LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Lehrstuhl für Datenbanksysteme und Data Mining
Sebastian Schmoll

Managing Massive Multiplayer Online Games

Tutorial 20.06.2018

HMM: Evaluation / Detection

The Hidden Markov Model (HMM) $M = \{S, B, D, F\}$ with $S = \{A, B, C\}$, $B = \{\clubsuit, \heartsuit, \spadesuit\}$ is given.

$$D = \begin{pmatrix} \times & - & A & B & C \\ - & 0 & 1/3 & 1/3 & 1/3 \\ A & 1/4 & 1/4 & 1/4 & 1/4 \\ B & 1/4 & 0 & 1/4 & 1/2 \\ C & 1/4 & 1/4 & 1/2 & 0 \end{pmatrix} \quad F = \begin{pmatrix} \times & \clubsuit & \heartsuit & \spadesuit \\ A & 1/4 & 3/4 & 0 \\ B & 0 & 0 & 1 \\ C & 0 & 1/4 & 3/4 \end{pmatrix}$$

- Compute the probability of the observation $\clubsuit, \heartsuit, \spadesuit$ inductively with help of the forward-variable
- Determine with help of the Viterbi-Algorithm which sequence of states most probably produced the observation $\clubsuit, \heartsuit, \spadesuit$

HMM: Evaluation / Detection - Forward-Variable

$$\alpha_A(1) = 1/3 \cdot 1/4 = 1/12$$
$$\alpha_B(1) = 0$$
$$\alpha_C(1) = 0$$

$$\alpha_A(2) = 1/12 \cdot 1/4 \cdot 3/4 = 1/64$$
$$\alpha_B(2) = 1/12 \cdot 1/4 \cdot 0 = 0$$
$$\alpha_C(2) = 1/12 \cdot 1/4 \cdot 1/4 = 1/192$$

$$\alpha_A(3) = 1/64 \cdot 1/4 \cdot 0 + 1/192 \cdot 1/4 \cdot 0 = 0$$
$$\alpha_B(3) = 1/64 \cdot 1/4 \cdot 1 + 1/192 \cdot 1/2 \cdot 1 = 5/786$$
$$\alpha_C(3) = 1/64 \cdot 1/4 \cdot 3/4 + 1/192 \cdot 0 \cdot 3/4 = 3/1024$$

$$\Rightarrow P(\clubsuit, \heartsuit, \spadesuit) = \alpha_A(3) \cdot d_{A,-} + \alpha_B(3) \cdot d_{B,-} + \alpha_C(3) \cdot d_{C,-} = 29/12288$$

HMM: Evaluation / Detection - Viterbi

Determine with help of the Viterbi-Algorithm which sequence of states most probably produced the observation ♣, ♡, ♠

$$\delta_A(1) = d_{-,A} f_{A,\clubsuit} = 1/12 \qquad \psi_A(1) = 0$$
$$\delta_B(1) = d_{-,B} f_{B,\clubsuit} = 0 \qquad \psi_B(1) = 0$$
$$\delta_C(1) = d_{-,C} f_{C,\clubsuit} = 0 \qquad \psi_C(1) = 0$$

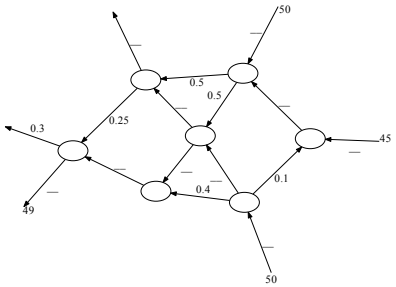
$$\delta_A(2) = \max\{\delta_A(1) \cdot d_{A,A}, \delta_B(1) \cdot d_{B,A}, \delta_C(1) \cdot d_{C,A}\} \cdot f_{A,\heartsuit} = 1/64 \qquad \psi_A(2) = A$$
$$\delta_B(2) = \max\{\delta_A(1) \cdot d_{A,B}, \delta_B(1) \cdot d_{B,B}, \delta_C(1) \cdot d_{C,B}\} \cdot f_{B,\heartsuit} = 0 \qquad \psi_B(2) = A$$
$$\delta_C(2) = \max\{\delta_A(1) \cdot d_{A,C}, \delta_B(1) \cdot d_{B,C}, \delta_C(1) \cdot d_{C,C}\} \cdot f_{C,\heartsuit} = 1/192 \qquad \psi_C(2) = A$$

$$\delta_A(3) = \max\{1/64 \cdot 1/4, 1/192 \cdot 1/4\} \cdot 0 = 0 \qquad \psi_A(3) = A$$
$$\delta_B(3) = \max\{1/64 \cdot 1/4, 1/192 \cdot 1/2\} \cdot 1 = 1/256 \qquad \psi_B(3) = A$$
$$\delta_C(3) = \max\{1/64 \cdot 1/4, 1/192 \cdot 0\} \cdot 3/4 = 3/1024 \qquad \psi_C(3) = A$$

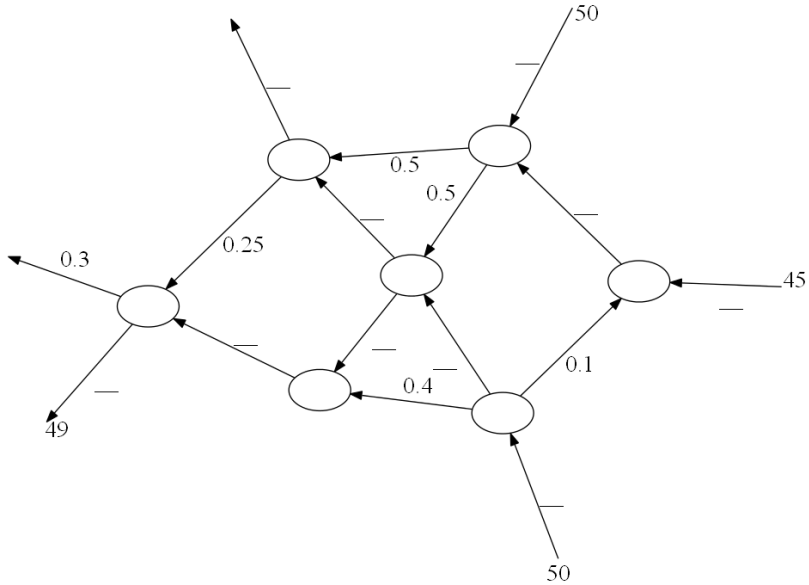
$$\delta(4) = \max\{1/256 \cdot 1/4, 3/1024 \cdot 1/4\} \approx \max\{0.0009766, 0.0007324\} \qquad \psi(4) = B$$

$$\Rightarrow \arg\max P(S_1 S_2 S_3 \mid \clubsuit, \heartsuit, \spadesuit) = AAB$$

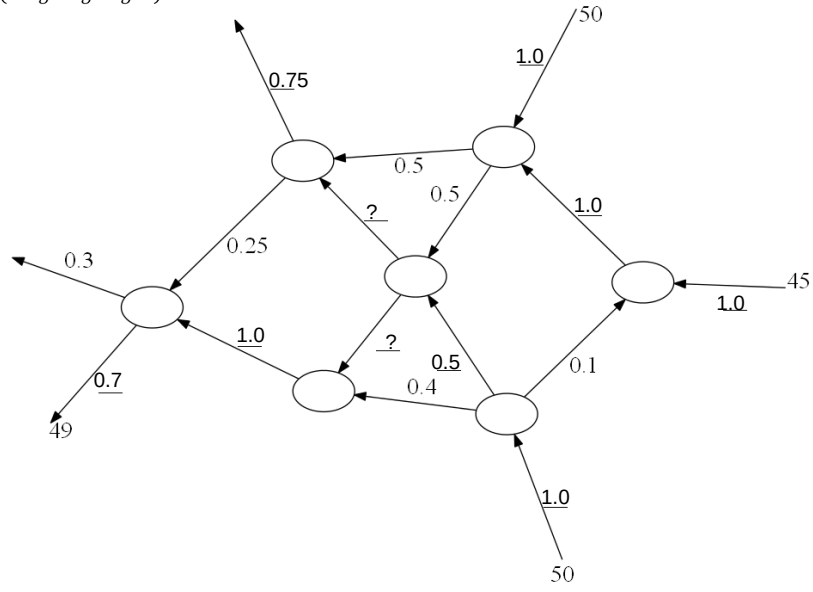
Homogeneous Poisson Models

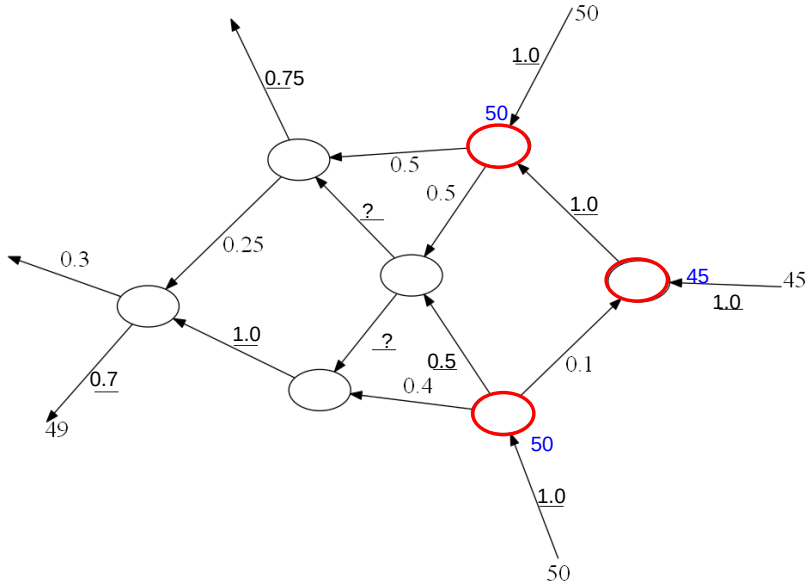


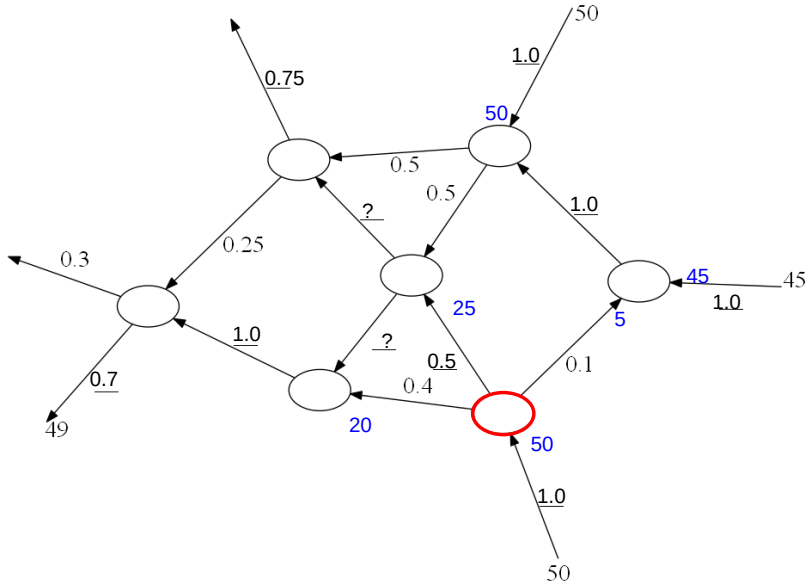
Labels of entering edges indicate the number of characters entering the represented area. Labels of the other edges indicate the probability that a character decides upon the corresponding way. Assume that the motion inside the road network follows a homogeneous poisson process. Calculate the missing probabilities and for every edge the expected number of characters who are located at every time on the way represented by the edge.

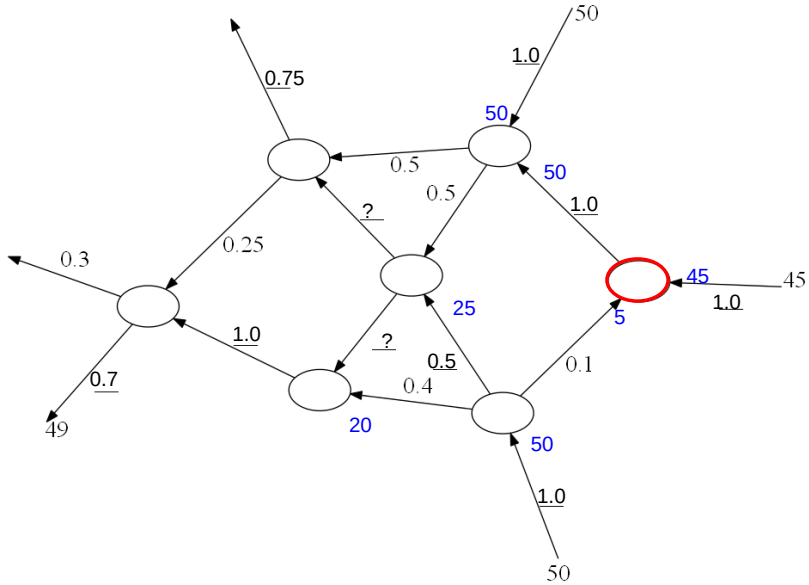

50
0.5
0.5
0.25
0.3
45
0.1
0.4
49
50

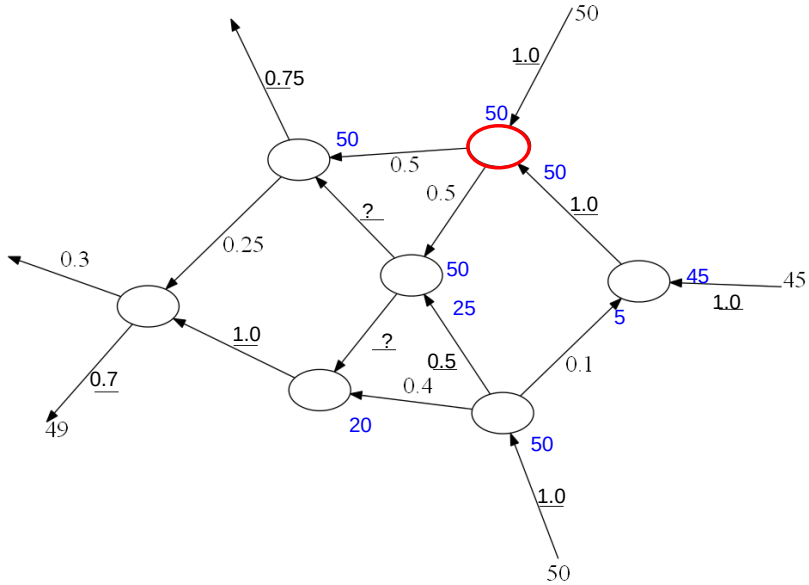
$$\sum(outgoing\ edges) = 1$$

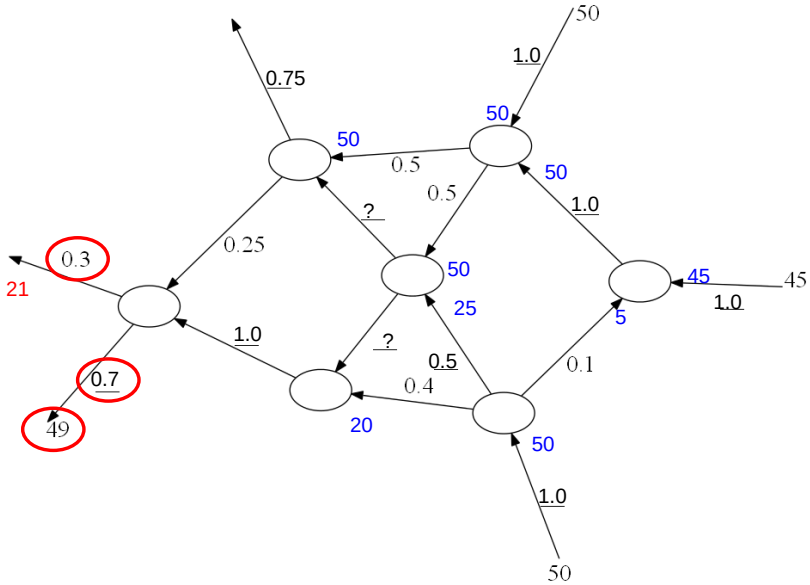



50
1.0
0.75
50
0.5
0.5
1.0
?
0.25
0.3
45
45
1.0
1.0
?
0.5
0.1
0.7
49
0.4
50
1.0
50

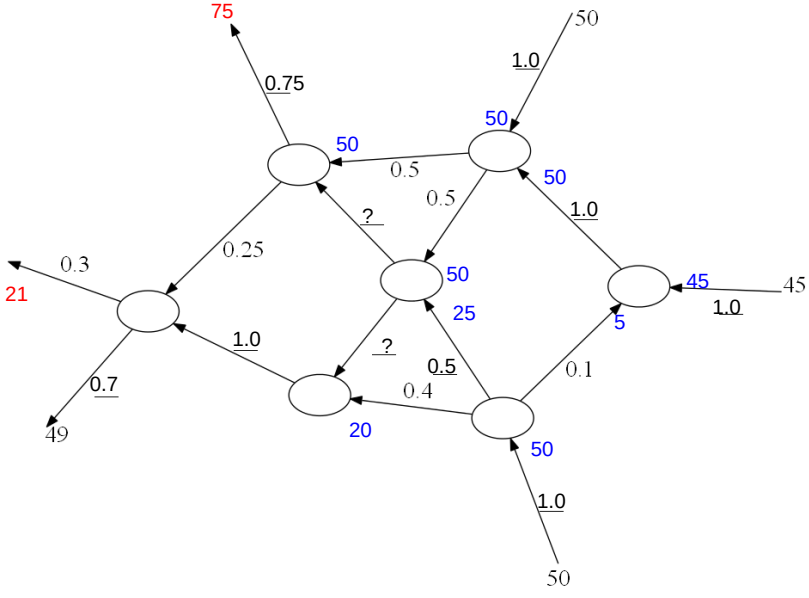

50
1.0
0.75
50
0.5
0.5
?
1.0
0.25
0.3
45
45
1.0
25
5
?
1.0
0.5
0.1
0.7
0.4
49
20
50
1.0
50

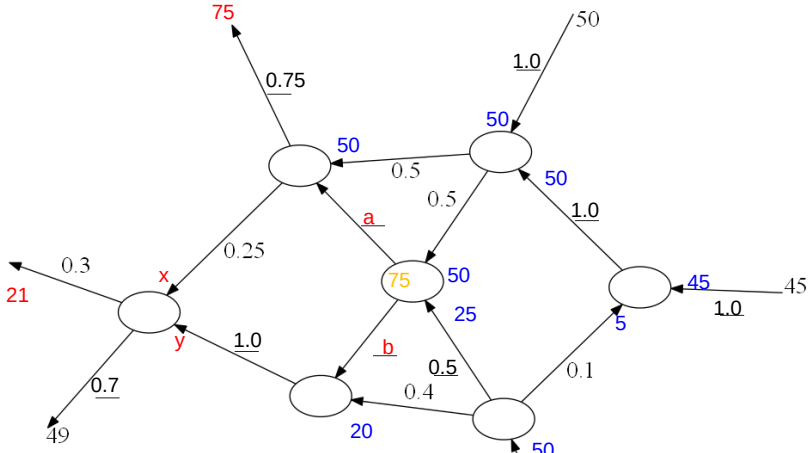

50
1.0
0.75
50
0.5
0.5
50
?
1.0
0.25
0.3
45
45
1.0
25
5
?
1.0
0.5
0.1
0.7
0.4
49
20
50
1.0
50


50
1.0
0.75
50
50
0.5
0.5
50
?
1.0
0.25
0.3
50
45
45
1.0
25
5
?
1.0
0.5
0.1
0.7
0.4
49
20
50
1.0
50

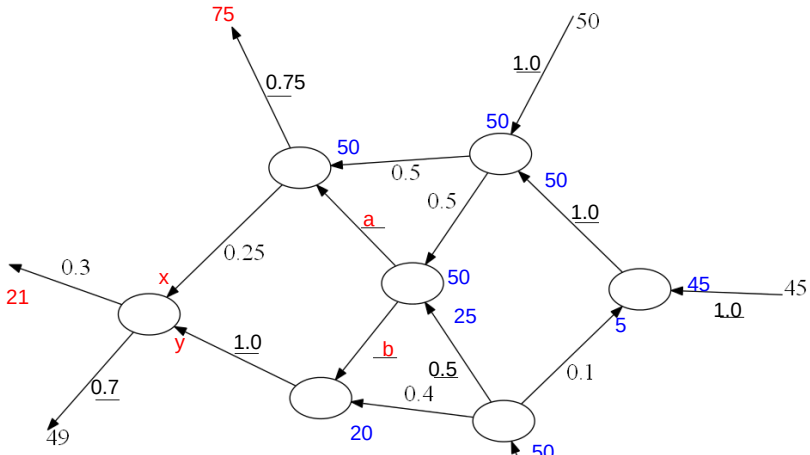

50
1.0
0.75
50
50
0.5
0.5
50
1.0
?
0.25
0.3
21
50
45
45
1.0
25
5
?
1.0
0.5
0.1
0.7
0.4
49
20
50
1.0
50

$$\sum \text{all ingoing edges} = \sum \text{all outgoing edges}$$



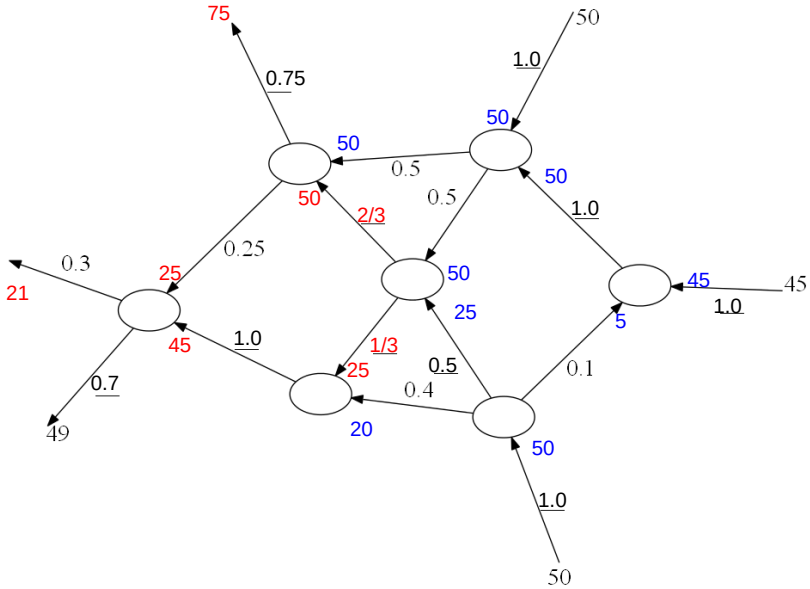


x = 0.25*(50+75a)
y = 1.0 * (20+75b)
x+y = 70
a+b = 1.0

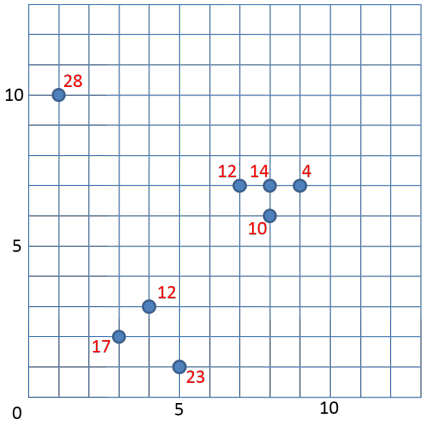


x = 0.25*(50+75a)
y = 1.0 * (20+75b)
x+y = 70
a+b = 1.0

a = 2/3
b = 1/3
y = 45
x = 25


75
0.75
50
1.0
50
50
0.5
50
2/3
0.5
50
1.0
0.3
21
25
0.25
50
45
45
1.0
25
5
45
1.0
1/3
25
0.5
0.1
0.7
49
0.4
20
50
1.0
50

Spatial Outlier Detection



In the following relevant spatial positions (e.g. starting positions in an FPS or frequent camp positions in an MMORPG) are given. For every position a score is given additionally which depicts semantic information about the quality of the position (e.g. the average number of frags in an FPS or the average count of EP/coins per hour in an MMORPG).

Find the three strongest outliers in this dataset applying the Point Outlier Detection Algorithm with $k = 2$. Use the absolute score difference as weighting function.

Parameters:

k = #neighbors [here: = 2]

m = #(outliers to look for) [here: = 3]

Approach:

a) Compute *kNN*-graph (directed edge from p to q exists if and only if *q* is a *kNN* of *p*) [here: using the euclidean distance]

b) For every edge in the *kNN*-graph calculate its weight [here: the absolute score difference]

c) Beginning with the highest weighted edge, delete edges with descending weights from the graph

d) If a point is isolated (having neihter in- nor out-going edges) after the successive removal of edges, it is an outlier

e) Proceed like that until *m* outliers are found

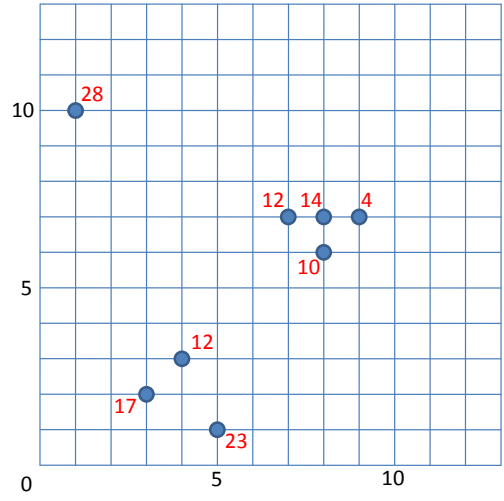
Possible options for equality of *kNN*-distance:

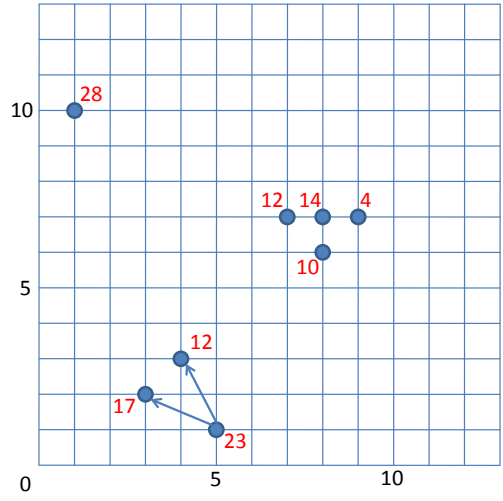
1) Add all points with equal *kNN*-distance to the *kNN*-set (potentially *#kNN >=k*)
2) Choose non-deterministically one of the points with equal *kNN*-distance (=> *#kNN=k*)

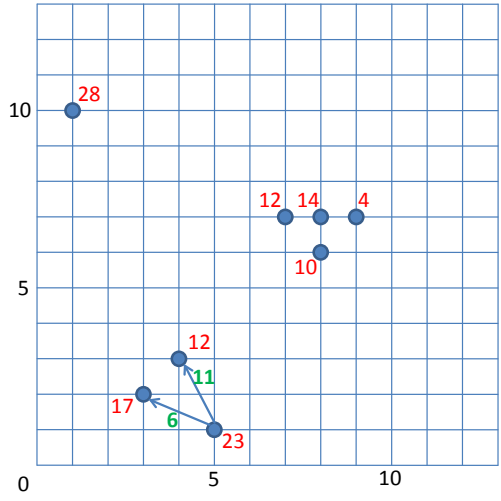
Possible options for equality of weighting of edges in the kNN-graph:

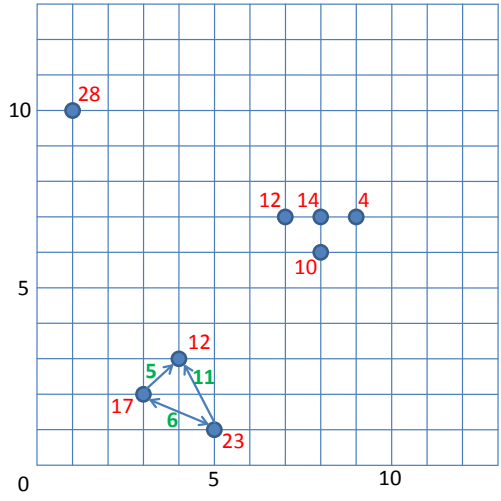
1) Delete all edges with equal weight at once
2) Delete one of the edges non-deterministically

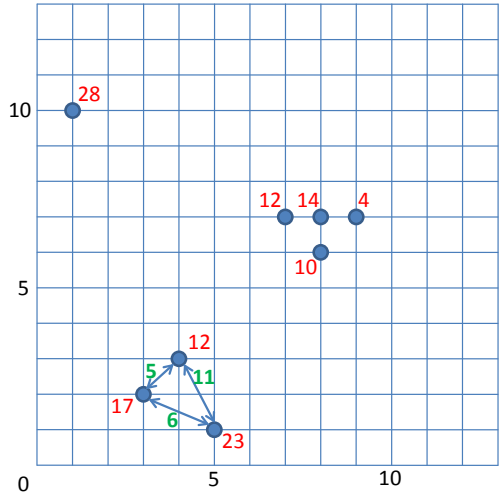
We choose strategy 1) and 1)

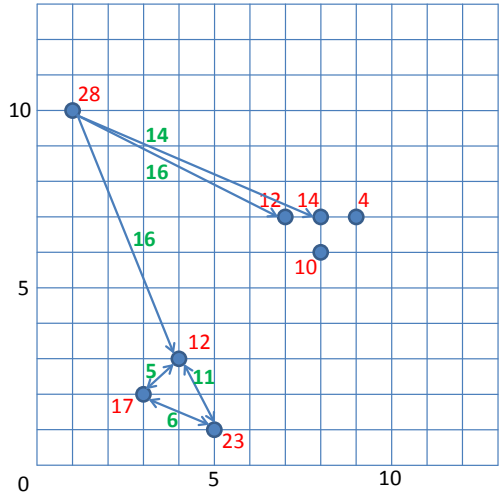

28
12
14
4
10
12
17
23
10
5
0
5
10

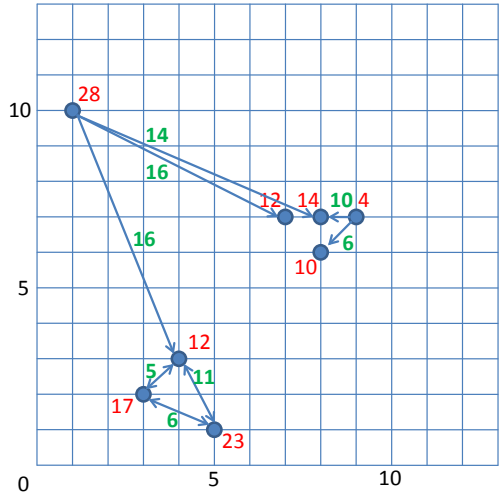

28
10
12
14
4
10
5
12
17
23
0
5
10

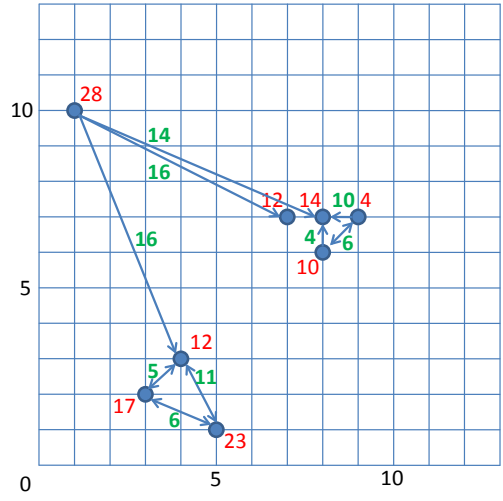

28
10
12
14
4
10
5
12
11
17
6
23
0
5
10

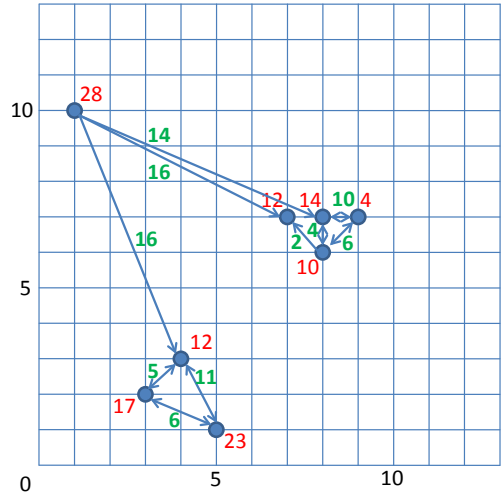

28
10
12
14
4
10
5
12
5
11
17
6
23
0
5
10


28
10
12
14
4
10
5
12
5
11
17
6
23
0
5
10

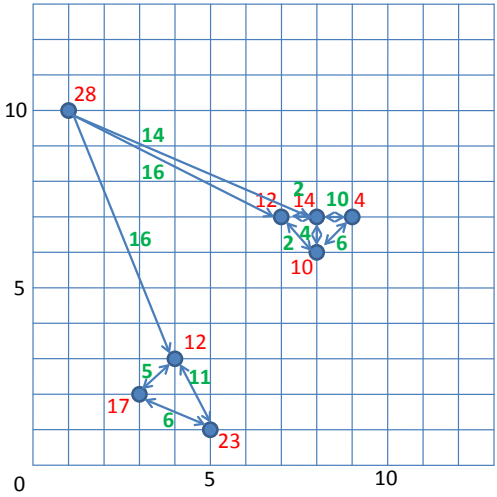

28
14
16
12
14
4
16
10
12
5
11
17
6
23
10
5
0
5
10

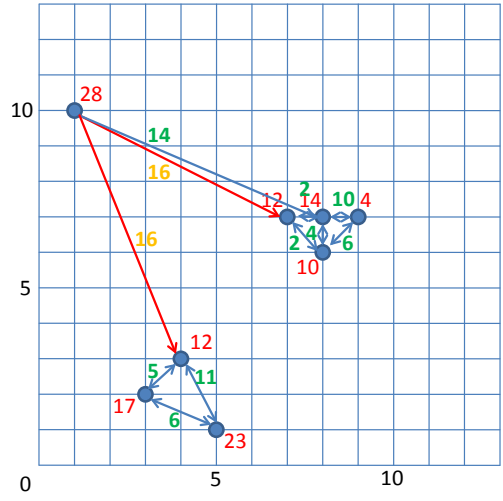

28
14
16
12
14
10
4
6
10
16
12
5
11
17
6
23
10
5
0
5
10

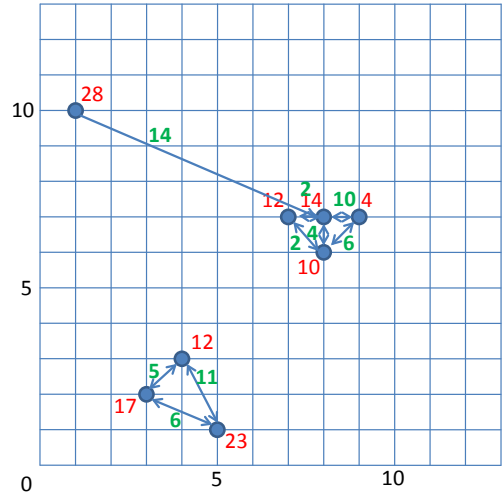

28
14
16
12
14
10
4
4
6
10
16
12
5
11
17
6
23
10
5
0
5
10

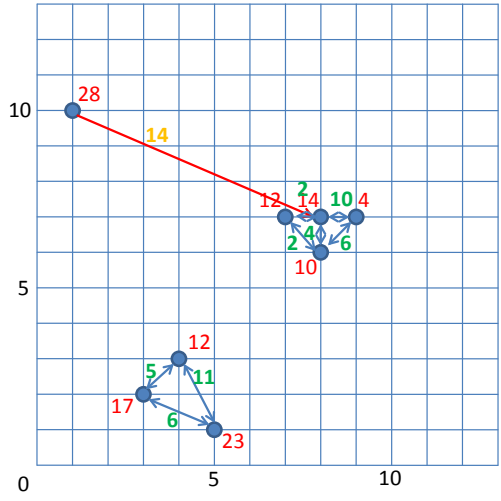

28
14
16
16
12
14
10
4
2
4
6
10
12
5
11
17
6
23
10
5
0
5
10

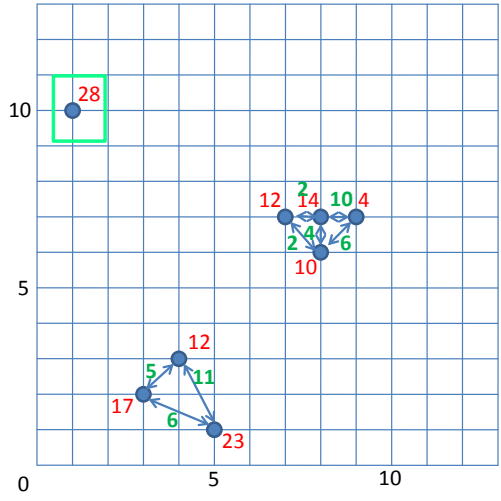
Construction of neighborhood graph finished
Now delete the edges with highest weights (=score difference) in a descending order

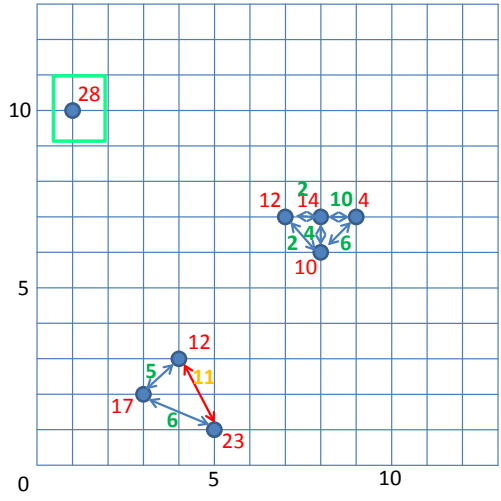


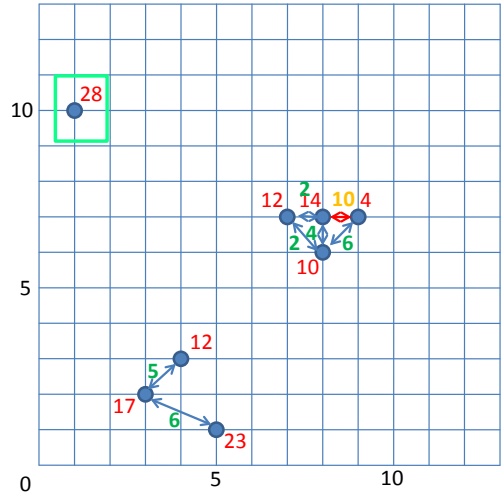

28
14
16
16
12
14
10
4
2
4
2
6
10
12
5
11
17
6
23
10
5
0
5
10

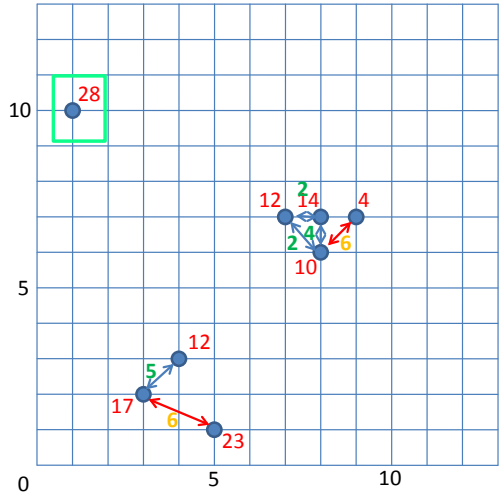

28
14
12
14
10
4
2
4
2
6
10
12
5
11
17
6
23
10
5
0
5
10


28
14
12
14
2
10
4
2
4
6
10
12
5
11
17
6
23
10
5
0
5
10

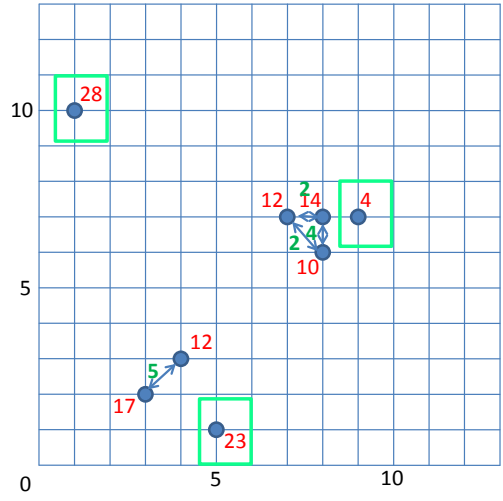

28
10
5
0
5
10
12
14
10
4
2
4
2
6
10
12
5
11
17
6
23


28
10
5
0
5
10
12
14
10
4
2
4
2
6
10
12
5
11
17
6
23


28
10
5
0
5
10
12
14
10
4
2
4
2
6
10
12
5
17
6
23


28
10
5
0
5
10
12
14
4
2
2
4
6
10
12
5
17
6
23

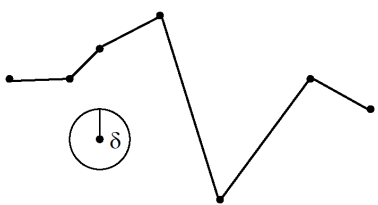
The three strongest outliers are determined



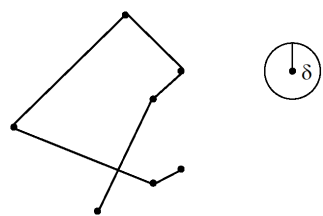
Exercise 10-3 *Compression of trajectories*

Approximate the following trajectories with the Douglas Peucker Algorithm

a)



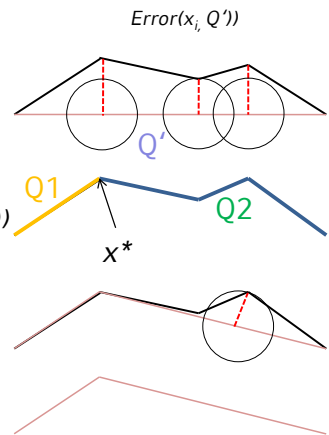
b)



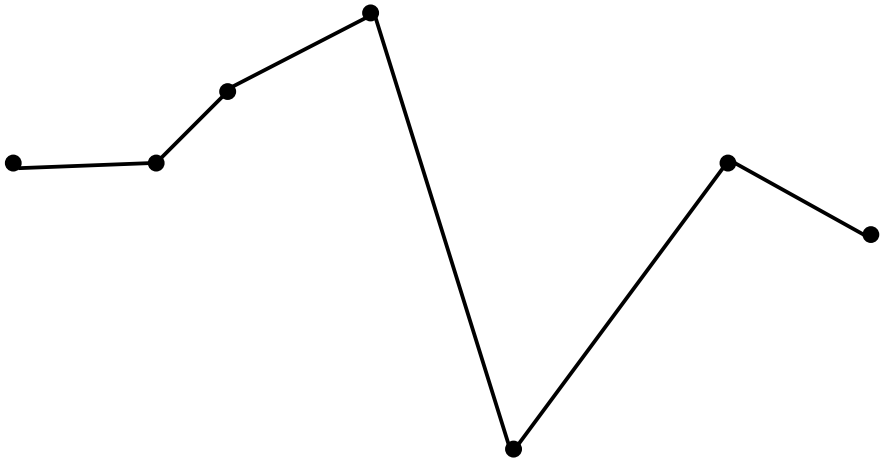
Douglas-Peucker Algorithm

Given: A trajectory $Q=((x_1,t_1), \ldots, (x_l,t_l))$ of l length.
Searched: Q' with | Q' |<< l and approximation error smaller than δ .
Algorithm:

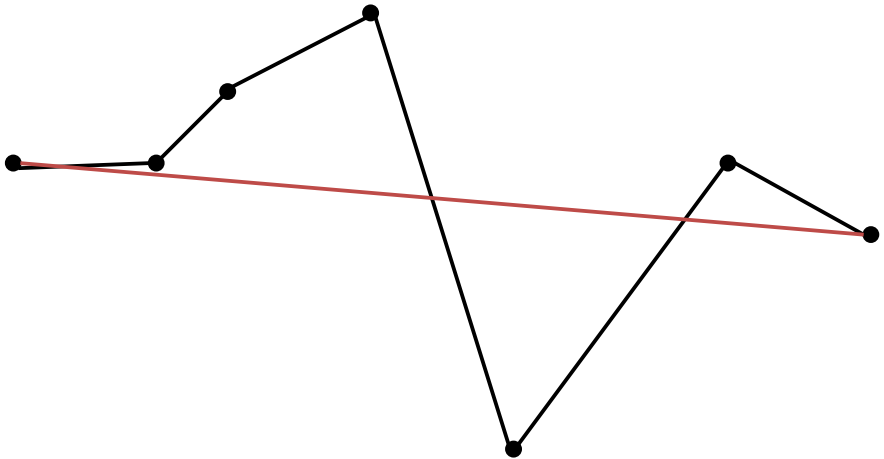
DP(Q, δ)
Q' = $((x_1,t_1), (x_l,t_l))$
FOR ALL (x_i,t_i) *in Q*
  IF $Error(x_i, Q') > \delta$ THEN
    determine x^* with $\max(Error(x_i, Q'))$
    $(Q1,Q2) = split(Q,x^*)$
    RETURN DP(Q1, δ) DP(Q2, δ)
ENDFOR
RETURN Q'



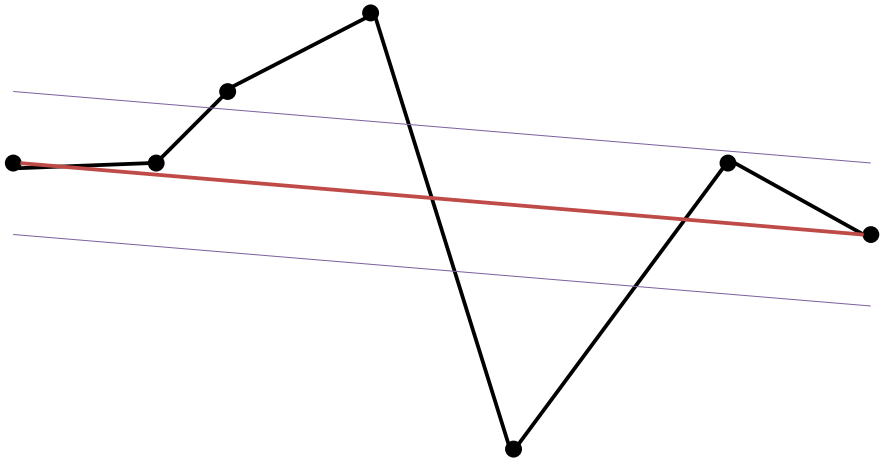
10-3 a)



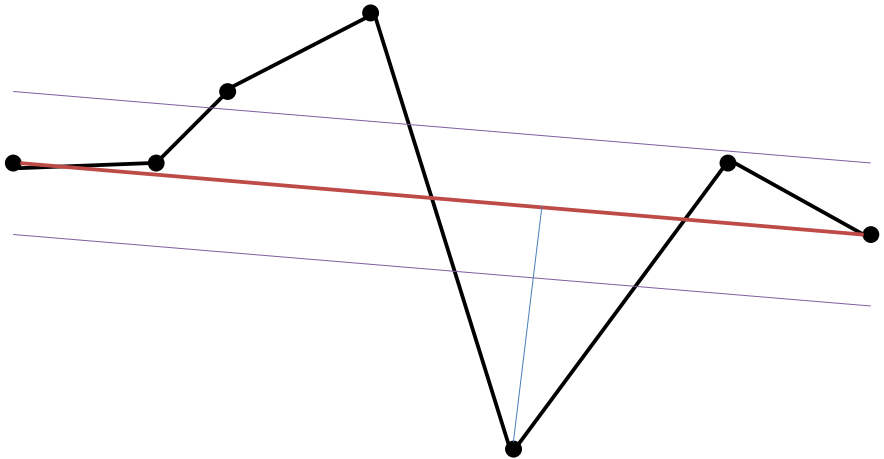
10-3 a)



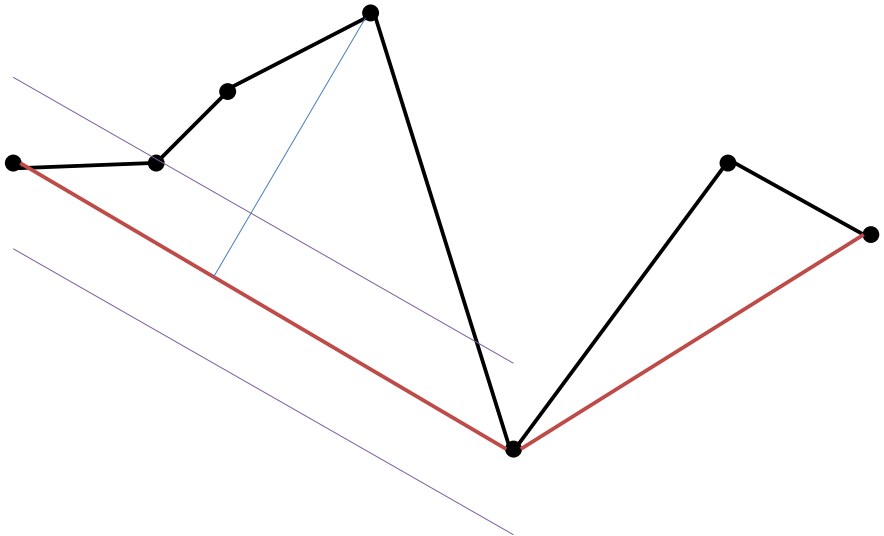
10-3 a)



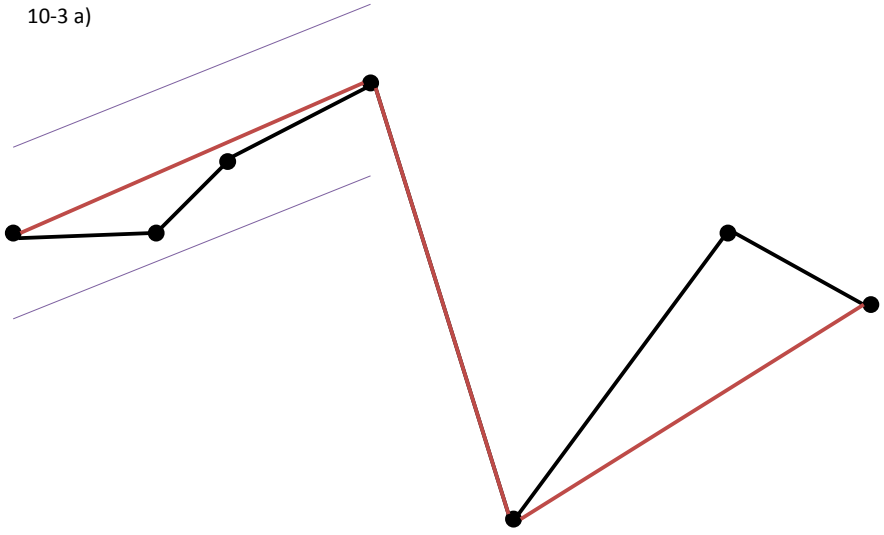
10-3 a)



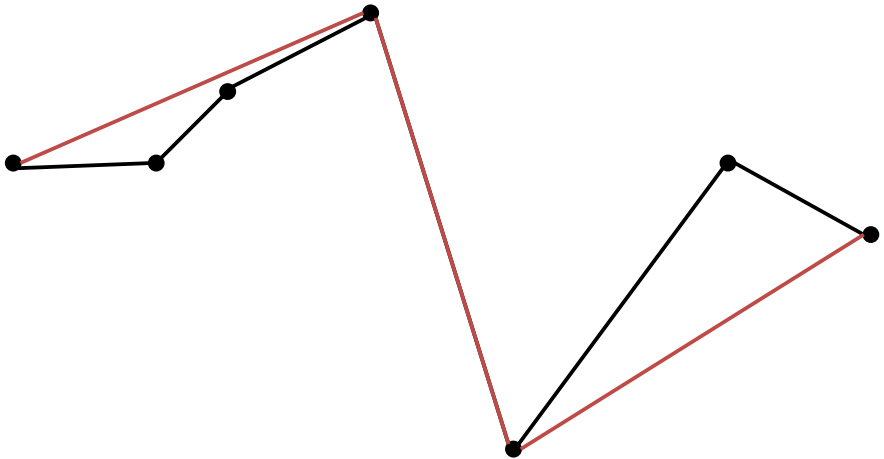
10-3 a)



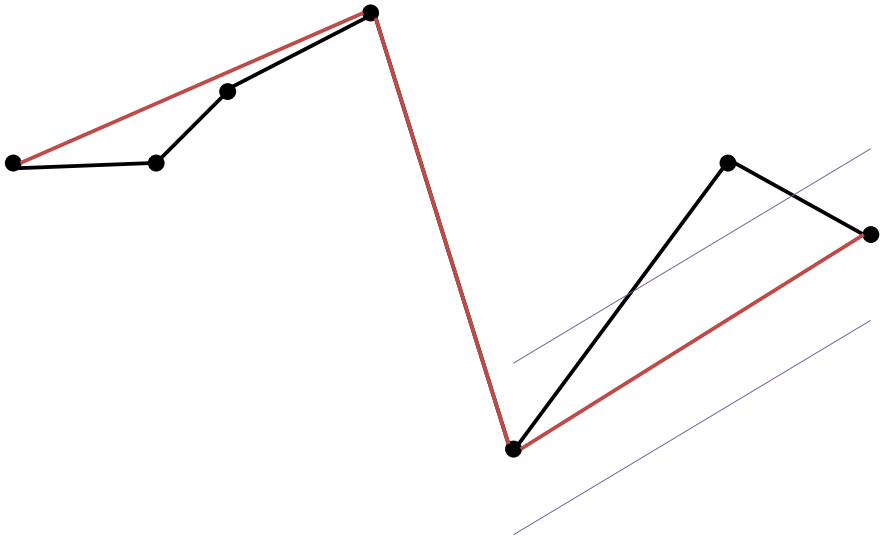
10-3 a)



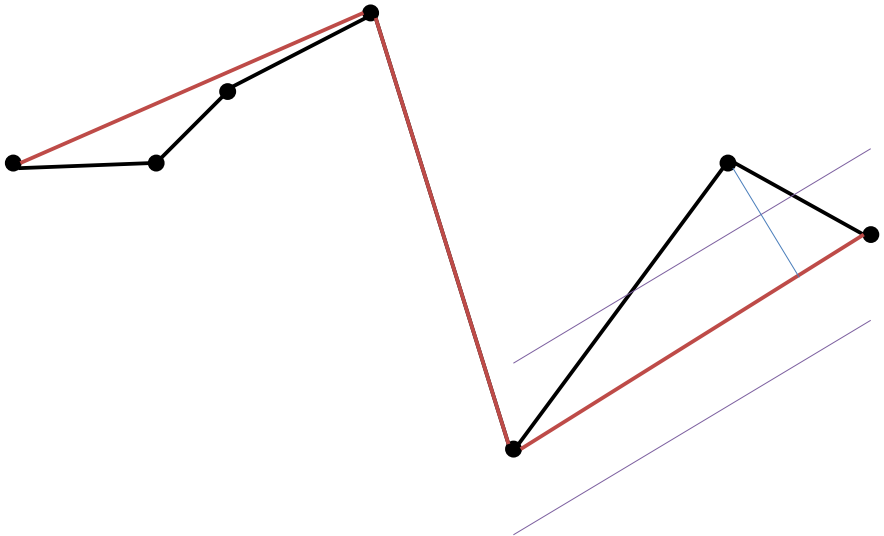
10-3 a)



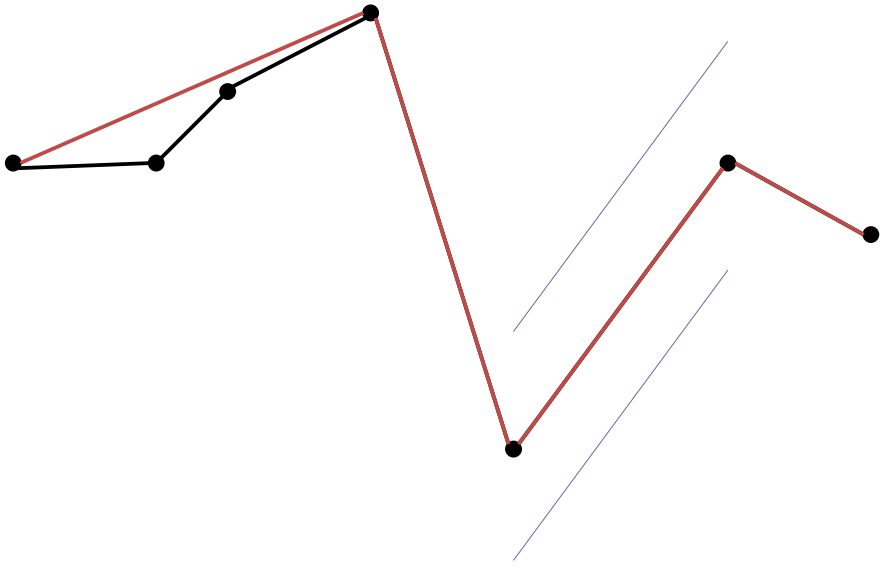
10-3 a)



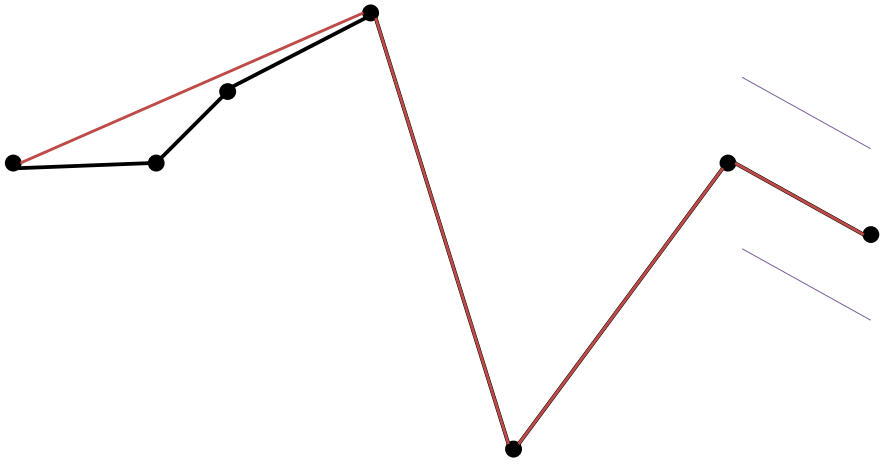
10-3 a)



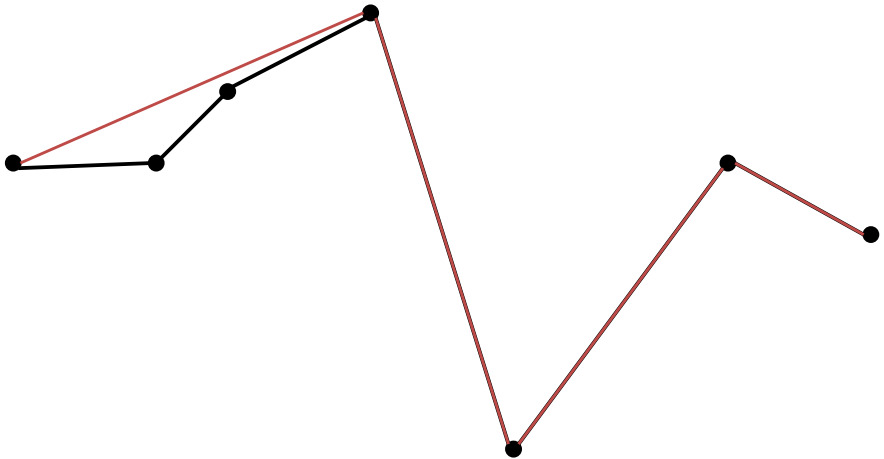
10-3 a)



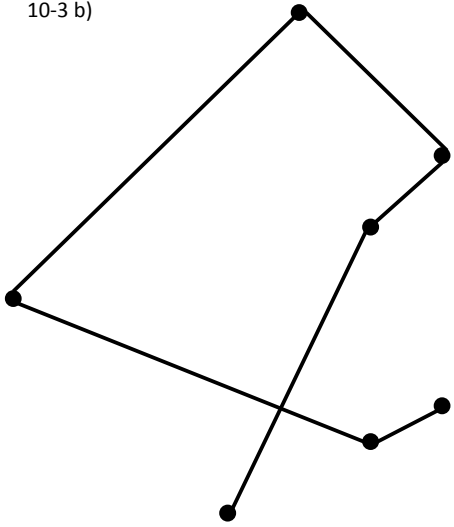
10-3 a)



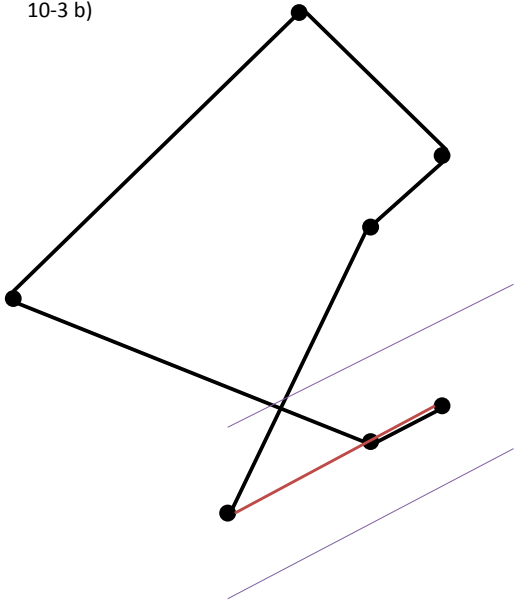
10-3 a) Result



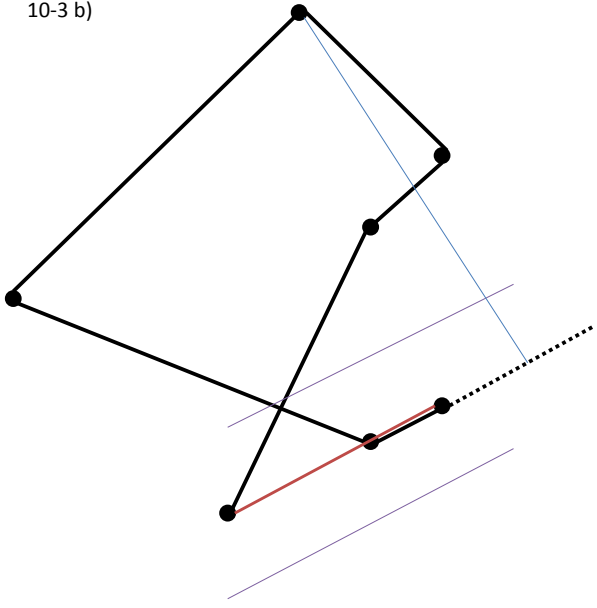
10-3 b)



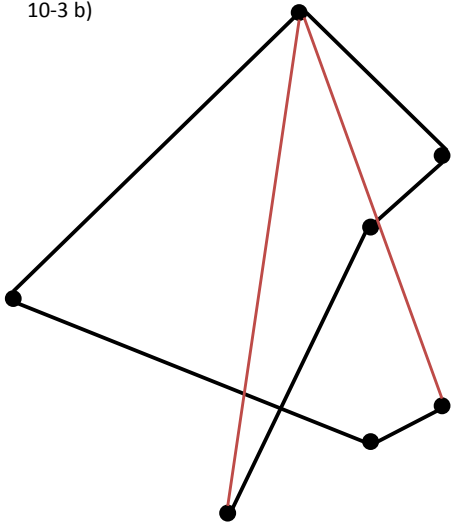
10-3 b)



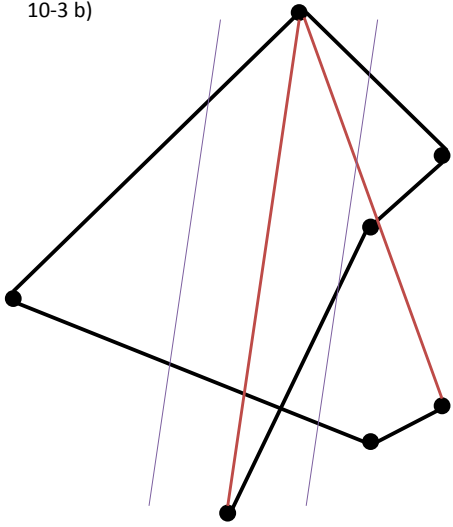
10-3 b)



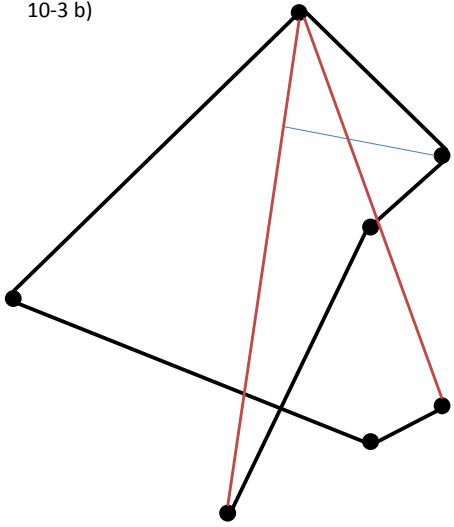
10-3 b)



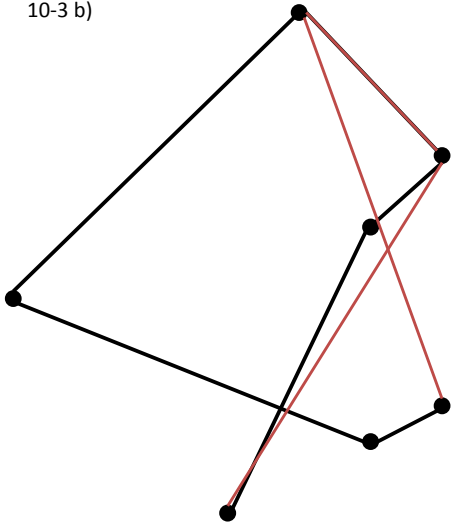
10-3 b)



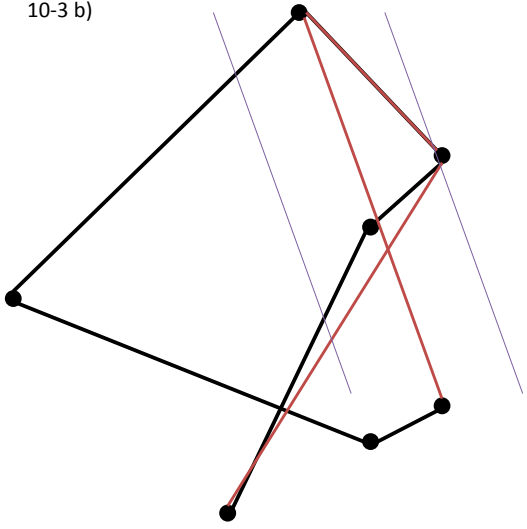
10-3 b)



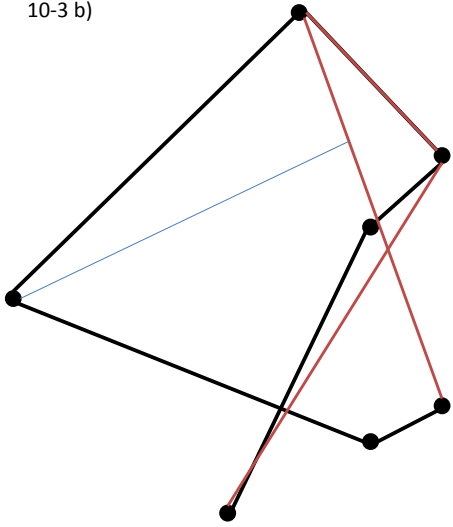
10-3 b)



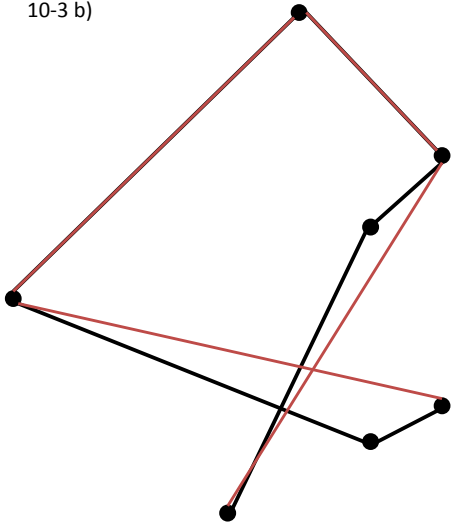
10-3 b)



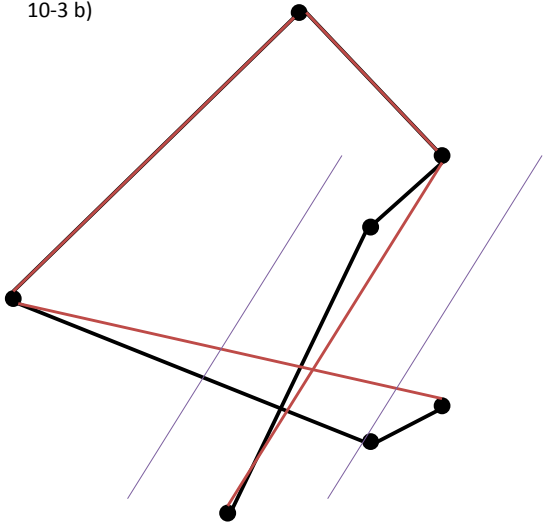
10-3 b)



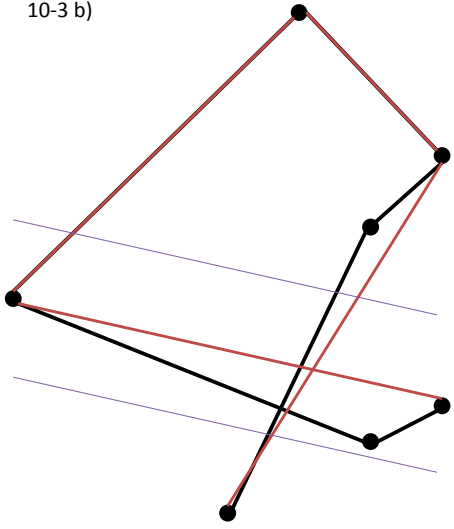
10-3 b)



10-3 b)



10-3 b)



10-3 b) Result