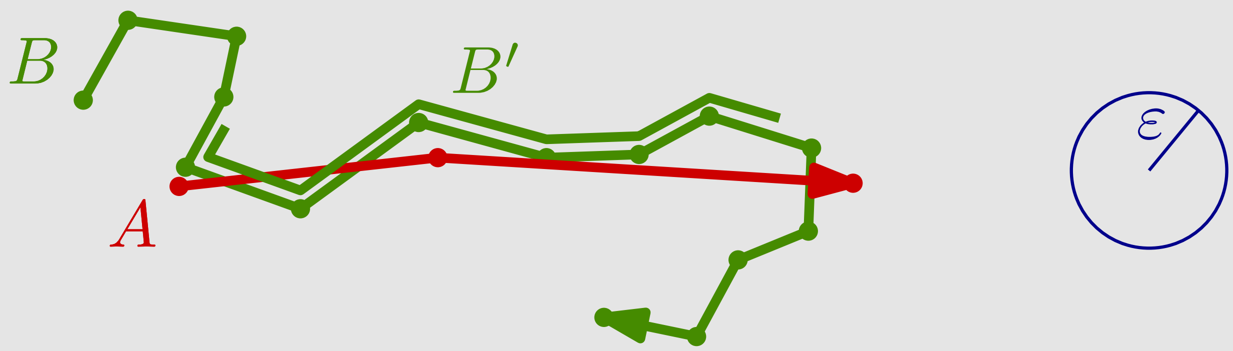
Algorithm:
Compute decomposition into free and forbidden regions, and check if monotone path exists in free region.

forbidden region in diagram:
(a, b) such that $d(a, b) > \varepsilon$

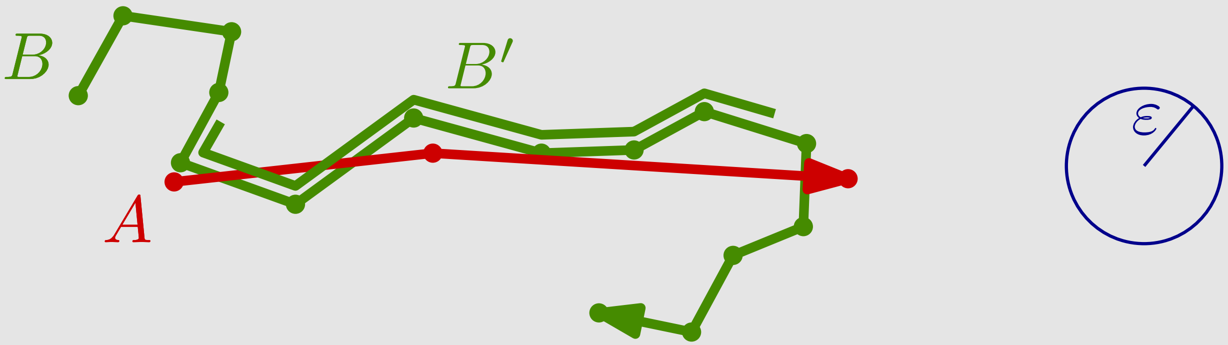
Partial matching: what if we want to decide if there is a connected subcurve of B similar to A?

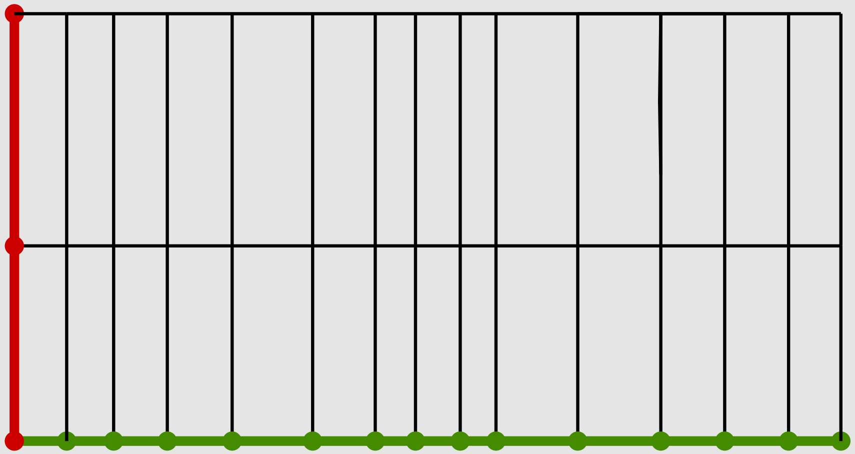


Partial matching: what if we want to decide if there is a connected subcurve of B similar to A?

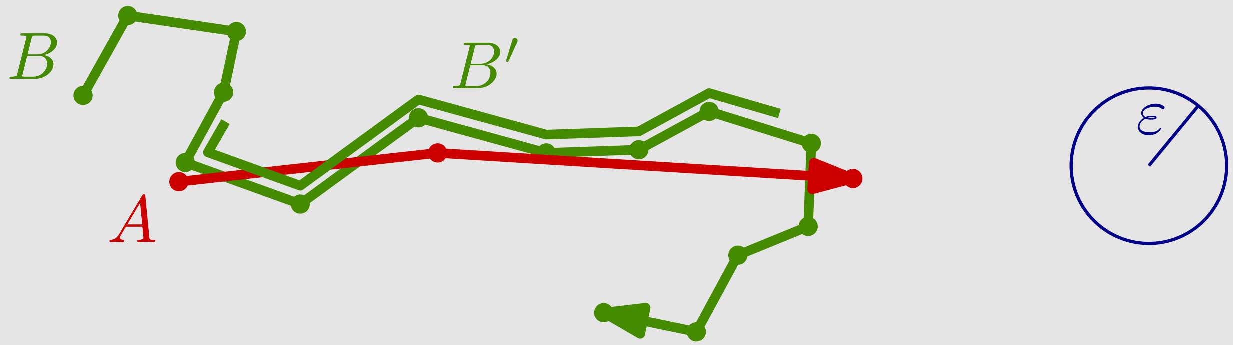


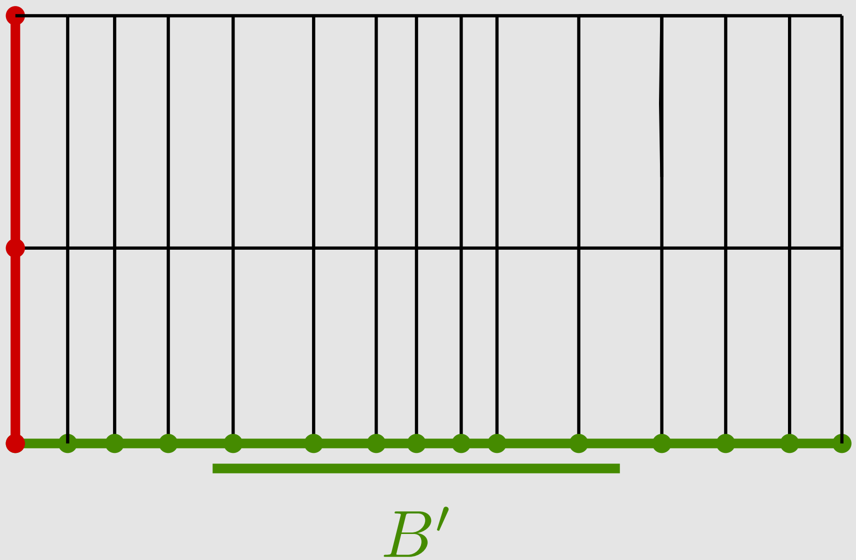
Partial matching: what if we want to decide if there is a connected subcurve of B similar to A?



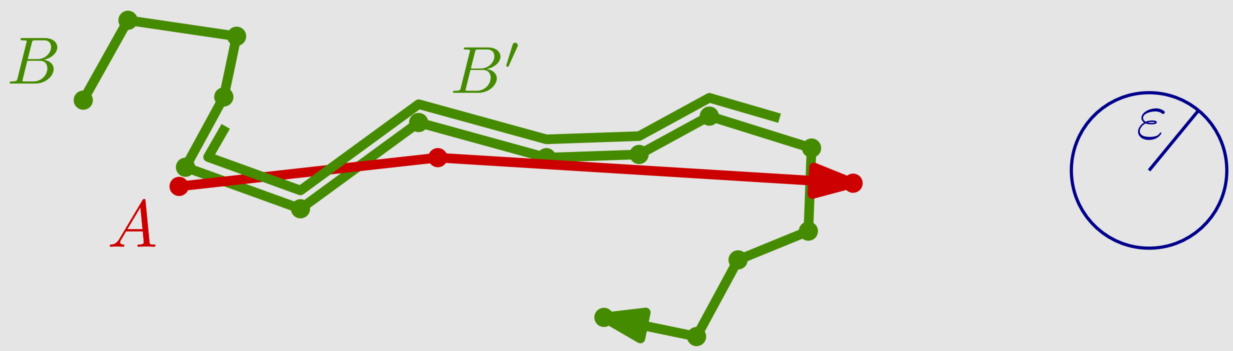


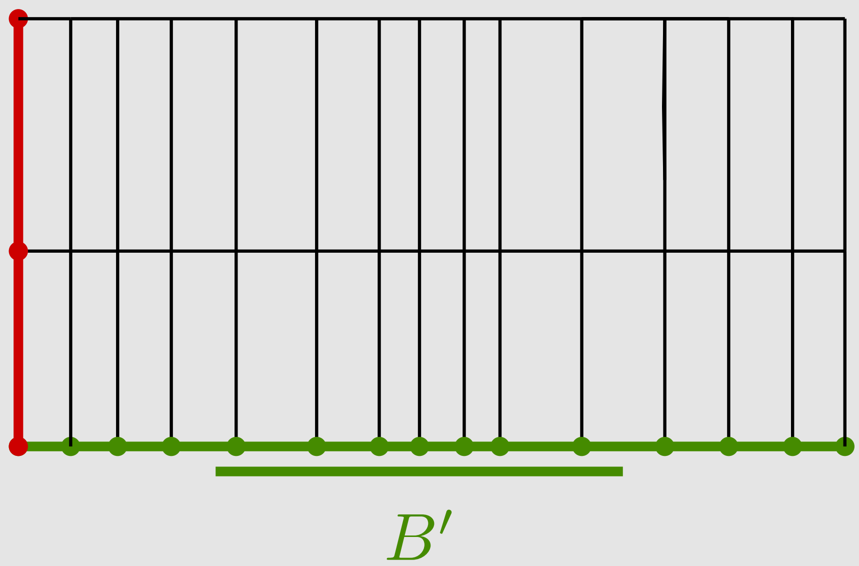
Partial matching: what if we want to decide if there is a connected **subcurve** of B similar to A?





Partial matching: what if we want to decide if there is a connected subcurve of B similar to A?

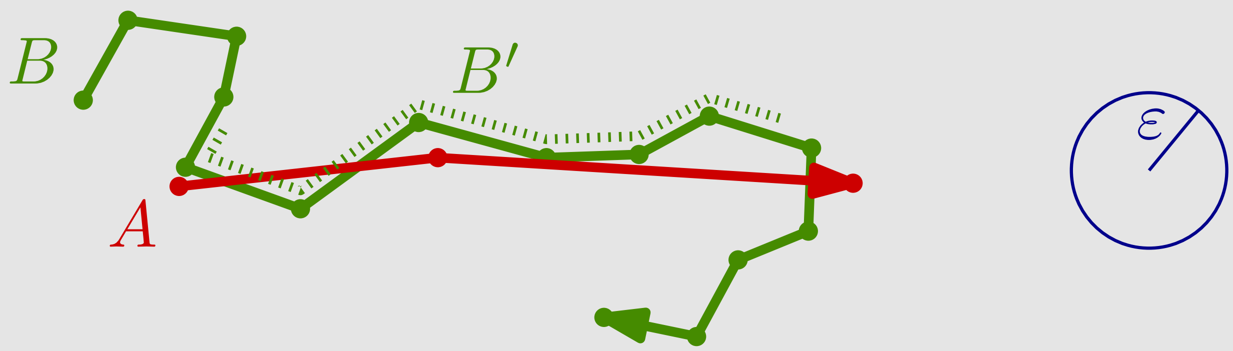


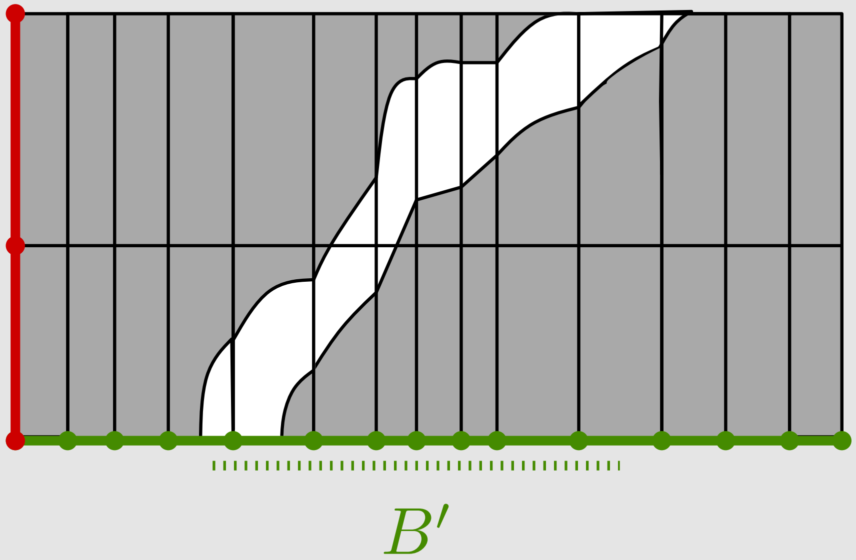


mapping from $B' \subseteq B$ to A

path from bottom to top in diagram

Partial matching: what if we want to decide if there is a connected subcurve of B that is similar to A?



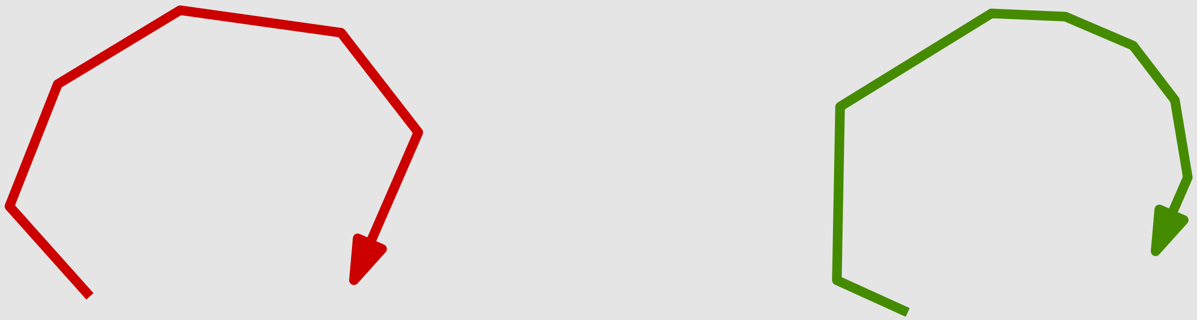


mapping from $B' \subseteq B$ to A

$\Updownarrow$

path from bottom to top in diagram

what if exact locations are not important?

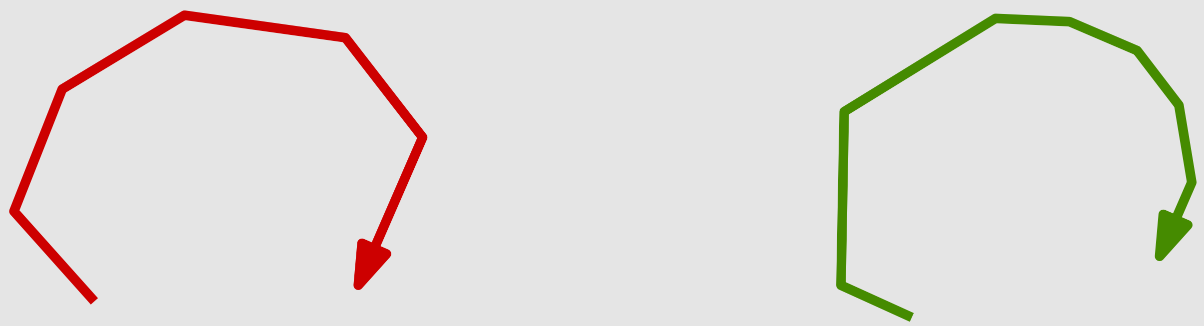


Recall definition of Fréchet distance:

$$\text{Fréchet-dist}(A, B) = \min_{\mu} \max_{a \in A} d(a, \mu(a)),$$

where min is taken over all continuous 1-to-1 mappings

what if exact locations are not important?



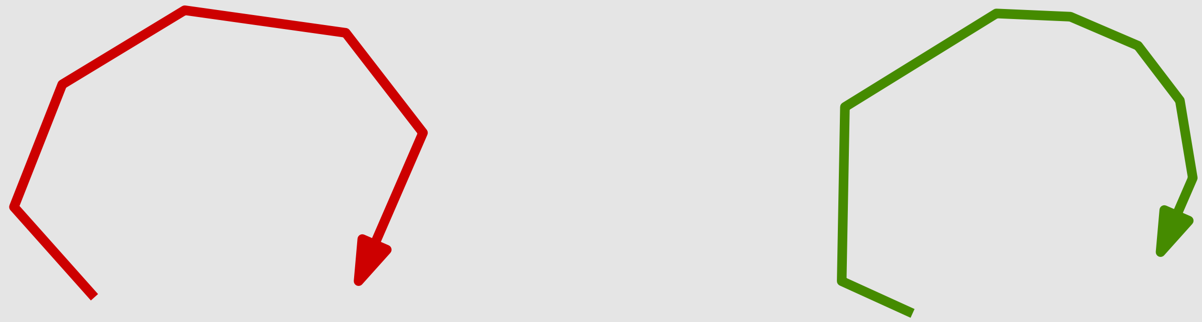
Recall definition of Fréchet distance:

$$\text{Fréchet-dist}(A, B) = \min_{\mu} \max_{a \in A} \boxed{d(a, \mu(a))},$$

where min is taken over all continuous 1-to-1 mappings

Euclidean distance

what if exact locations are not important?



Recall definition of Fréchet distance:

$$\text{Fréchet-dist}(A, B) = \min_{\mu} \max_{a \in A} d(a, \mu(a)),$$

where min is taken over all continuous 1-to-1 mappings

difference in direction of movement ~~Euclidean distance~~

Fr'echet distance: summary

- good measuring similarity of curves (e.g. trajectories)
- can be computed fairly efficiently for polygonal curves
- many variations (partial matching, directional Fréchet distance, . . .)