2IL76
Algorithms for Geographic Data

Spring 2015

Lecture 1b: Similarity

Trajectories

- Model for the movement of a (point) object;

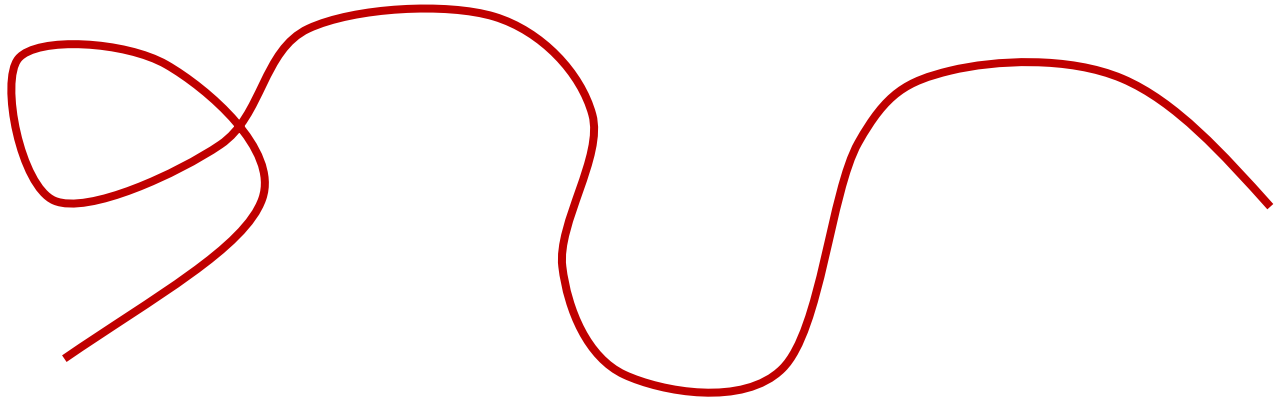
  f : [time interval] → 2D or 3D

Trajectories

- Model for the movement of a (point) object;

  f : [time interval] → 2D or 3D

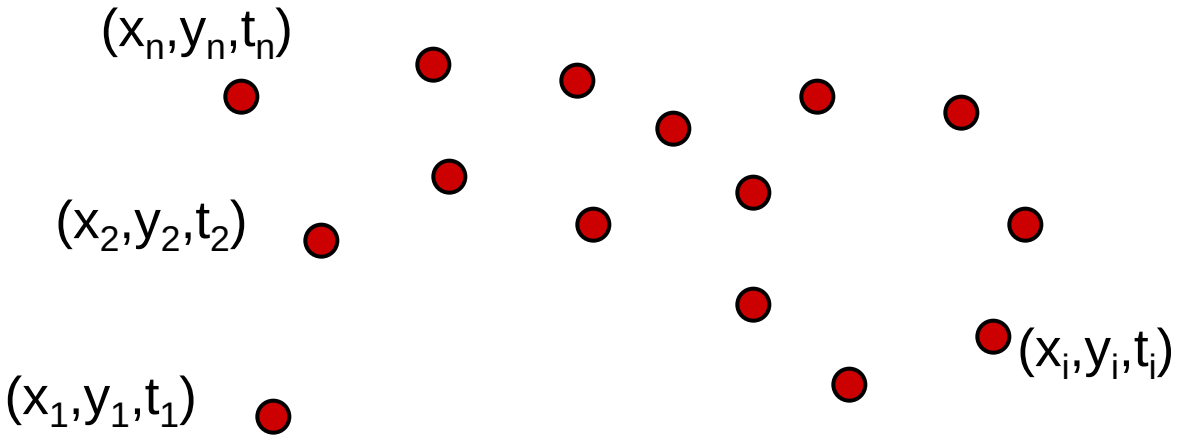
- The path of a trajectory is just any curve



Trajectories vs trajectory data

Trajectory data

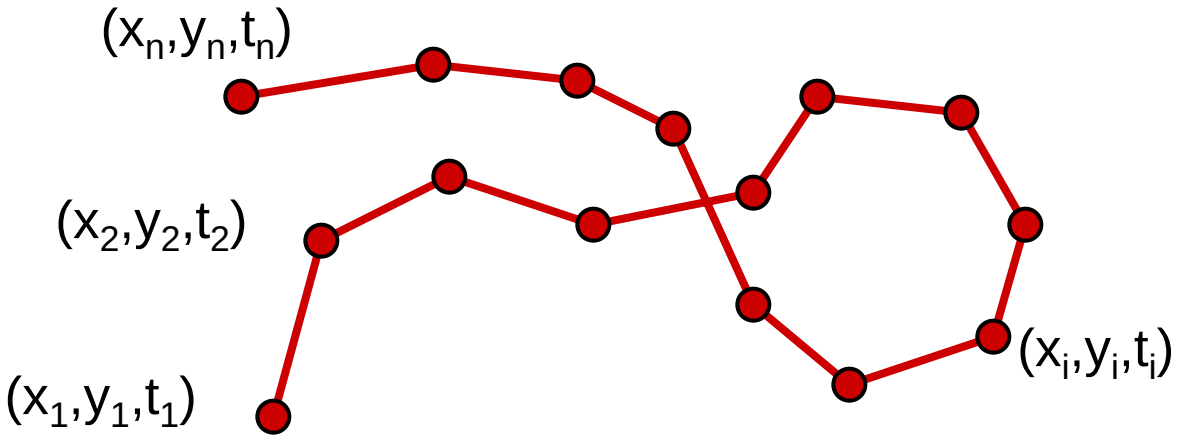
- The data as it is acquired by e.g. GPS:
  sequence of triples (spatial plus time-stamp);
  quadruples for trajectories in 3D



Trajectory data

- Typical assumption for sufficiently densely sampled data: constant velocity between consecutive samples

  ➡ velocity/speed is a piecewise constant function



Trajectory data analysis

Questions

“How much time does a gull typically spend foraging on a trip from the colony and back?”

Trajectory data analysis

Questions

"How much time does a gull typically spend foraging on a trip from the colony and back?"

"If customers look at book display X, do they more often than average also go to and look at bookshelf Y?"

Trajectory data analysis

Questions

"How much time does a gull typically spend foraging on a trip from the colony and back?"

"If customers look at book display X, do they more often than average also go to and look at bookshelf Y?"

"Is this Alzheimer patient in need of assistance?"

Abstract / general purpose questions

Single trajectory

- simplification, cleaning
- segmentation into semantically meaningful parts
- finding recurring patterns (repeated subtrajectories)

Two trajectories

- similarity computation
- subtrajectory similarity

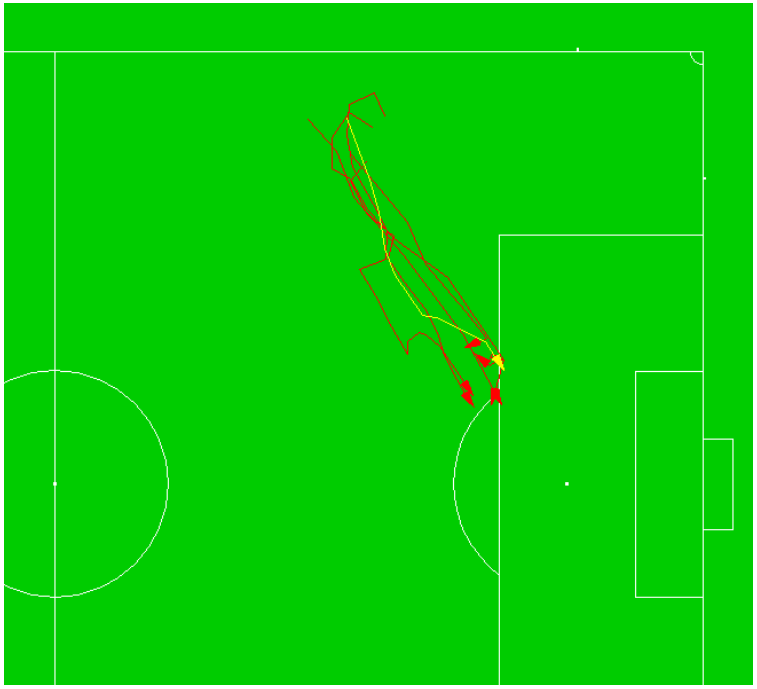
Multiple trajectories

- clustering, outliers
- flocking/grouping pattern detection
- finding a typical trajectory or computing a mean/median trajectory
- visualization

Fundamental tools: clustering

When are two trajectories similar?

In many cases the spatial similarity is much more important than the temporal aspect.

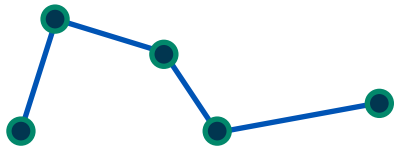


Similarity of curves … many measures

- Agrawal, Lin, Sawhney & Shim, 1995
- Alt & Godau, 1995
- Yi, Jagadish & Faloutsos, 1998
- Gaffney & Smyth, 1999
- Kalpakis, Gada & Puttagunta, 2001
- Vlachos, Gunopulos & Kollios, 2002
- Gunopoulos, 2002
- Mamoulis, Cao, Kollios, Hadjieleftheriou, Tao & Cheung, 2004
- Chen, Ozsu & Oria, 2005
- Nanni & Pedreschi, 2006
- Van Kreveld & Luo, 2007
- Gaffney, Robertson, Smyth, Camargo & Ghil, 2007
- Lee, Han & Whang, 2007
- Buchin, Buchin, van Kreveld & Luo, 2009
- Agarwal, Aranov, van Kreveld, Löffler & Silveira, 2010
- Sankaraman, Agarwal, Molhave, Pan and Boedihardjo, 2013
- ...

Similarity measures

- Input: Two polygonal curves P and Q of size m in R^d
- Approach: Map curve into a point in dm-dimensional space



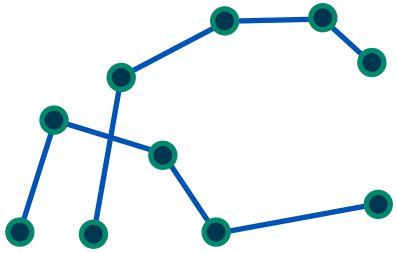
2D

$\Rightarrow$

10D

Similarity measures

- Input: Two polygonal curves P and Q of size m in R^d
- Approach: Map curve into a point in dm-dimensional space



2D

$\Rightarrow$

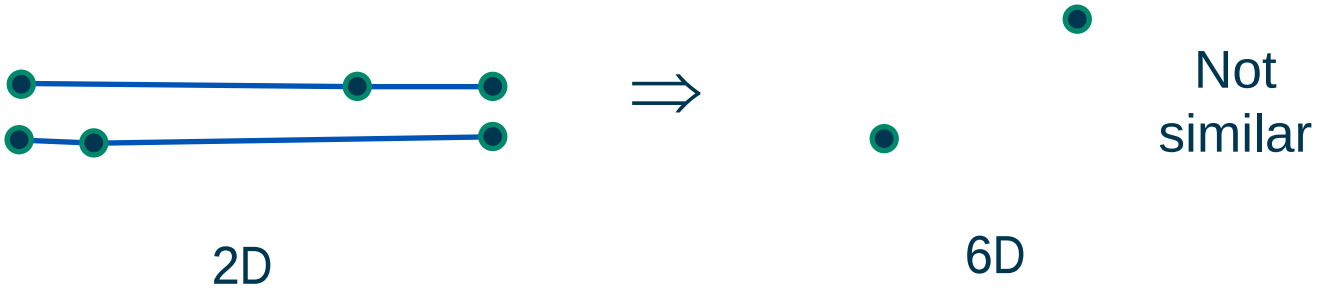
10D

Similar?

Similarity measures

- Input: Two polygonal curves P and Q of size m in R^d
- Approach: Map curve into a point in dm-dimensional space

- Drawbacks: P and Q must have same number points only takes the vertices into consideration



Similarity measures

- Input: Two polygonal curves P and Q of size m in R^d
- Approach: Map curve into a point in dm-dimensional space

- Drawbacks: P and Q must have same number points only takes the vertices into consideration

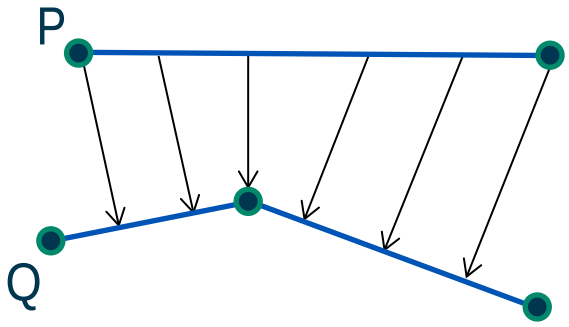
- Question: How does this relate to the measures discussed in the tutorial?

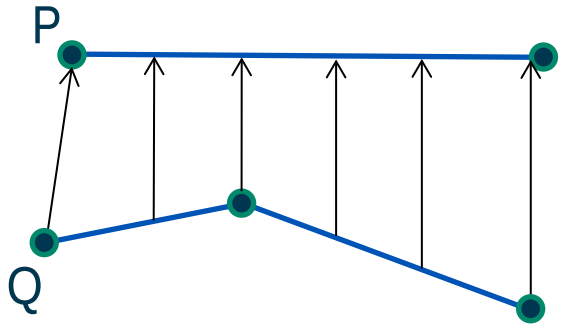
Similarity measures: Hausdorff

- Input: Two polygonal curves P and Q in R^d
- Distance: Hausdorff distance

$$H(P \rightarrow Q) = \max_{p \in P} \min_{q \in Q} |p-q|$$

$$d_H(P,Q) = \max [H(P \rightarrow Q), H(Q \rightarrow P)]$$



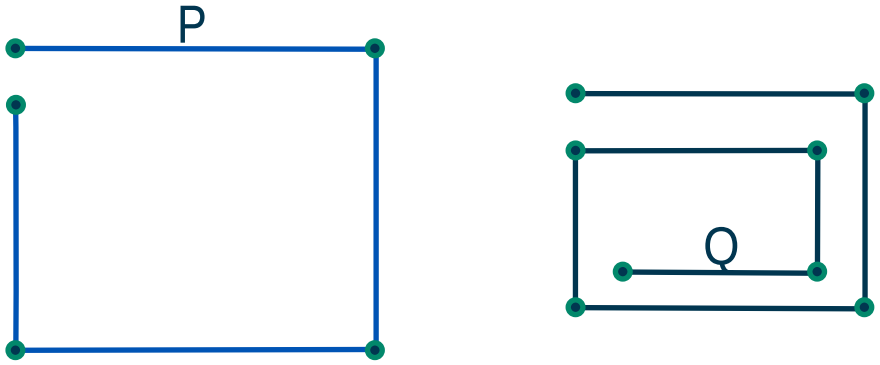


Similarity measures: Hausdorff

- Input: Two polygonal curves P and Q in R^d
- Distance: Hausdorff distance

$$H(P \rightarrow Q) = \max_{p \in P} \min_{q \in Q} |p-q|$$

$$d_H(P,Q) = \max [H(P \rightarrow Q), H(Q \rightarrow P)]$$

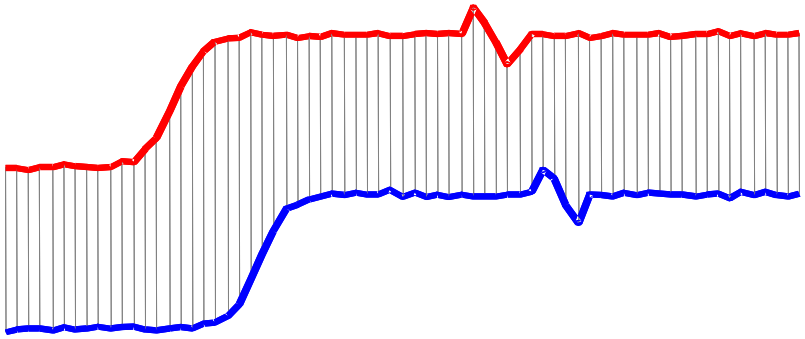


- Drawbacks: Fail to capture the distance between curves.

Similarity measures: Aligning sequences

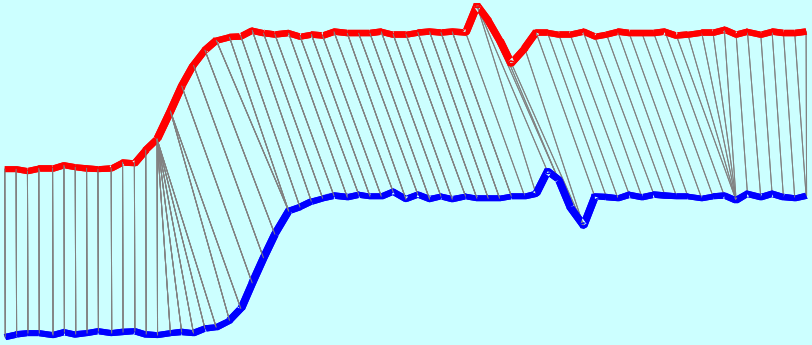
- Input: Two polygonal curves P and Q in R^d
- Other interesting measures are:
  - Dynamic Time Warping (DTW)
  - Longest Common Subsequence (LCSS)
  - Model-Driven matching (MDM)

Dynamic Time Warping (DTW)



Euclidean Distance

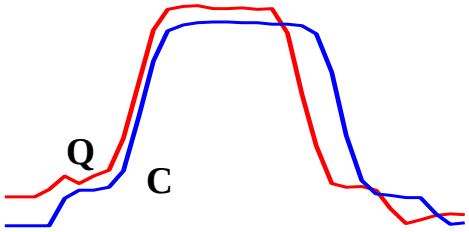
One-to-one alignments



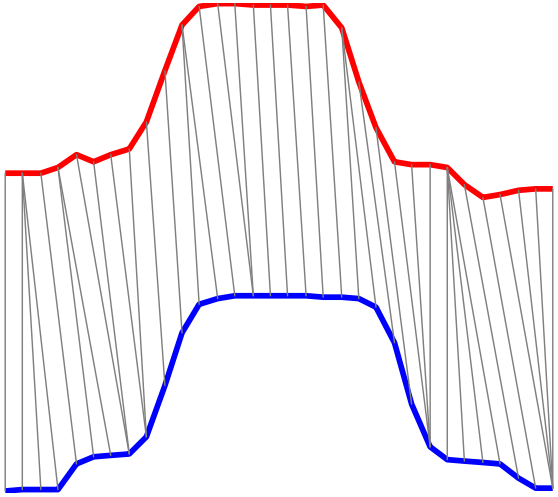
Time Warping Distance

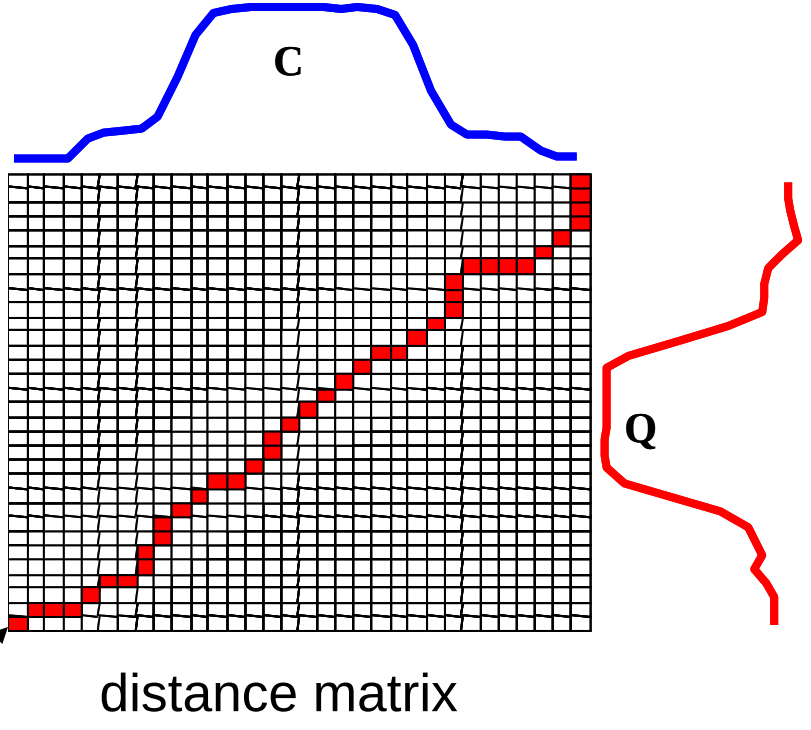
Non-linear alignments are allowed

Calculating DTW



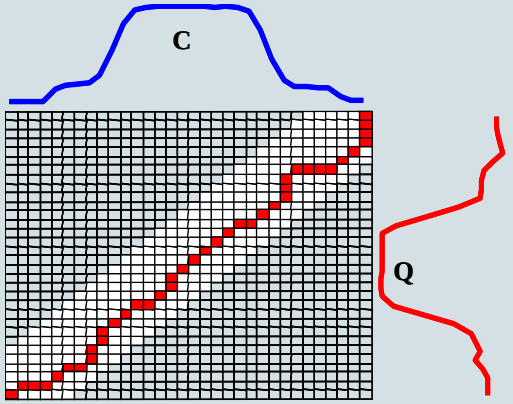
$$DTW(Q,C) = \min\left\{ \sqrt{\sum_{k=1}^{K} w_k} \right.$$





5 steps in designing dynamic-programming algorithms

1. define subproblems [#subproblems]
2. guess first choice [#choices]
3. give recurrence for the value of an optimal solution
   [time/subproblem treating recursive calls as $\Theta(1)$]
   i. define subproblem in terms of a few parameters
   ii. define variable $m[..]$ = value of optimal solution for subproblem
   iii. relate subproblems by giving recurrence for $m[..]$
4. algorithm: fill in table for $m[..]$ in suitable order (or recurse & memoize)
5. solve original problem



Running time: #subproblems * time/subproblem

Correctness:

(i) correctness of recurrence: relate OPT to recurrence

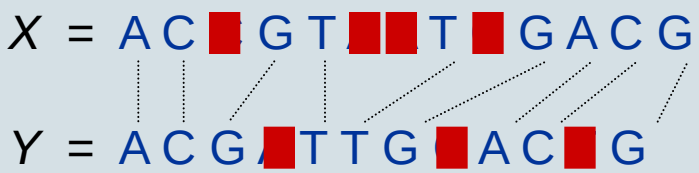
(ii) correctness of algorithm: induction using (i)

Longest common subsequence

substring of a string:
string that can be obtained by omitting zero or more characters

string: A C C A T T G
example substrings: [redacted]C C A T [redacted] or A C C [redacted] or …

LCS(X,Y)
= a longest common subsequence of strings X and Y
= *a* longest string that is a subsequence of *X* and a subsequence of *Y*



LCS(*X*,*Y*) = ACGTTGACG
(or ACGTTCACG …)

The LCS problem

Input: strings $X = x_1, x_2, \ldots, x_n$ and $Y = y_1, y_2, \ldots, y_m$

Output: (the length of) a longest common subsequence of X and Y

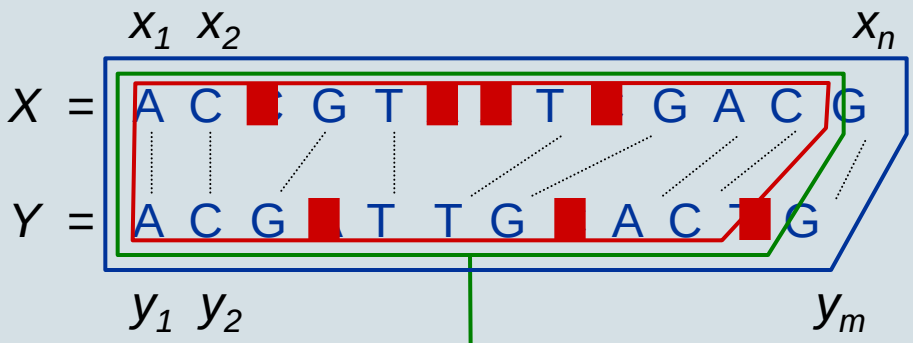
So we have

valid solution = common subsequence

profit = length of the subsequence (to be maximized)

Let's study the structure of an optimal solution

- what are the choices that need to be made?
- what are the subproblems that remain? do we have optimal substructure?



LCS for $x_1 \ldots x_{n-1}$ and $y_1 \ldots y_{m-1}$

so: optimal substructure

first choice:

is last character of *X* matched to last character of *Y*?

or is last character of *X* unmatched ?

or is last character of *Y* unmatched ?

greedy choice ?

overlapping subproblems ?

5 steps in designing dynamic-programming algorithms

1. define subproblems 
2. guess first choice 
3. give recurrence for the value of an optimal solution

4. algorithm: fill in table for c [..] in suitable order (or recurse & memoize)
5. solve original problem

3. Give recursive formula for the value of an optimal solution
   i. define subproblem in terms of a few parameters

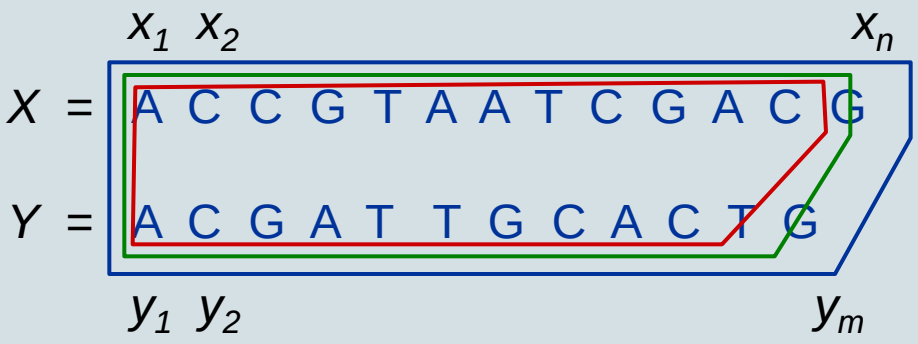
Find LCS of $X_i := x_1 \ldots x_i$ and $Y_j := y_1 \ldots y_j$,
for parameters i, j with $0 \le i \le n$ and $0 \le j \le m$?

X_0 is empty string

   ii. define variable $c[..]$ = value of optimal solution for subproblem

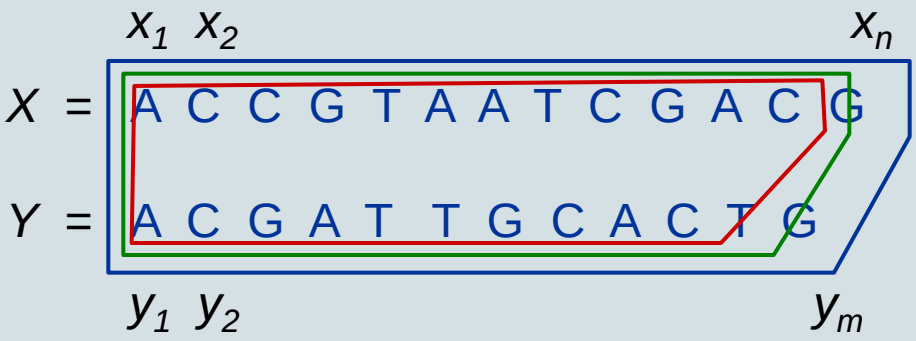
$c[i,j]$ = length of LCS of X_i and Y_j

   iii. give recursive formula for $c[..]$



$X_i := x_1 \ldots x_i$ and $Y_j := y_1 \ldots y_j$, for $0 \le i \le n$ and $0 \le j \le m$

we want recursive formula for $c[i,j]$ = length of LCS of X_i and Y_j



Lemma: $c[i,j] =$

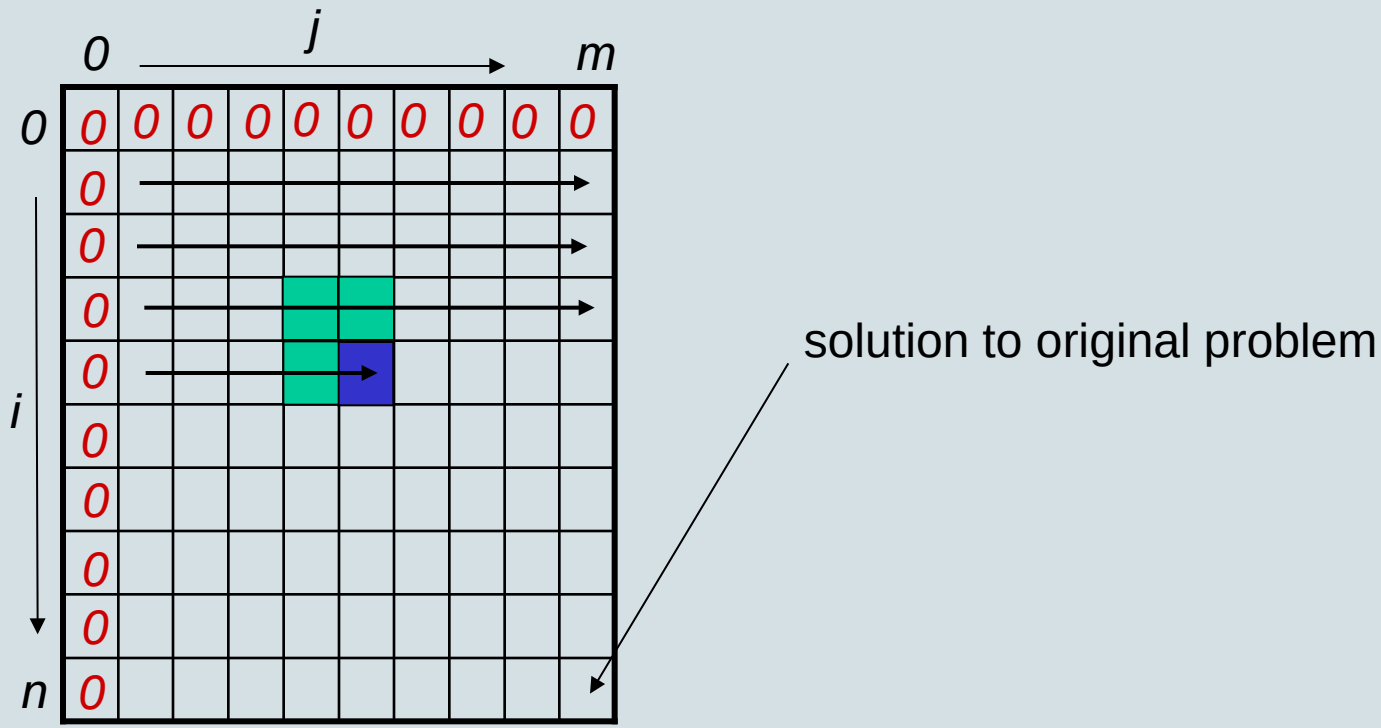
$$\begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1,j-1] + 1 & \text{if } i,j > 0 \text{ and } x_i = y_j \\ \max\{ c[i-1,j], c[i,j-1] \} & \text{if } i,j > 0 \text{ and } x_i \neq y_j \end{cases}$$

5 steps in designing dynamic-programming algorithms

1. define subproblems 
2. guess first choice 
3. give recurrence for the value of an optimal solution 
4. algorithm: fill in table for c [..] in suitable order (or recurse & memoize)
5. solve original problem

4. Algorithm: fill in table for c [..] in suitable order

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1,j-1] + 1 & \text{if } i,j > 0 \text{ and } x_i = y_j \\ \max\{ c[i-1,j], c[i,j-1] \} & \text{if } i,j > 0 \text{ and } x_i \neq y_j \end{cases}$$

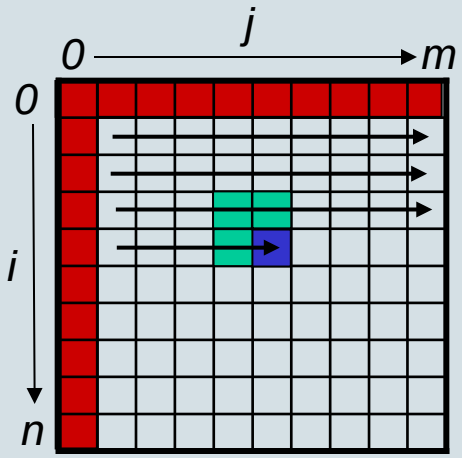


4. Algorithm: fill in table for c [..] in suitable order

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1,j-1] + 1 & \text{if } i,j > 0 \text{ and } x_i = y_j \\ \max\{ c[i-1,j], c[i,j-1] \} & \text{if } i,j > 0 \text{ and } x_i \neq y_j \end{cases}$$

LCS-Length(*X*, *Y*)

1. $n \leftarrow length[X]; m \leftarrow length[Y]$
2. **for** $i \leftarrow 0$ **to** n **do** $c[i,0] \leftarrow 0$
3. **for** $j \leftarrow 0$ **to** m **do** $c[0,j] \leftarrow 0$
4. **for** $i \leftarrow 1$ **to** n
5. **do for** $j \leftarrow 1$ **to** m
6. **do if** $X[i] = Y[j]$
7. **then** $c[i,j] \leftarrow c[i-1,j-1] + 1$
8. **else** $c[i,j] \leftarrow \max\{ c[i-1,j], c[i,j-1] \}$
9. **return** $c[n,m]$



Analysis of running time

LCS-Length(*X*, *Y*)

Θ(1)	1.	$n \leftarrow length[X]; m \leftarrow length[Y]$
Θ(n)	2.	**for** $i \leftarrow 0$ **to** n **do** $c[i,0] \leftarrow 0$
Θ(m)	3.	**for** $j \leftarrow 0$ **to** m **do** $c[0,j] \leftarrow 0$
	4.	**for** $i \leftarrow 1$ **to** n
	5.	**do for** $j \leftarrow 1$ **to** m
	6.	**do if** $X[i] = Y[j]$
	7.	**then** $c[i,j] \leftarrow c[i-1,j-1] + 1$
	8.	**else** $c[i,j] \leftarrow \max \{ c[i-1,j], c[i,j-1] \}$
Θ(1)	9.	**return** $c[n,m]$

Lines 4 – 8: $\sum_{1 \le i \le n} \sum_{1 \le j \le m} \Theta(1) = \Theta(nm)$

5 steps in designing dynamic-programming algorithms

1. define subproblems ✓
2. guess first choice ✓
3. give recurrence for the value of an optimal solution ✓
4. algorithm: fill in table for c [..] in suitable order (or recurse & memoize) ✓
5. solve original problem

   Approaches for going from optimum value to optimum solution:

   i. store choices that led to optimum (e.g. s[i,j] for matrix multiplication)

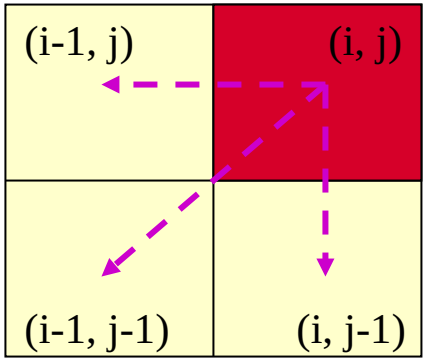
   ii. retrace/deduce sequence of solutions that led to optimum backwards (next slide)

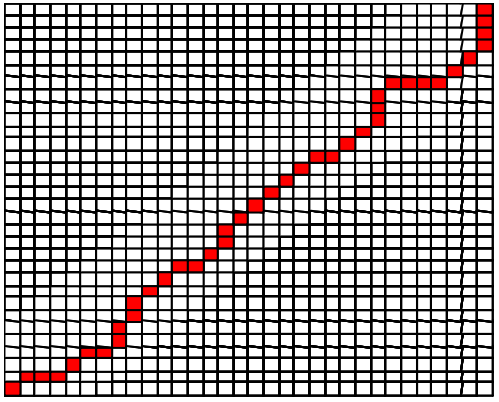
Calculating DTW

- Each warping path *w* can be found using dynamic programming to evaluate the following recurrence:

$$\gamma(i, j) = d(q_i, c_j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}$$

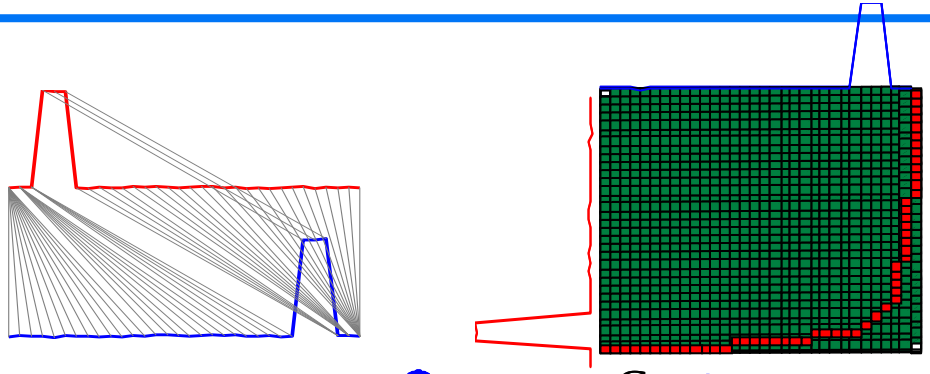
- where γ*(i, j)* is the cumulative distance of the distance *d(i, j)* and its minimum cumulative distance among the adjacent cells

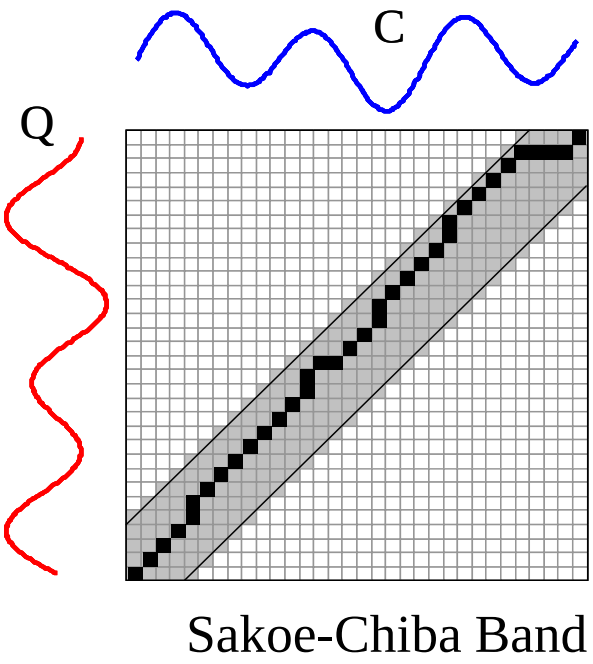




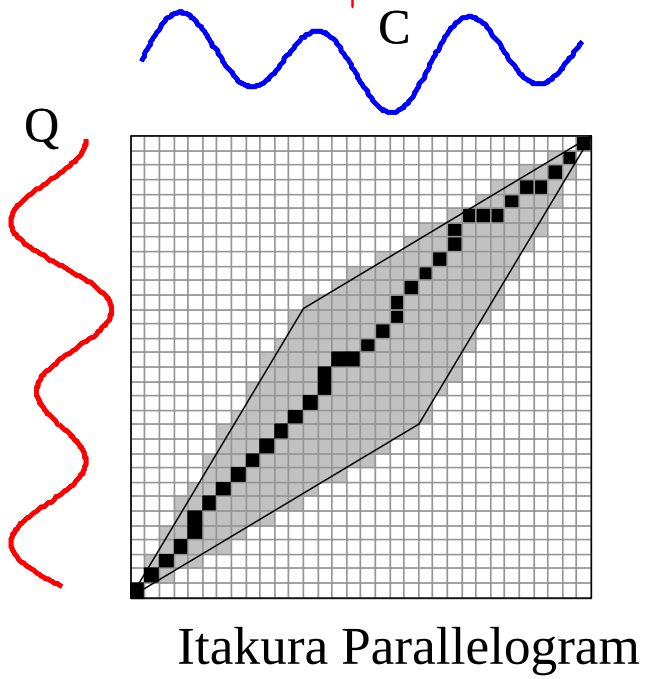
Incorporating time

- To align points with similar time stamp: limit warping





Sakoe-Chiba Band



Itakura Parallelogram

- Also useful for other measures in this lecture

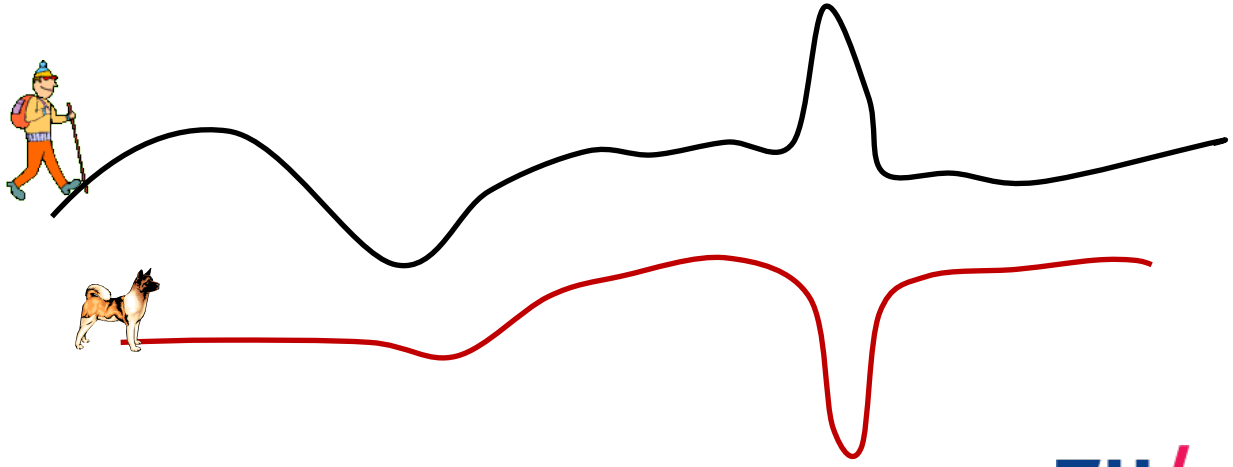
Similarity measures: Aligning sequences

- Input: Two polygonal curves P and Q in R^d
- Other interesting measures are:
  - Dynamic Time Warping (DTW)
  - Longest Common Subsequence (LCSS)
  - Model-Driven matching (MDM)
- often work well
- Drawback: Non-metrics (triangle inequality does not hold) which makes clustering complicated.

Similarity measure: Fréchet Distance

- Fréchet Distance measures the similarity of two curves.

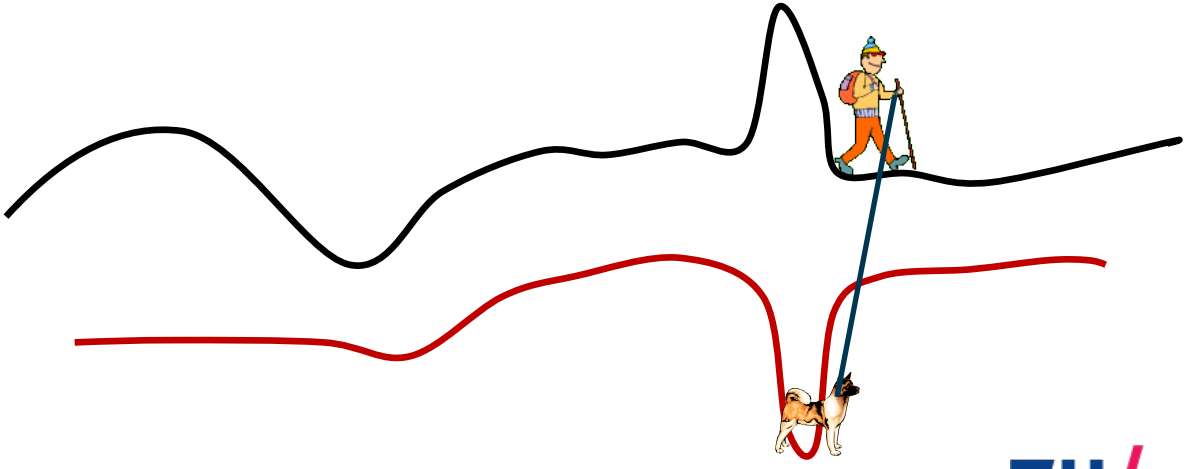
- Dog walking example
  - Person is walking his dog (person on one curve and the dog on other)
  - Allowed to control their speeds but not allowed to go backwards!
  - Fréchet distance of the curves: **minimal leash length** necessary for both to walk the curves from beginning to end



Similarity measure: Fréchet Distance

- Fréchet Distance measures the similarity of two curves.

- Dog walking example
  - Person is walking his dog (person on one curve and the dog on other)
  - Allowed to control their speeds but not allowed to go backwards!
  - Fréchet distance of the curves: **minimal leash length** necessary for both to walk the curves from beginning to end



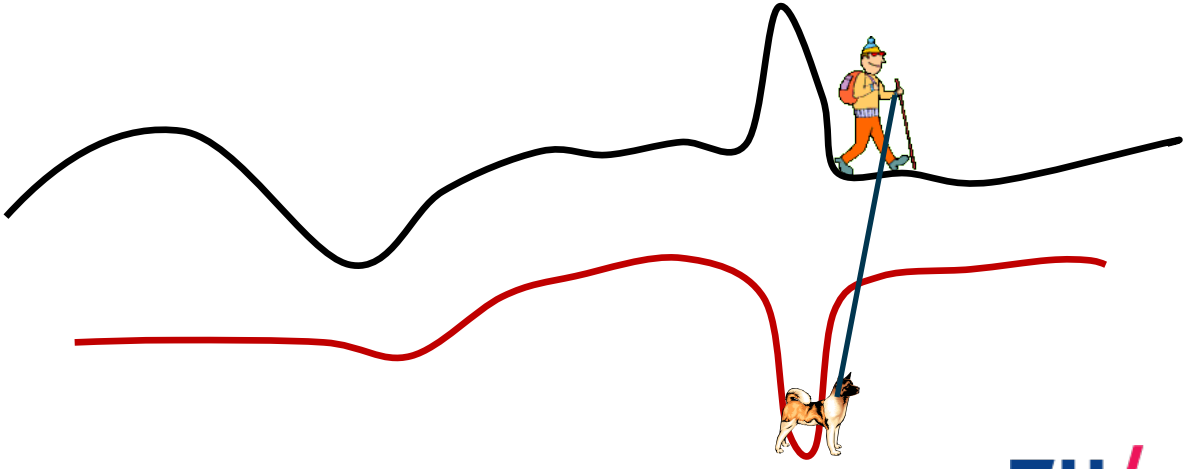
Fréchet Distance

- Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

- The Fréchet distance between P and Q is:

$$\delta_F(P,Q) = \inf_{\substack{\alpha:[0,1]\to P \\ \beta:[0,1]\to Q}} \max_{t\in[0,1]} |P(\alpha(t)) - Q(\beta(t))|$$

- where α and β range over all continuous non-decreasing reparametrizations with $\alpha(0)=p_1$, $\alpha(1)=p_n$, $\beta(0)=q_1$ and $\beta(1)=q_m$.

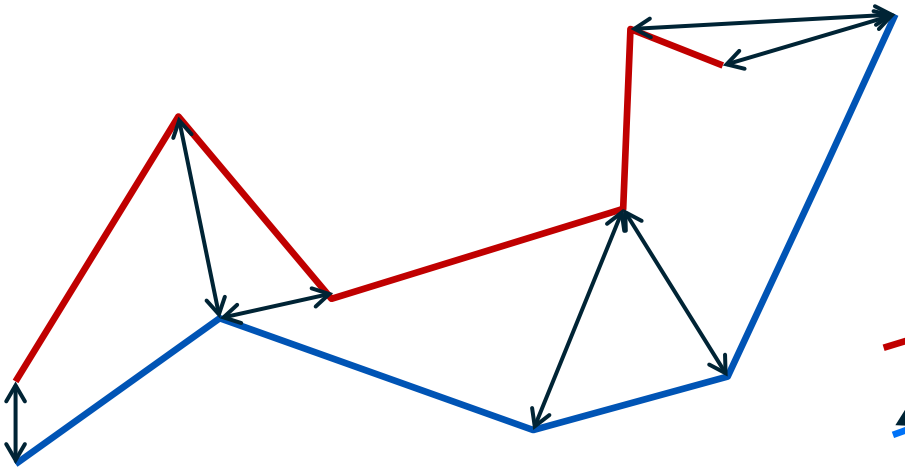


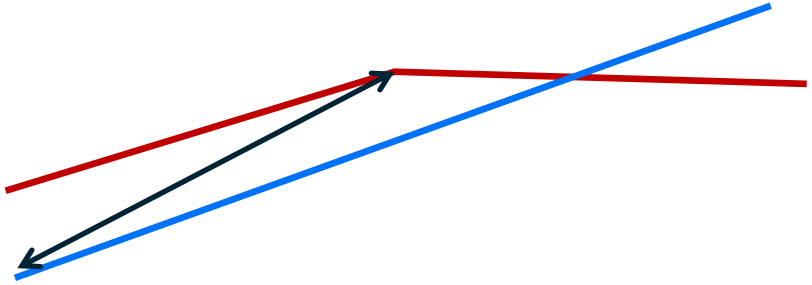
Discrete Fréchet Distance

- The discrete Fréchet distance considers only positions of the leash where its endpoints are located at vertices of the two polygonal curves and never in the interior of an edge.

- Recurrence:

$$\gamma(i, j) = \max\{d(q_i, c_j), \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}\}$$



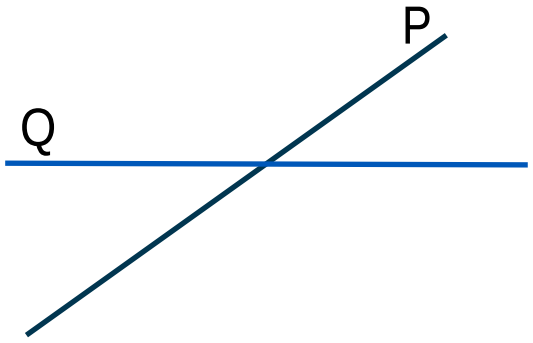


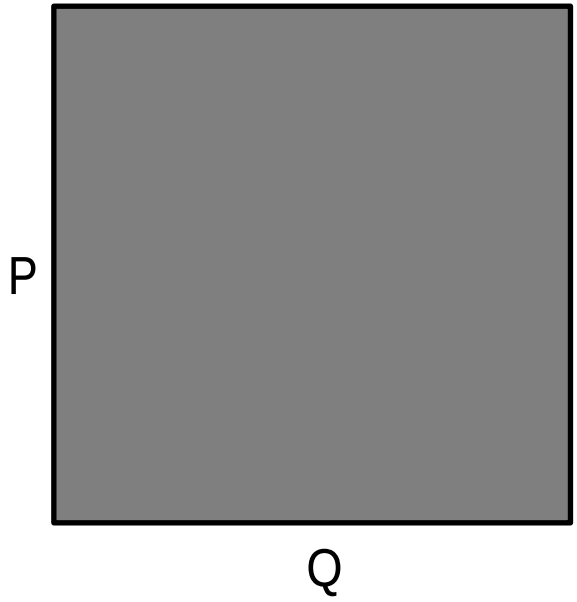
May give large errors!

Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

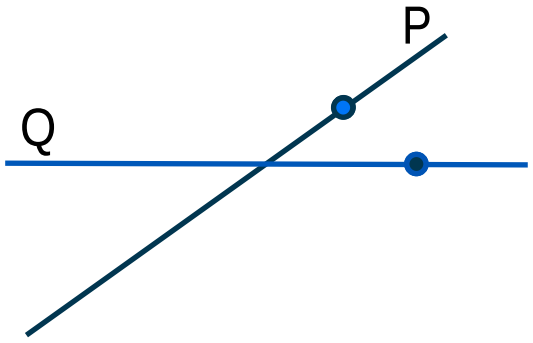


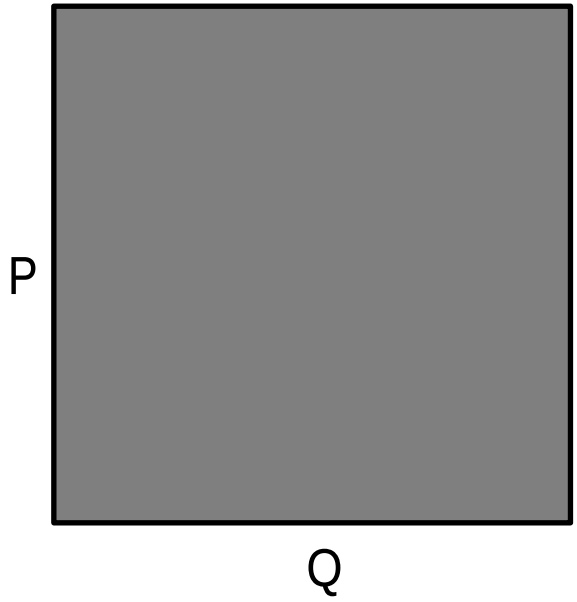


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

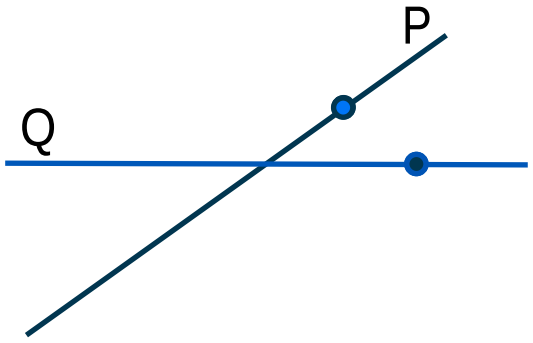


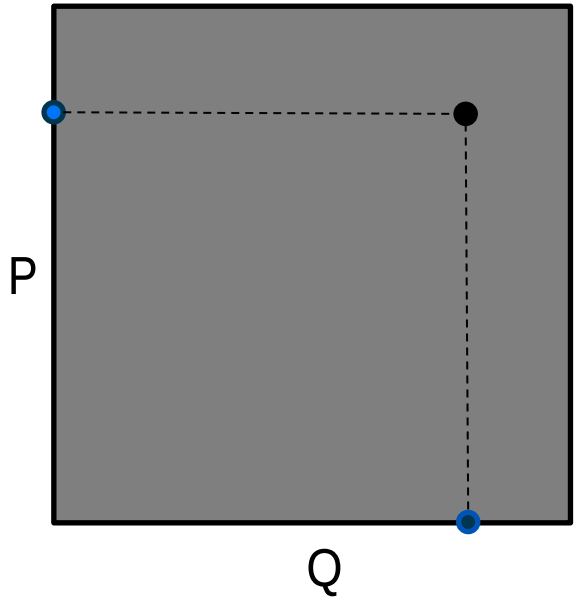


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

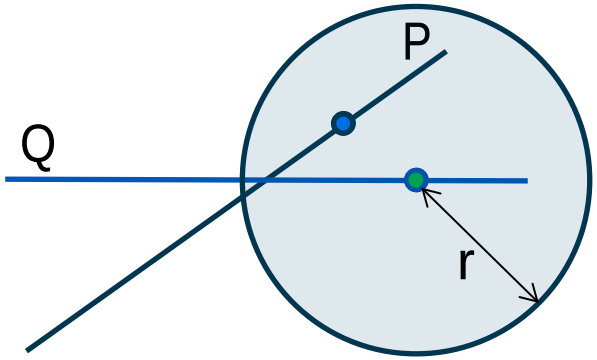


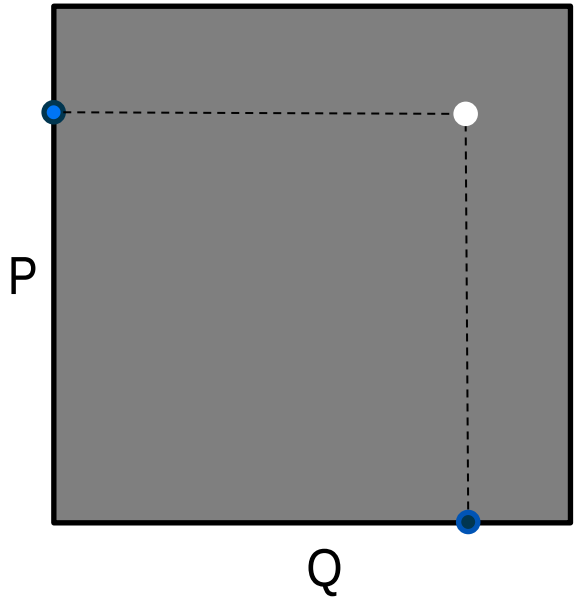


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

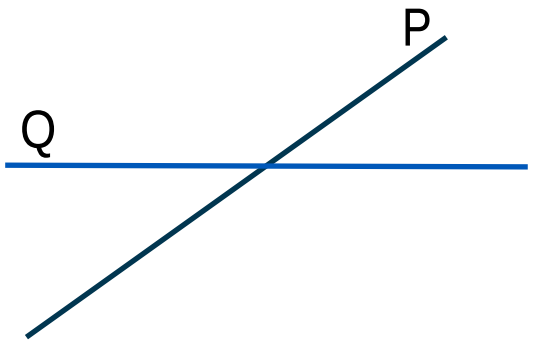


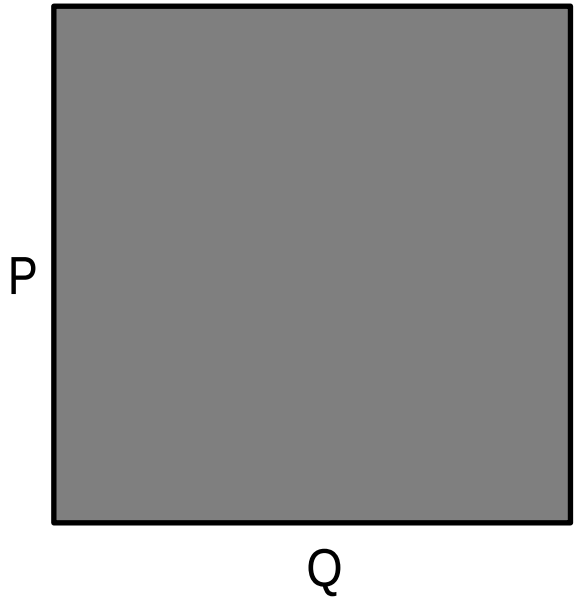


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

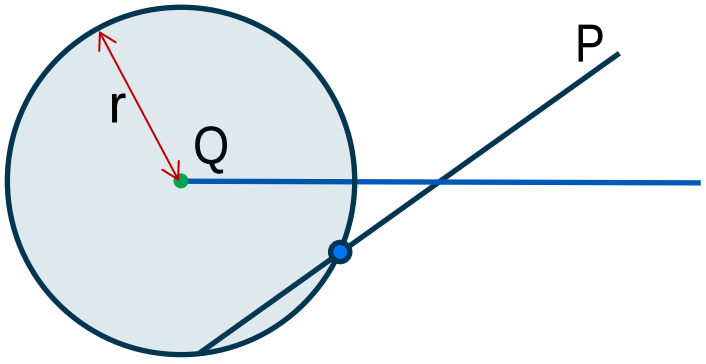


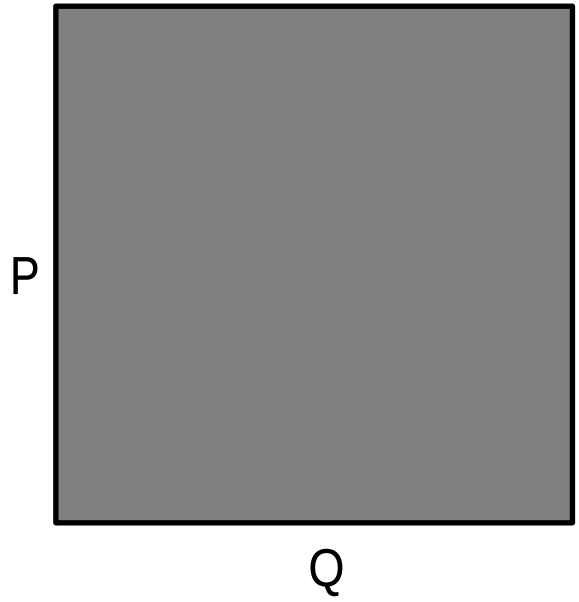


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

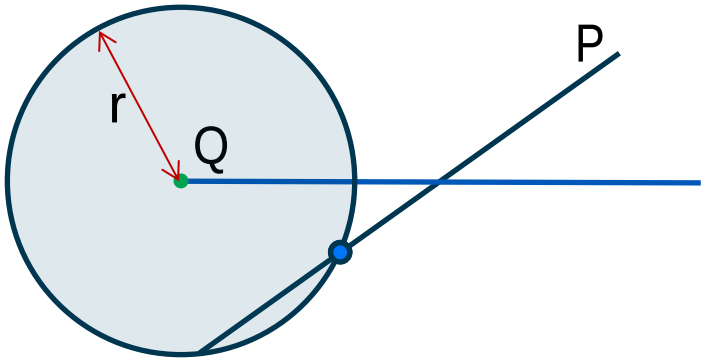


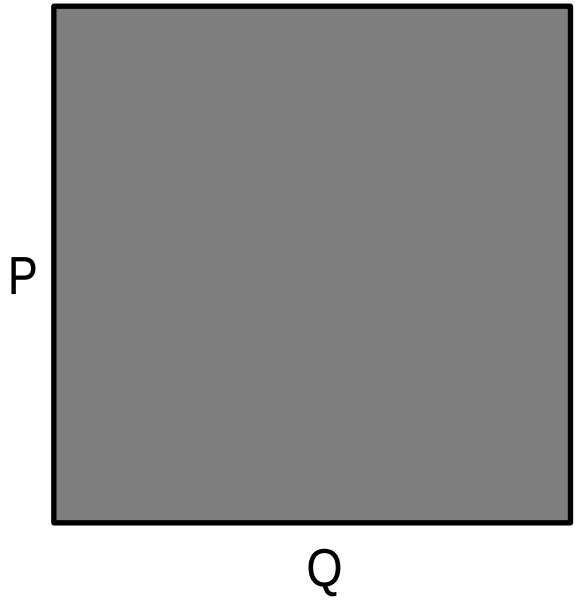


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

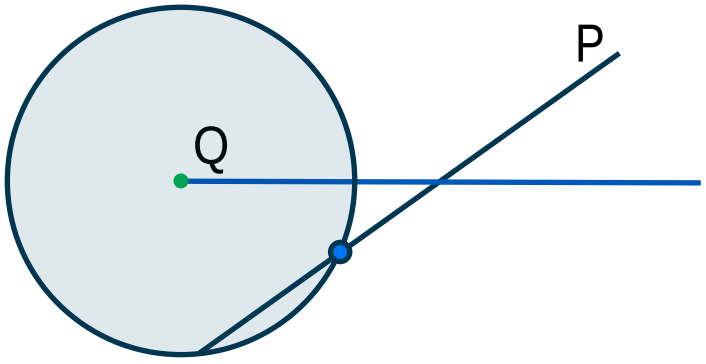


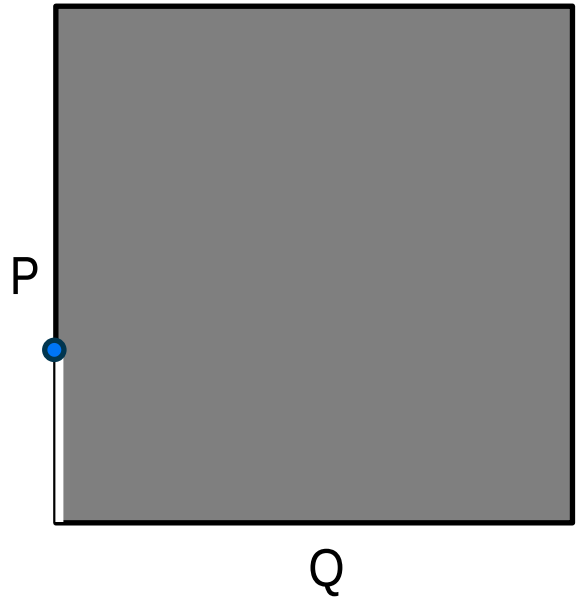


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

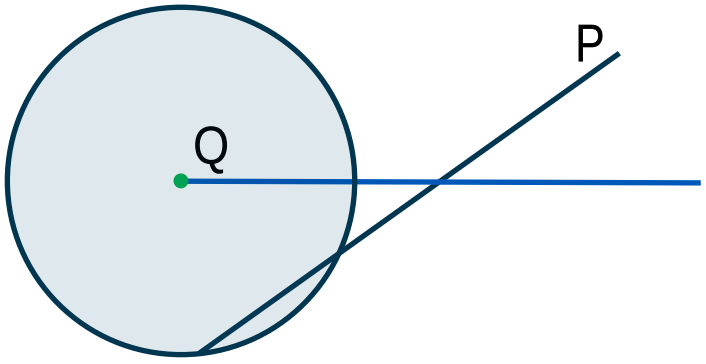


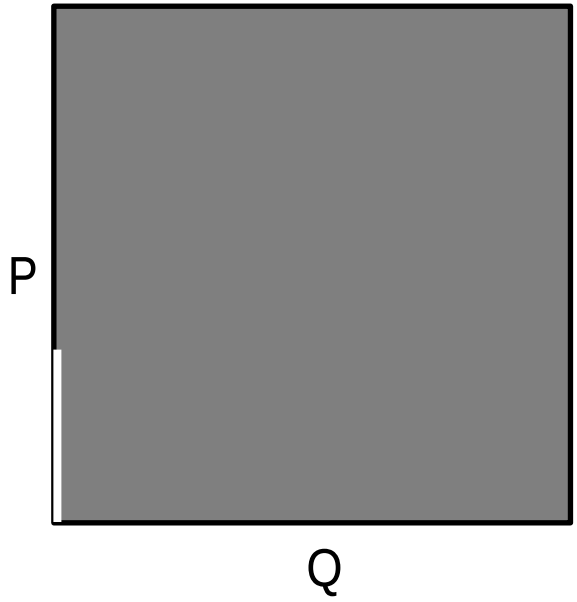


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

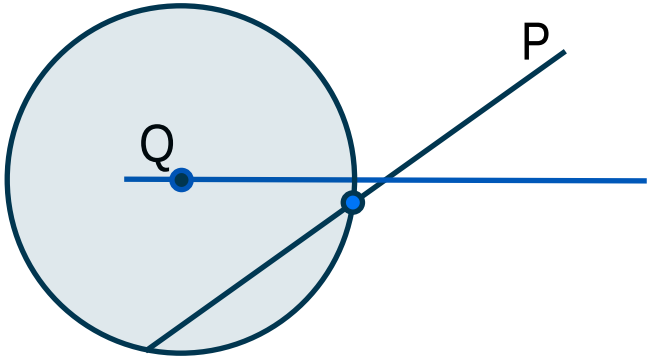


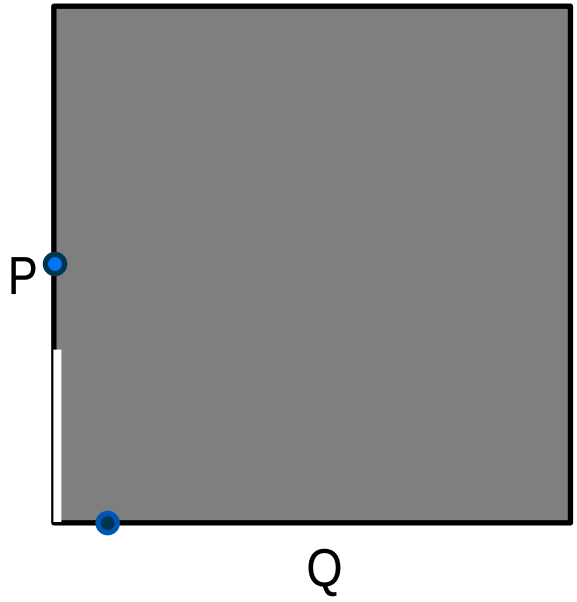


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

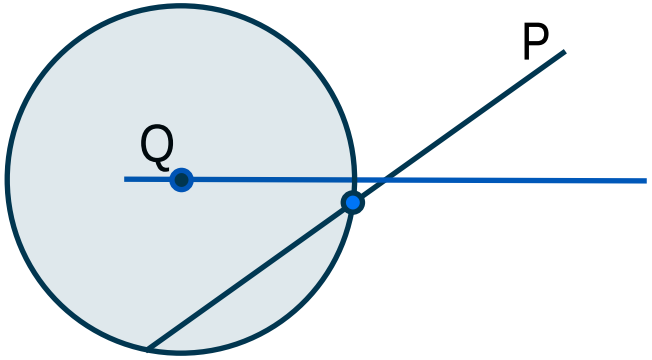


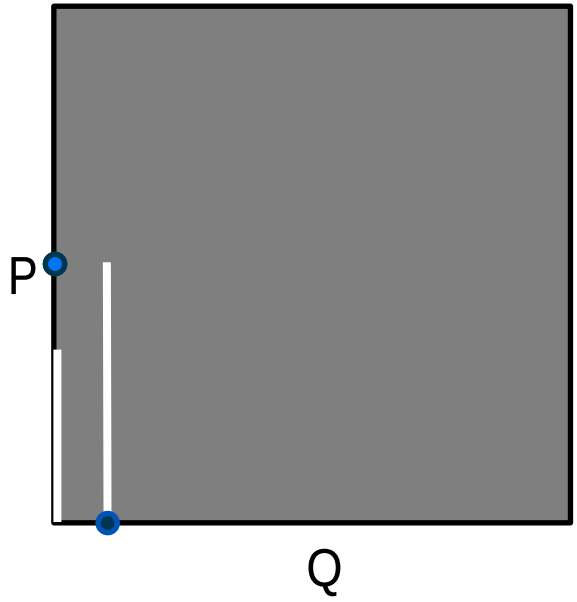


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

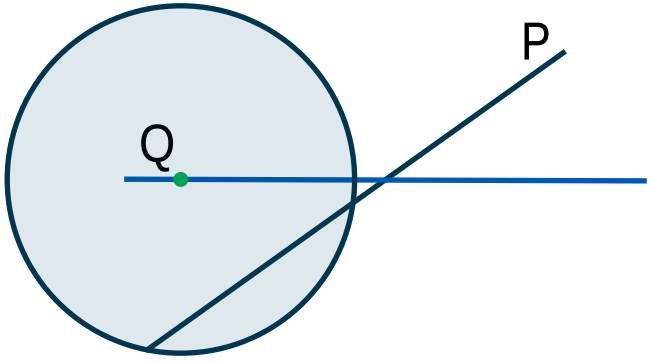


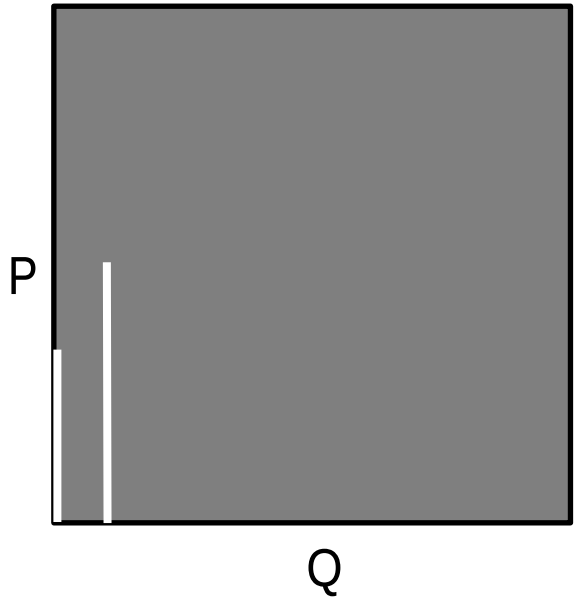


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

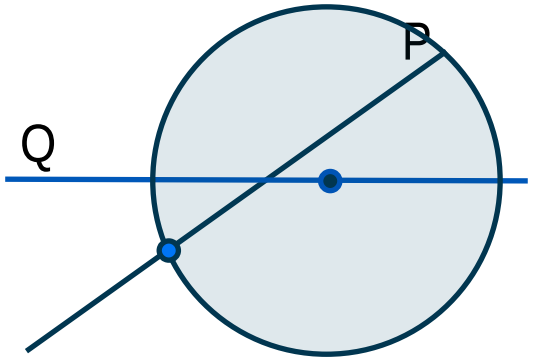


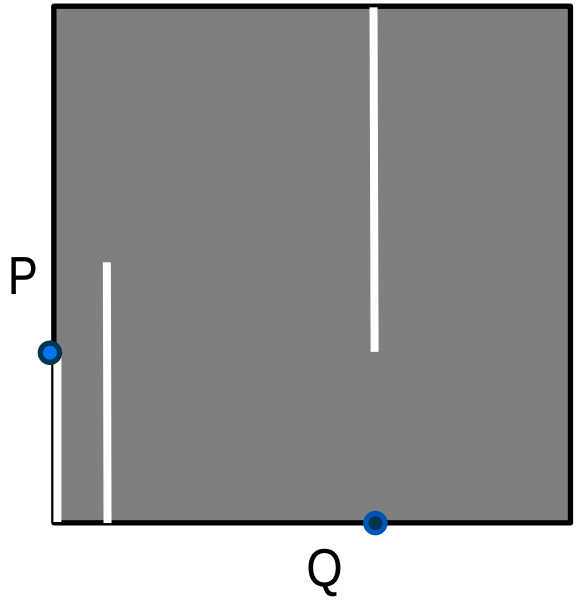


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

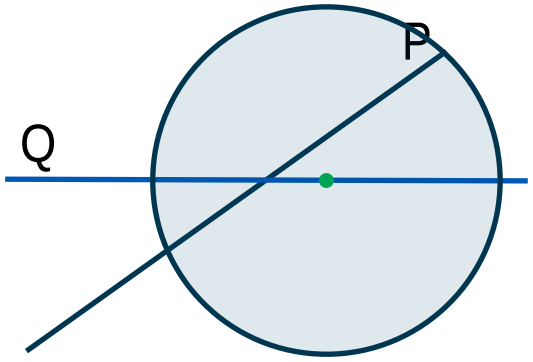


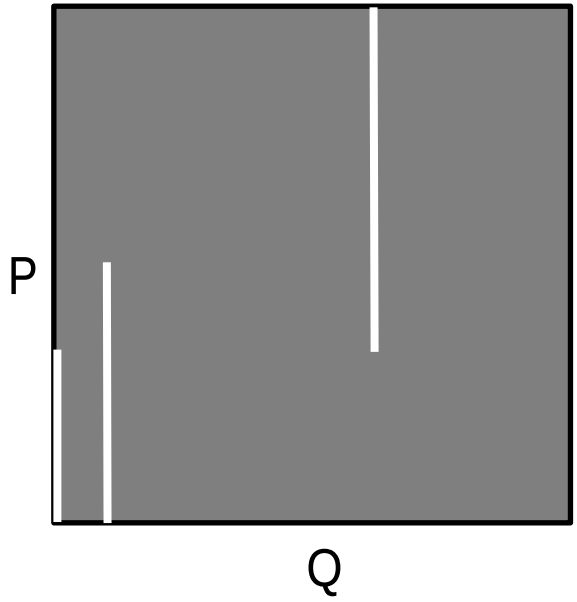


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

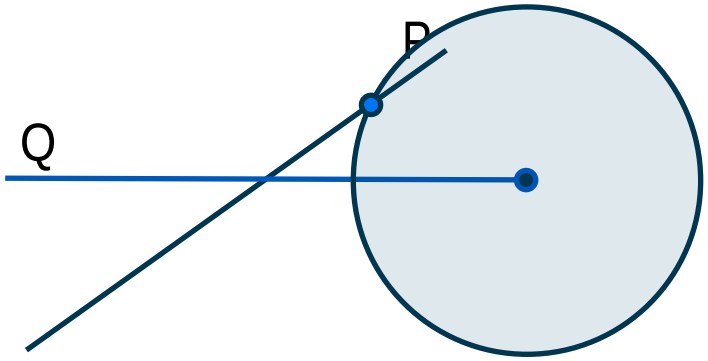


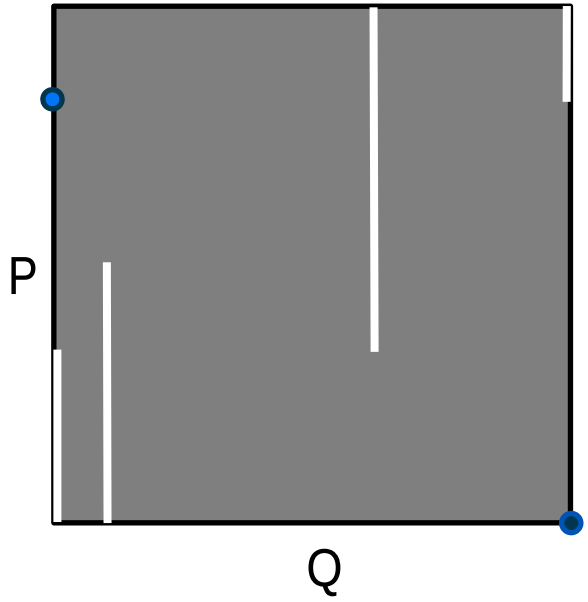


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

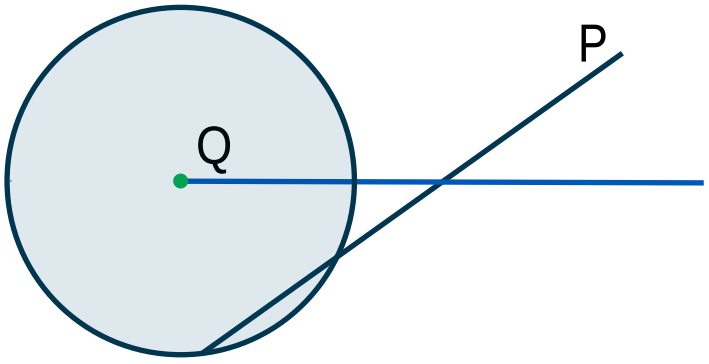


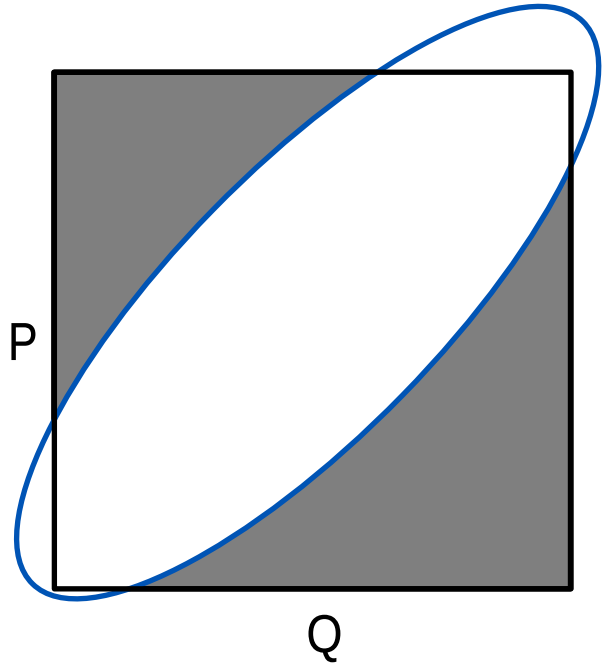


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?



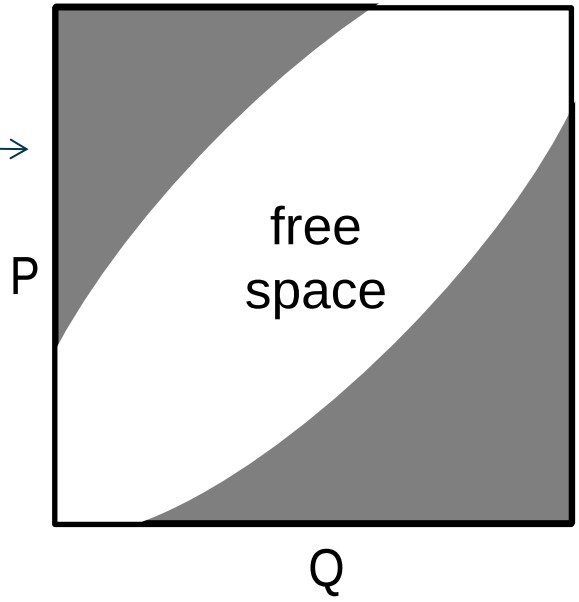


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

Freespace diagram of P and Q

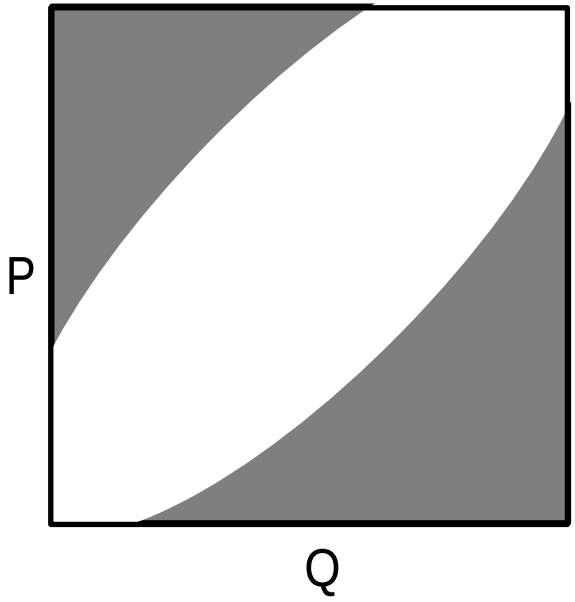


Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

- How do we find a reparametrization between P and Q that realises the Fréchet distance r?



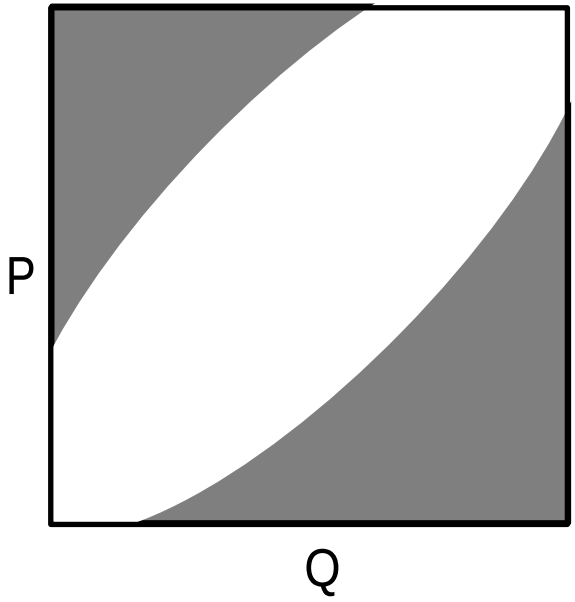
Compute the Fréchet distance

Input: Two polygonal chains $P=\langle p_1, \ldots, p_n \rangle$ and $Q=\langle q_1, \ldots, q_m \rangle$ in R^d.

Decision problem: $\delta_F(P, Q) \leq r$?

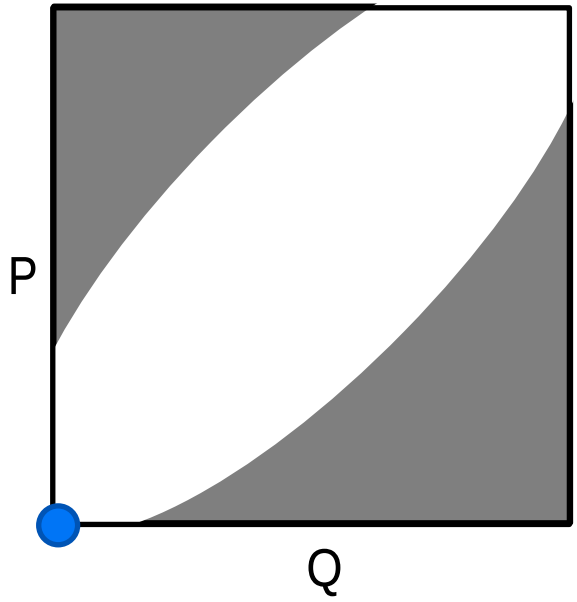
- How do we find a reparametrization between P and Q that realises the Fréchet distance r?

- Recall that it has to be continuous, non-decreasing, $q_1 \leftrightarrow p_1$ and $q_m \leftrightarrow p_n$.



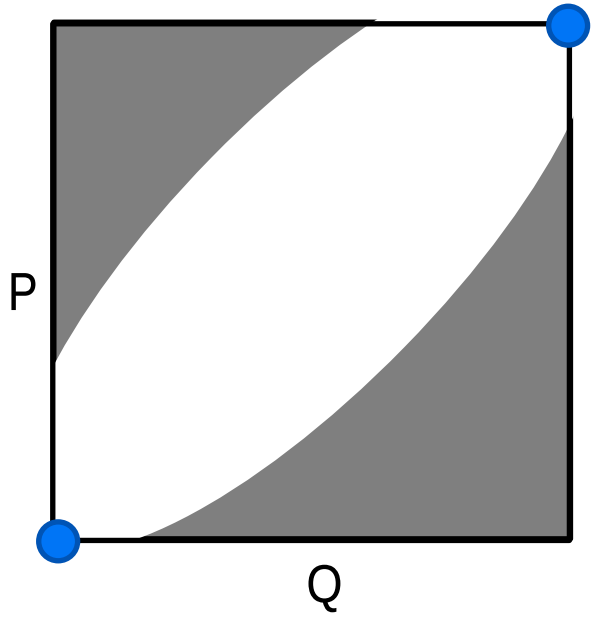
Compute the Fréchet distance

- $q_1 \leftrightarrow p_1$



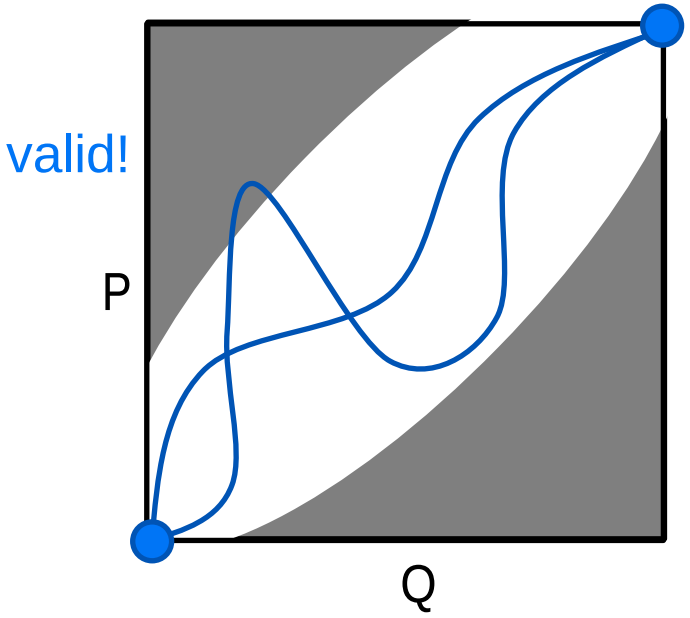
Compute the Fréchet distance

- $q_1 \leftrightarrow p_1$
- $q_m \leftrightarrow p_n$



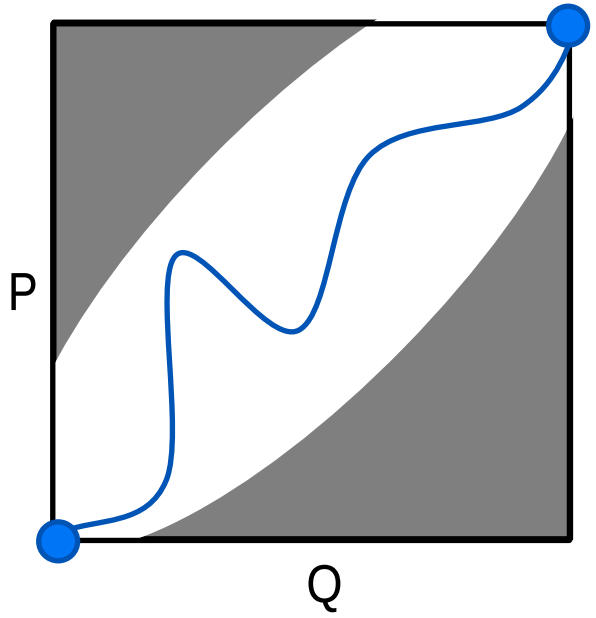
Compute the Fréchet distance

- $q_1 \leftrightarrow p_1$
- $q_m \leftrightarrow p_n$
- must lie in the free space



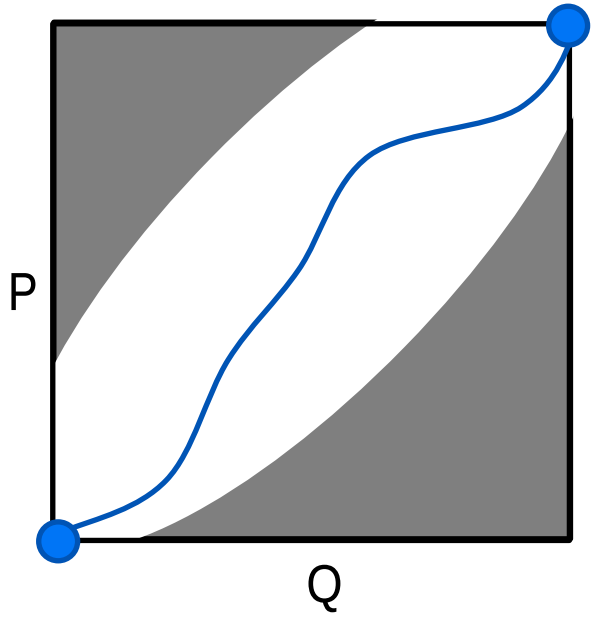
Compute the Fréchet distance

- $q_1 \leftrightarrow p_1$
- $q_m \leftrightarrow p_n$
- must lie in the free space
- continuous



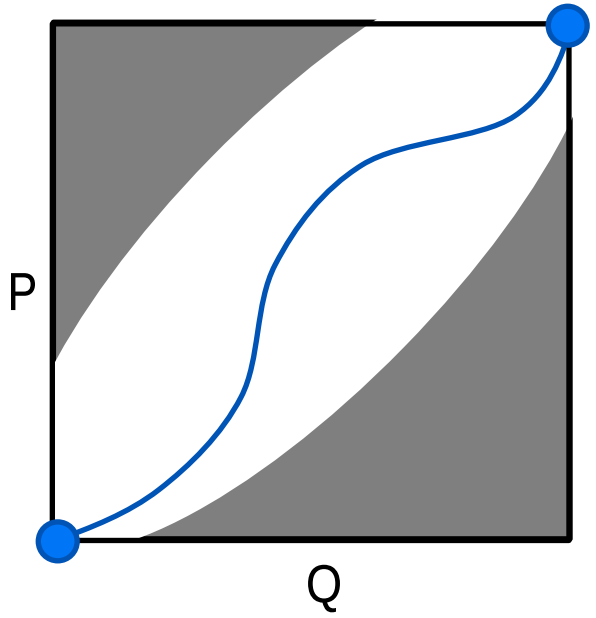
Compute the Fréchet distance

- $q_1 \leftrightarrow p_1$
- $q_m \leftrightarrow p_n$
- must lie in the free space
- continuous
- non-decreasing



Compute the Fréchet distance

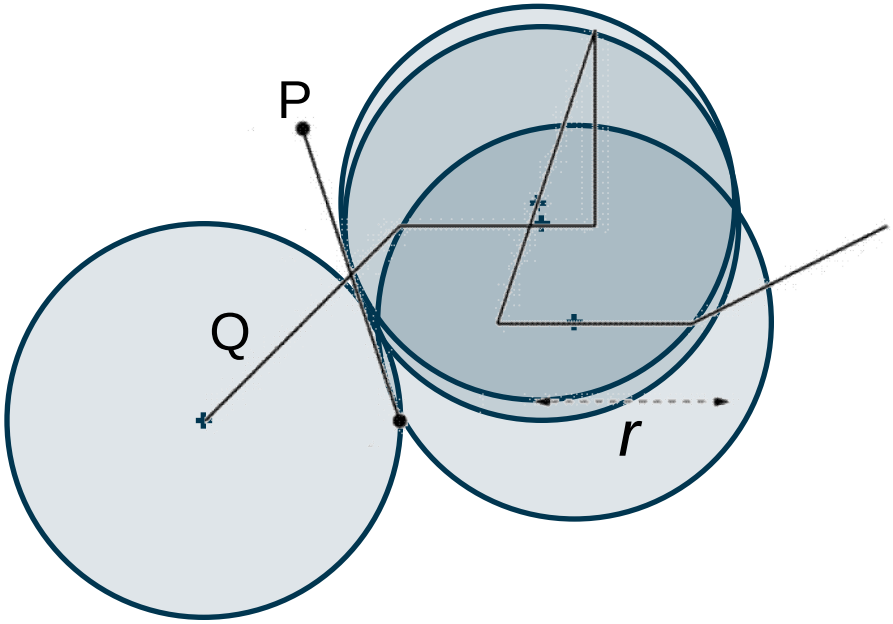
- $q_1 \leftrightarrow p_1$
- $q_m \leftrightarrow p_n$
- must lie in the free space
- continuous
- non-decreasing

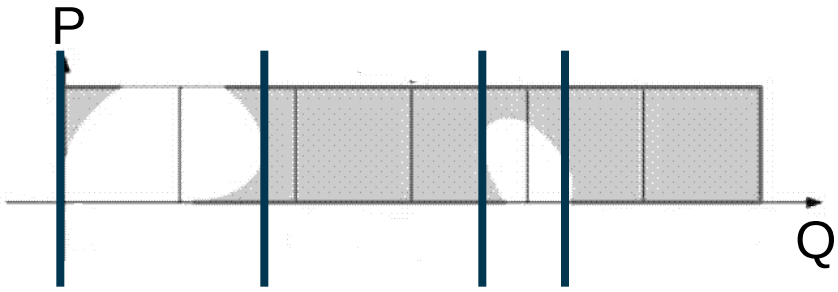


- If there exists an xy-non-decreasing path inside the freespace from the bottom left to the top right corner of the diagram then the Fréchet distance is at most r.

Free Space Diagram

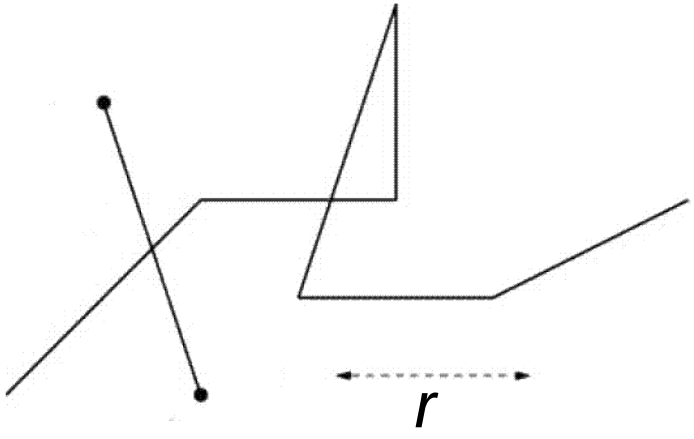
- Decision version
  - Is the Fréchet distance between two paths at most r?

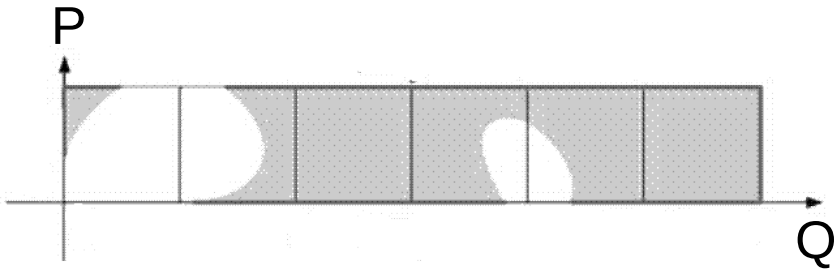




Free Space Diagram

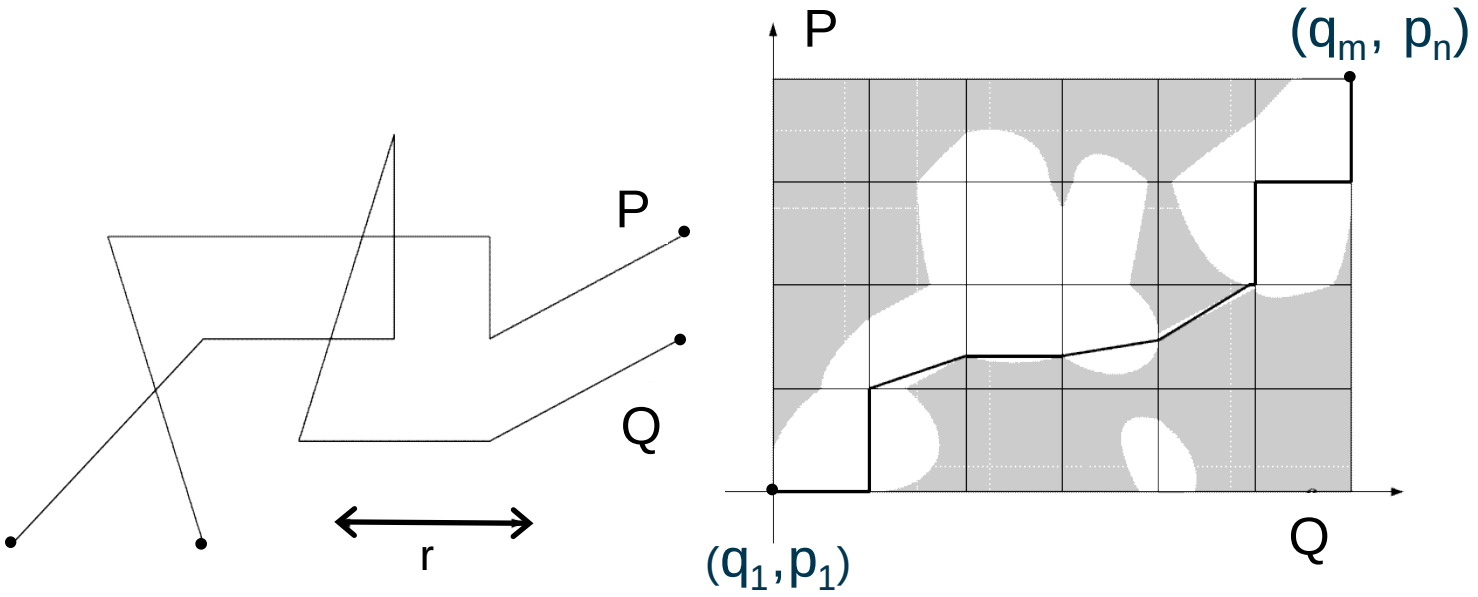
- Decision version
  - Is the Fréchet distance between two paths at most r?





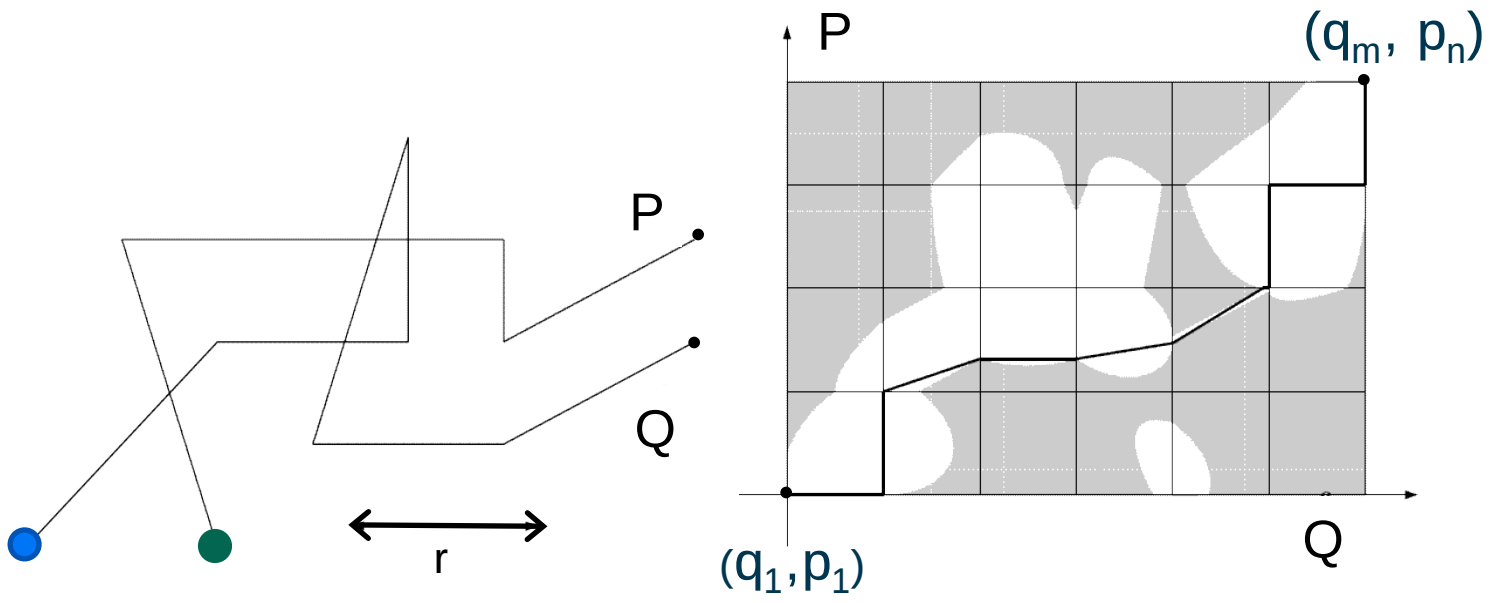
Free Space Diagram

- If there exists a non-decreasing path in the free space from (q_1,p_1) to (q_m,p_n) which is non-decreasing in both coordinates, then curves P and Q have Fréchet distance at most r.



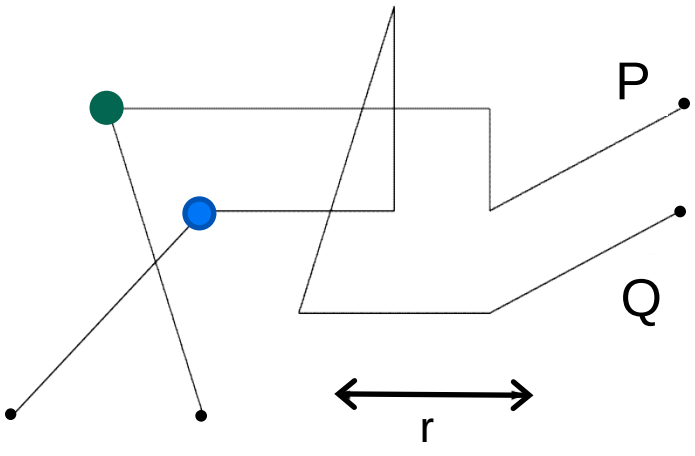
Free Space Diagram

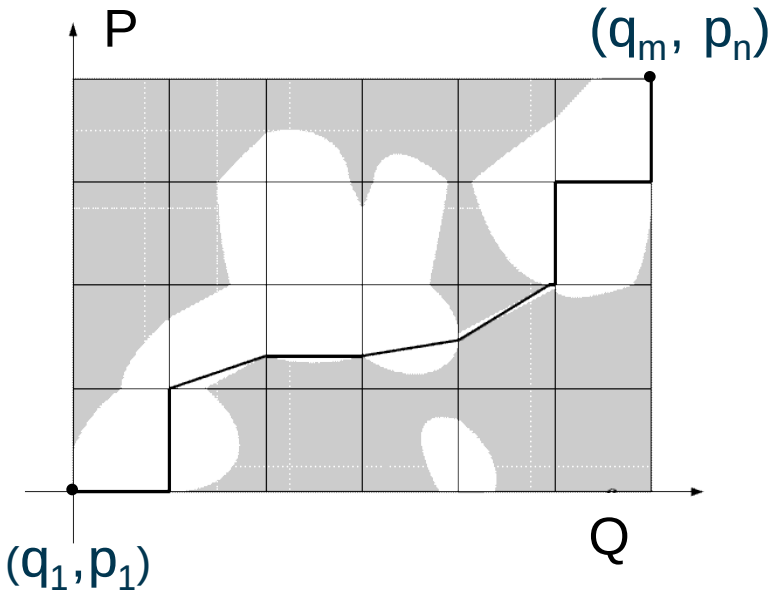
- If there exists a non-decreasing path in the free space from (q_1,p_1) to (q_m,p_n) which is non-decreasing in both coordinates, then curves P and Q have Fréchet distance at most r.



Free Space Diagram

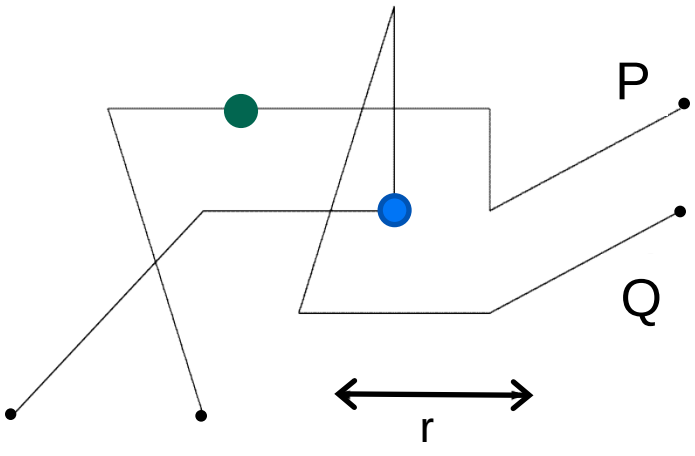
- If there exists a non-decreasing path in the free space from (q_1,p_1) to (q_m,p_n) which is non-decreasing in both coordinates, then curves P and Q have Fréchet distance at most r.

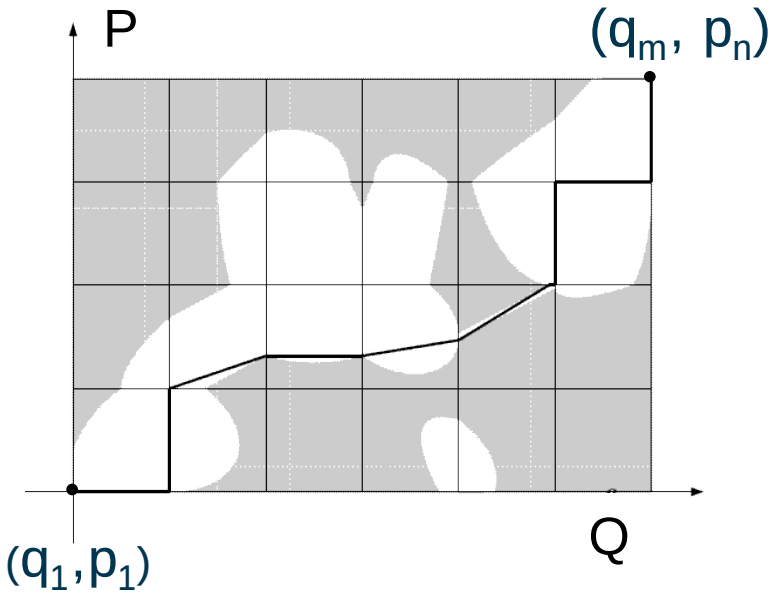




Free Space Diagram

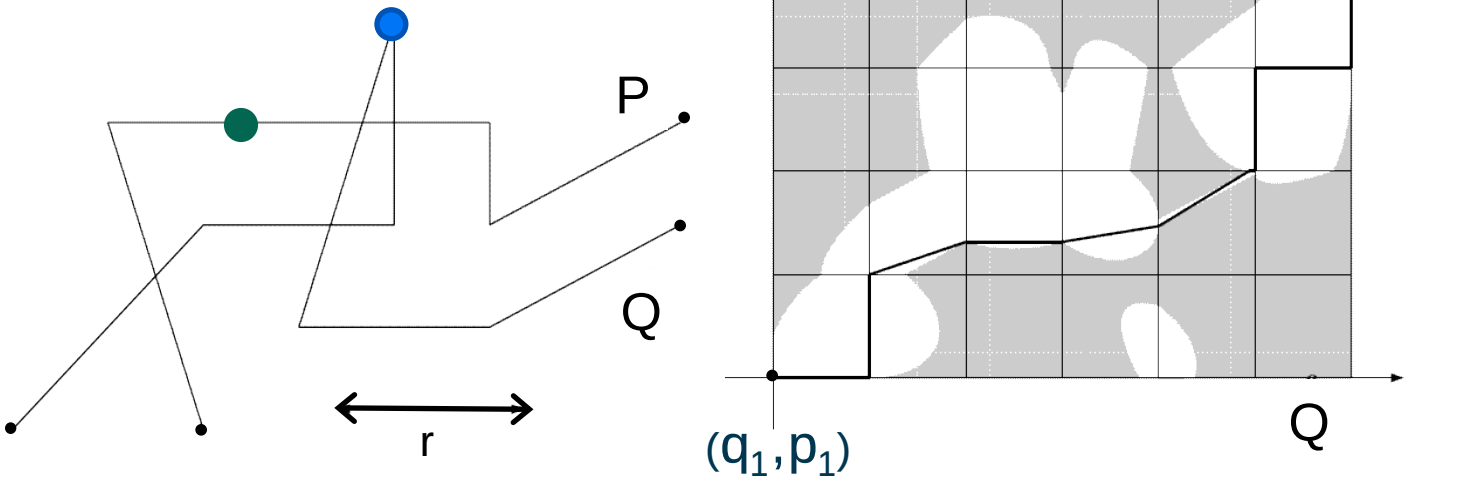
- If there exists a non-decreasing path in the free space from (q_1,p_1) to (q_m,p_n) which is non-decreasing in both coordinates, then curves P and Q have Fréchet distance at most r.





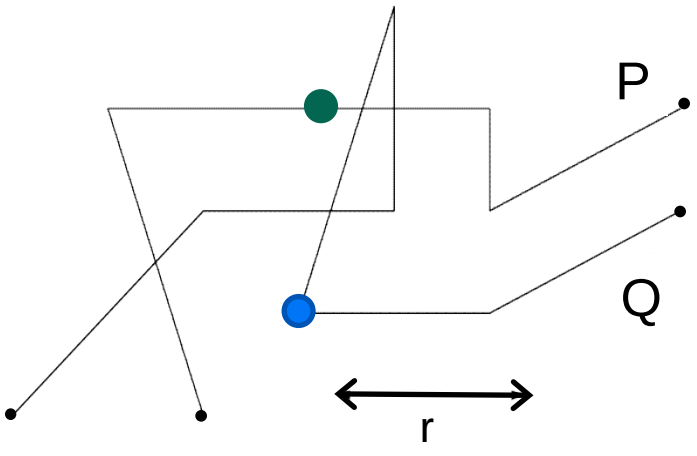
Free Space Diagram

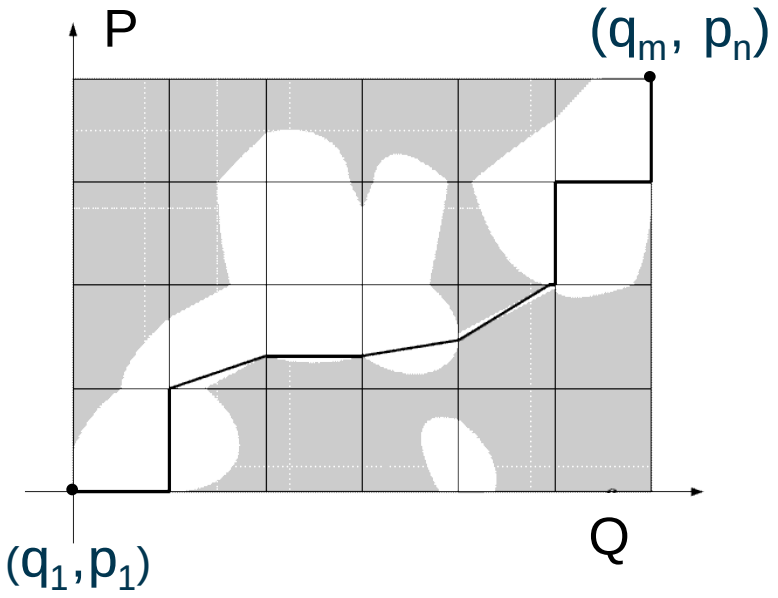
- If there exists a non-decreasing path in the free space from (q_1,p_1) to (q_m,p_n) which is non-decreasing in both coordinates, then curves P and Q have Fréchet distance at most r.



Free Space Diagram

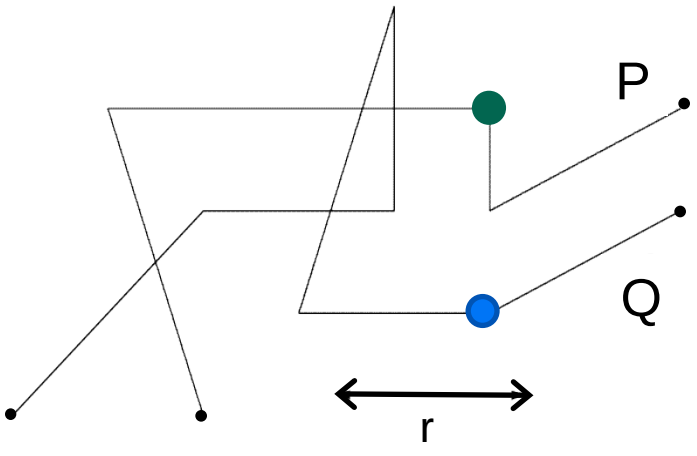
- If there exists a non-decreasing path in the free space from (q_1,p_1) to (q_m,p_n) which is non-decreasing in both coordinates, then curves P and Q have Fréchet distance at most r.

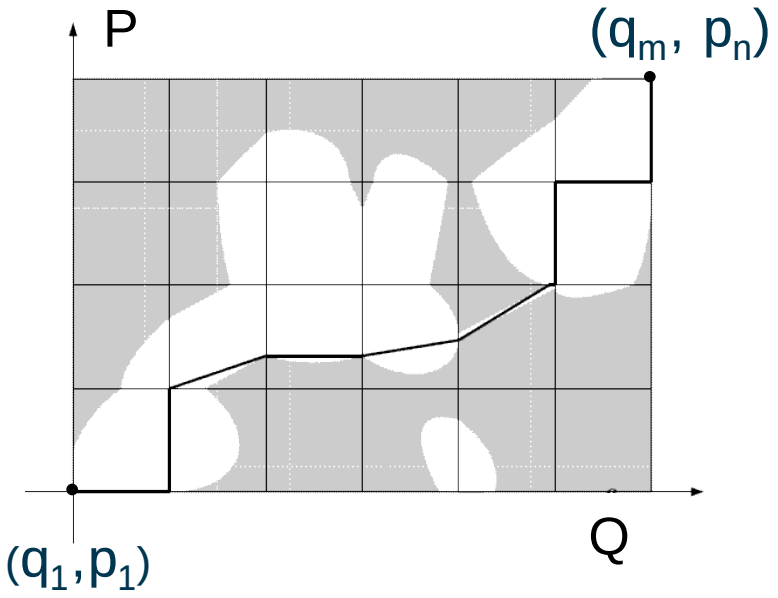




Free Space Diagram

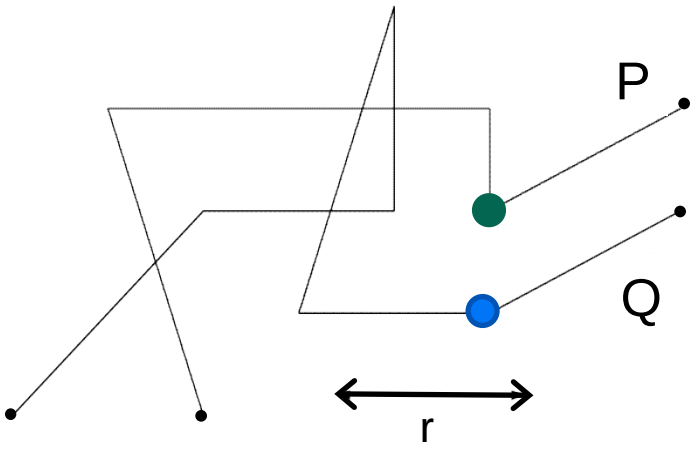
- If there exists a non-decreasing path in the free space from (q_1,p_1) to (q_m,p_n) which is non-decreasing in both coordinates, then curves P and Q have Fréchet distance at most r.

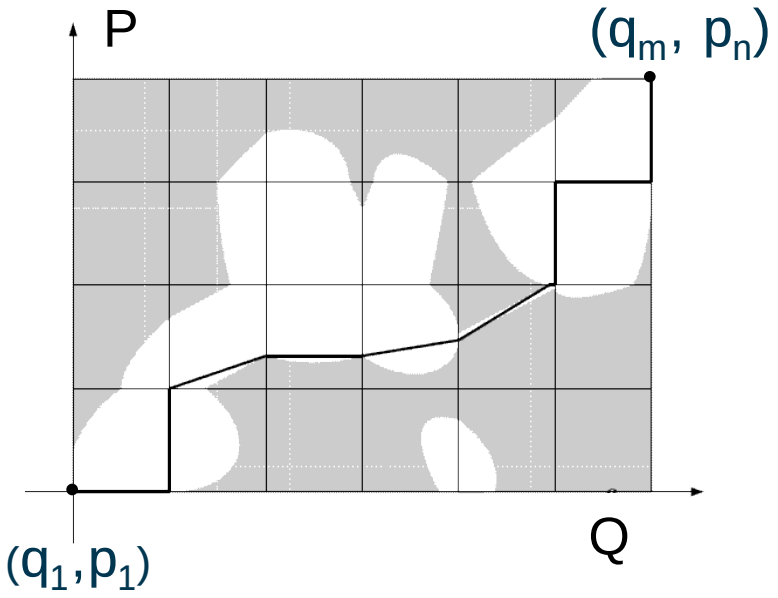




Free Space Diagram

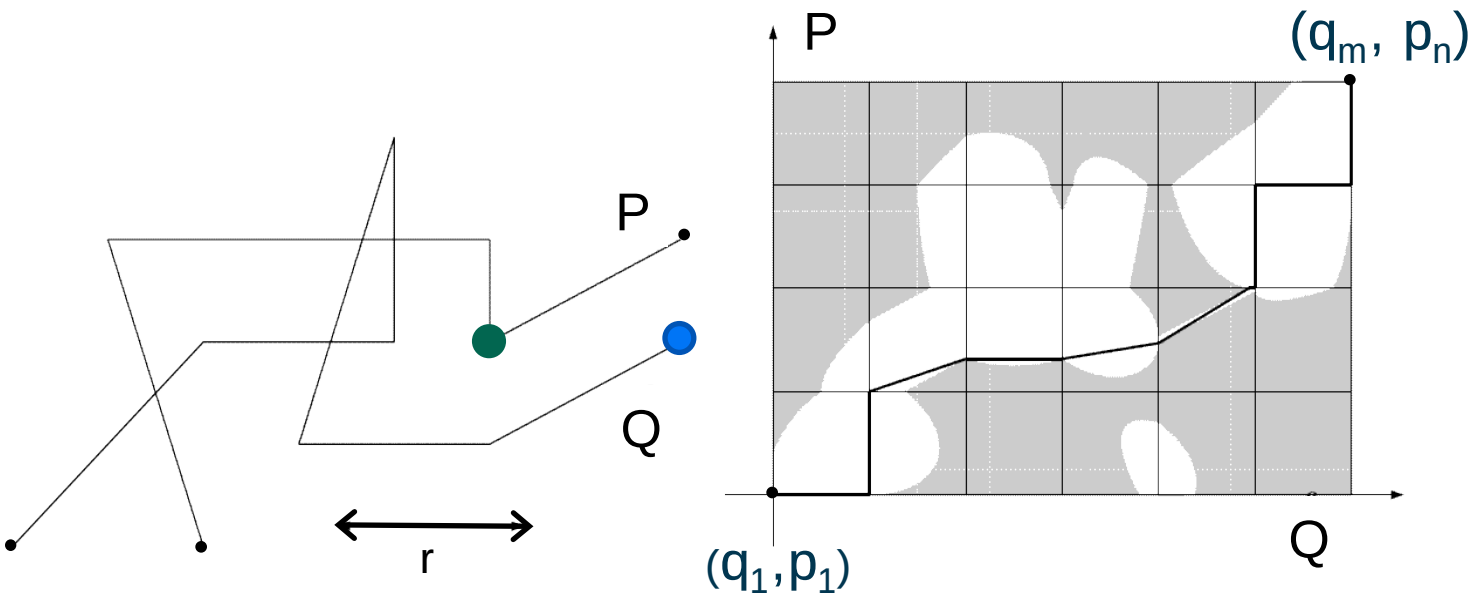
- If there exists a non-decreasing path in the free space from (q_1,p_1) to (q_m,p_n) which is non-decreasing in both coordinates, then curves P and Q have Fréchet distance at most r.





Free Space Diagram

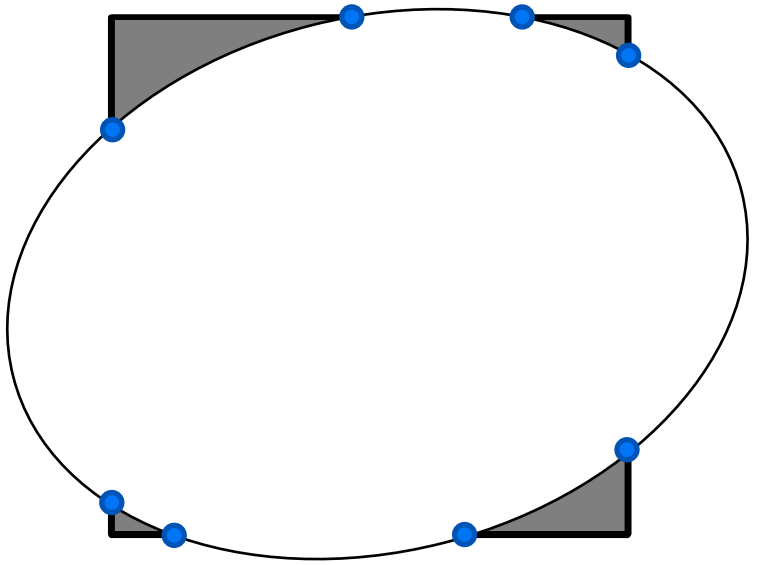
- If there exists a non-decreasing path in the free space from (q_1,p_1) to (q_m,p_n) which is non-decreasing in both coordinates, then curves P and Q have Fréchet distance at most r.



Decision algorithm: free space diagram

Algorithm:

1. Compute Free Space diagram
   O(mn) time



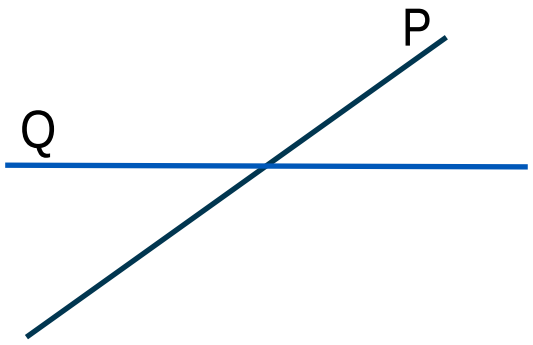
A square and an ellipsoid can intersect in at most 8 points.
(critical points)

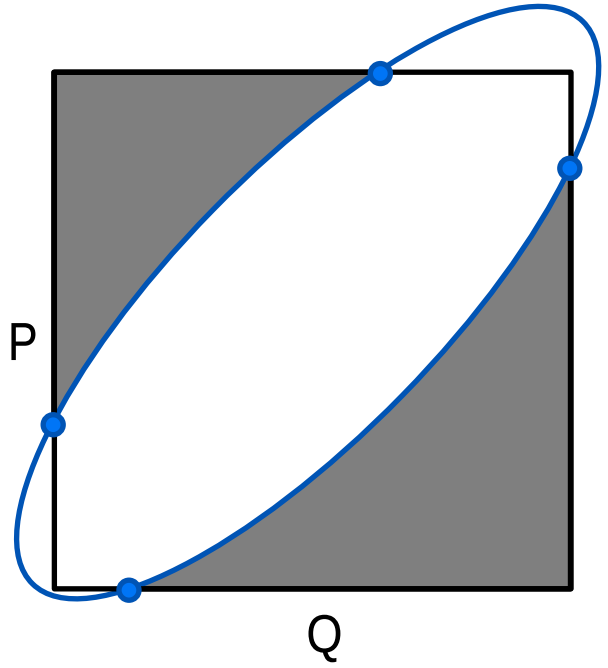
The free space in a cell is a convex region.

$\Rightarrow$ One cell can be constructed in time O(1).

Compute the Fréchet distance

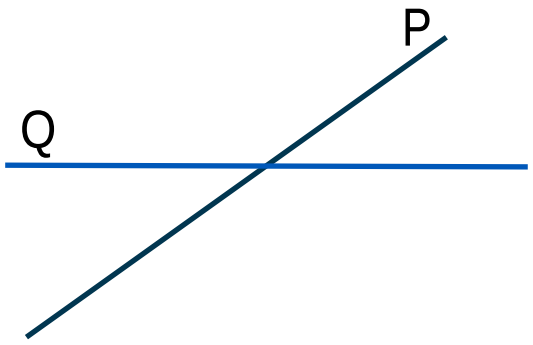
- One cell can be constructed in time O(1). How?
- Note that we do not need the exact shape. We only need the intersection points between the cell boundary and the ellipsoid.

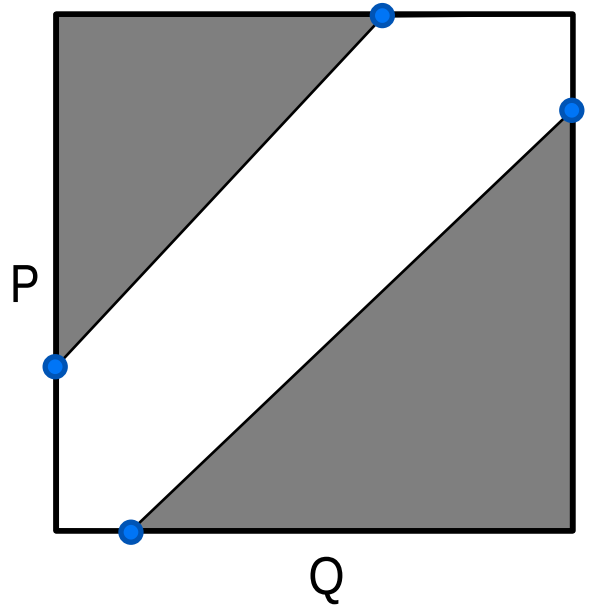




Compute the Fréchet distance

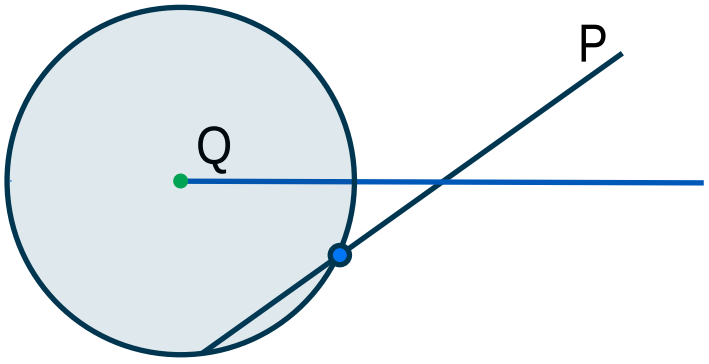
- One cell can be constructed in time O(1). How?
- Note that we do not need the exact shape. We only need the intersection points between the cell boundary and the ellipsoid.

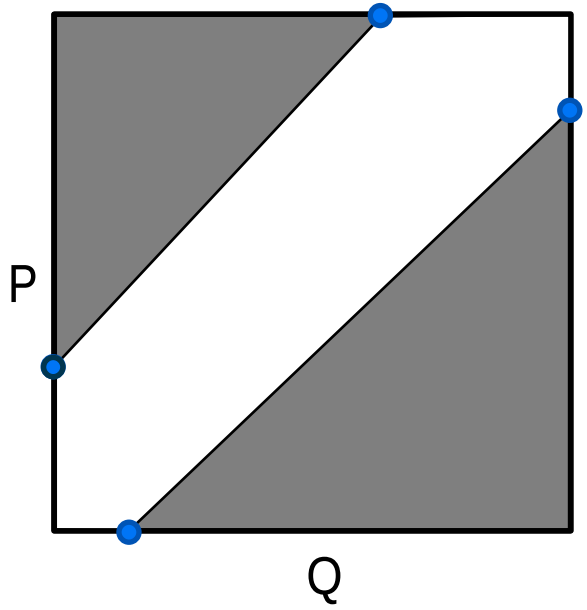




Compute the Fréchet distance

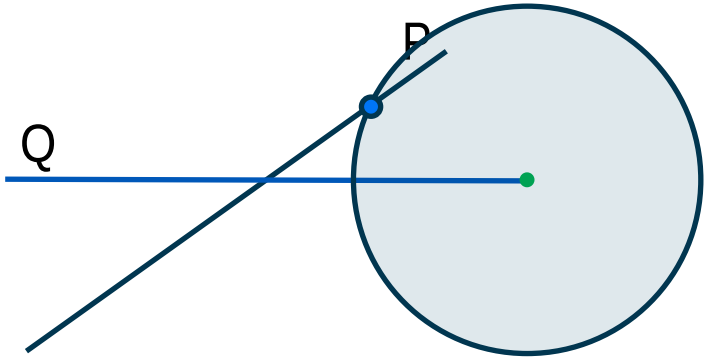
- One cell can be constructed in time O(1). How?
- Note that we do not need the exact shape. We only need the intersection points between the cell boundary and the ellipsoid.

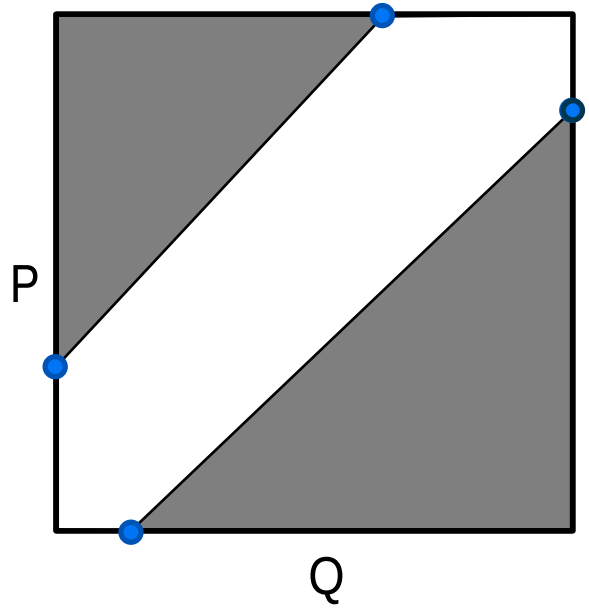




Compute the Fréchet distance

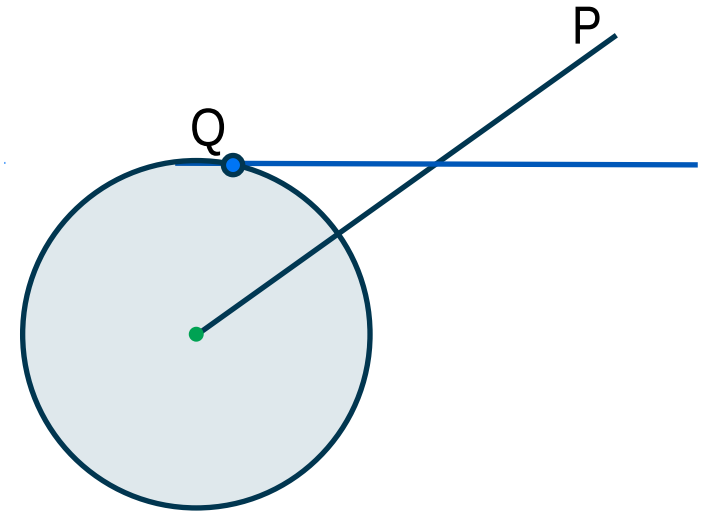
- One cell can be constructed in time O(1). How?
- Note that we do not need the exact shape. We only need the intersection points between the cell boundary and the ellipsoid.

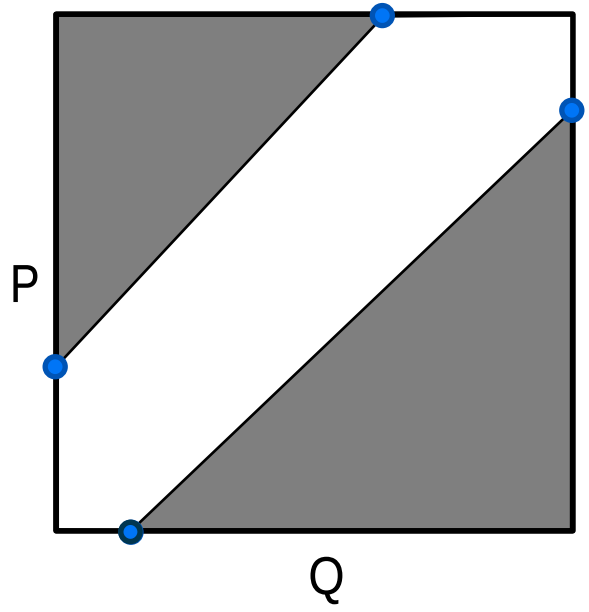




Compute the Fréchet distance

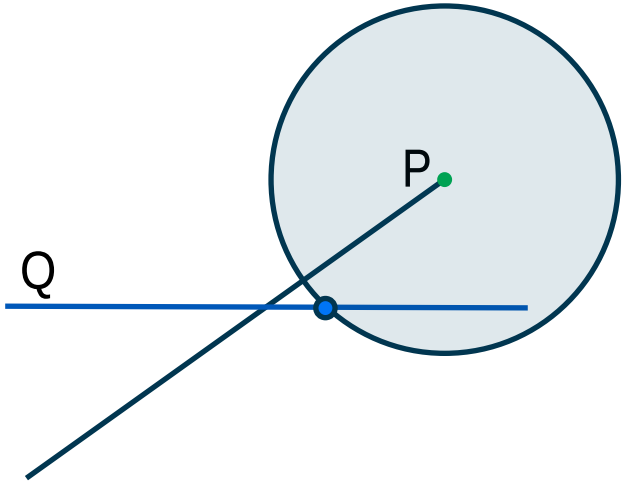
- One cell can be constructed in time O(1). How?
- Note that we do not need the exact shape. We only need the intersection points between the cell boundary and the ellipsoid.

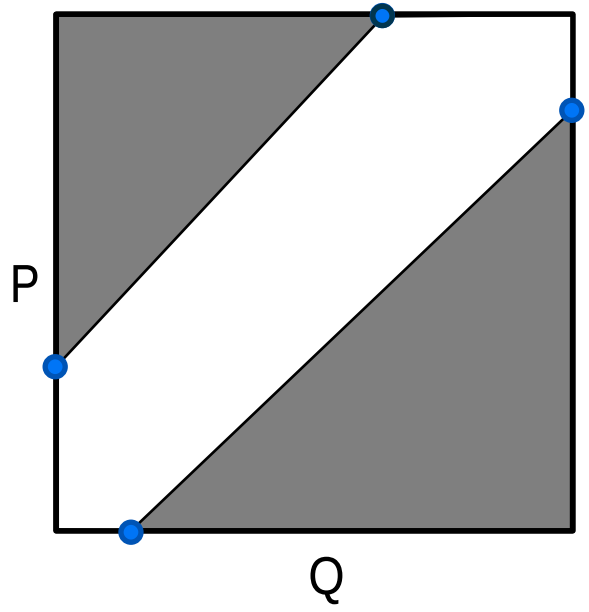




Compute the Fréchet distance

- One cell can be constructed in time O(1). How?
- Note that we do not need the exact shape. We only need the intersection points between the cell boundary and the ellipsoid.

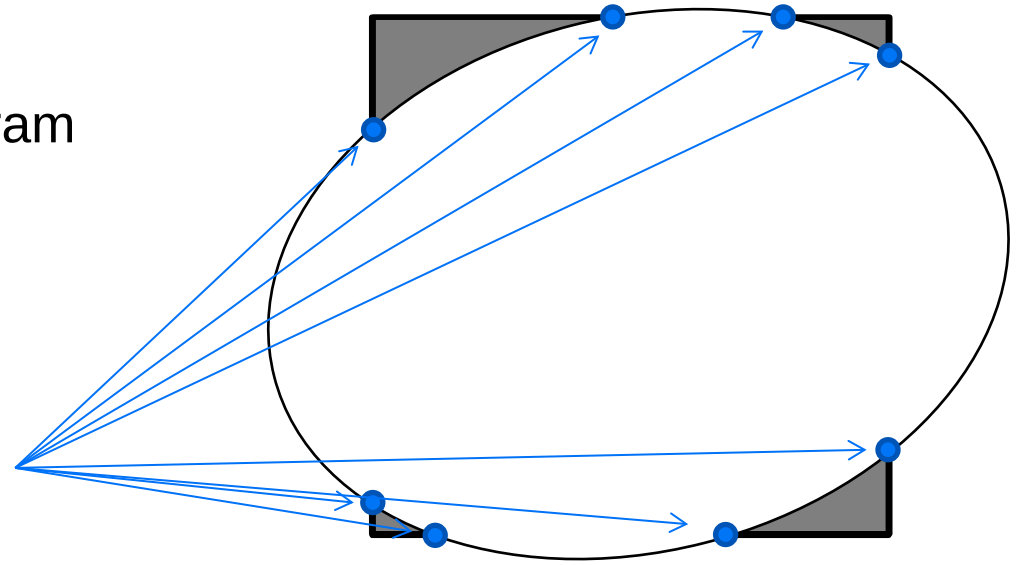




Decision algorithm: free space algorithm

Algorithm:

1. Compute Free Space diagram
   O(mn) time



A square and an ellipsoid can intersect in at most 8 points.

The free space in a cell is a convex region.

$\Rightarrow$ One cell can be constructed in time O(1).

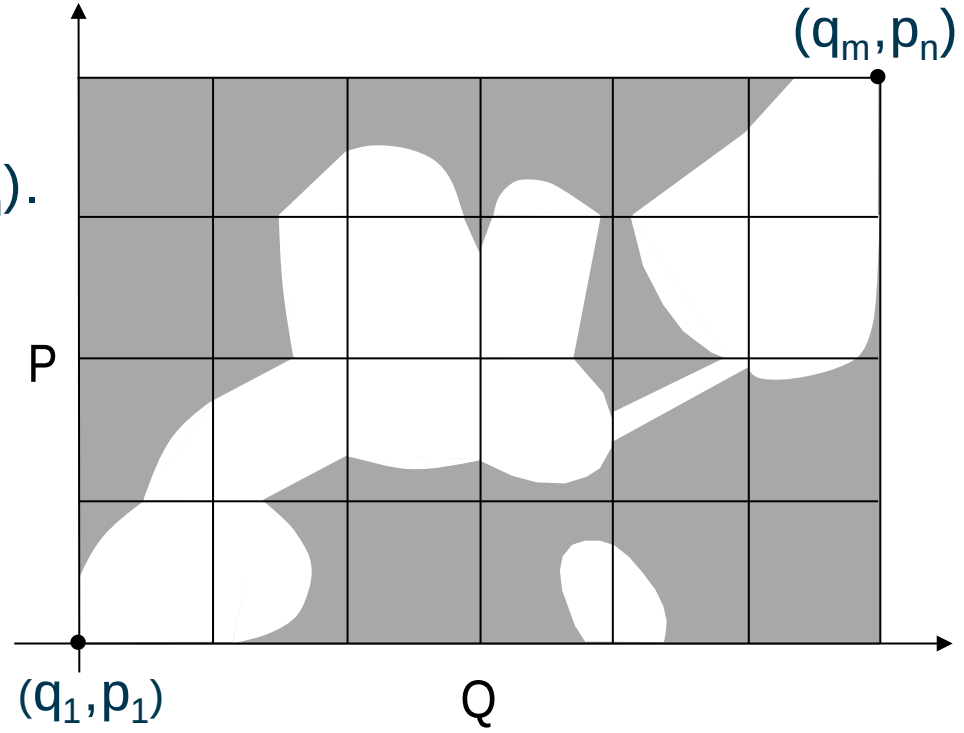
Decision algorithm: compute path

Algorithm:

1. Compute Free Space diagram

   O(mn) time

2. Compute a non-xy-decreasing path from (q_1,p_1) to (q_m,p_n).



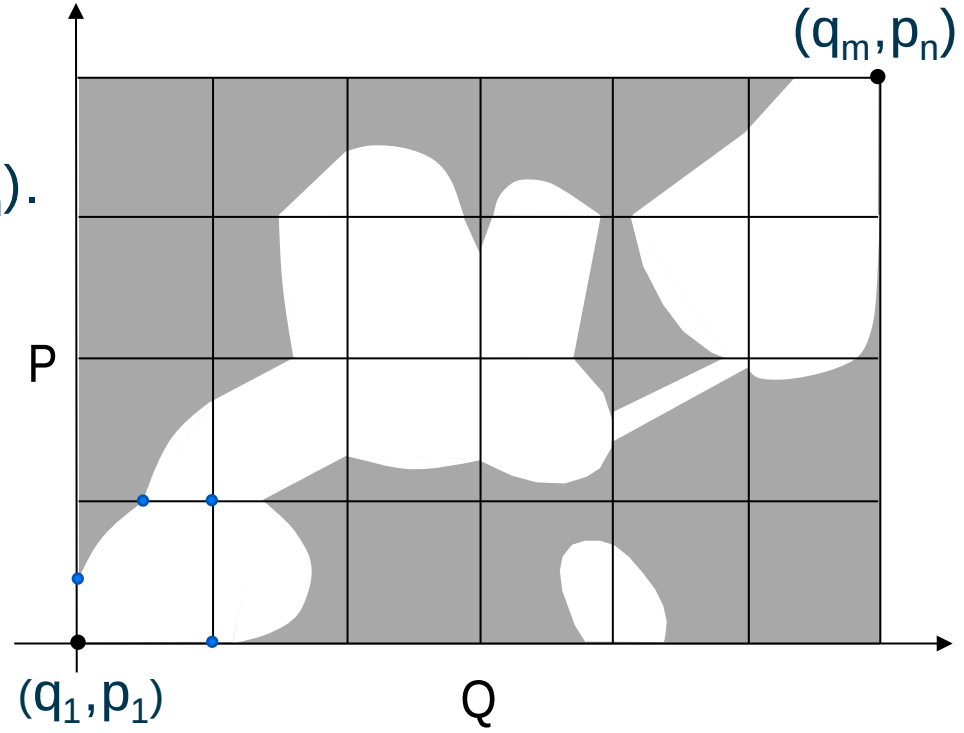
Decision algorithm: compute path

Algorithm:

1. Compute Free Space diagram

   O(mn) time

2. Compute a non-xy-decreasing path from (q_1,p_1) to (q_m,p_n).



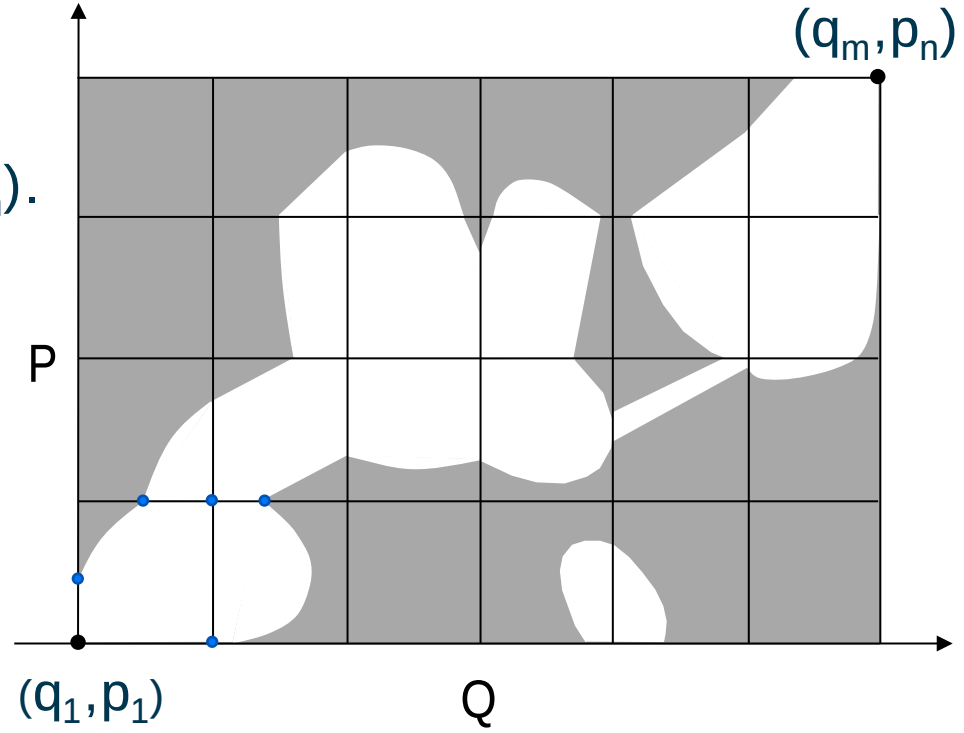
Decision algorithm: compute path

Algorithm:

1. Compute Free Space diagram

   O(mn) time

2. Compute a non-xy-decreasing path from (q_1,p_1) to (q_m,p_n).



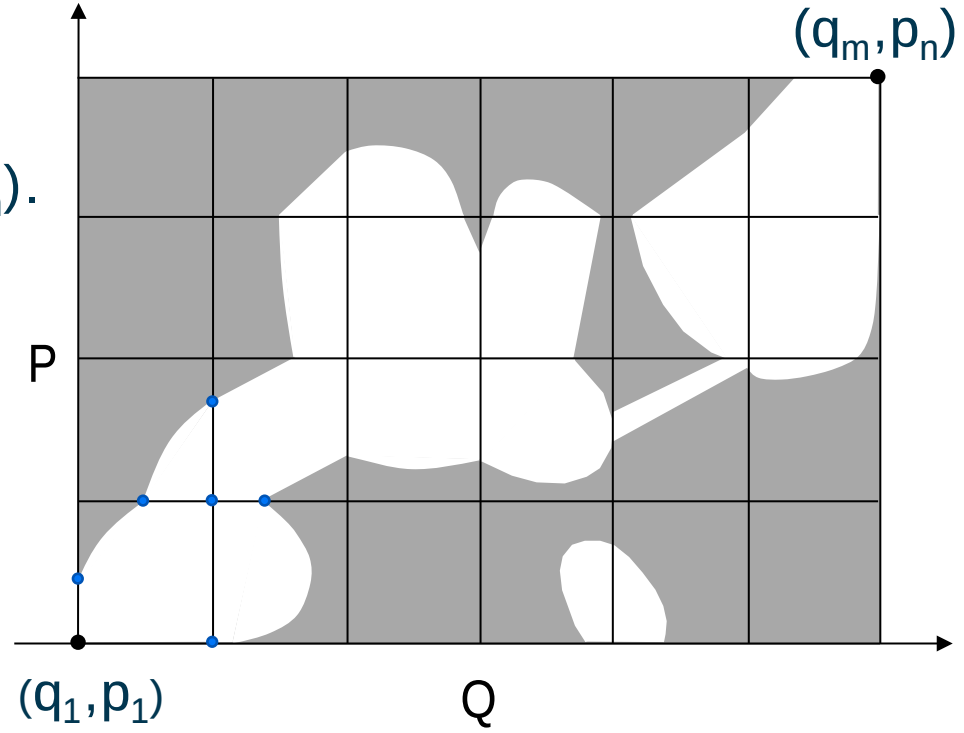
Decision algorithm: compute path

Algorithm:

1. Compute Free Space diagram

   O(mn) time

2. Compute a non-xy-decreasing path from (q_1,p_1) to (q_m,p_n).

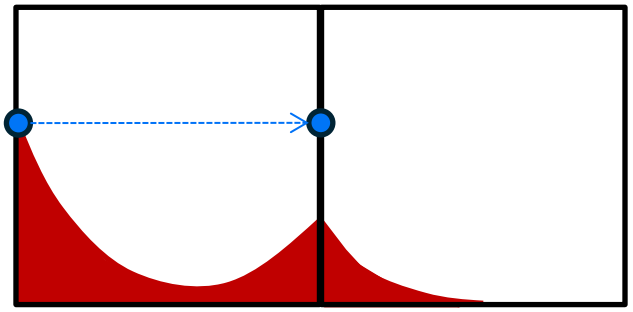


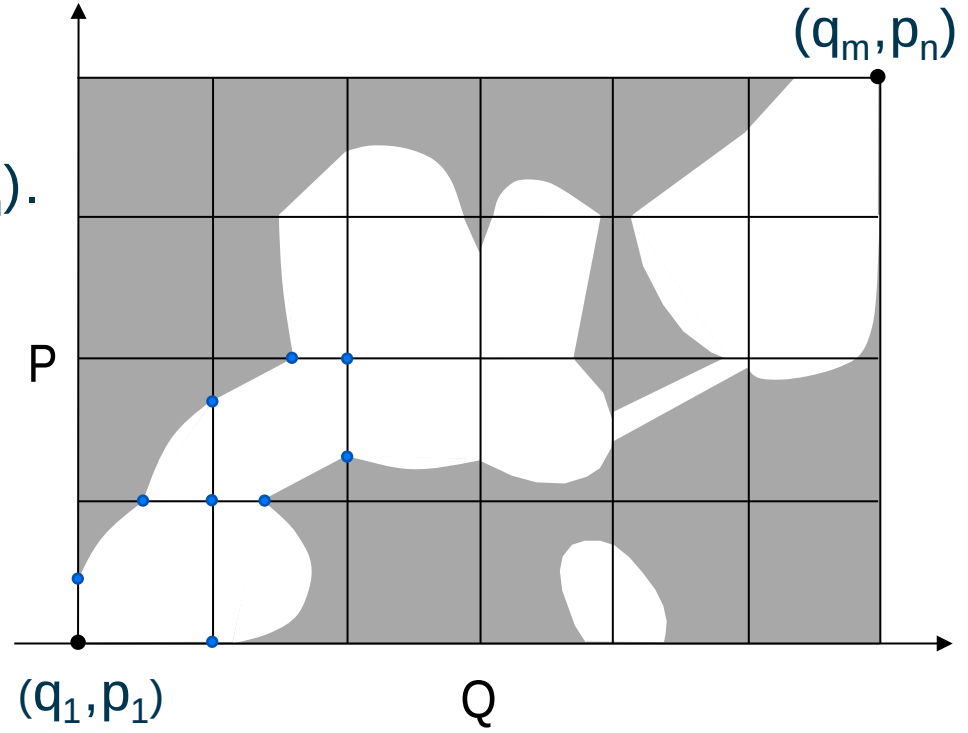
Decision algorithm: compute path

Algorithm:

1. Compute Free Space diagram
   O(mn) time

2. Compute a non-xy-decreasing path from (q_1,p_1) to (q_m,p_n).





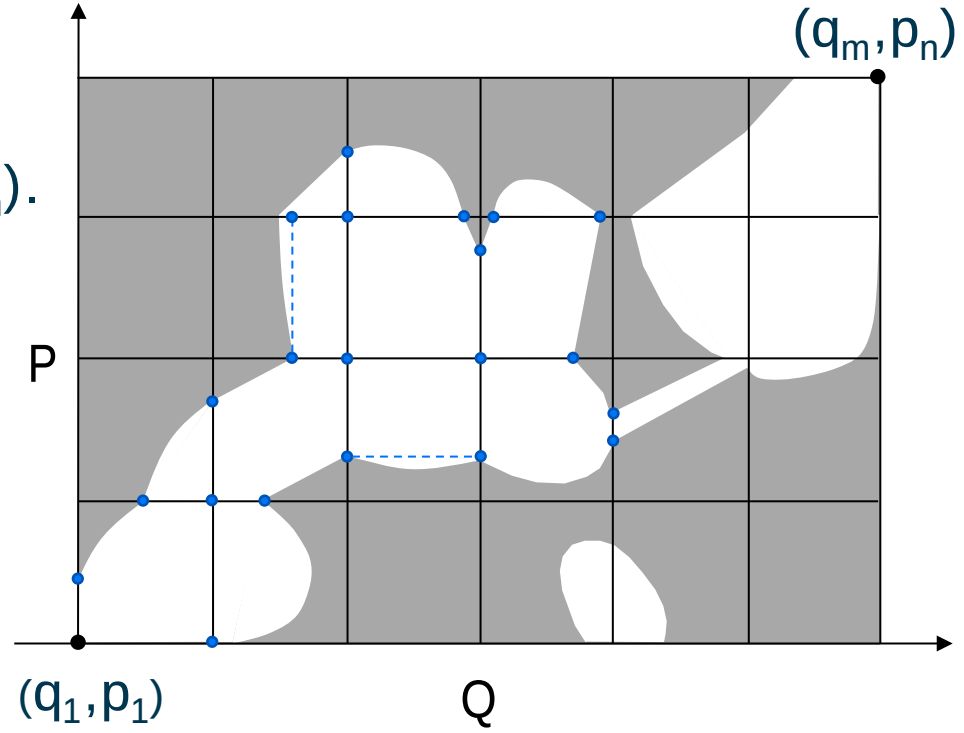
Decision algorithm: compute path

Algorithm:

1. Compute Free Space diagram

   O(mn) time

2. Compute a non-xy-decreasing path from (q_1,p_1) to (q_m,p_n).

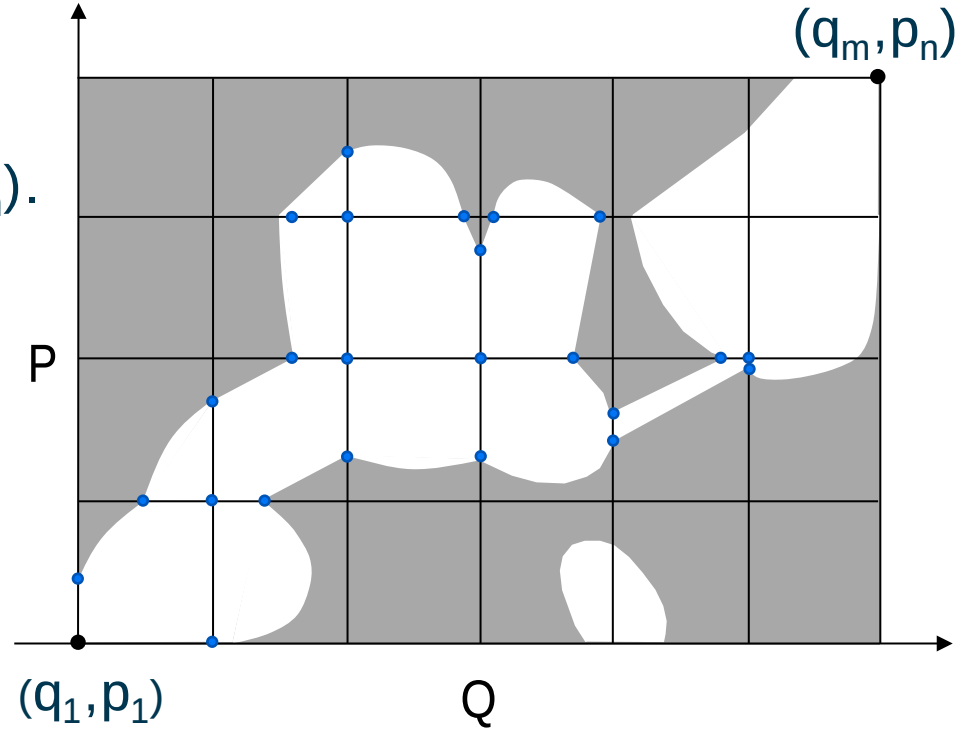


Decision algorithm: compute path

Algorithm:

1. Compute Free Space diagram
   O(mn) time

2. Compute a non-xy-decreasing path from (q_1,p_1) to (q_m,p_n).



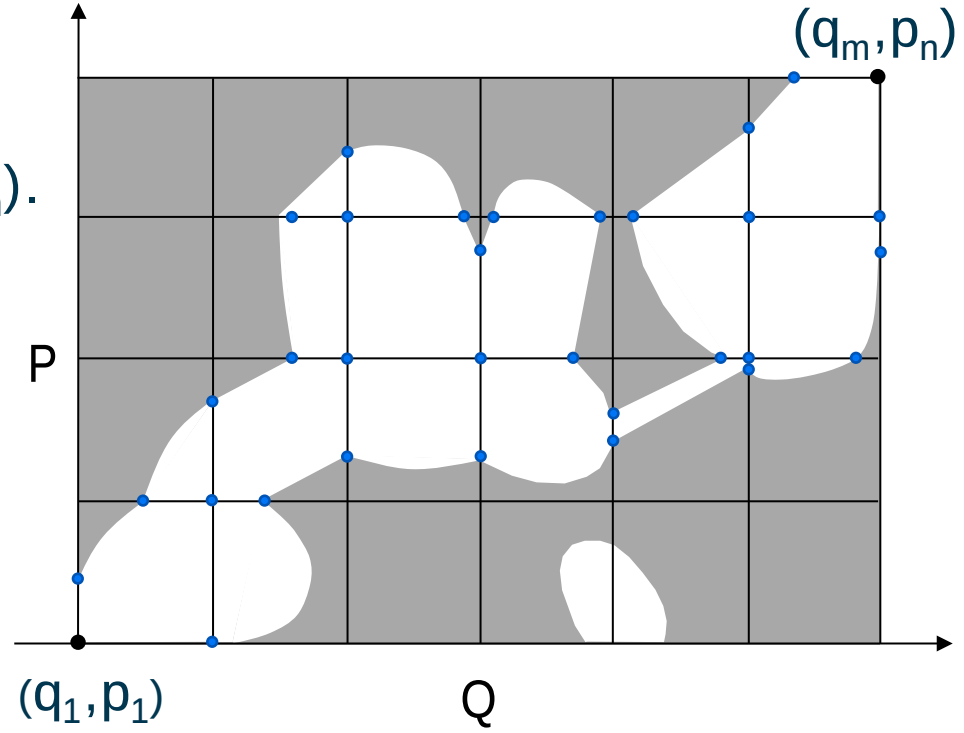
Decision algorithm: compute path

Algorithm:

1. Compute Free Space diagram

   O(mn) time

2. Compute a non-xy-decreasing path from (q_1,p_1) to (q_m,p_n).



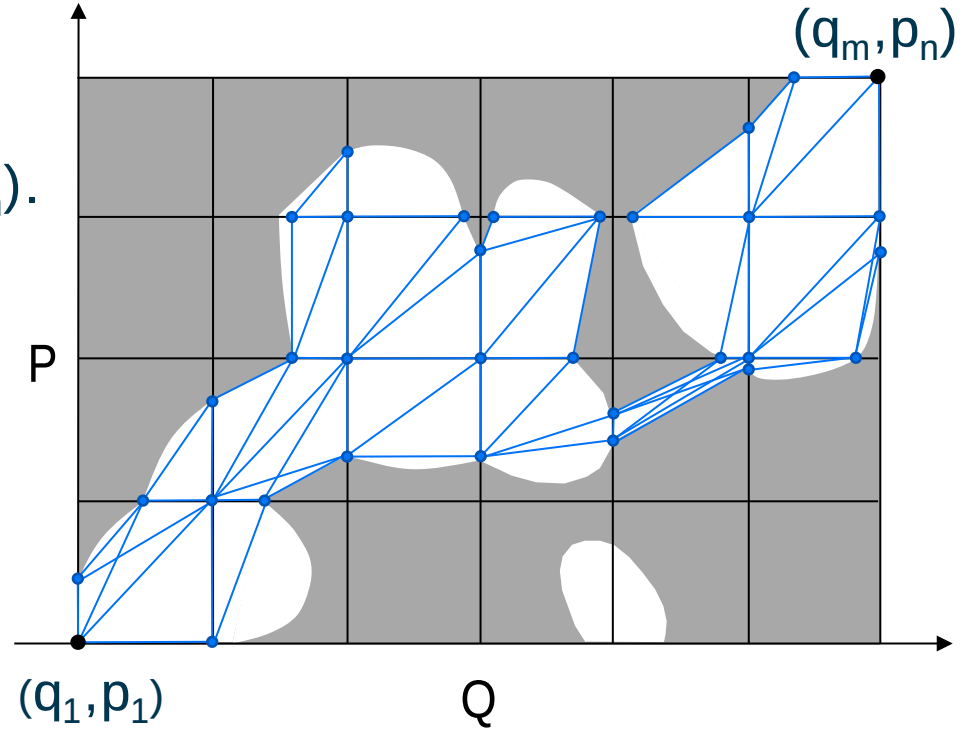
Decision algorithm: compute path

Algorithm:

1. Compute Free Space diagram

   O(mn) time

2. Compute a non-xy-decreasing path from (q_1,p_1) to (q_m,p_n).

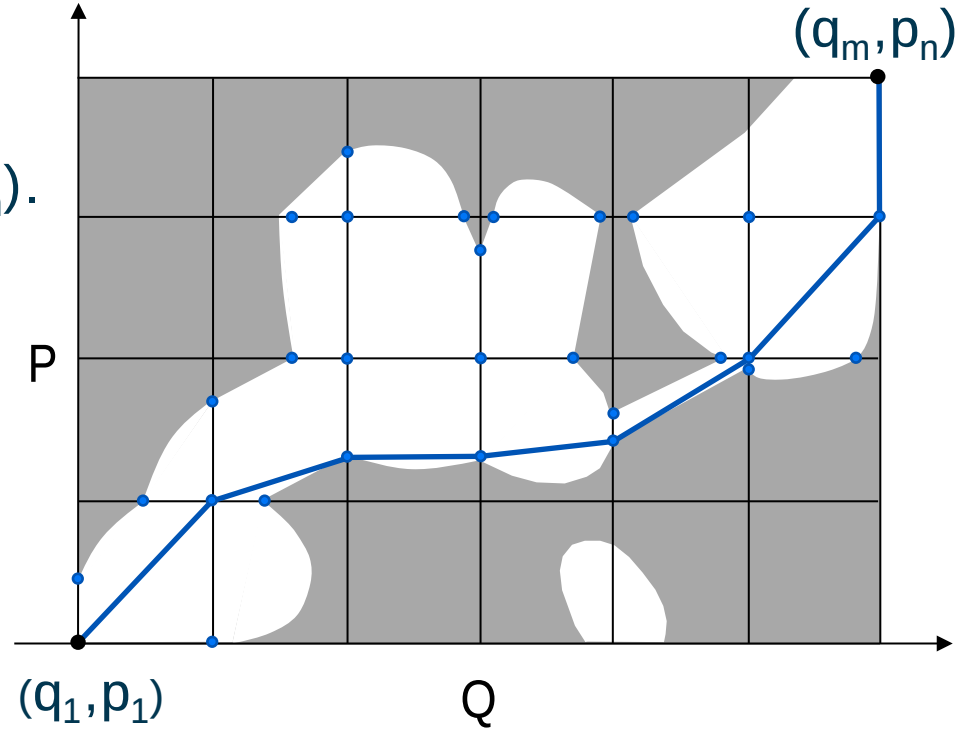


Decision algorithm: compute path

Algorithm:

1. Compute Free Space diagram
   O(mn) time

2. Compute a non-xy-decreasing path from (q_1,p_1) to (q_m,p_n).



Compute the Fréchet distance

- Theorem: Given two polygonal curves P, Q and $r \geq 0$ one can decide in O(mn) time, whether $\delta(P,Q) \leq r$.

- How can we use the algorithm to compute the Fréchet distance?

- Binary search on r?

- But what do we do binary search on?

Compute the Fréchet distance

- Theorem: Given two polygonal curves P, Q and $r \geq 0$ one can decide in O(mn) time, whether $\delta(P,Q) \leq r$.

- How can we use the algorithm to compute the Fréchet distance?

- In practice: Determine r bit by bit

$\Rightarrow$ O(mn log “accuracy”) time.

Compute the Fréchet distance

Assume r=0 and the let r grow.

$\Rightarrow$ The free space will become larger and larger.

Aim: Determine the smallest r such that it contains a montone path from (q_1,p_1) to (q_m,p_n).

This can only occur in one of the following cases:

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

Compute the Fréchet distance

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

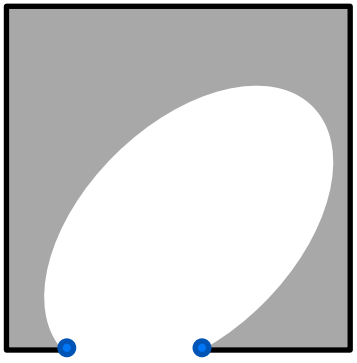
Case 1 is easy to test in constant time.

Compute the Fréchet distance

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

Case 1 is easy to test in constant time.

Case 2:

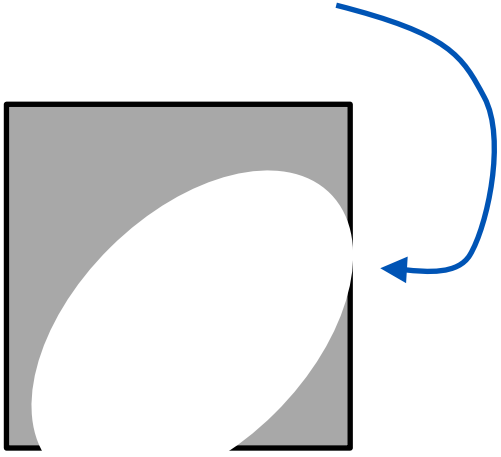


Compute the Fréchet distance

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

Case 1 is easy to test in constant time.

Case 2:

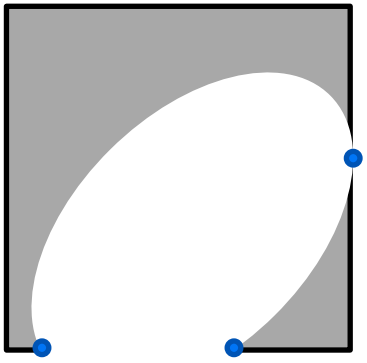


Compute the Fréchet distance

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

Case 1 is easy to test in constant time.

Case 2:

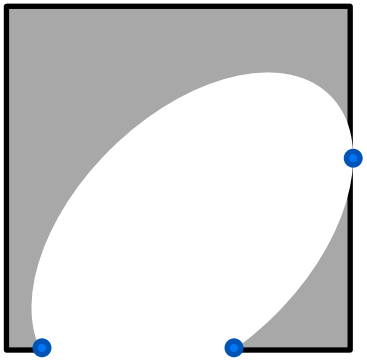


Compute the Fréchet distance

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

Case 1 is easy to test in constant time.

Case 2:



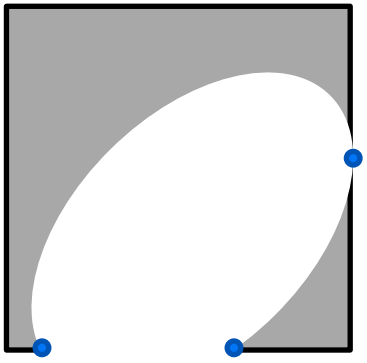
Number of events?

Compute the Fréchet distance

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

Case 1 is easy to test in constant time.

Case 2:

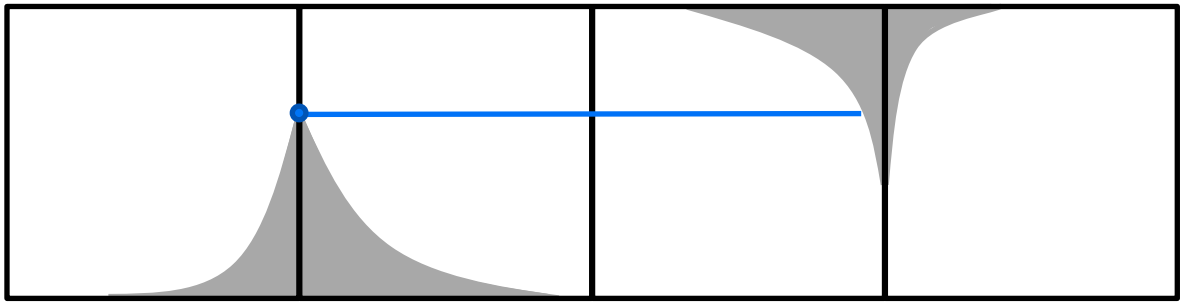


Number of events: O(mn)

Compute the Fréchet distance

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

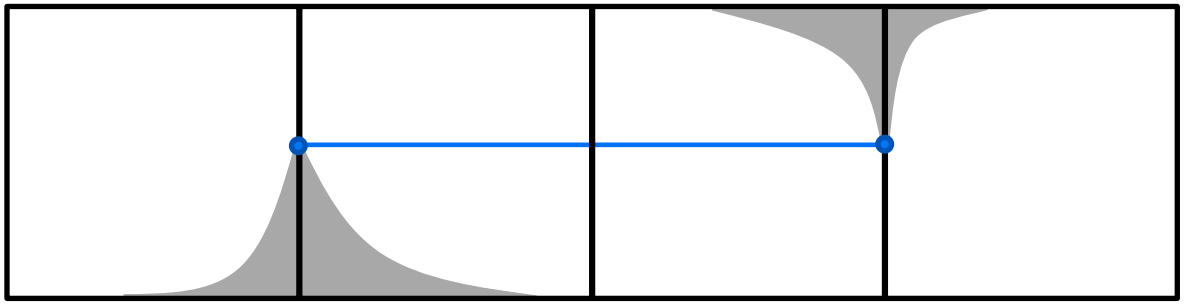
Case 1 is easy to test in constant time.
Case 2: O(mn) events
Case 3:



Compute the Fréchet distance

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

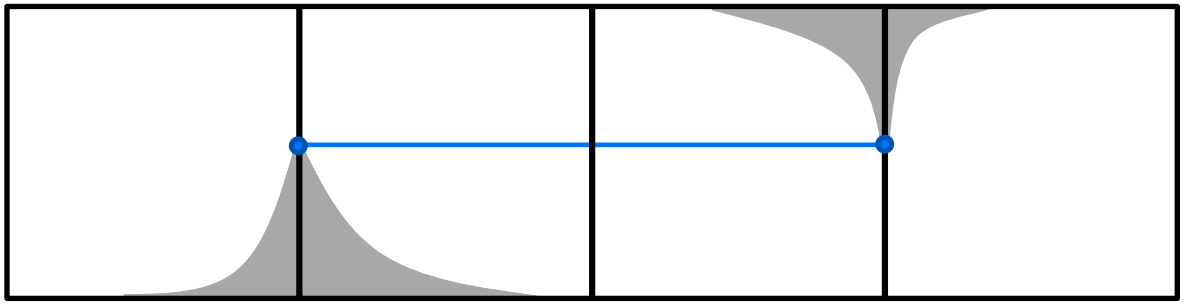
Case 1 is easy to test in constant time.
Case 2: O(mn) events
Case 3:



Compute the Fréchet distance

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

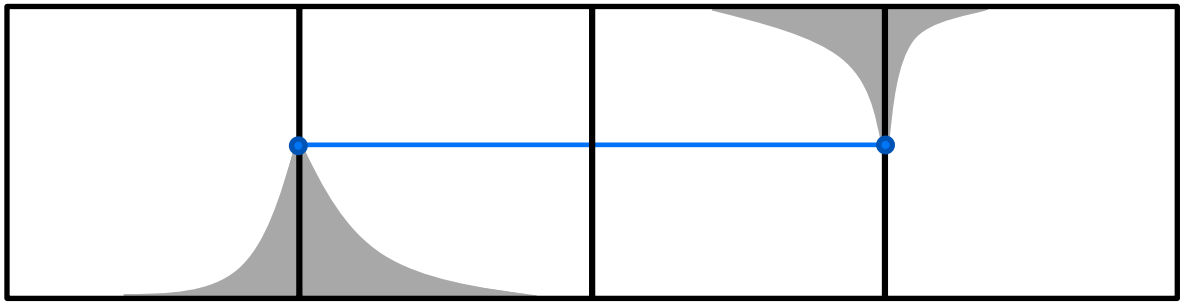
Case 1 is easy to test in constant time.
Case 2: O(mn) events
Case 3:



Compute the Fréchet distance

1. r is minimal with (q_1,p_1) to (q_m,p_n) in the free space,
2. r is minimal when a new vertical or horizontal passage opens up between two adjacent cells in the free space.
3. r is minimal when a new vertical or horizontal passage opens up between two non-adjacent cells in the free space.

Case 1 is easy to test in constant time.
Case 2: O(mn) events
Case 3:



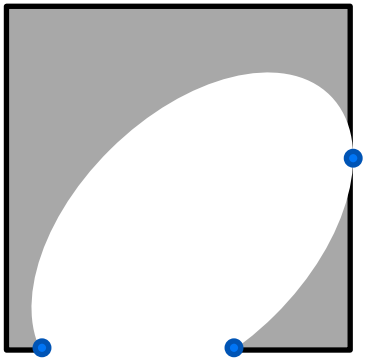
Compute the Fréchet distance

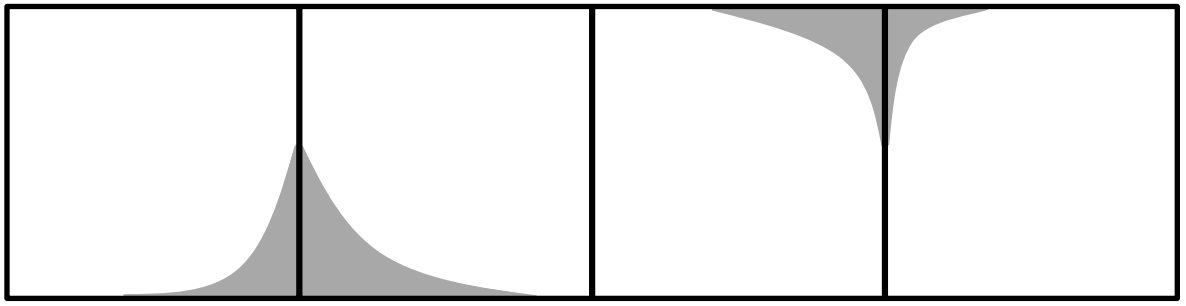
Case 1: easy to test in constant time.

Case 2: O(mn) possible events

Case 3: $O(m^2n+mn^2)$ possible events

Perform binary search on the events $\Rightarrow$ $O((m^2n+mn^2) \log mn)$ time.





Compute the Fréchet distance

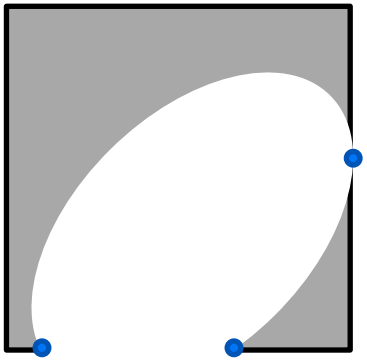
Case 1: easy to test in constant time.

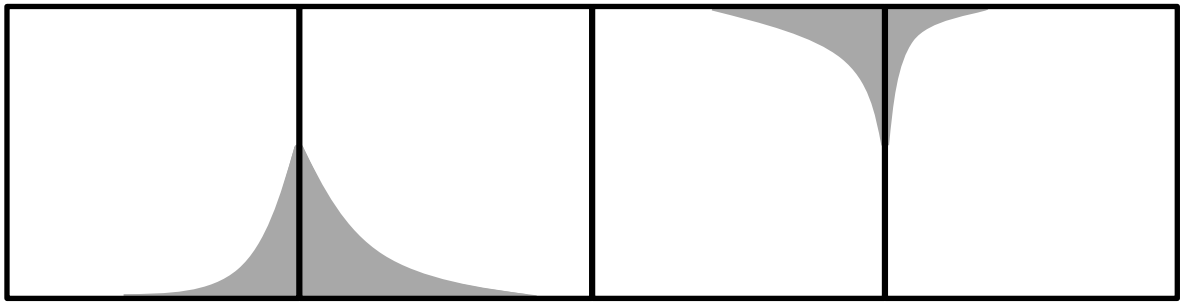
Case 2: O(mn) possible events

Case 3: $O(m^2n+mn^2)$ possible events

Improvement: Use parametric search [Meggido'83, Cole'88]

$\Rightarrow$ O(mn log mn) time.





Parametric search

The decision version F(r) of the Fréchet distance depends on the parameter r, and it is monotone in r.

Aim: Find minimum r such that F(r) is true.

Our algorithm A_s can solve F(r) in T_s=O(mn) time.

Meggido'83 and Cole'88 (simplified):
If there exists a parallel algorithm A_p that can solve F(r) in T_p time using P processors then there exists a sequential algorithm with running time $O(PT_p+T_pT_s+T_s \log P)$.

For our problem we get:

O(mn log mn) time using a parallel sorting network
[n processors and O(log n) time]

Summary: Similarity of Trajectories

- Many different similarity measures. Which is best depends on the application and the data.
- We use the Fréchet distance, or variants of it.
- Theorem: Given two polygonal curves P, Q and $r \geq 0$ one can decide in O(mn) time, whether $\delta(P,Q) \leq r$.
- Theorem: The Fréchet distance between two polygonal curves P and Q can be computed in O(mn log mn) time.

References

H. Alt and M. Godau. *Computing the Fréchet distance between two polygonal curves.* IJCGA, 1995.