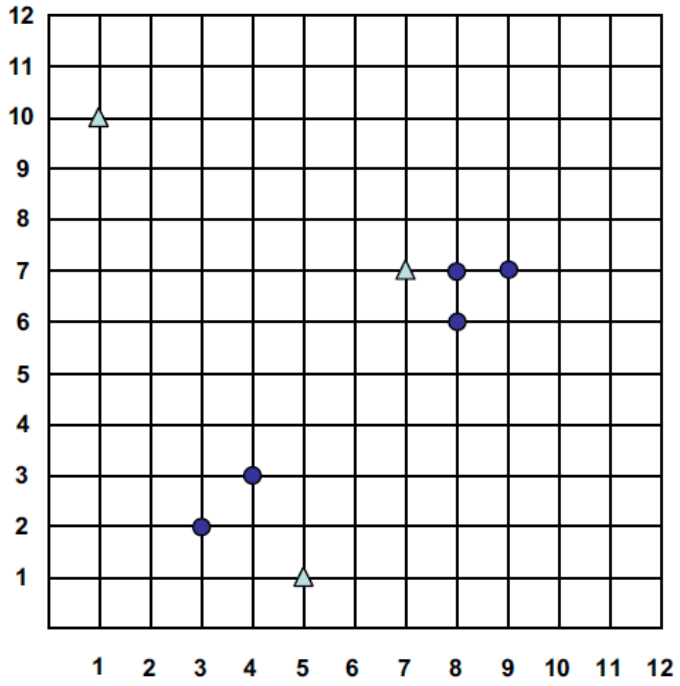
MMMO- Sheet 6

Exercise 6-1 *Supervised Learning: Instance-based learning: classification with kNN*

The following data set with 8 points (e.g. two-dimensionally feature vectors) is given. The triangles build one class and the circles build the other.

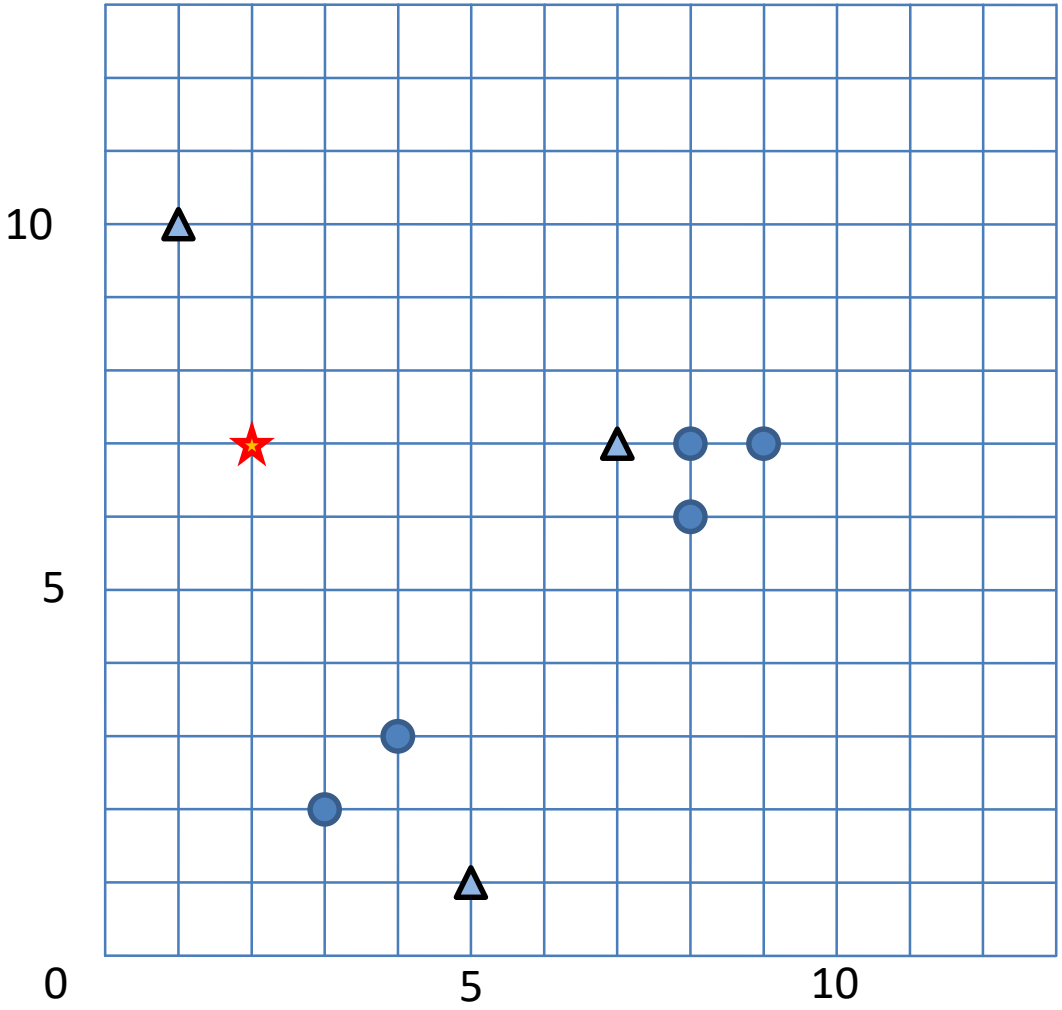


In the following the classes of data points should be determined with the k-nearest neighbors algorithm. As distance function between two points the Manhattan distance (l_1 norm) shall be used:

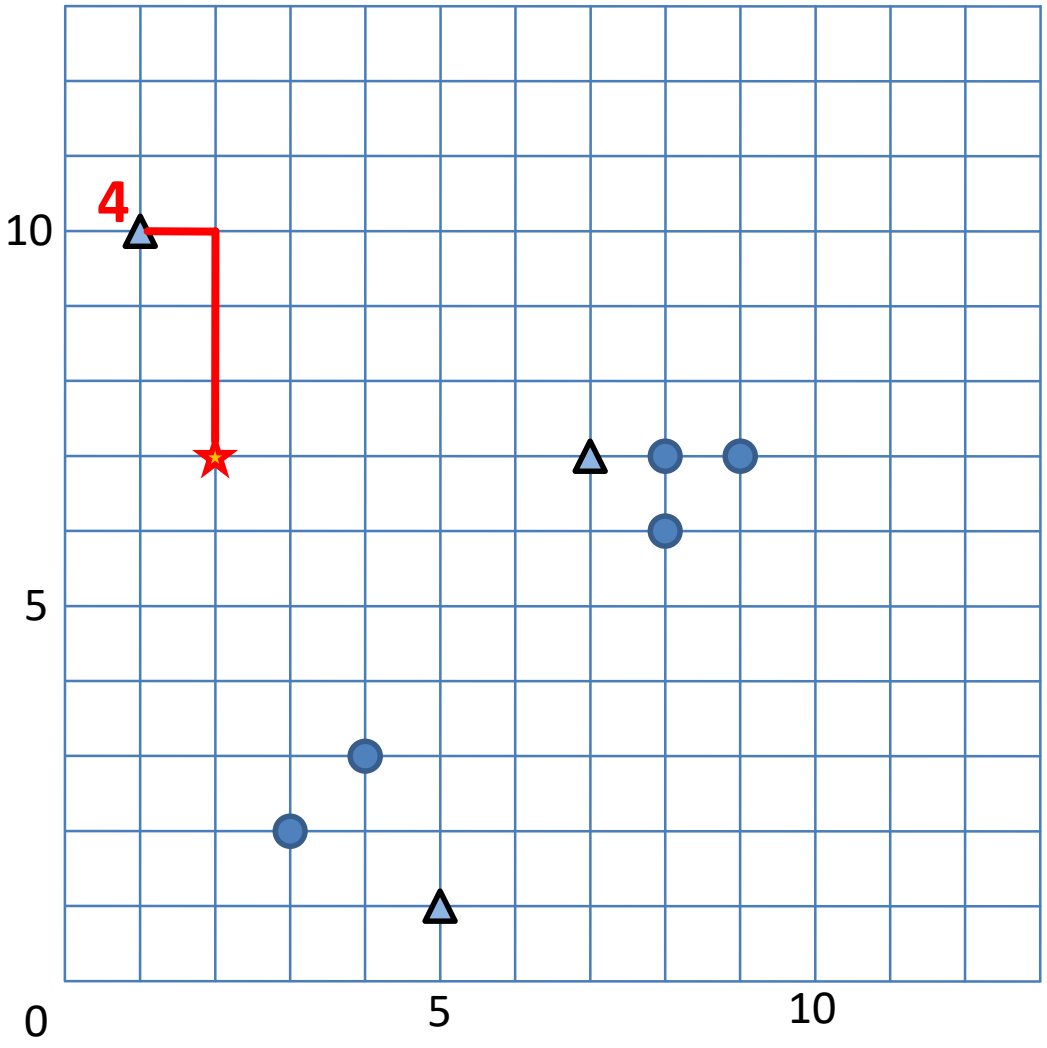
$$L_1(x, y) = \sum_{i=1}^{d} |x_i - y_i|$$

6-1 a)

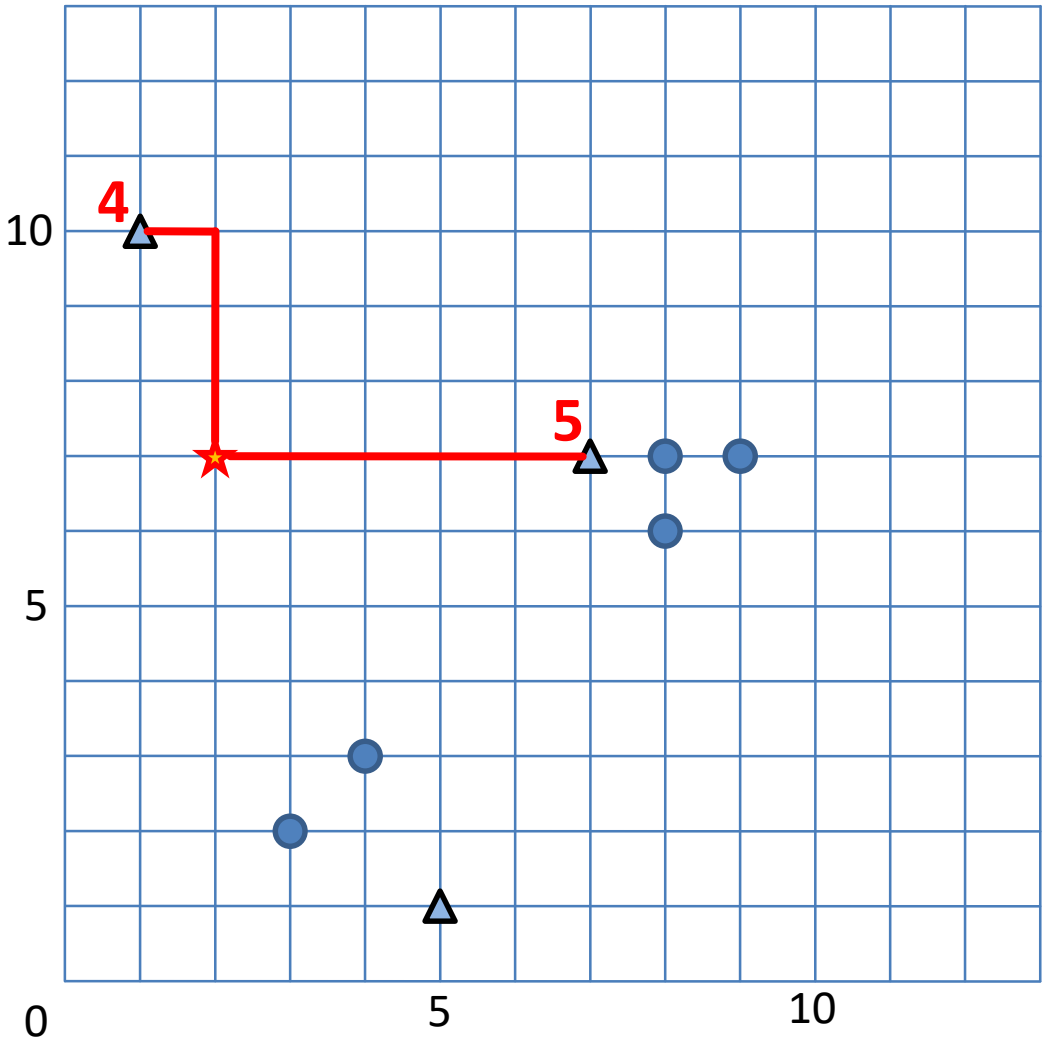
Determine the class of point (2,7) for k = 2 using the class of majority of its k-nearest neighbors, i.e. the point is assigned to the class which occurs most often among its k-nearest neighbors.



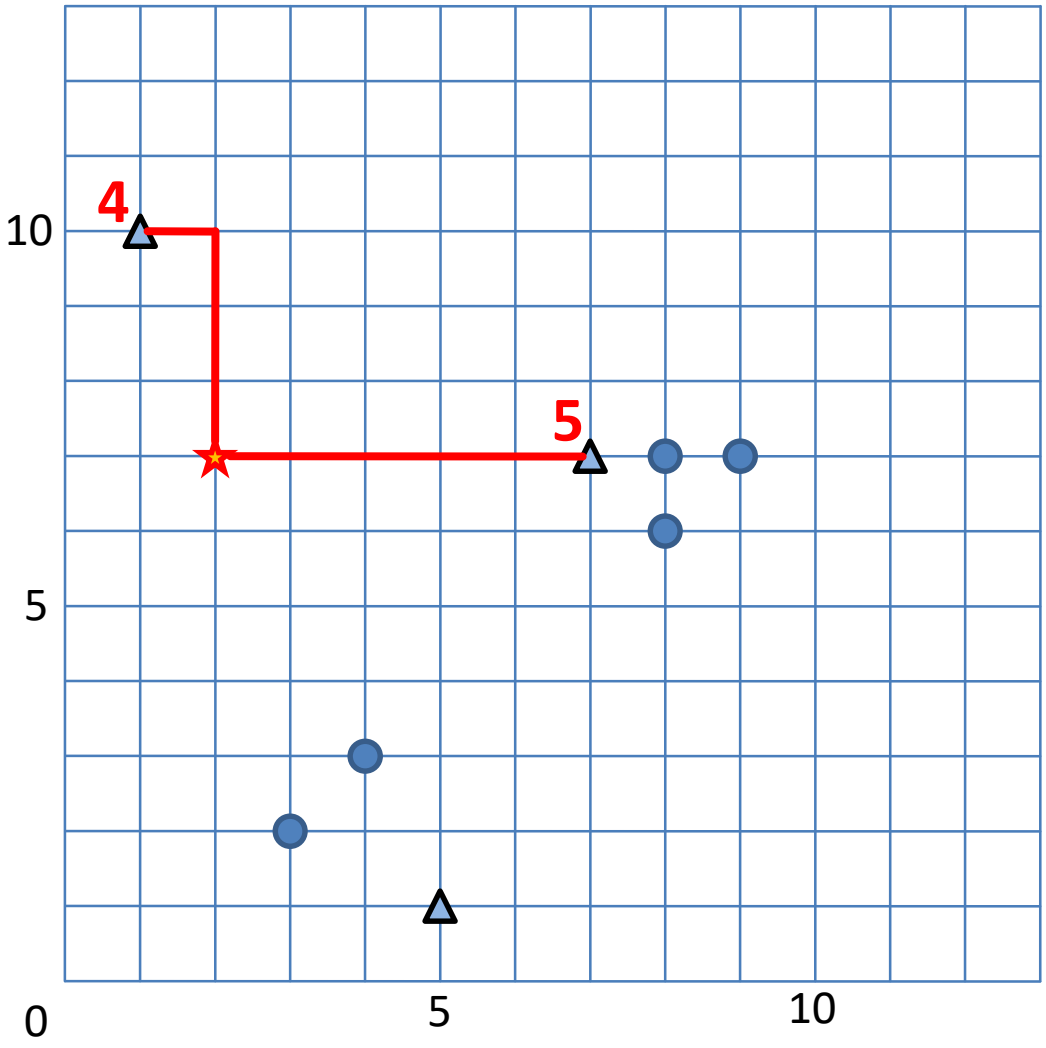
6-1a)



6-1a)

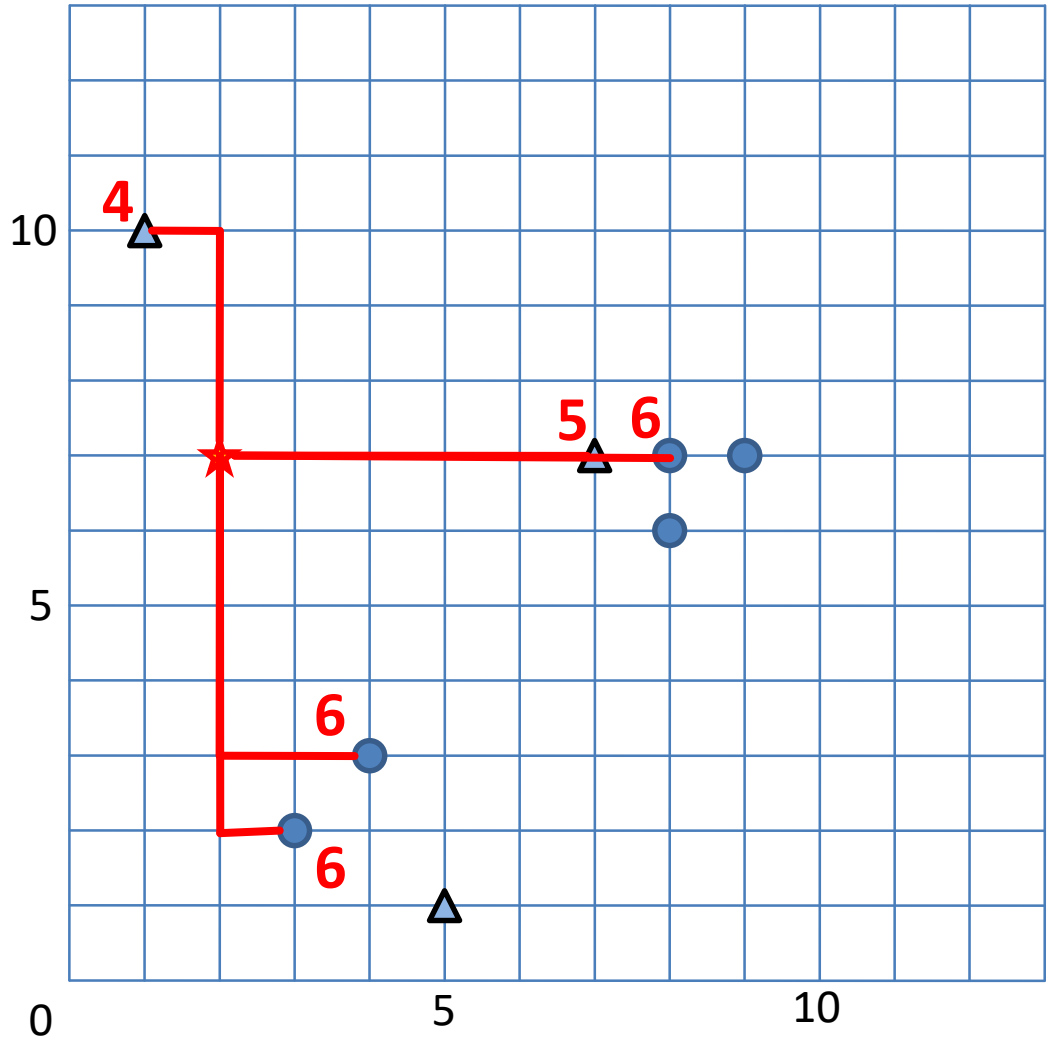


6-1a) Thus, for k=2 ★ is classified as △



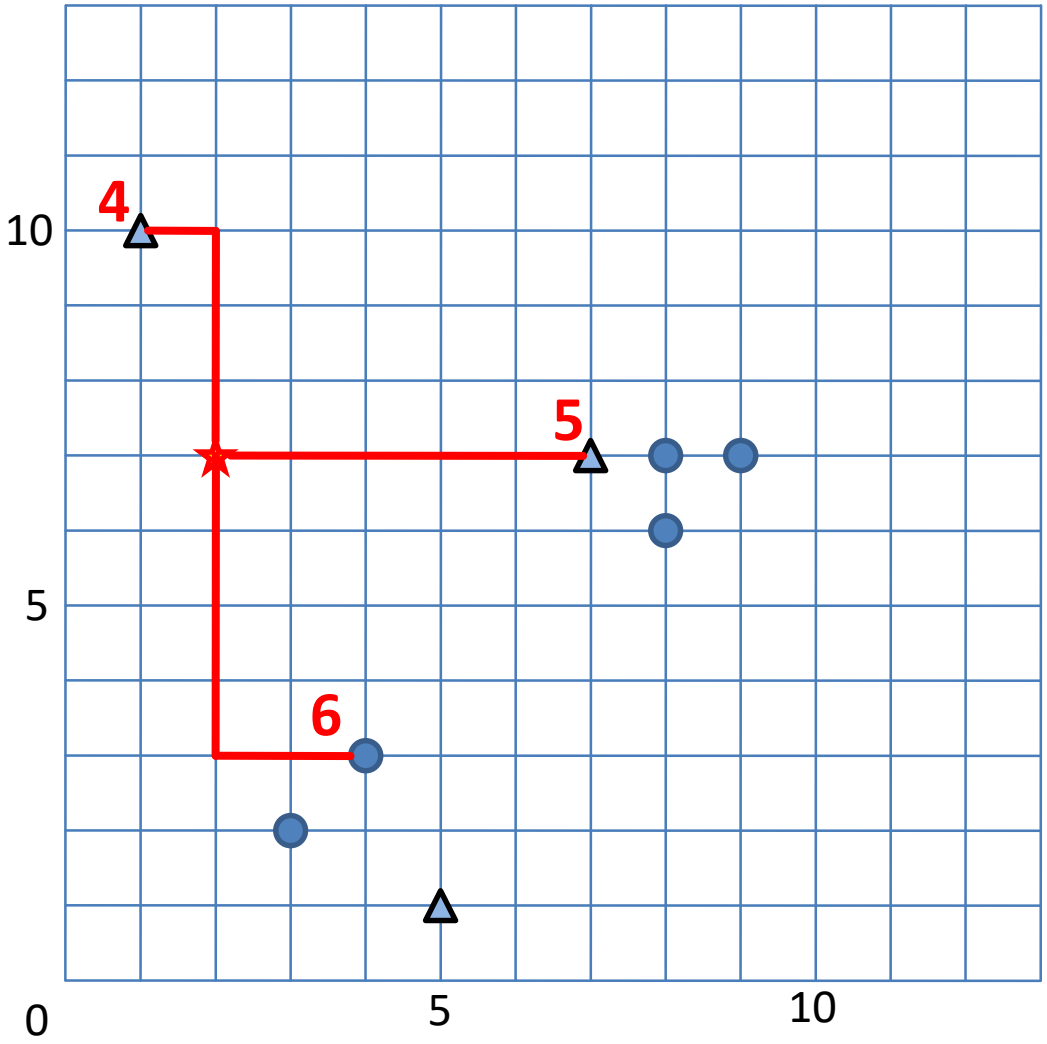
6-1b) Determine the class of point (2,7) for k=3 using the class of majority of its k-nearest neighbors.

=> Special case: several points have the distance 6 to the query point



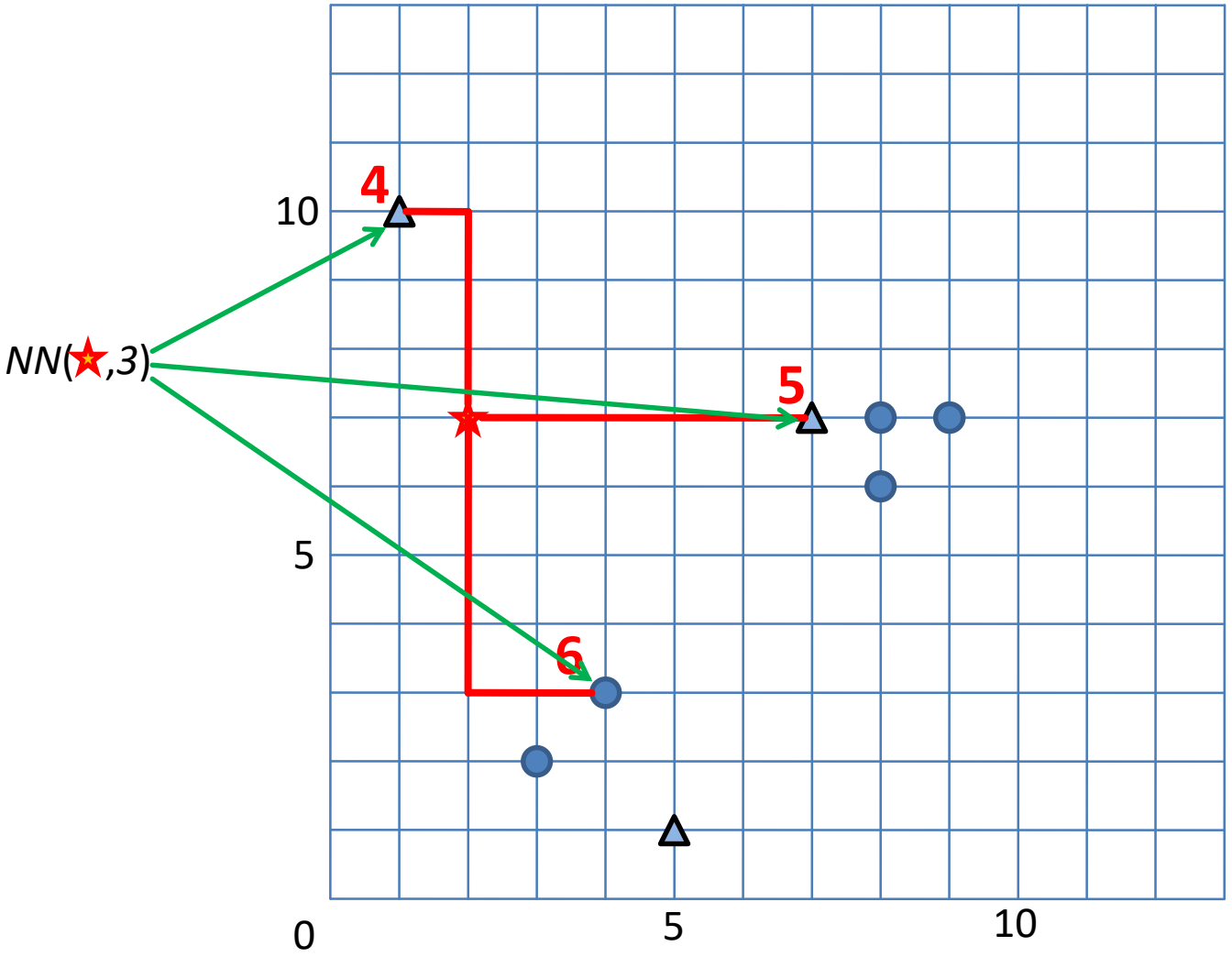
6-1b) Alternative 1: Nondeterministic definition of kNN: Set $NN(q,k) \subseteq$ DB with <u>exactly</u> k objects such that:

$$\forall o \in NN(q,k), \forall o' \in DB - NN(q,k) : dist(q,o) \leq dist(q,o')$$



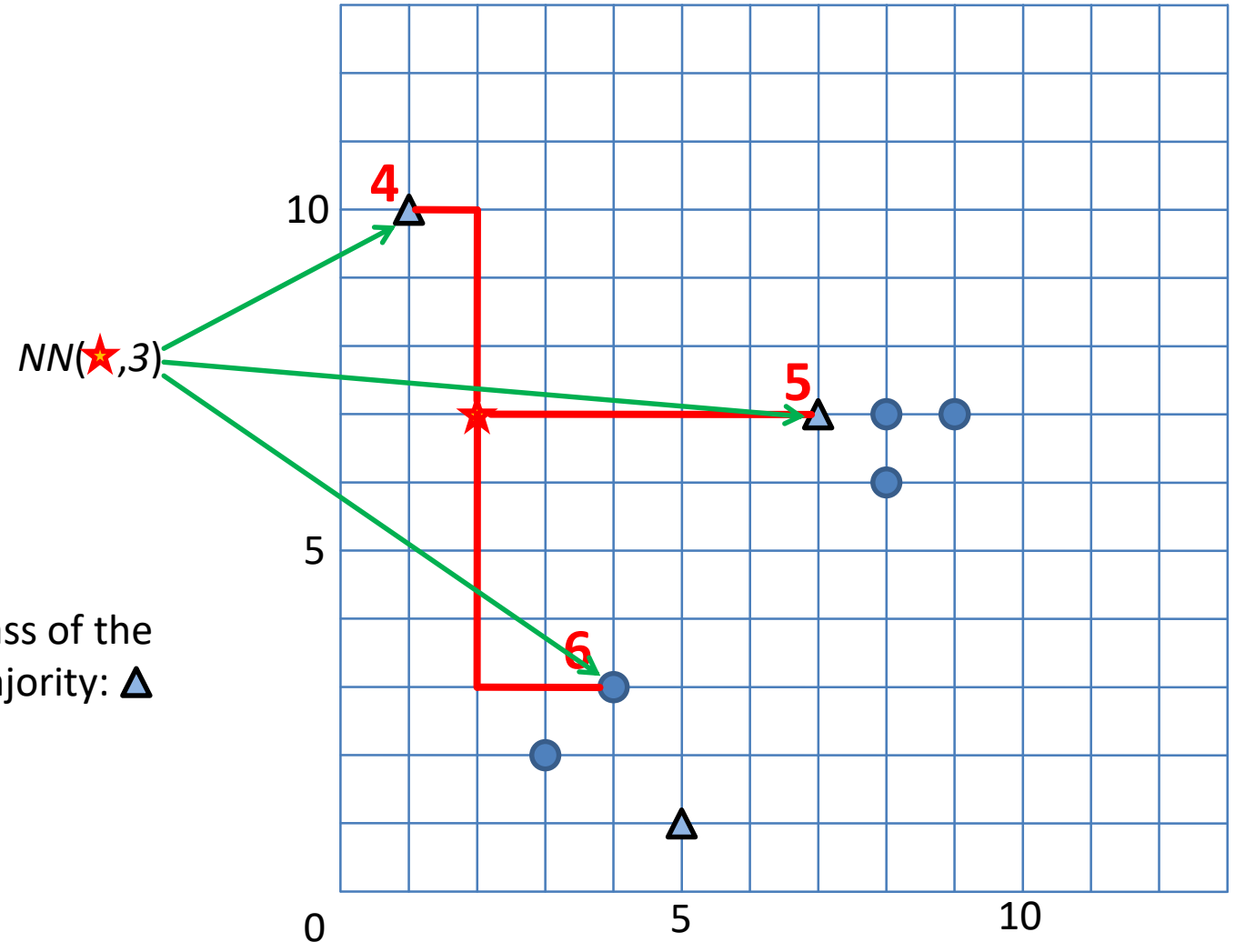
6-1b) Nondeterministic definition of kNN: Set $NN(q,k) \subseteq$ DB with <u>exactly</u> k objects such that:

$$\forall o \in NN(q,k), \forall o' \in DB - NN(q,k) : dist(q,o) \leq dist(q,o')$$



6-1b) Nondeterministic definition of kNN: Set $NN(q,k) \subseteq$ DB with <u>exactly</u> k objects such that:

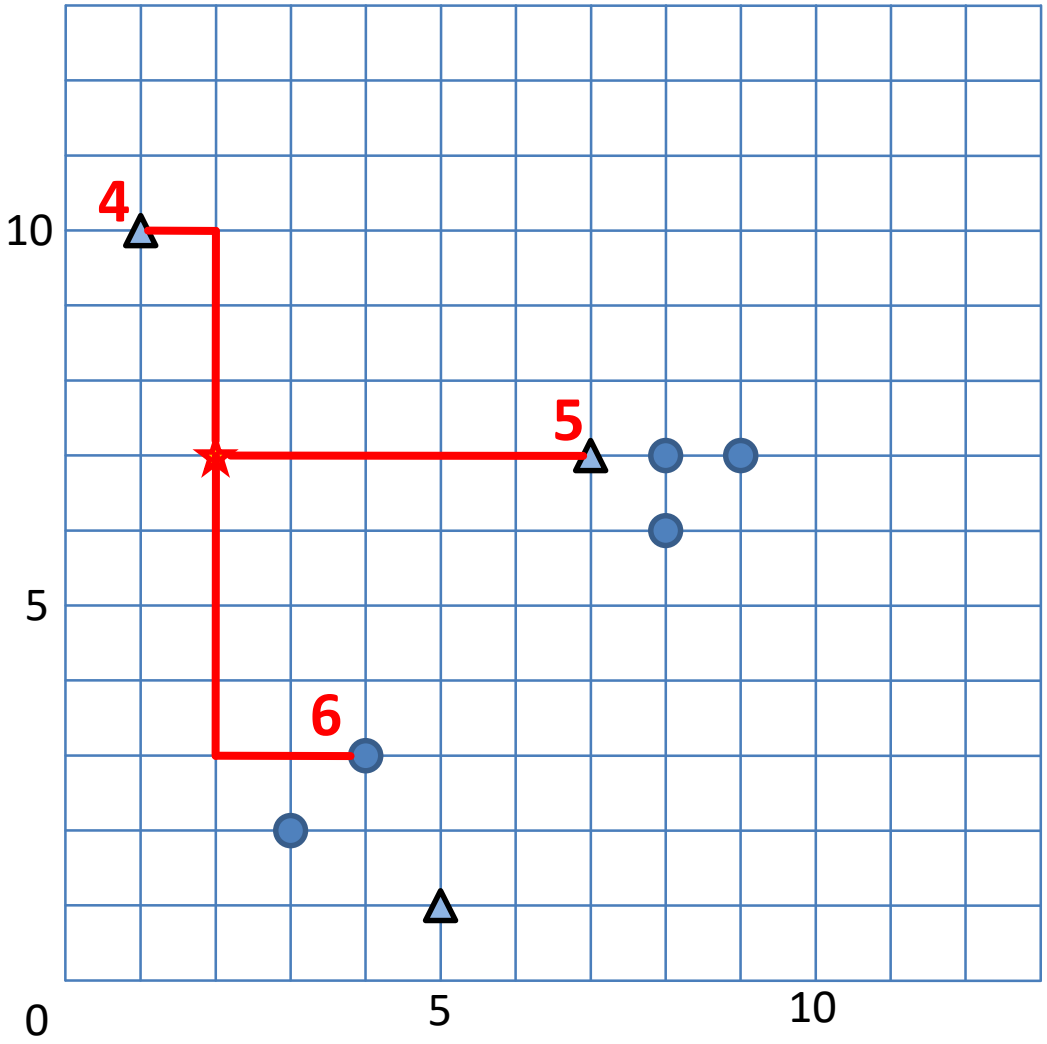
$$\forall o \in NN(q,k), \forall o' \in DB - NN(q,k) : dist(q,o) \leq dist(q,o')$$



Class of the majority: △

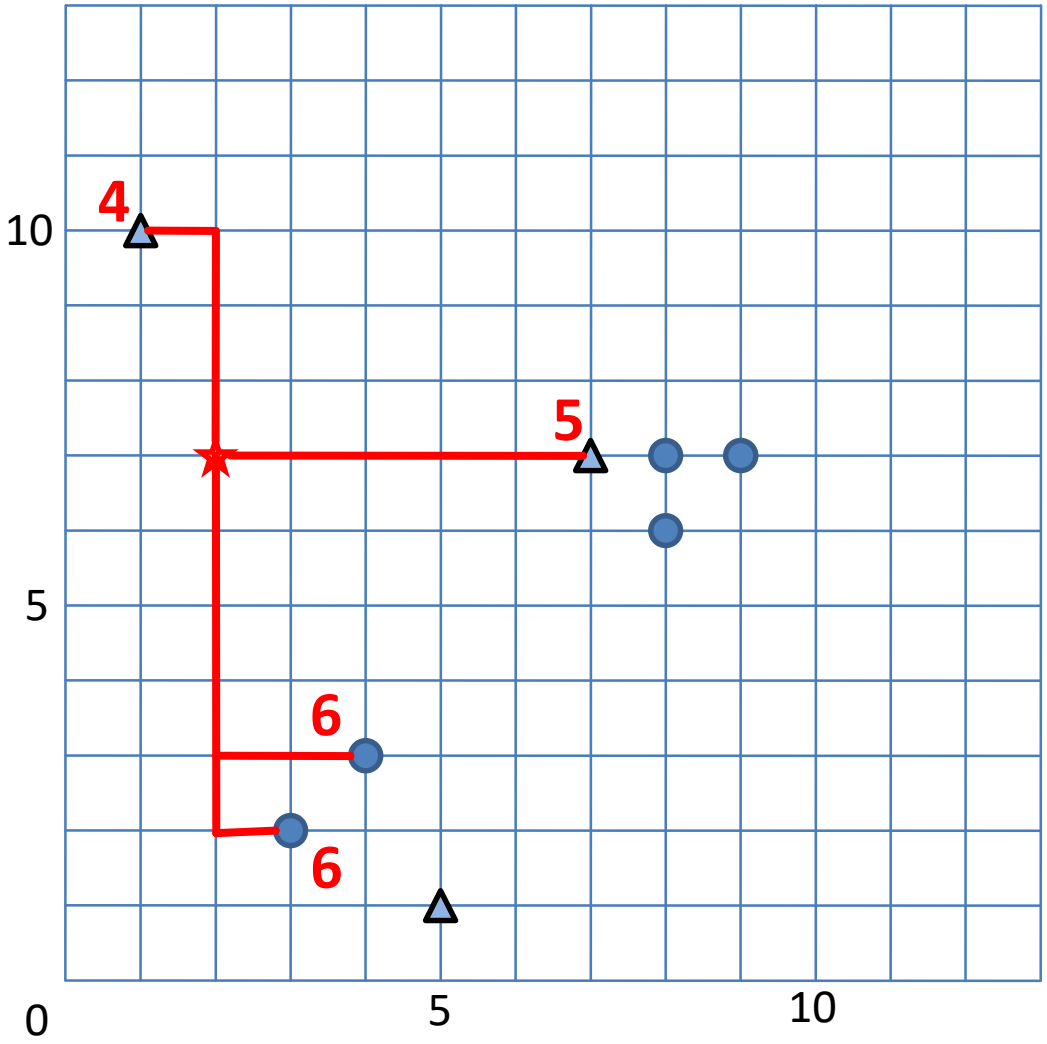
6-1b) Deterministic definition of kNN: Set $NN(q,k) \subseteq DB$ with <u>at least</u> k objects such that:

$$\forall o \in NN(q,k), \forall o' \in DB - NN(q,k) : dist(q,o) < dist(q,o')$$



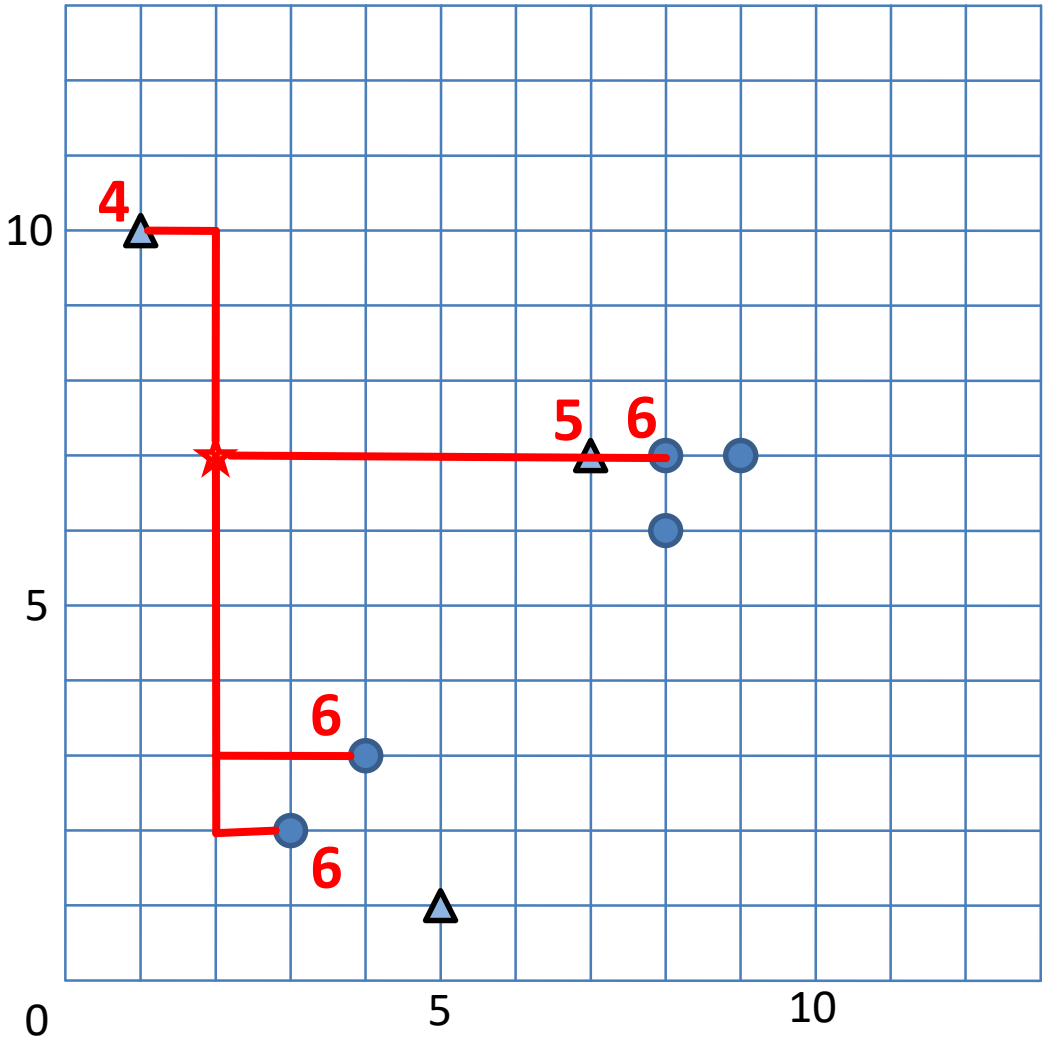
6-1b) Deterministic definition of kNN: Set $NN(q,k) \subseteq$ DB with <u>at least</u> k objects such that:

$$\forall o \in NN(q,k), \forall o' \in DB - NN(q,k) : dist(q,o) < dist(q,o')$$



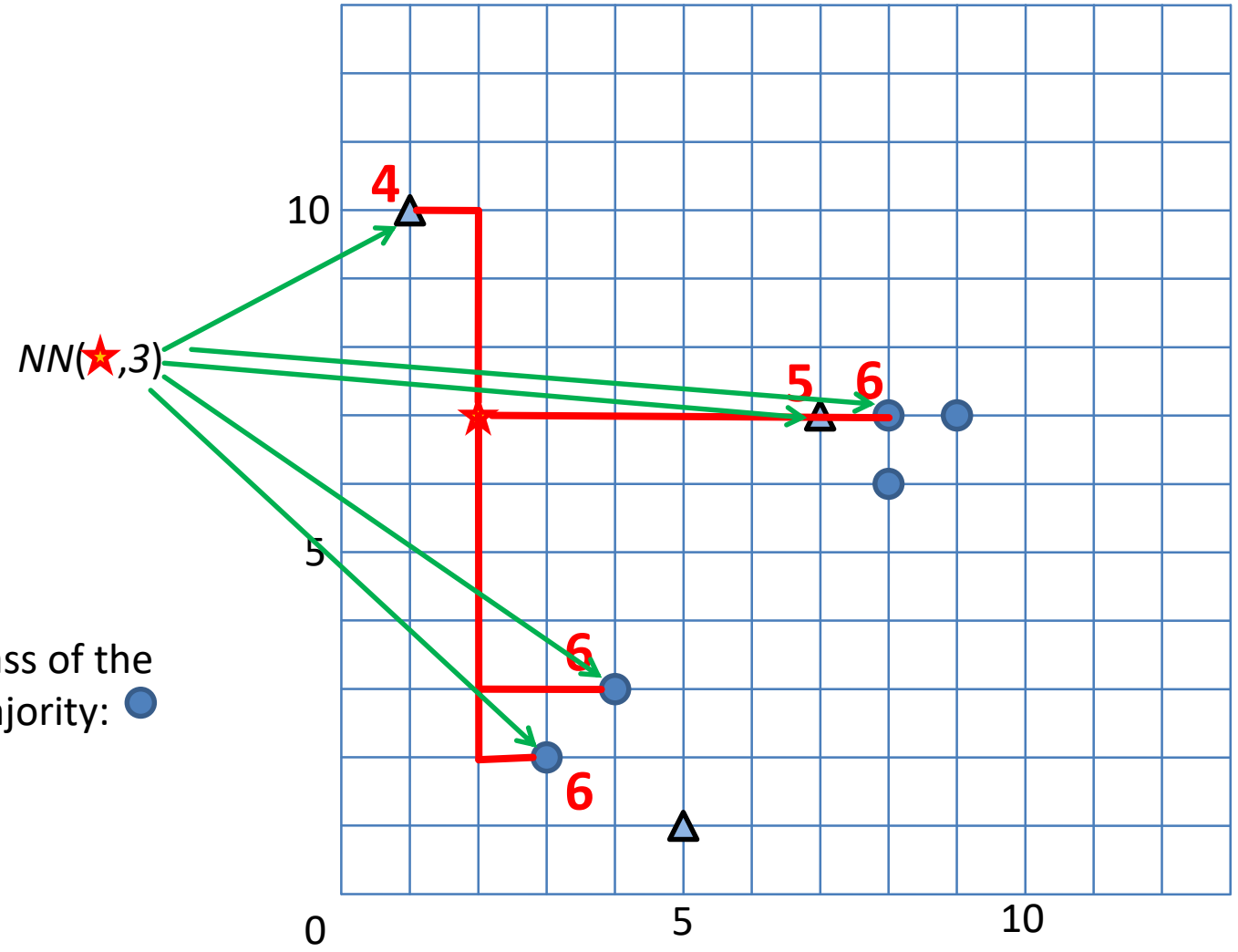
6-1b) Deterministic definition of kNN: Set $NN(q,k) \subseteq DB$ with <u>at least</u> k objects such that:

$$\forall o \in NN(q,k), \forall o' \in DB - NN(q,k) : dist(q,o) < dist(q,o')$$



6-1b) Deterministic definition of kNN: Set $NN(q,k) \subseteq DB$ with <u>at least</u> k objects such that:

$$\forall o \in NN(q,k), \forall o' \in DB - NN(q,k) : dist(q,o) < dist(q,o')$$

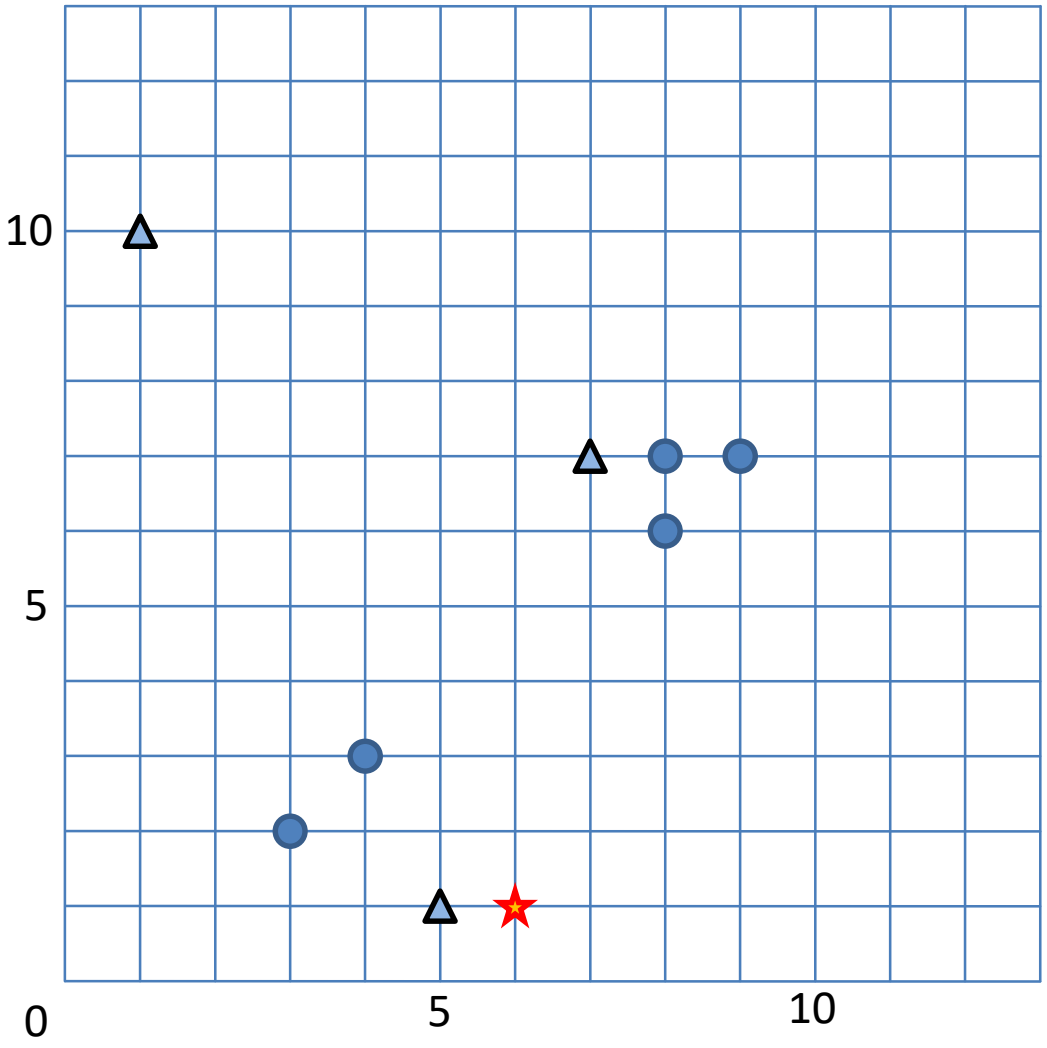


Class of the majority:

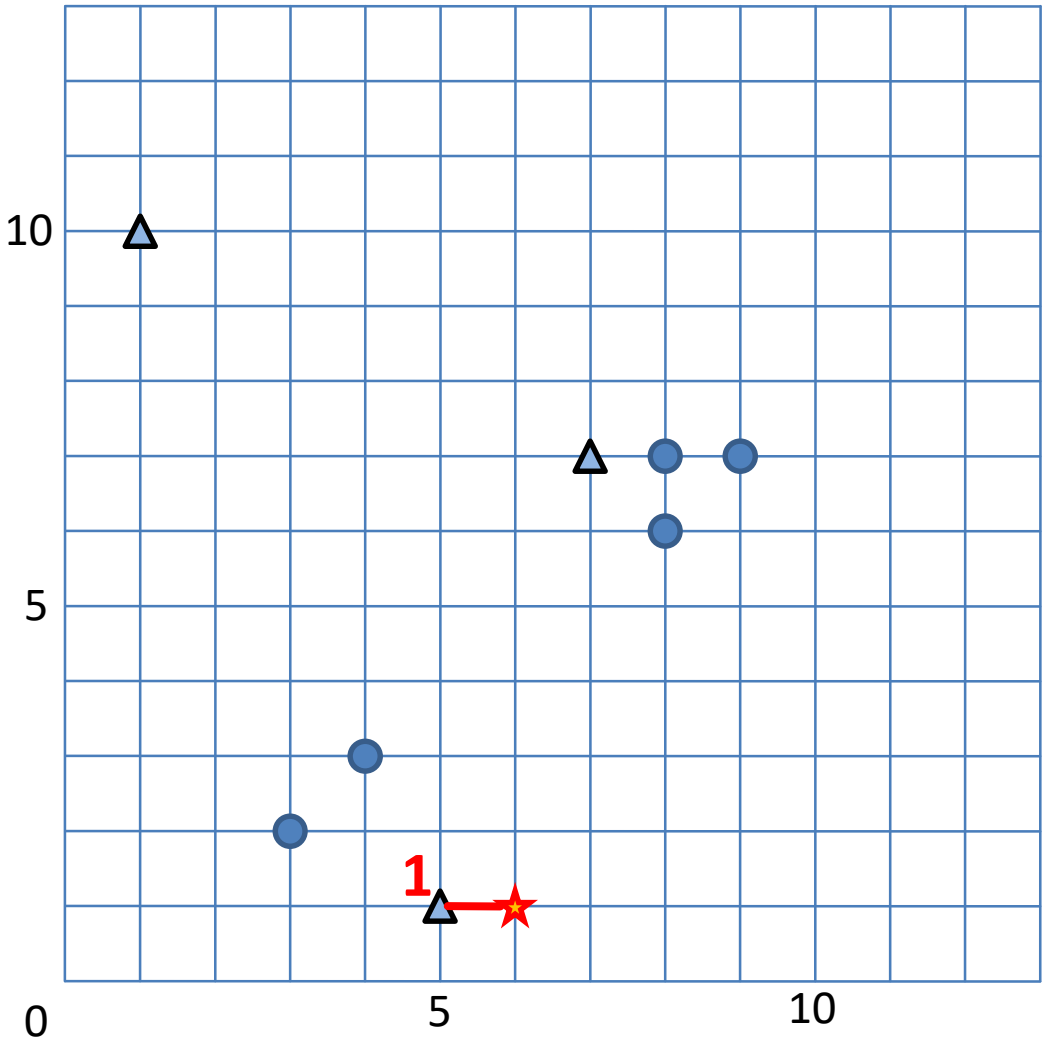
6-1c) Determine the class of point (2,7) for k=5 using the class of majority of its k-nearest neighbors.

Analogously to the deterministic alternative of 6-1 b)

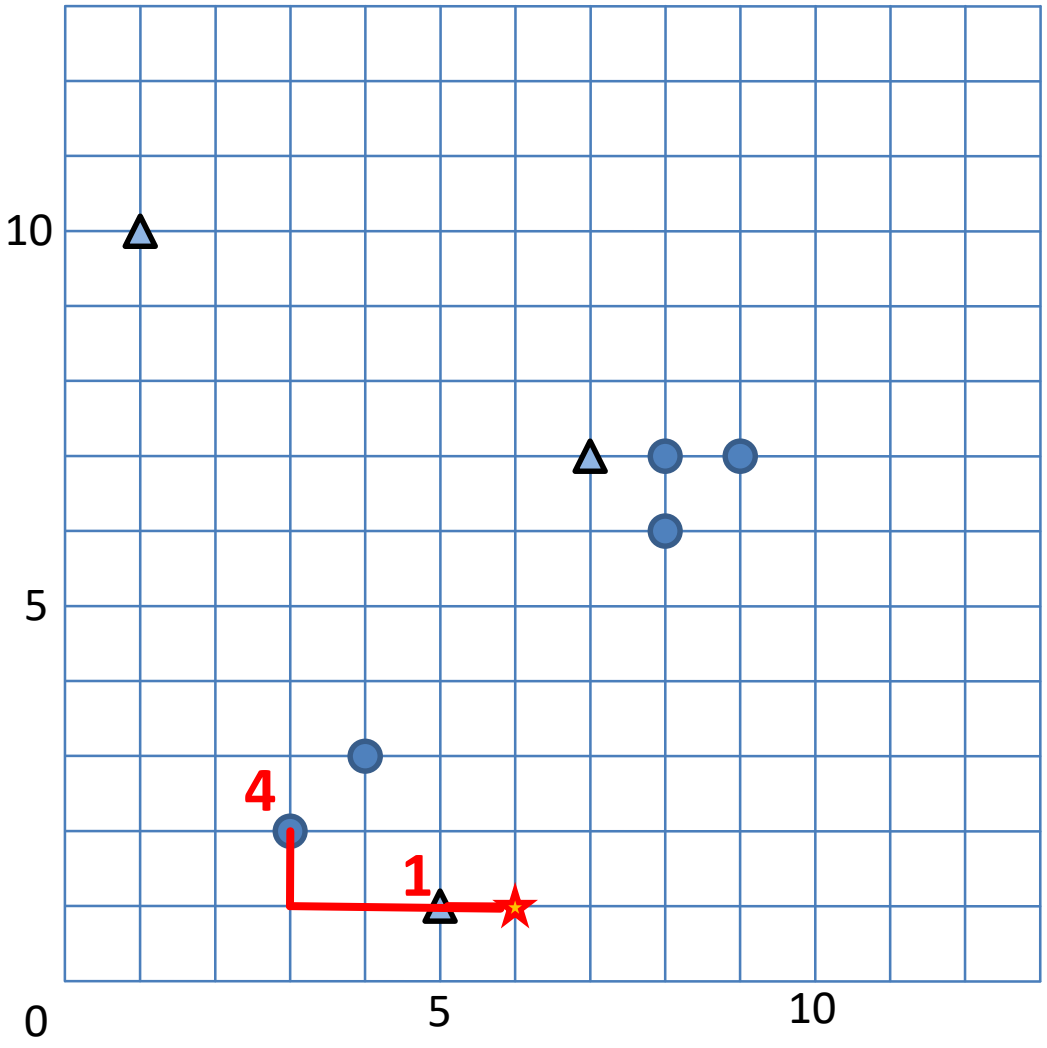
6-1d) Determine the class of point (6,1) for k=3 using the class of majority of its k-nearest neighbors.



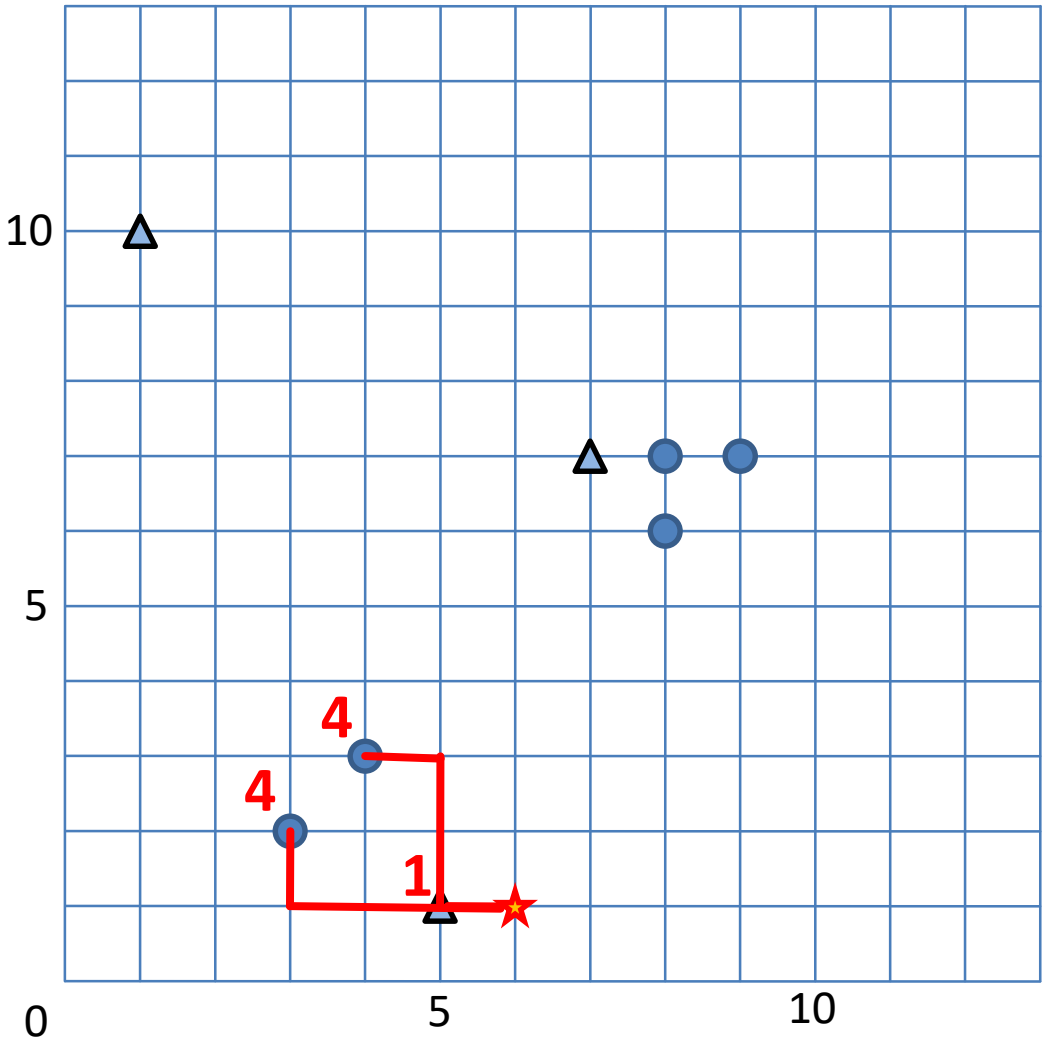
6-1d)



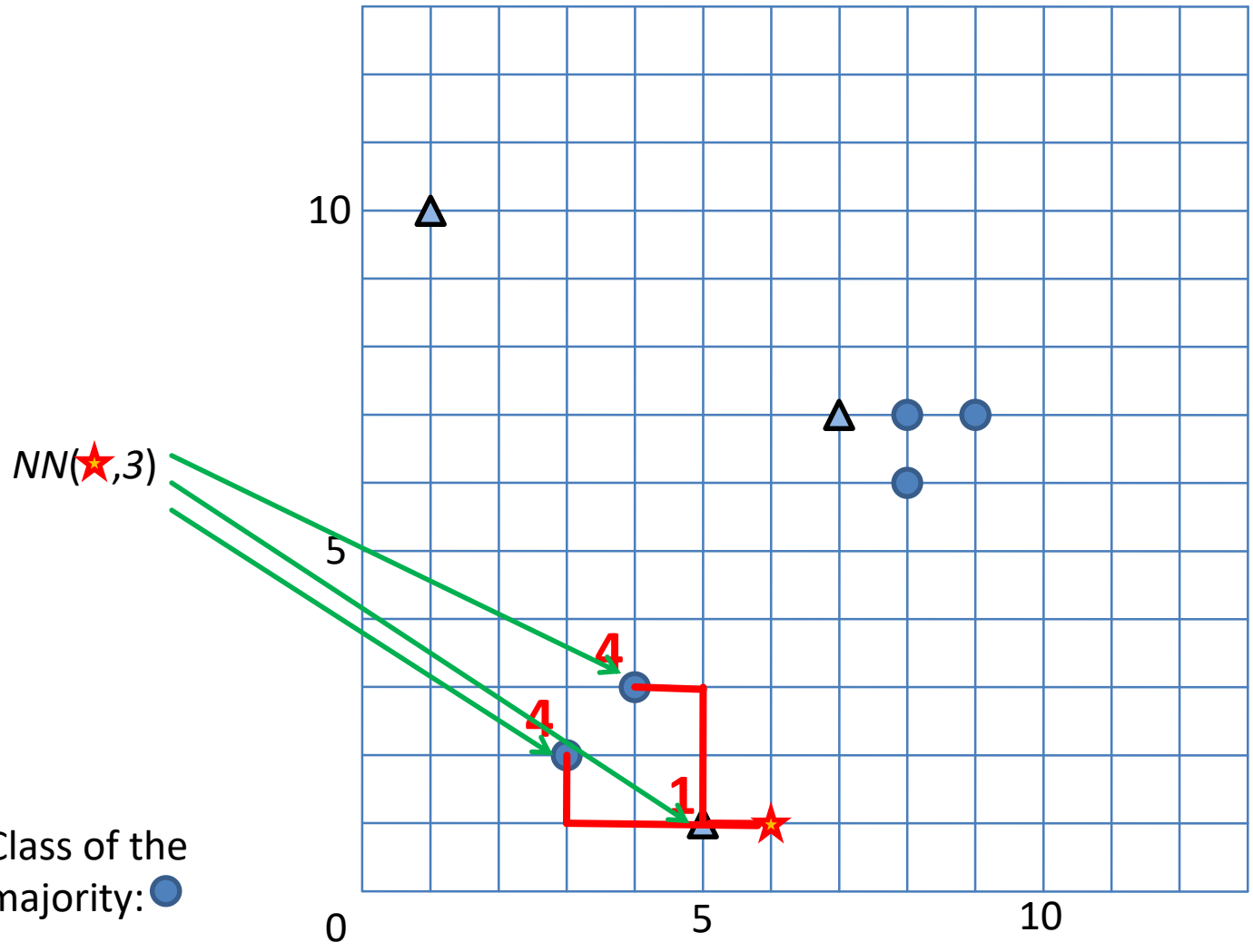
6-1d)



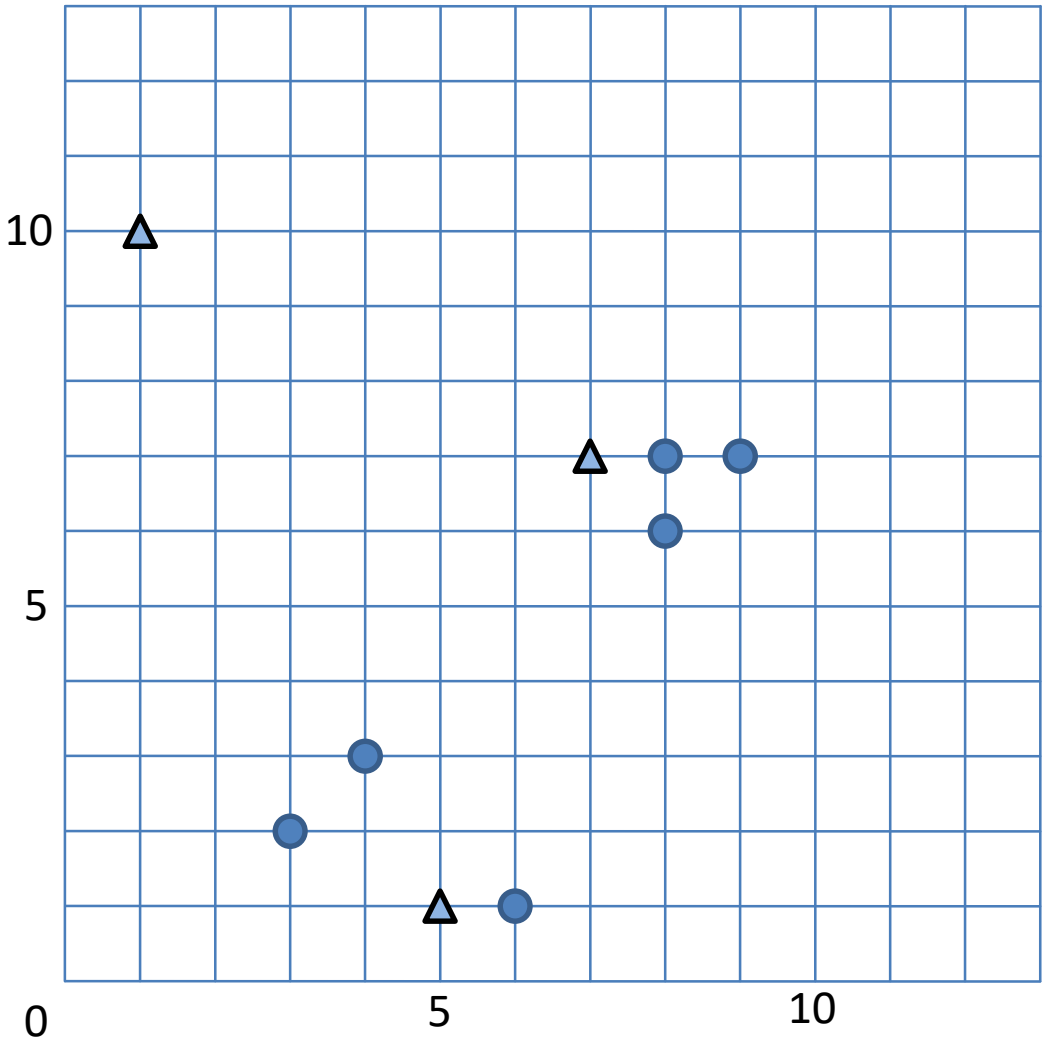
6-1d)



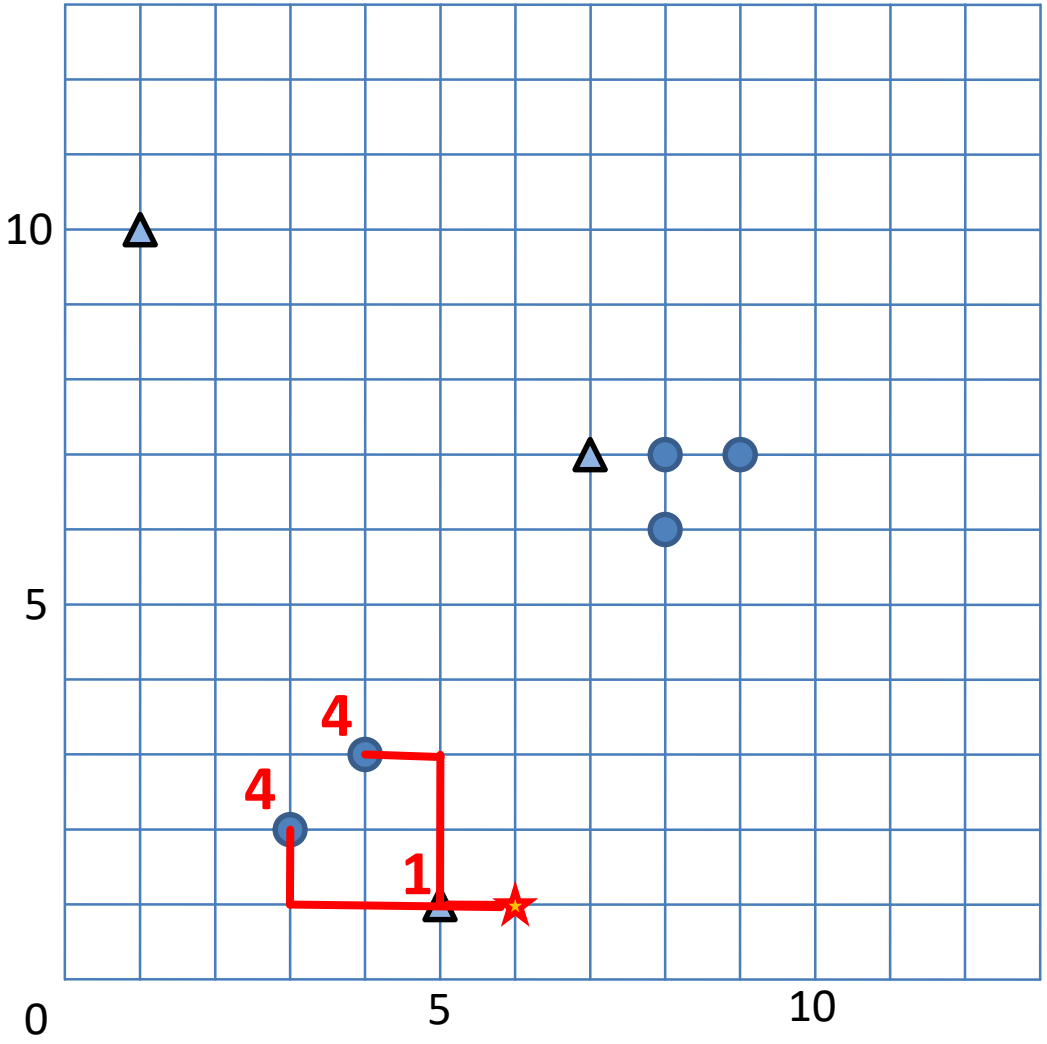
6-1d)



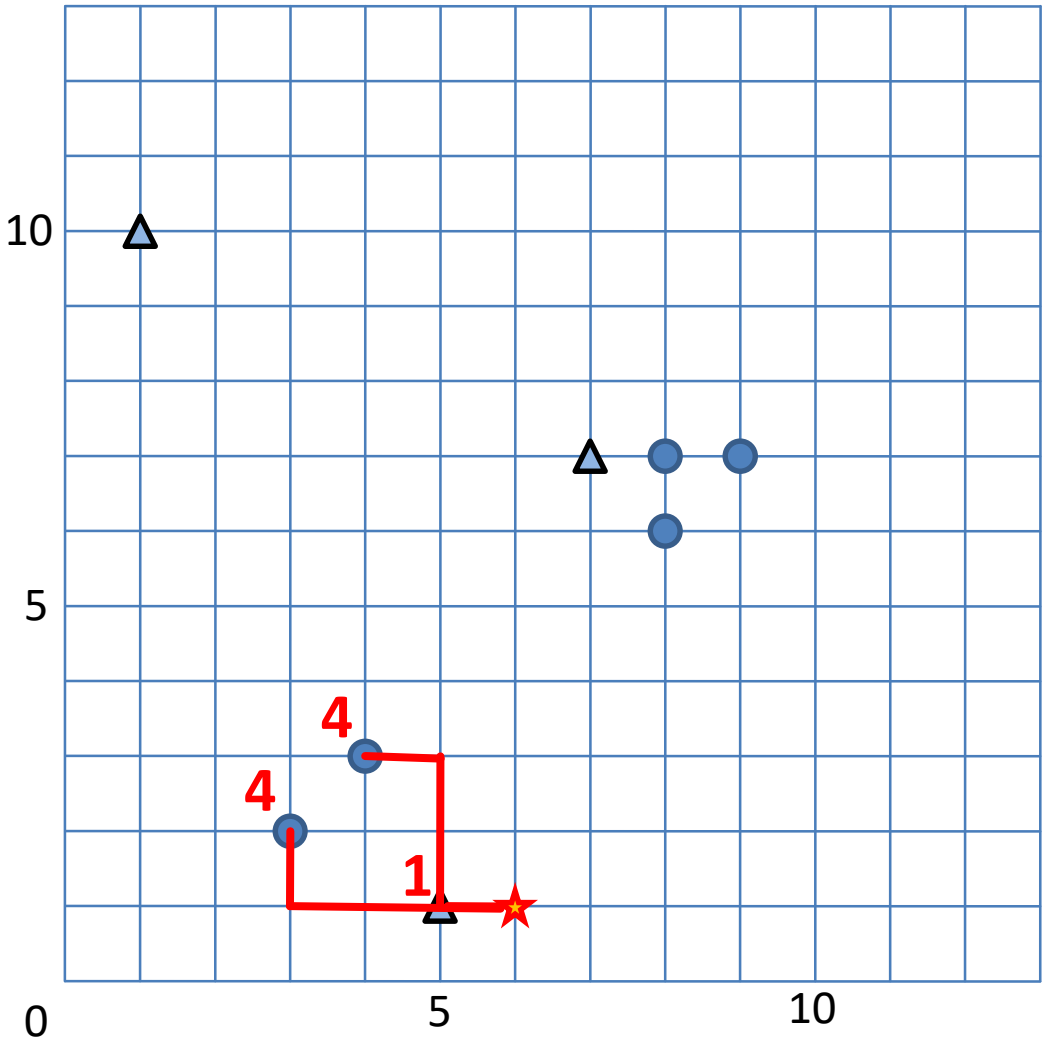
6-1d)



6-1e) Determine the class of point (6,1) for k=3 using the class of majority of its k-nearest neighbors weighting the classes with inverse Manhattan distance.



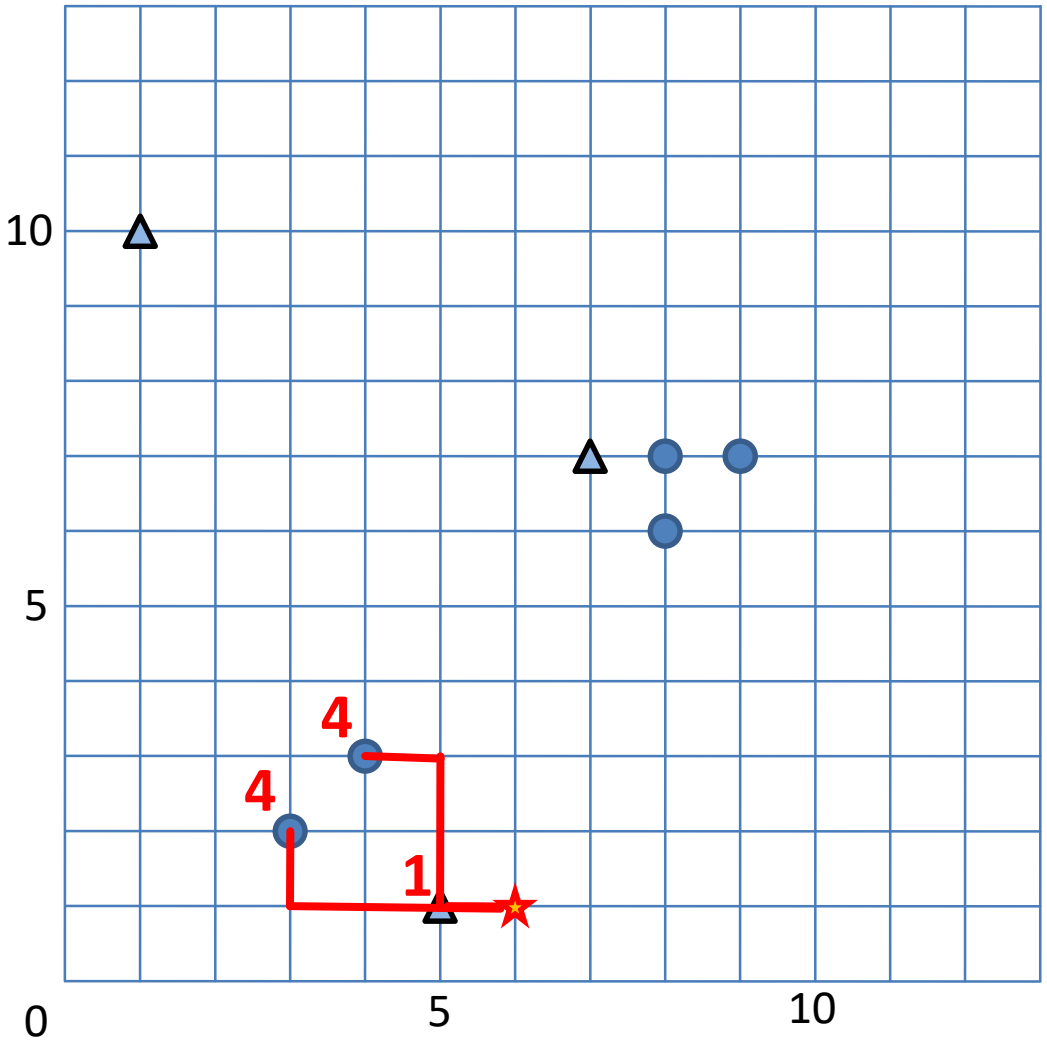
6-1e)



Weighting(●) = ¼ + ¼ = ½

Weighting(△) = 1/1 = 1

6-1e)

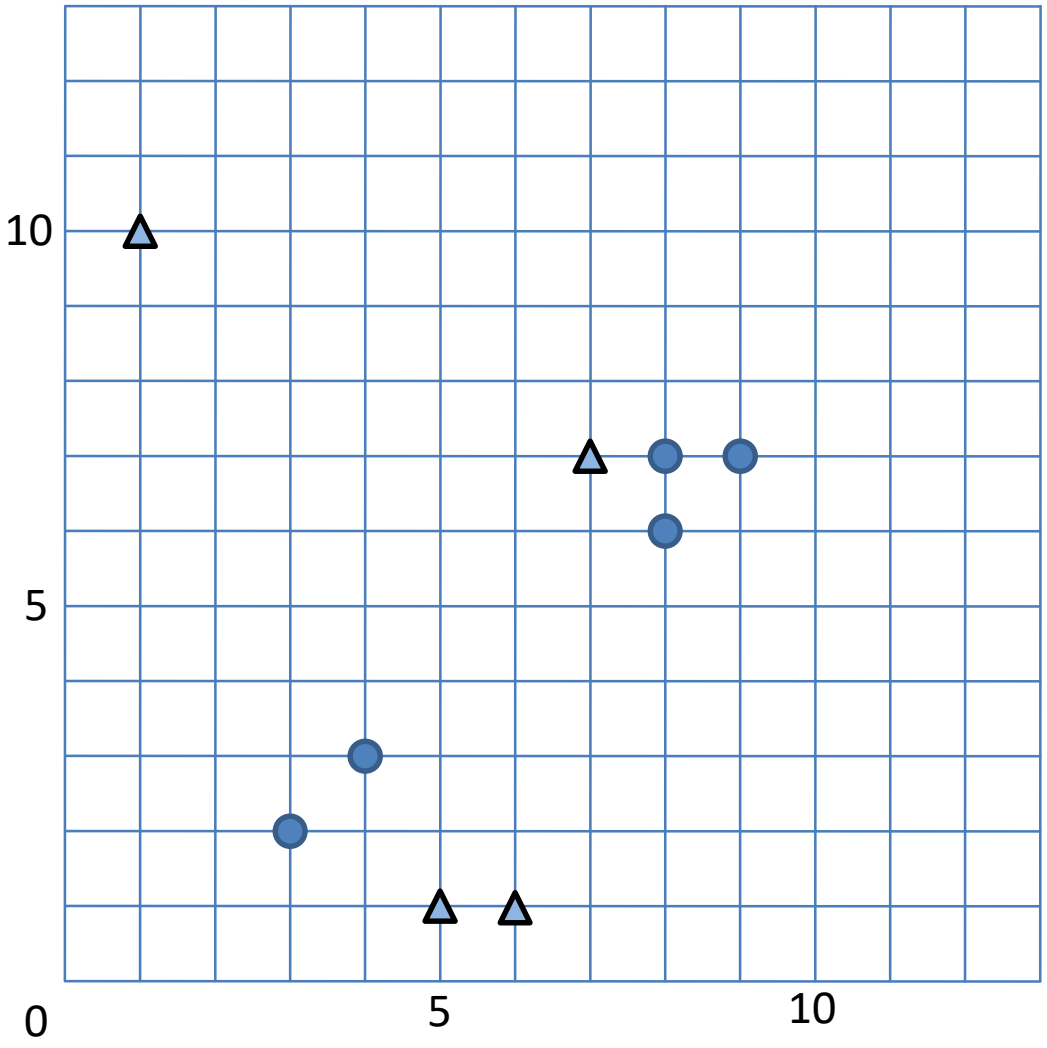


Weighting(●) = ¼ + ¼ = ½

Weighting(△) = 1/1 = 1

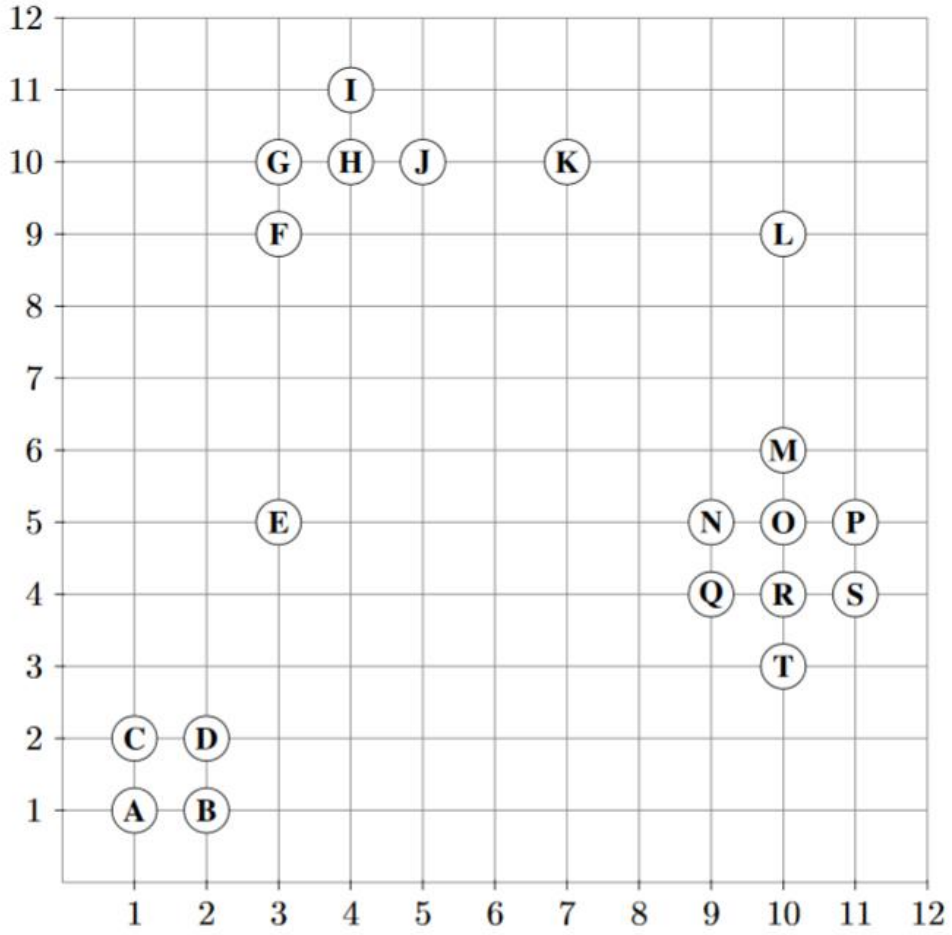
Highest weight: △

6-1e)



Exercise 6-2 *Unsupervised Learning: Clustering with DBSCAN*

The following dataset is given:



Cluster this dataset using DBSCAN. Use the Manhattan distance as distance function and the parameters $\epsilon = 1.1$ and $minPts = 3$.

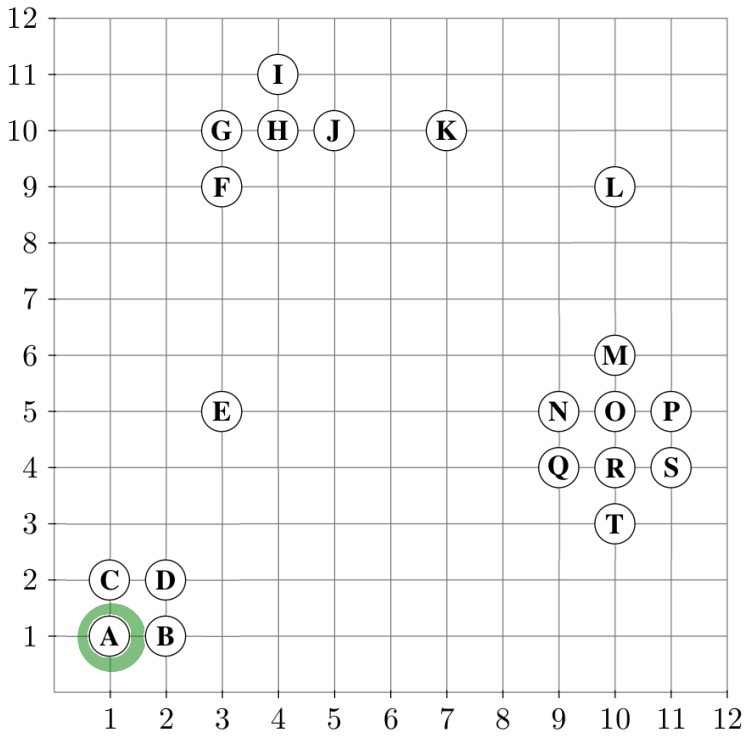
Algorithm DBSCAN

```
DBSCAN(SetOfPoints DB, Real ε, Integer MinPts)
  // At the beginning all objects are unclassified
  // o.ClId = UNCLASSIFIED for all o ∈ DB

   ClusterId := nextId(NOISE);
   for i from 1 to |DB| do
      Object := DB.get(i);
      if Object.ClId = UNCLASSIFIED then
         if ExpandCluster(DB, Object, ClusterId, ε,
          MinPts)
         then ClusterId:=nextId(ClusterId);
```

Algorithm DBSCAN

```
ExpandCluster(DB, StartObject, ClusterId, ε, MinPts): Boolean
seeds:= RQ(StartObject, ε);
if |seeds| < MinPts then // StartObject is no kernal object
  StartObjekt.ClId := NOISE;
  return false;
// else: StartObject is a kernal object
forall o ∈ seeds do o.ClId := ClusterId;
remove StartObject from seeds;
while seeds ≠ Empty do
  choose an object o from the set seeds;
  Neighbors := RQ(o, ε);
  if |Neighbors| ≥ MinPts then // o is a kernal object
    for i from 1 to |Neighbors| do
        p := Neighbors.get(i);
        if p.ClId in {UNCLASSIFIED, NOISE} then
              if p.ClId = UNCLASSIFIED then
                    add p to seeds;
              p.ClId := ClusterId;
  remove o from seeds;
return true;
```



Start: (A)

A.ClId = Unclassified

ExpandCluster (DB, A, 1, 1.1, 3)

Unclassified

(A) (B) (C) (D) (E) (F) (G) (H) (I) (J)
(K) (L) (M) (N) (O) (P) (Q) (R) (S) (T)

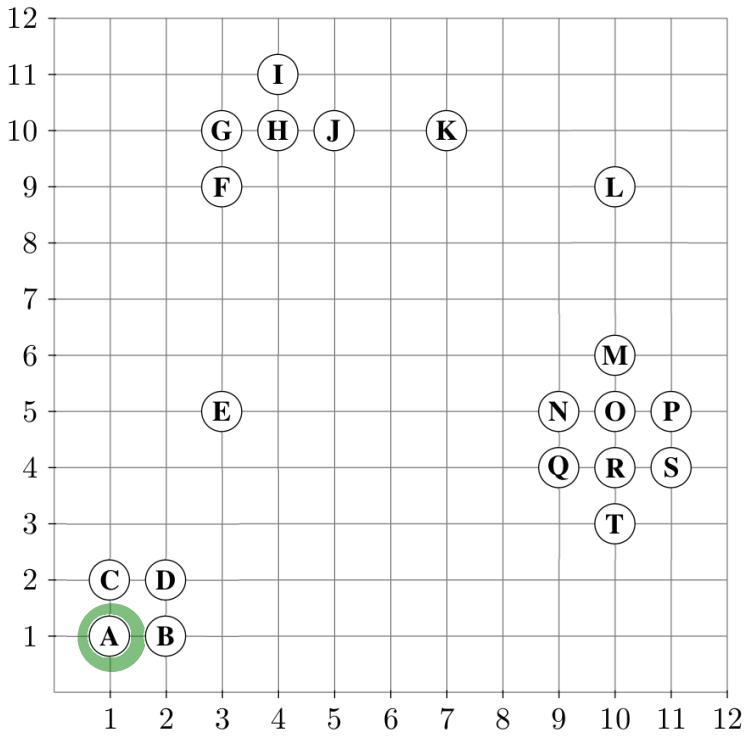
Noise

Seeds

Cluster 1:

Cluster 2:

Cluster 3:



Start: A

Seeds := RQ (A, 1.1)

Unclassified

A B C D E F G H I J

K L M N O P Q R S T

Noise

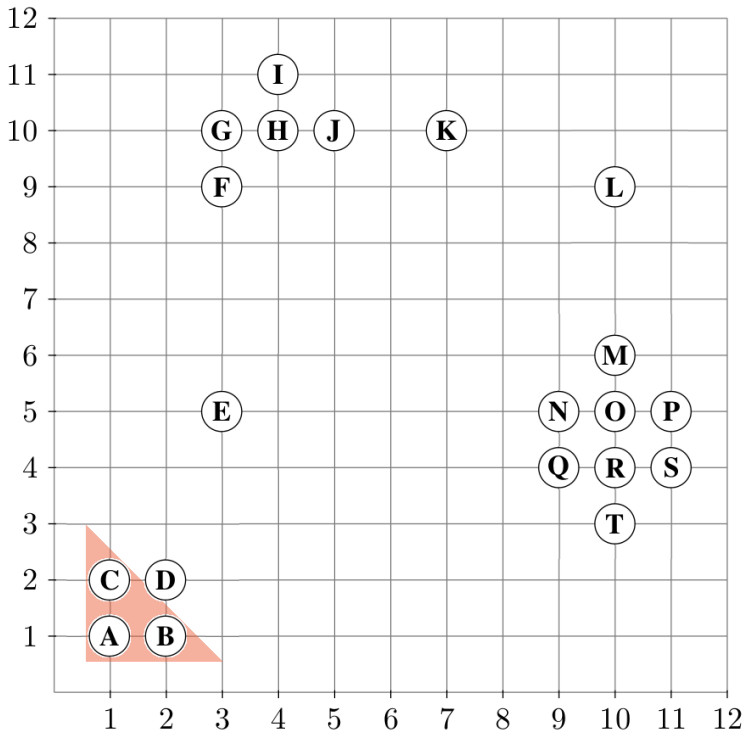
Seeds

A B C

Cluster 1:

Cluster 2:

Cluster 3:



Cluster: A B C

Forall o in Seeds: o.ClId := ClusterId
Remove starting object from Seeds

Unclassified

D E F G H I J
K L M N O P Q R S T

Noise

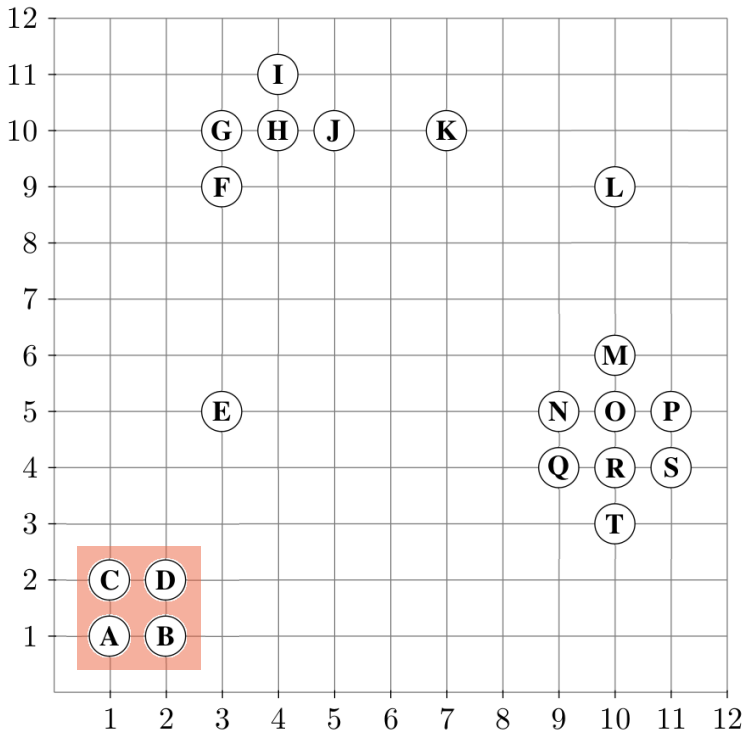
Seeds

B C

Cluster 1: A, B, C

Cluster 2:

Cluster 3:



Point: (B)

While Seeds != empty do
RQ (B, 1.1) = {A, B, D}

A.ClId = 1. finished
B.ClId = 1. finished
D.ClId = Unclassified →
Seeds += D
D.ClId = 1

Remove B from Seeds

Unclassified

E F G H I J
K L M N O P Q R S T

Noise

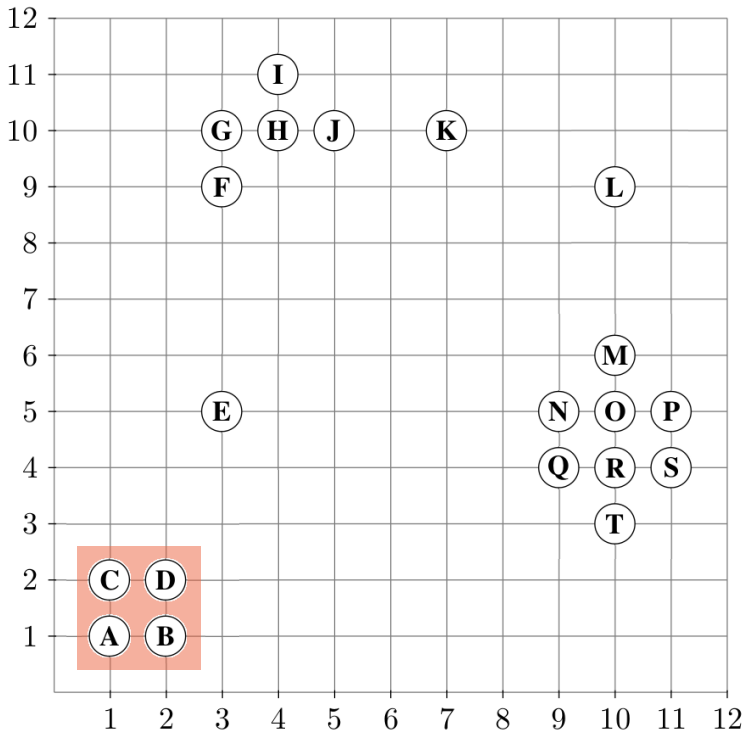
Seeds

C D

Cluster 1: A, B, C, D

Cluster 2:

Cluster 3:



Point: C

While Seeds != empty do
RQ (C, 1.1) = {A, C, D}

A.ClId = 1. finished
C.ClId = 1. finished
D.ClId = 1. finisehd

Remove C from Seeds

Unclassified

E F G H I J
K L M N O P Q R S T

Noise

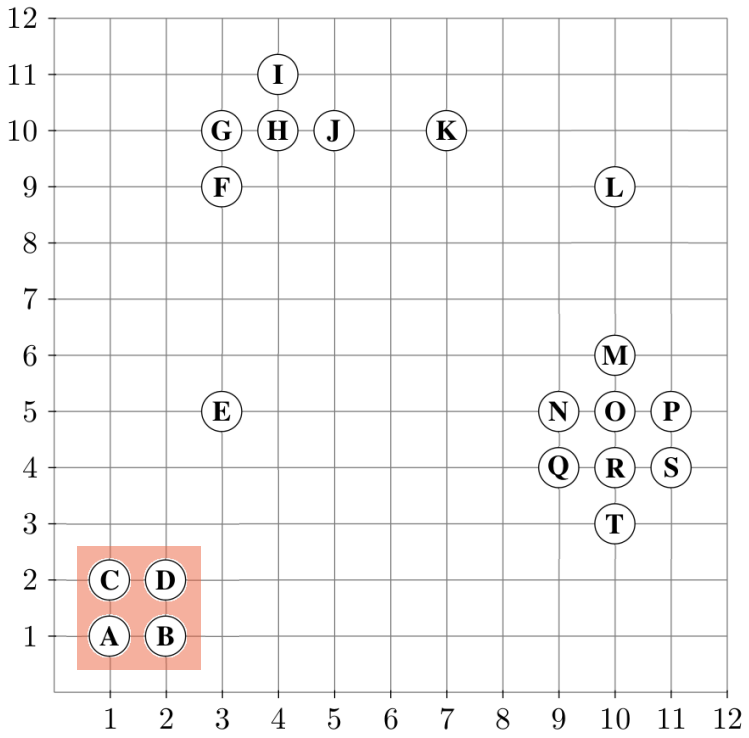
Seeds

D

Cluster 1: A, B, C, D

Cluster 2:

Cluster 3:



Point : (D)

While Seeds != empty do
RQ (D, 1.1) = {B, C, D}

B.ClId = 1. finished
C.ClId = 1. finished
D.ClId = 1. finished

Remove D from Seeds

Unclassified

E F G H I J

K L M N O P Q R S T

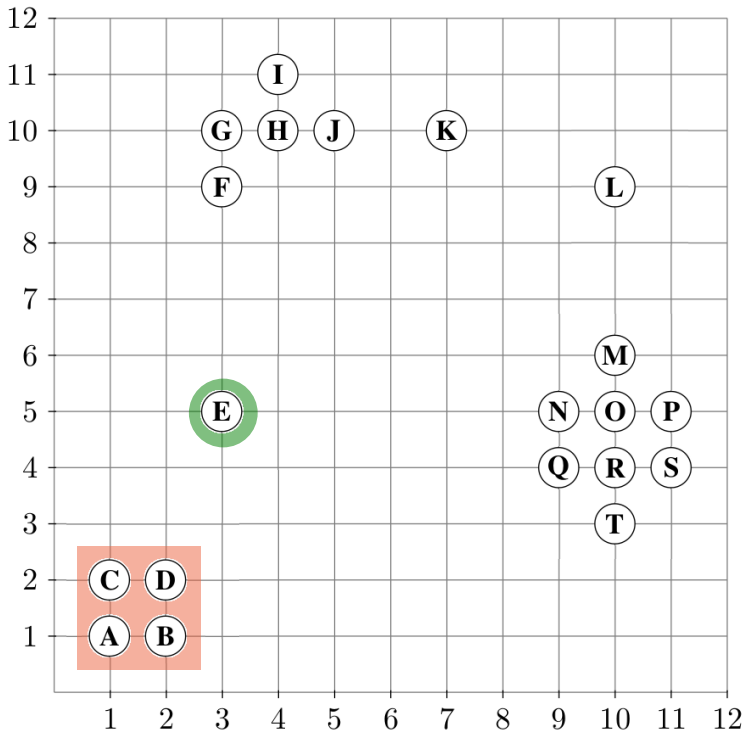
Noise

Seeds

Cluster 1: A, B, C, D

Cluster 2:

Cluster 3:



Start: E

E.ClId = Unclassified

ExpandCluster (DB, E, 2, 1.1, 3) = false

E.ClId := Noise

Unclassified

F G H I J

K L M N O P Q R S T

Noise

E

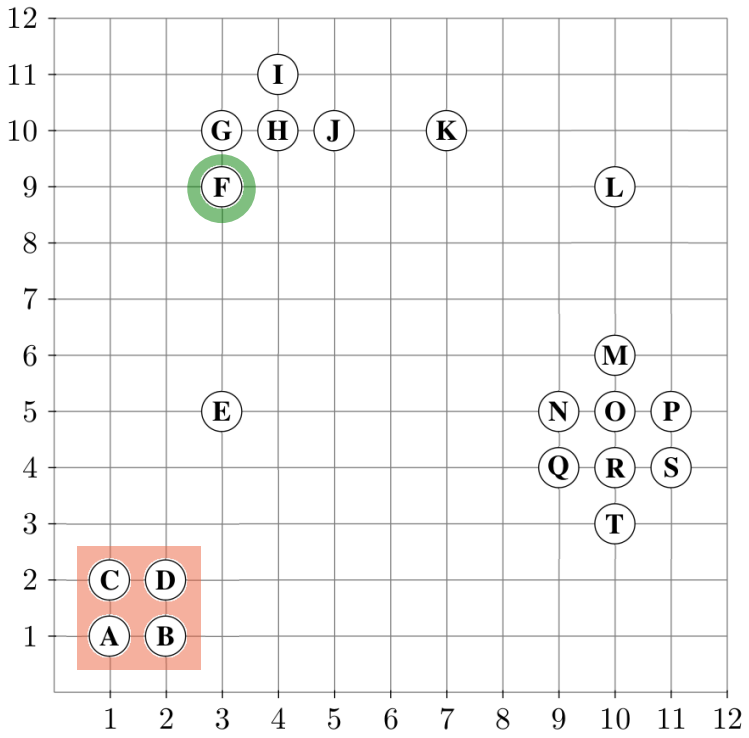
Seeds

E

Cluster 1: A, B, C, D

Cluster 2:

Cluster 3:



Start: F

F.ClId = Unclassified

ExpandCluster (DB, F, 2, 1.1, 3)
RQ (F, 1.1) = {F,G} → false

F.ClId := Noise

Unclassified

G H I J
K L M N O P Q R S T

Noise

E F

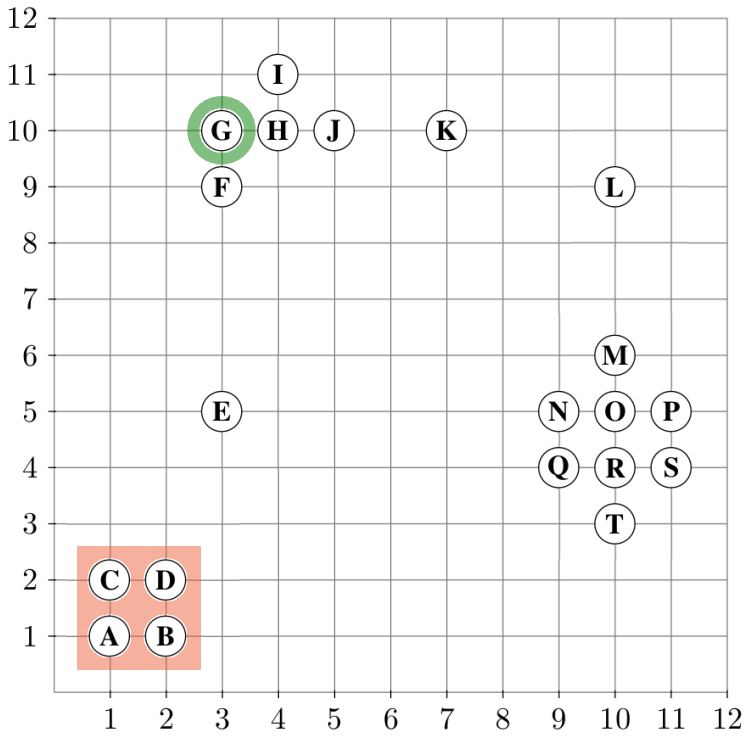
Seeds

F G

Cluster 1: A, B, C, D

Cluster 2:

Cluster 3:



Start: G

G.ClId = Unclassified

ExpandCluster (DB, G, 2, 1.1, 3)
RQ (G, 1.1) = {F,G,H}

Unclassified

G H I J
K L M N O P Q R S T

Noise

E F

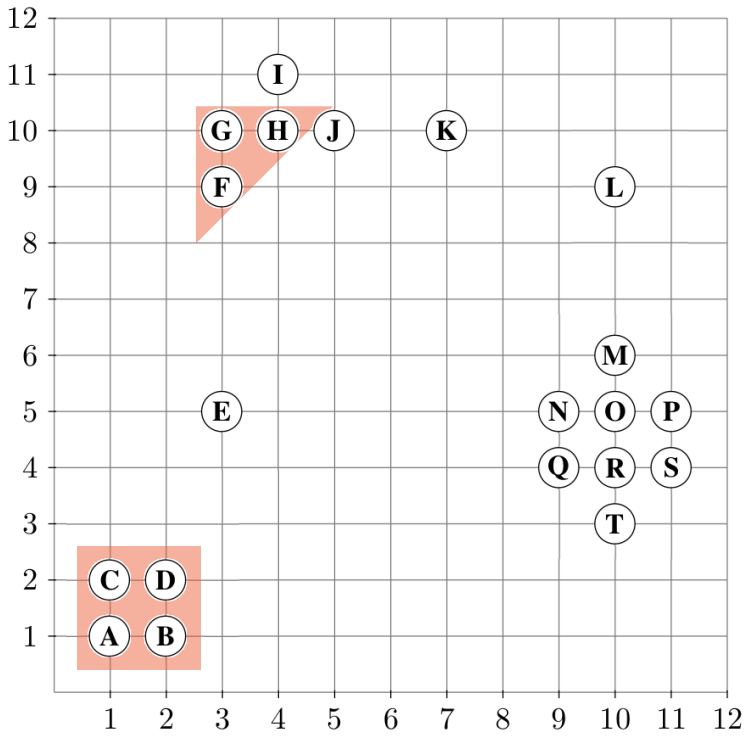
Seeds

F G H

Cluster 1: A, B, C, D

Cluster 2:

Cluster 3:



Cluster: F G H

Forall o in Seeds:
o.ClId := ClusterId
Remove G from Seeds

Unclassified

I J

K L M N O P Q R S T

Noise

E

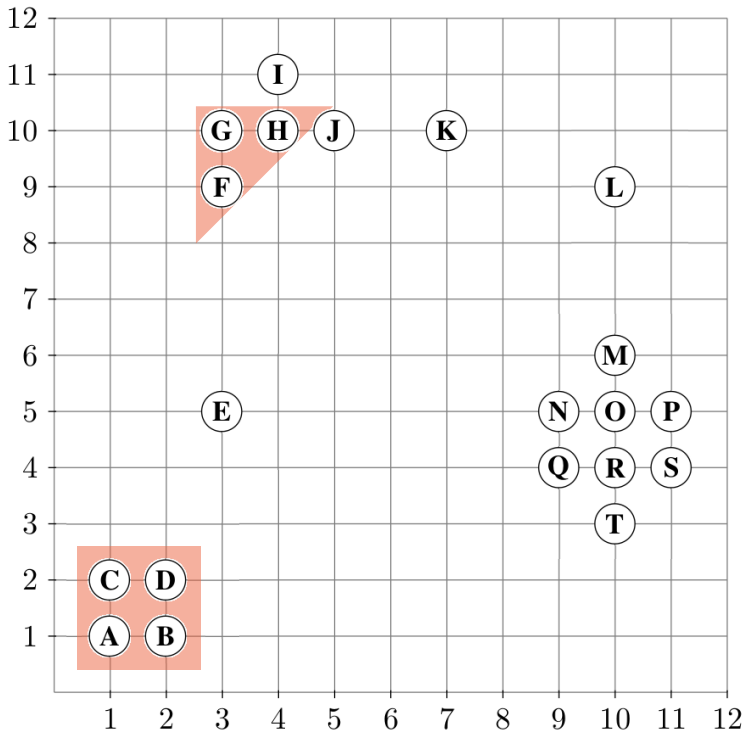
Seeds

F H

Cluster 1: A, B, C, D

Cluster 2: F, G, H

Cluster 3:



Point : (F)

While Seeds != empty do
RQ (F, 1.1) = {F, G}

F.CIId = 2. finished
G.CIId = 2. finished

Remove F from Seeds

Unclassified

I J

K L M N O P Q R S T

Noise

E

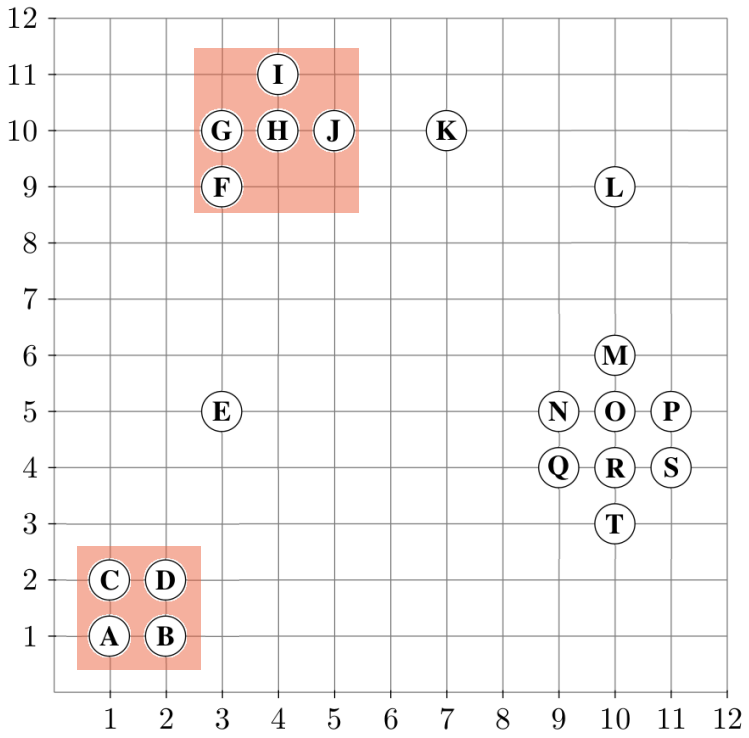
Seeds

H

Cluster 1: A, B, C, D

Cluster 2: F, G, H

Cluster 3:



Point : (H)

While Seeds != empty do
RQ (H, 1.1) = {G, H, I, J}

G.ClId = 2. finished
H.ClId = 2. finished
I.ClId = Unclassified → Seeds += I
J.ClId = Unclassified → Seeds += J
I.ClId := J.ClId := 2

Remove H from Seeds

Unclassified

(K) (L) (M) (N) (O) (P) (Q) (R) (S) (T)

Noise

(E)

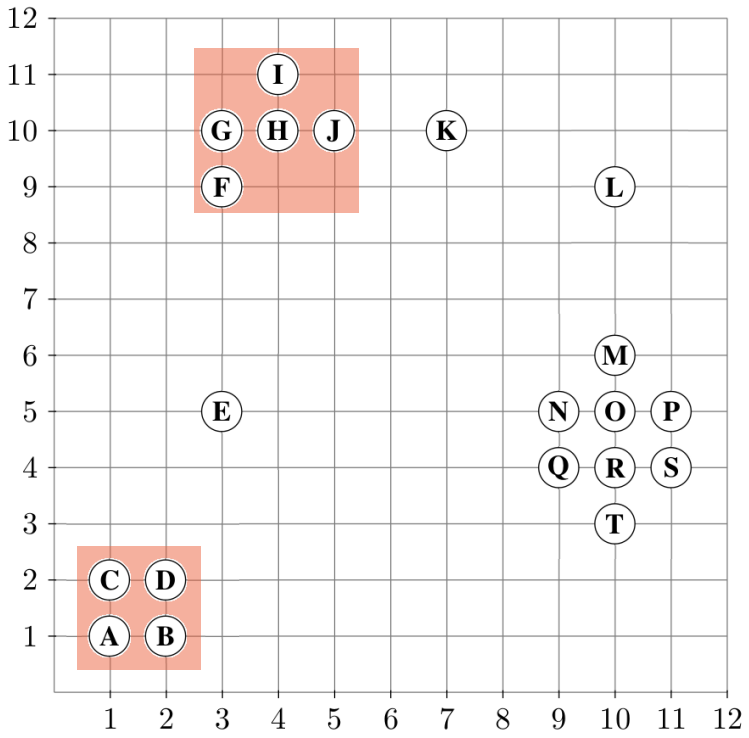
Seeds

(I) (J)

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3:



Point: I

While Seeds != empty do
RQ (I, 1.1) = {H, I}

H.ClId = 2. finished
I.ClId = 2. finished

Remove I from Seeds

Unclassified

K L M N O P Q R S T

Noise

E

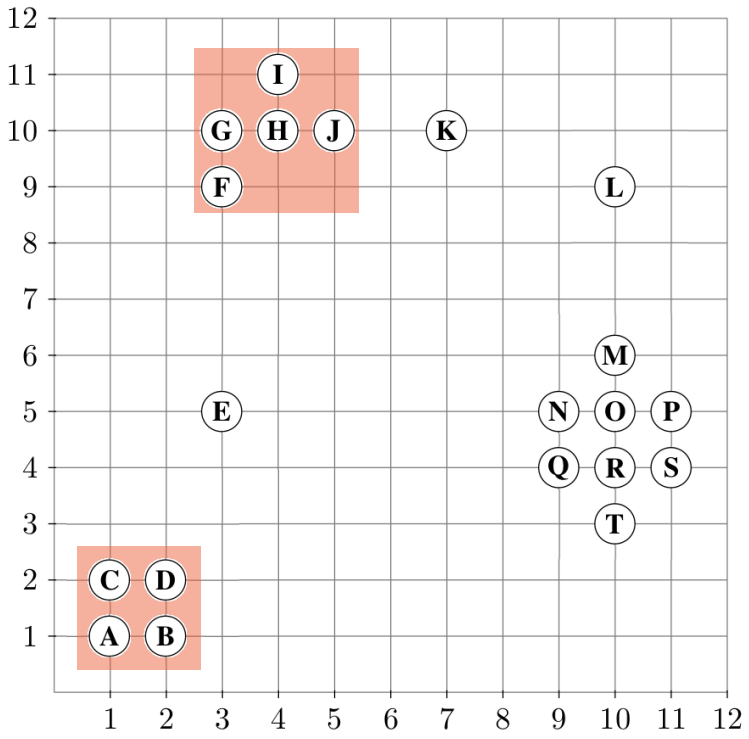
Seeds

J

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3:



Point: J

While Seeds != empty do
RQ (J, 1.1) = {H, J}

H.CIId = 2. finished
J.CIId = 2. finished

Remove J from Seeds

Unclassified

K L M N O P Q R S T

Noise

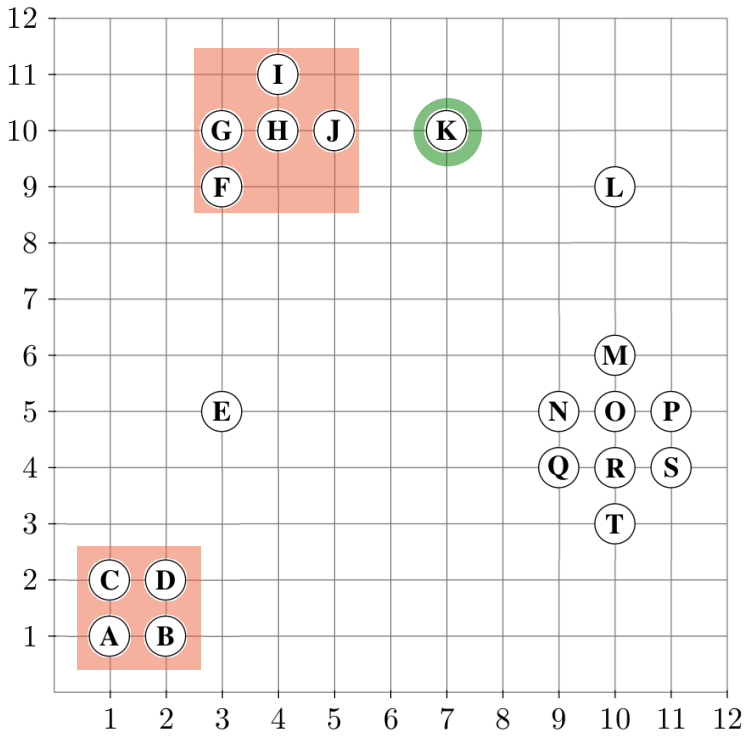
E

Seeds

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3:



Start: K

K.ClId = Unclassified

ExpandCluster (DB, K, 3, 1.1, 3) = false

K.ClId := Noise

Unclassified

L M N O P Q R S T

Noise

E K

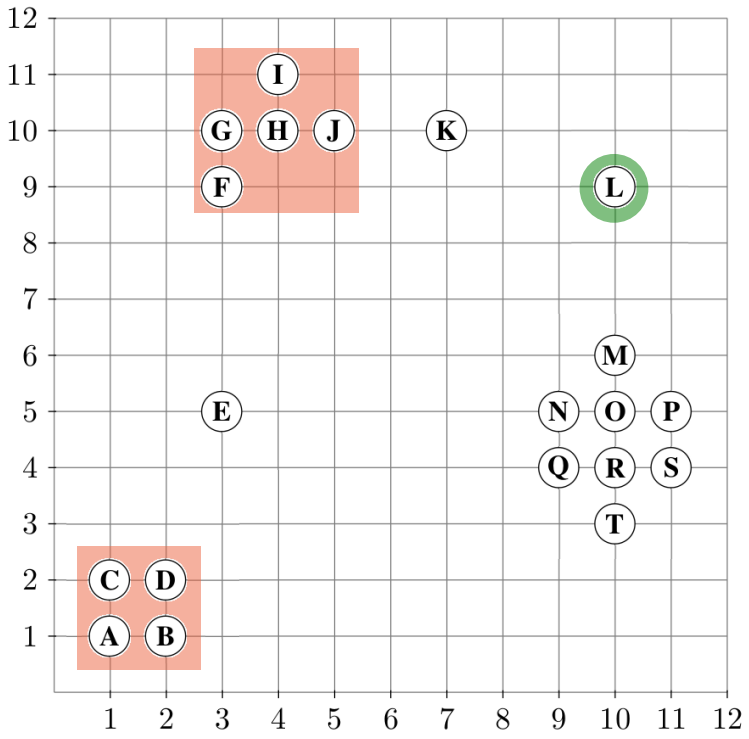
Seeds

K

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3:



Start: L

L.ClId = Unclassified

ExpandCluster (DB, L, 3, 1.1, 3) = false

L.ClId := Noise

Unclassified

M N O P Q R S T

Noise

E K L

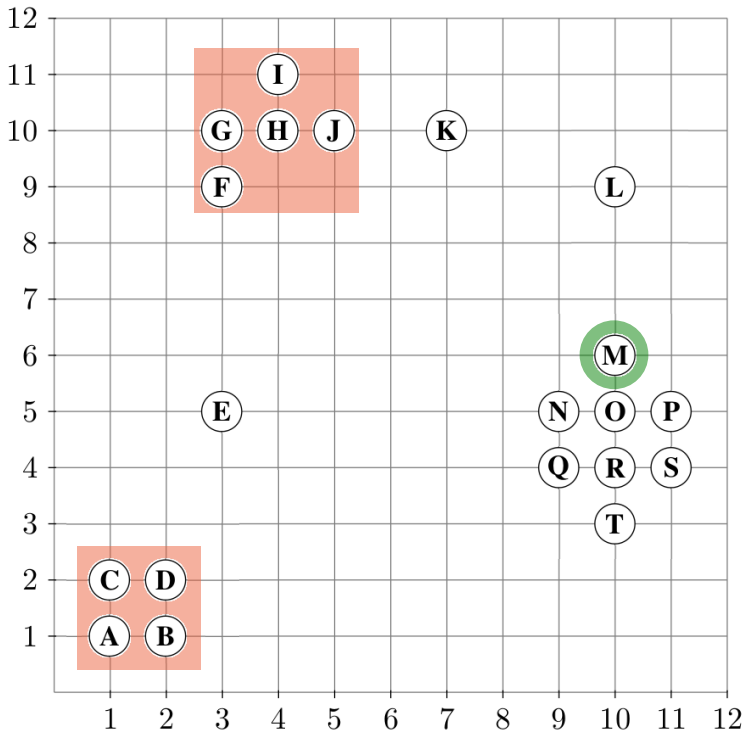
Seeds

L

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3:



Start: (M)

M.ClId = Unclassified

ExpandCluster (DB, M, 3, 1.1, 3)
RQ (M, 1.1) = {M, O} → false

M.ClId := Noise

Unclassified

(N) (O) (P) (Q) (R) (S) (T)

Noise

(E) (K) (L) (M)

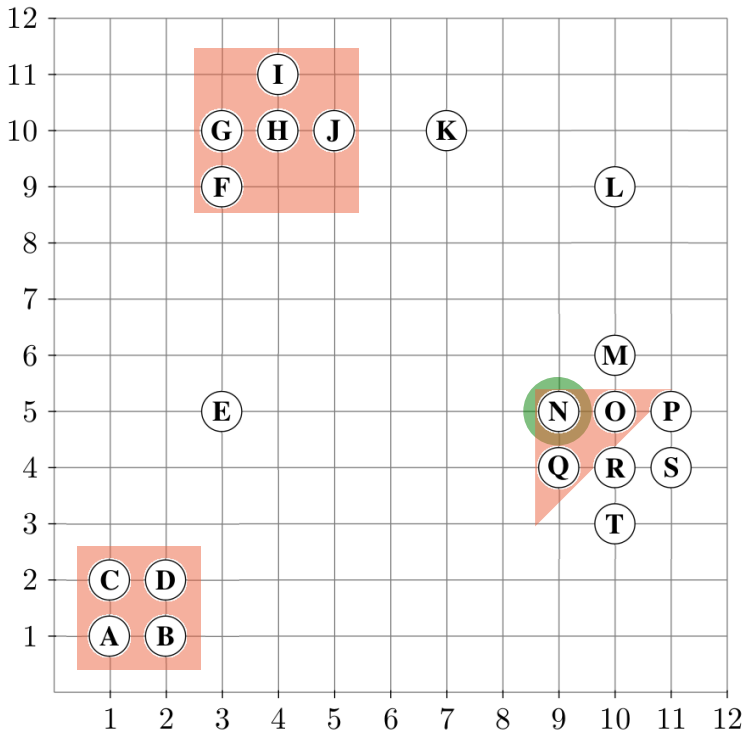
Seeds

(M) (O)

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3:



Start: N Cluster: N O Q

N.ClId = Unclassified

ExpandCluster (DB, N, 3, 1.1, 3)
RQ (M, 1.1) = {N, O, Q}

Forall o in Seeds:
o.ClId := ClusterId
Remove N from Seeds

Unclassified

P R S T

Noise

E K L M

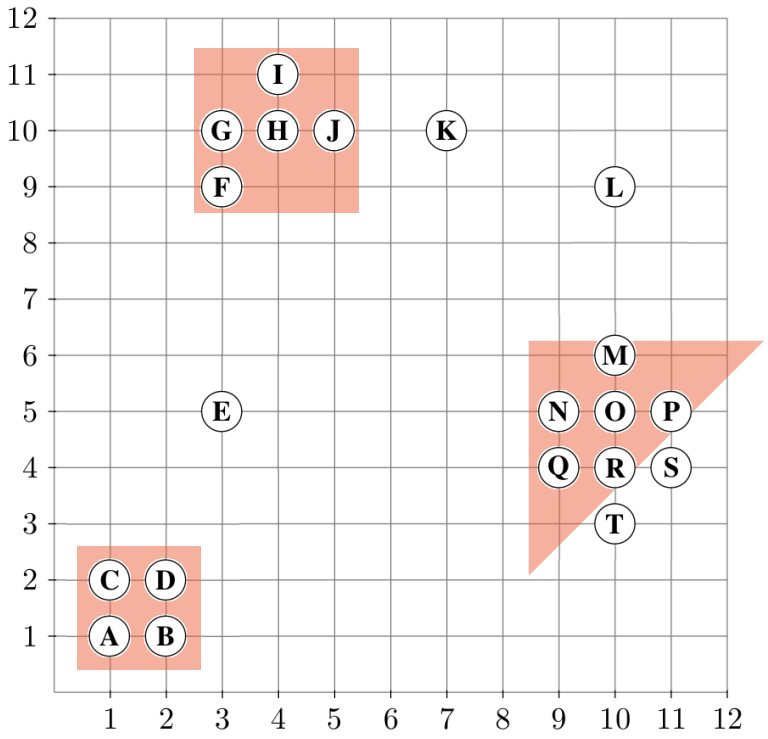
Seeds

O Q

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3: N, O, Q



Unclassified

S T

Noise

E K L

Seeds

P Q R

Point: O

While Seeds != empty do

RQ (O, 1.1) = {M, N, O, P, R}

M.ClId = Noise → M.ClId := 3

N. ClId = 3. finished

O.ClId = 3. finished

P.ClId = Unclassified → Seeds += P, P.ClId := 3

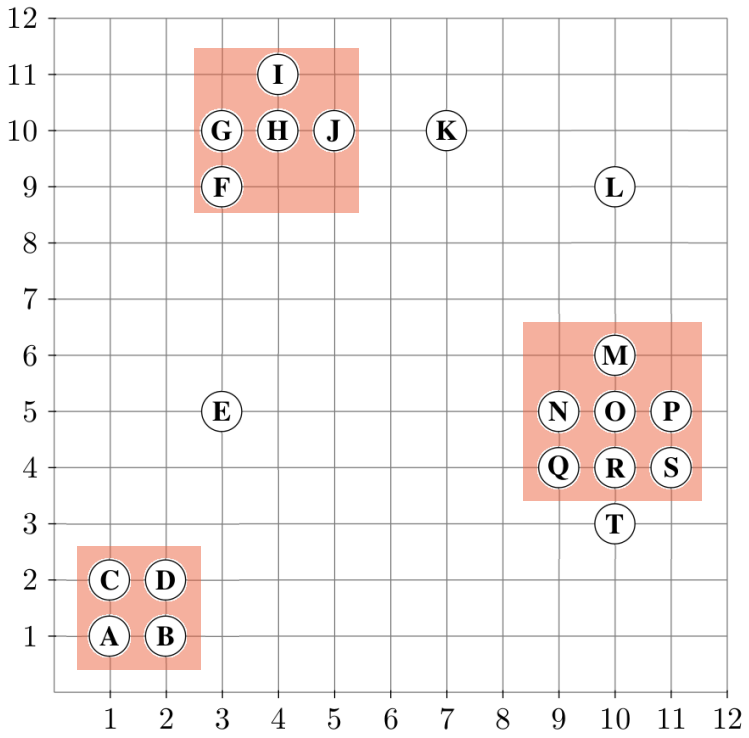
R.ClId = Unclassified → Seeds += R, R.ClId := 3

Remove O from Seeds

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3: M, N, O, P, Q, R



Unclassified

T

Noise

E K L

Seeds

Q R S

Point: P

While Seeds != empty do
RQ (P, 1.1) = {O, P, S}

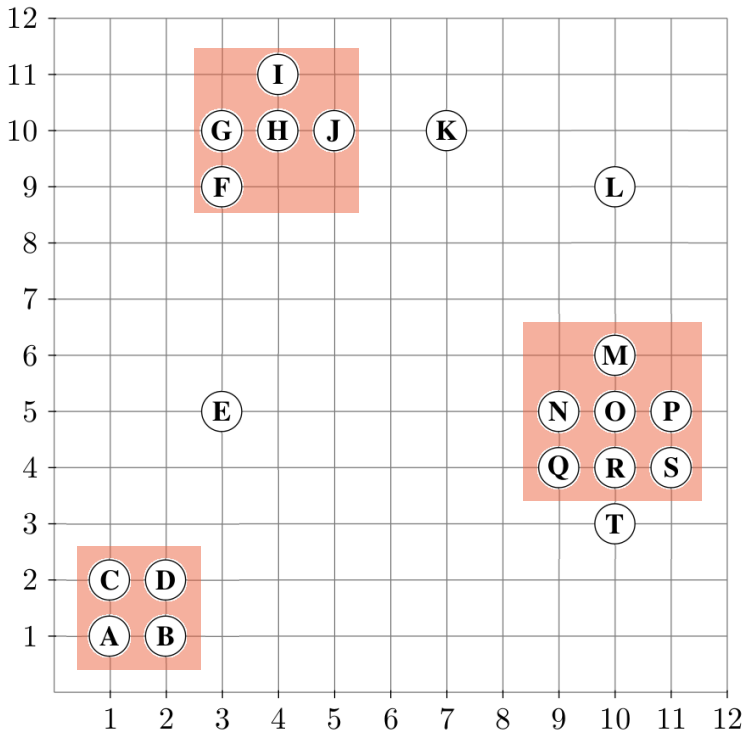
O.ClId = 3. finished
P. ClId = 3. finished
S.ClId = Unclassified → Seeds += S, S.ClId := 3

Remove P from Seeds

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3: M, N, O, P, Q, R, S



Point: Q

While Seeds != empty do
RQ (Q, 1.1) = {N, Q, R}

N.ClId = 3. finished
Q.ClId = 3. finished
R.ClId = 3. finished

Remove Q from Seeds

Unclassified

T

Noise

E K L

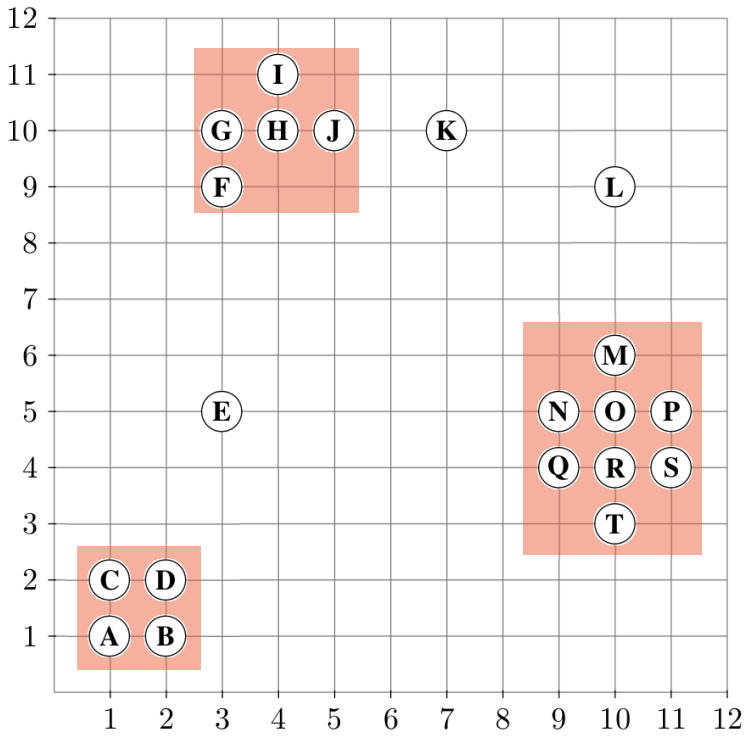
Seeds

R S

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3: M, N, O, P, Q, R, S



Unclassified

Noise

E K L

Seeds

S T

Point: R

While Seeds != empty do
RQ (R, 1.1) = {O, Q, R, S, T}

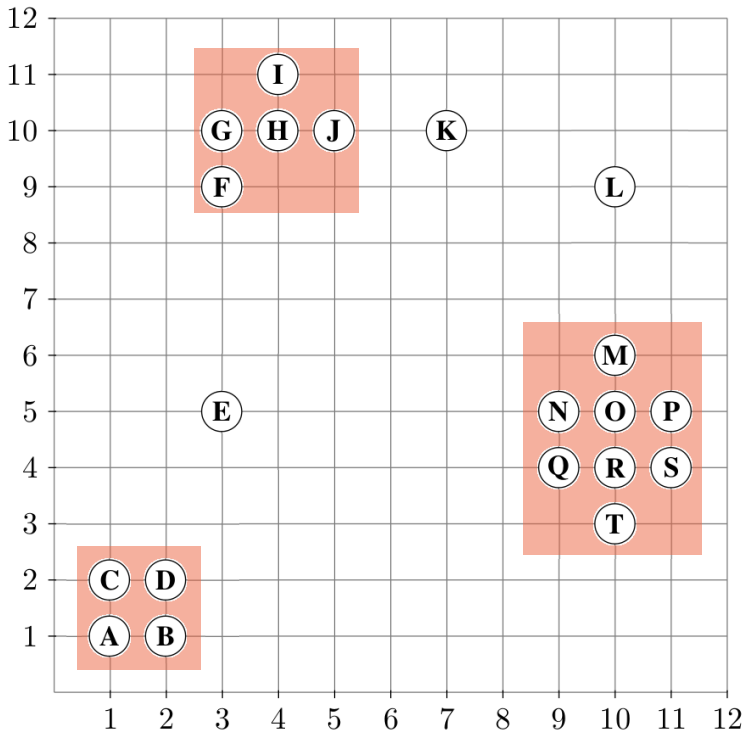
O.ClId = 3. finished
Q. ClId = 3. finished
R.ClId = 3. finished
S.ClId = 3. finished
T.ClId = Unclassified → Seeds += T; T.ClId := 3

Remove R from Seeds

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3: M, N, O, P, Q, R, S, T



Point: (S)

While Seeds != empty do
RQ (S, 1.1) = {P, R, S}

P.ClId = 3. finished
R. ClId = 3. finished
S.ClId = 3. finished

Remove S from Seeds

Unclassified

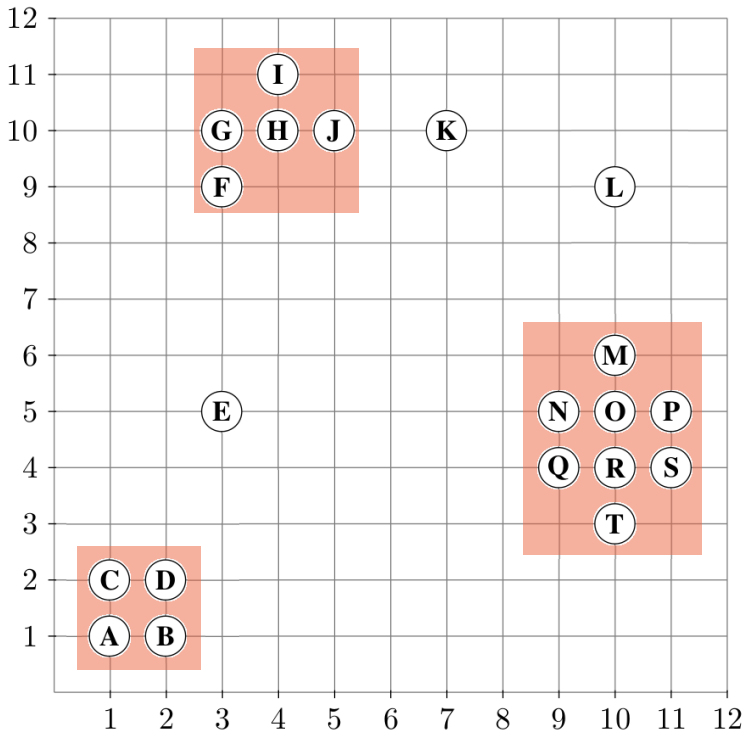
Noise
(E) (K) (L)

Seeds
(T)

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3: M, N, O, P, Q, R, S, T



Point: (T)

While Seeds != empty do
RQ (T, 1.1) = {R, T}

R.ClId = 3. finished
T. ClId = 3. finished

Remove T from Seeds

Unclassified

Noise

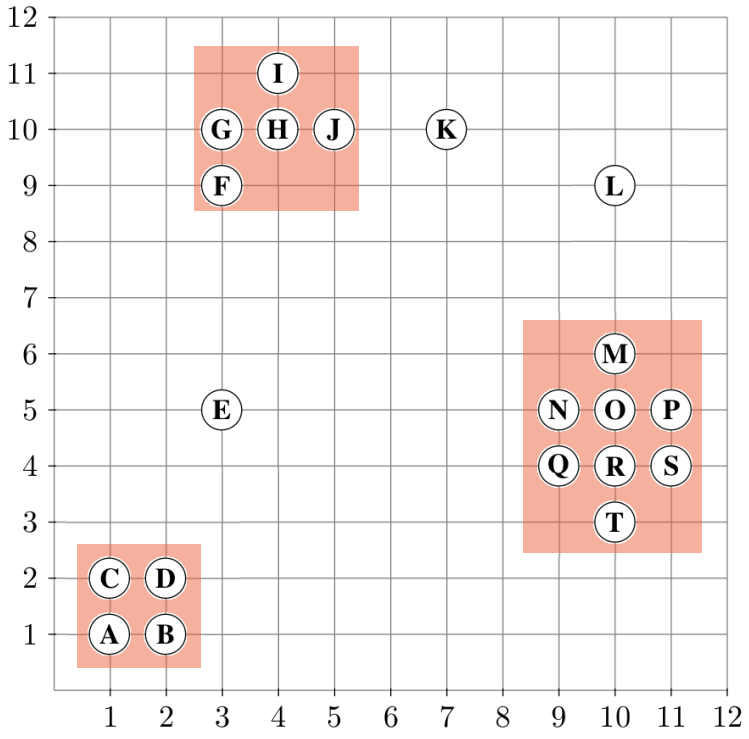
(E) (K) (L)

Seeds

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3: M, N, O, P, Q, R, S, T



Unclassified

Noise

E K L

Seeds

Cluster 1: A, B, C, D

Cluster 2: F, G, H, I, J

Cluster 3: M, N, O, P, Q, R, S, T