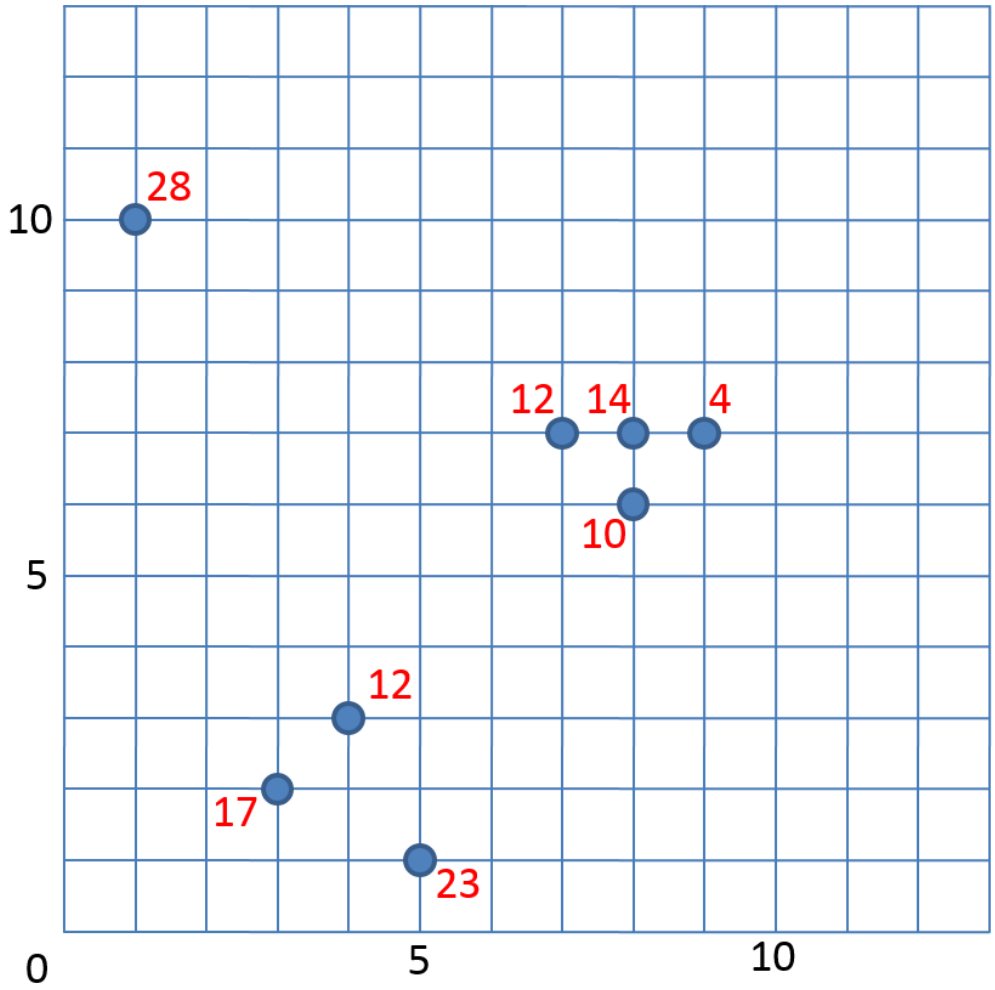
Exercise 9-2 *Spatial Outlier Detection*

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).



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

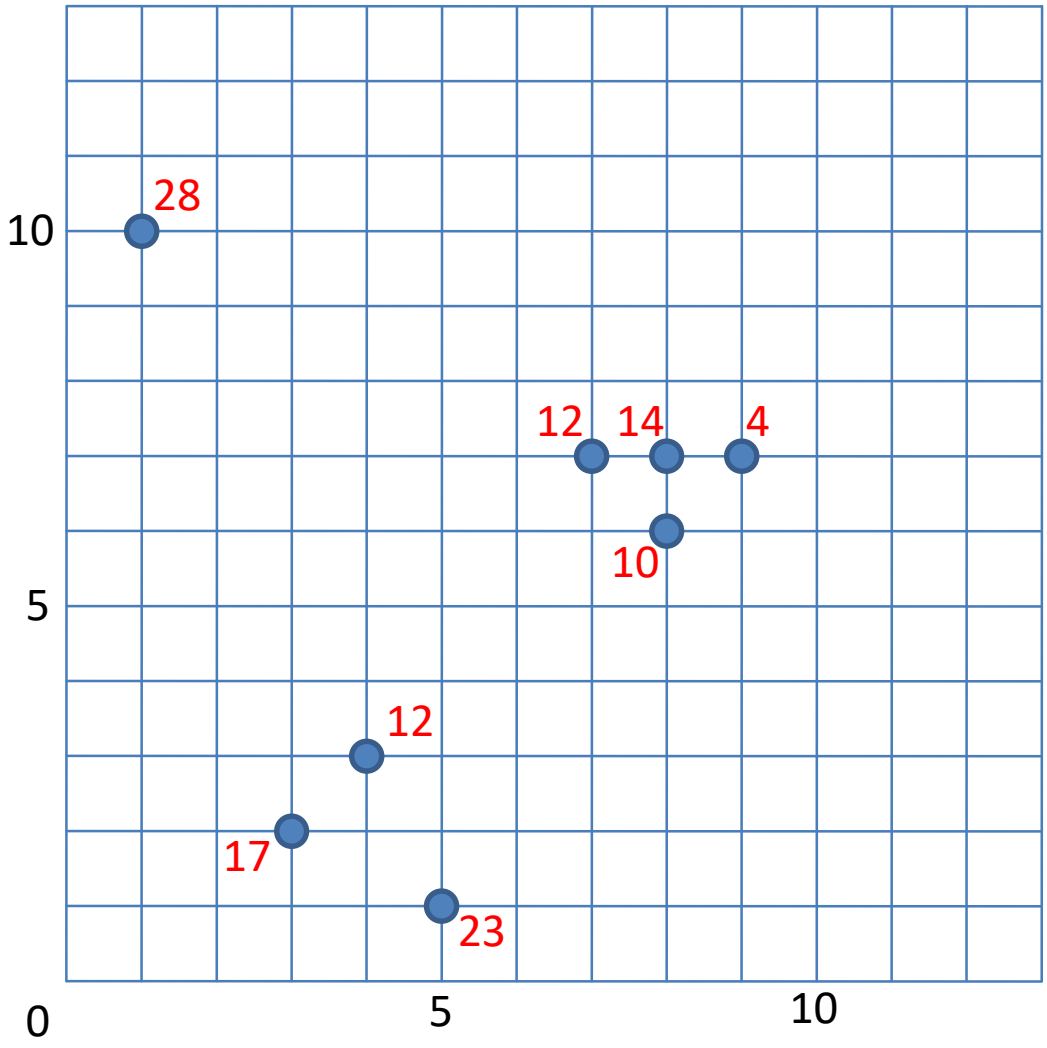
Possible options for equality of *kNN*-distance:

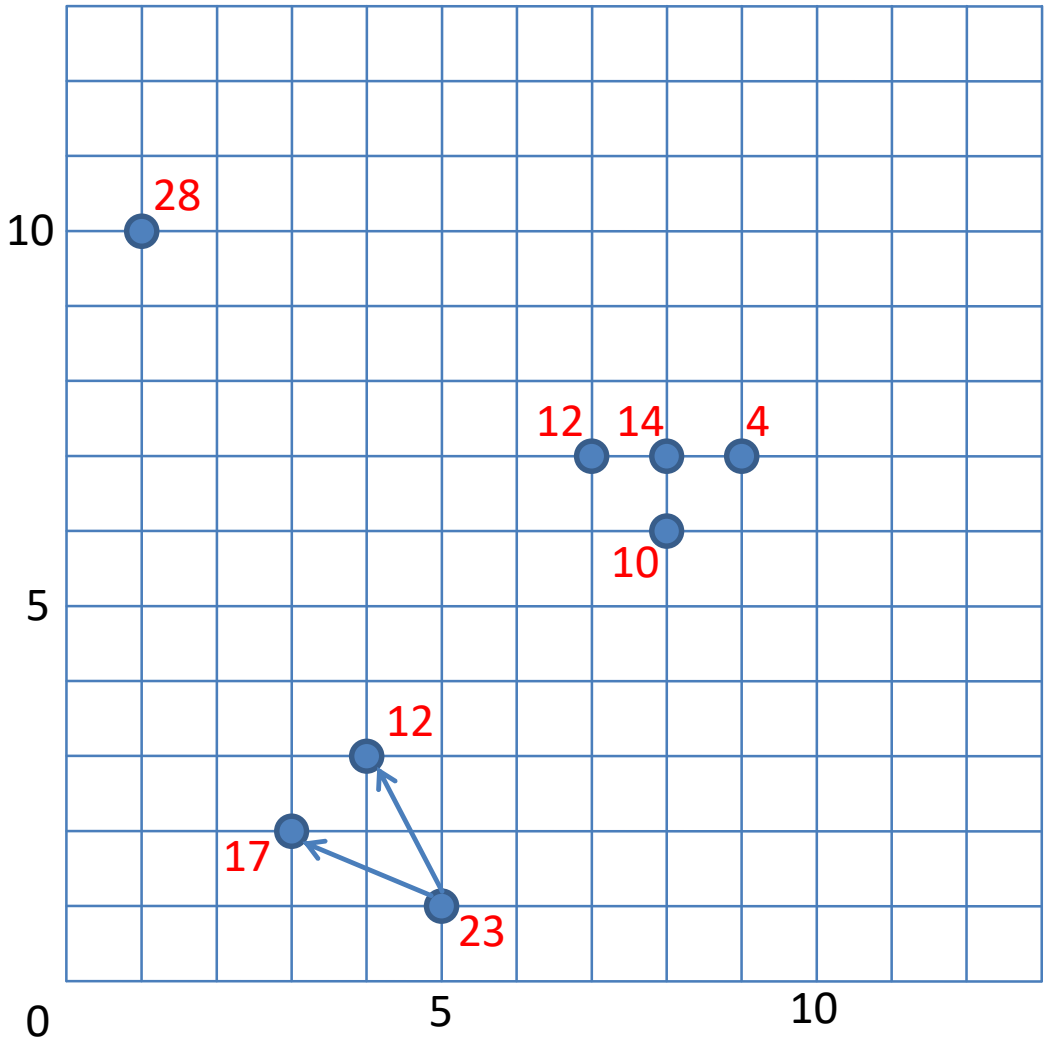
1) Add all points with equal *kNN*-distance to the *kNN*-set (potentially *#kNN >=k*)
2) Choose non-deterministically one of the points with equal *kNN*-distance (=> *#kNN=k*)

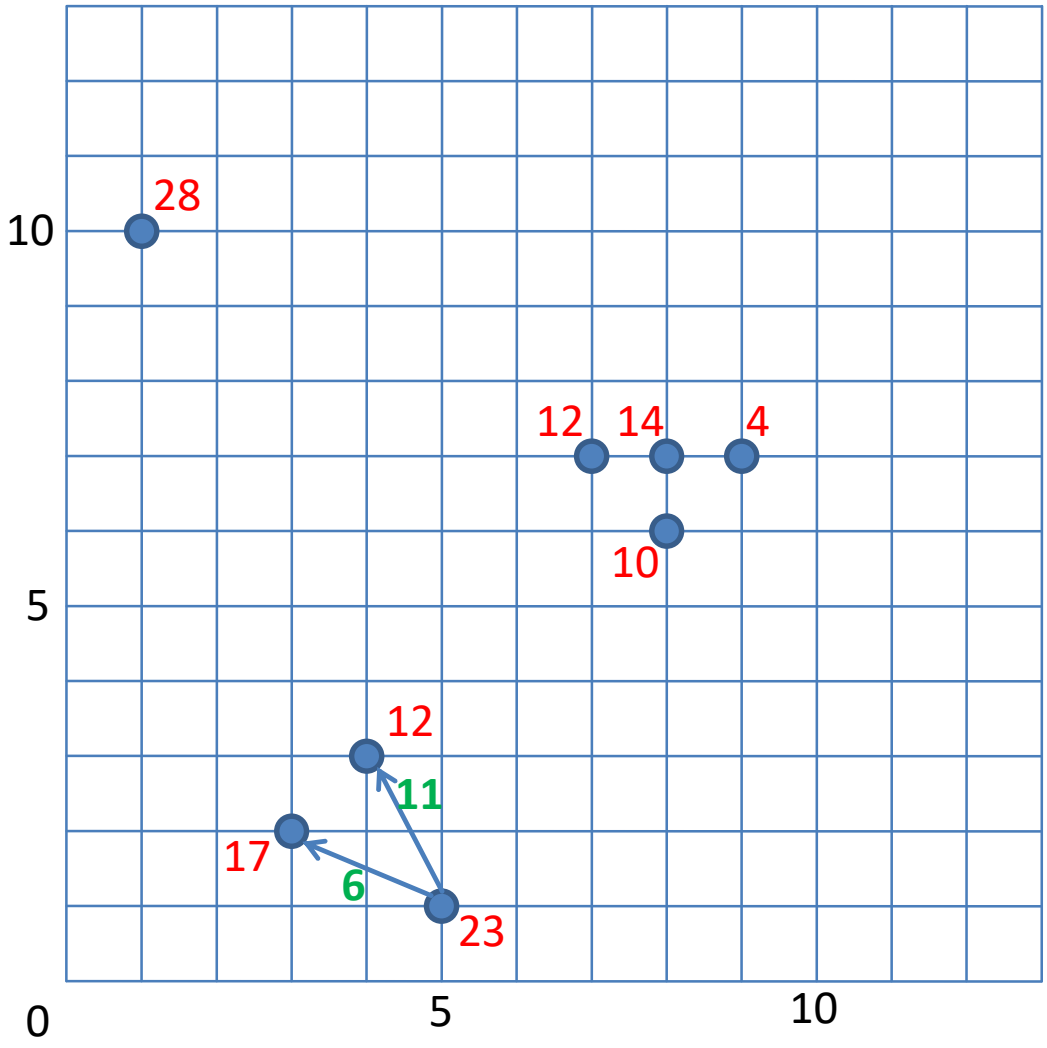
Possible options for equality of weighting of edges in the kNN-graph:

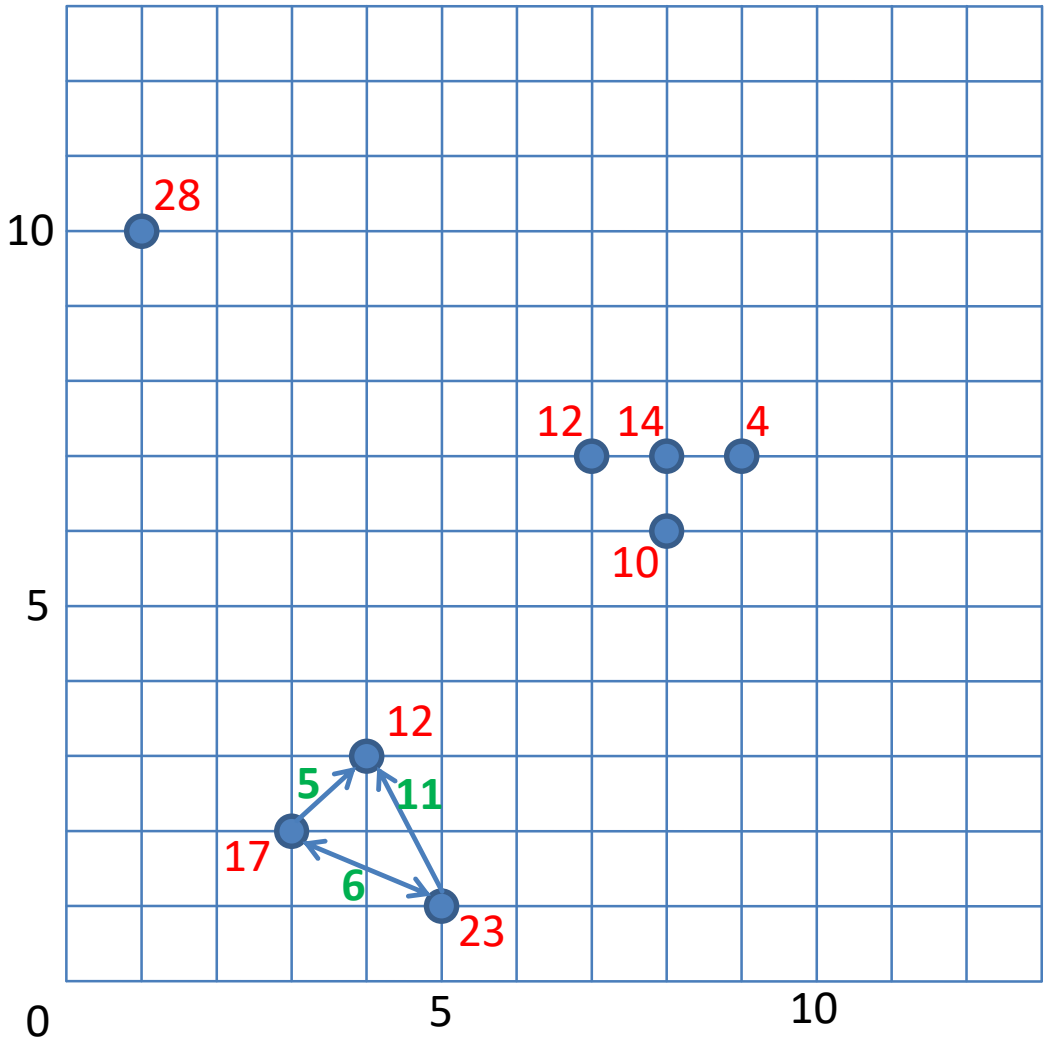
1) Delete all edges with equal weight at once
2) Delete one of the edges non-deterministically

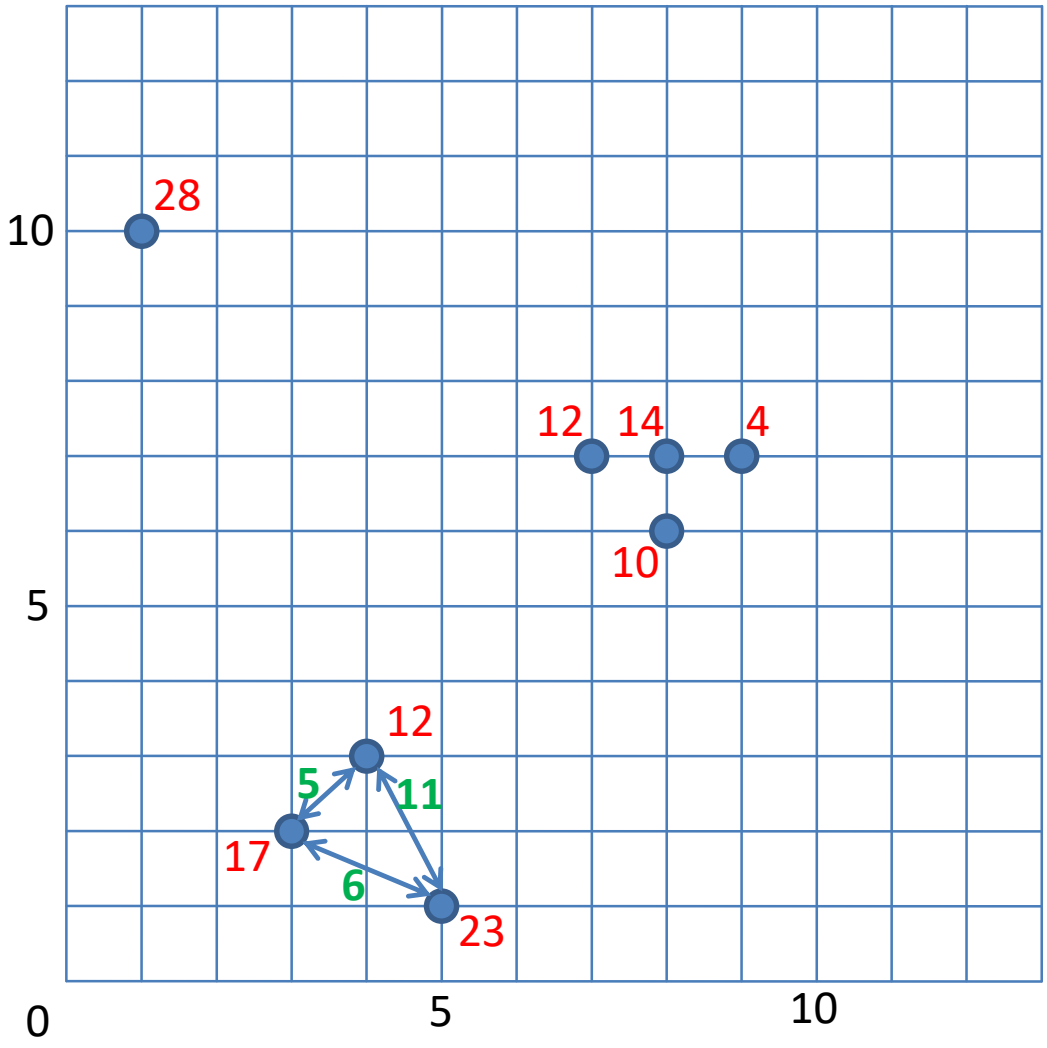
We choose strategy 1) and 1)

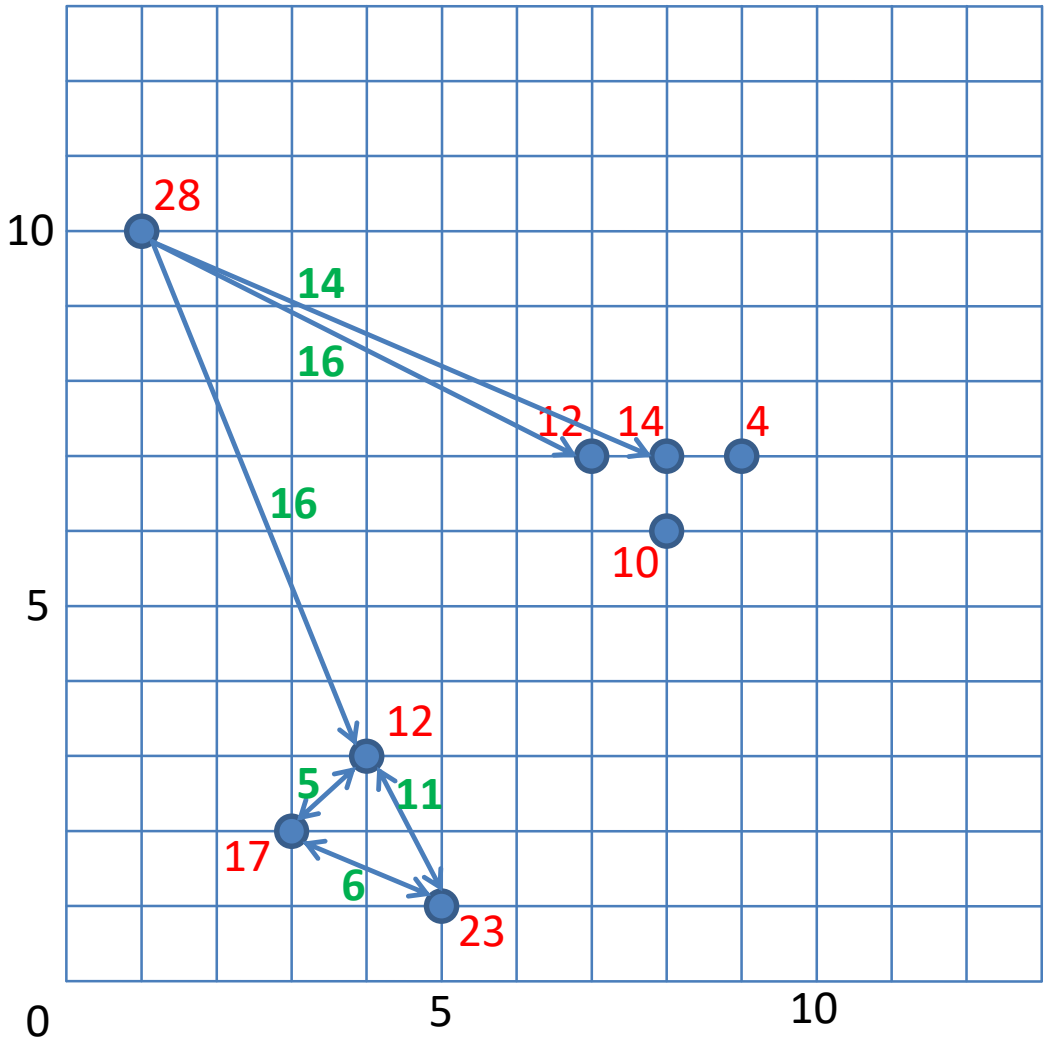

28
12
14
4
10
12
17
23
10
5
0
5
10

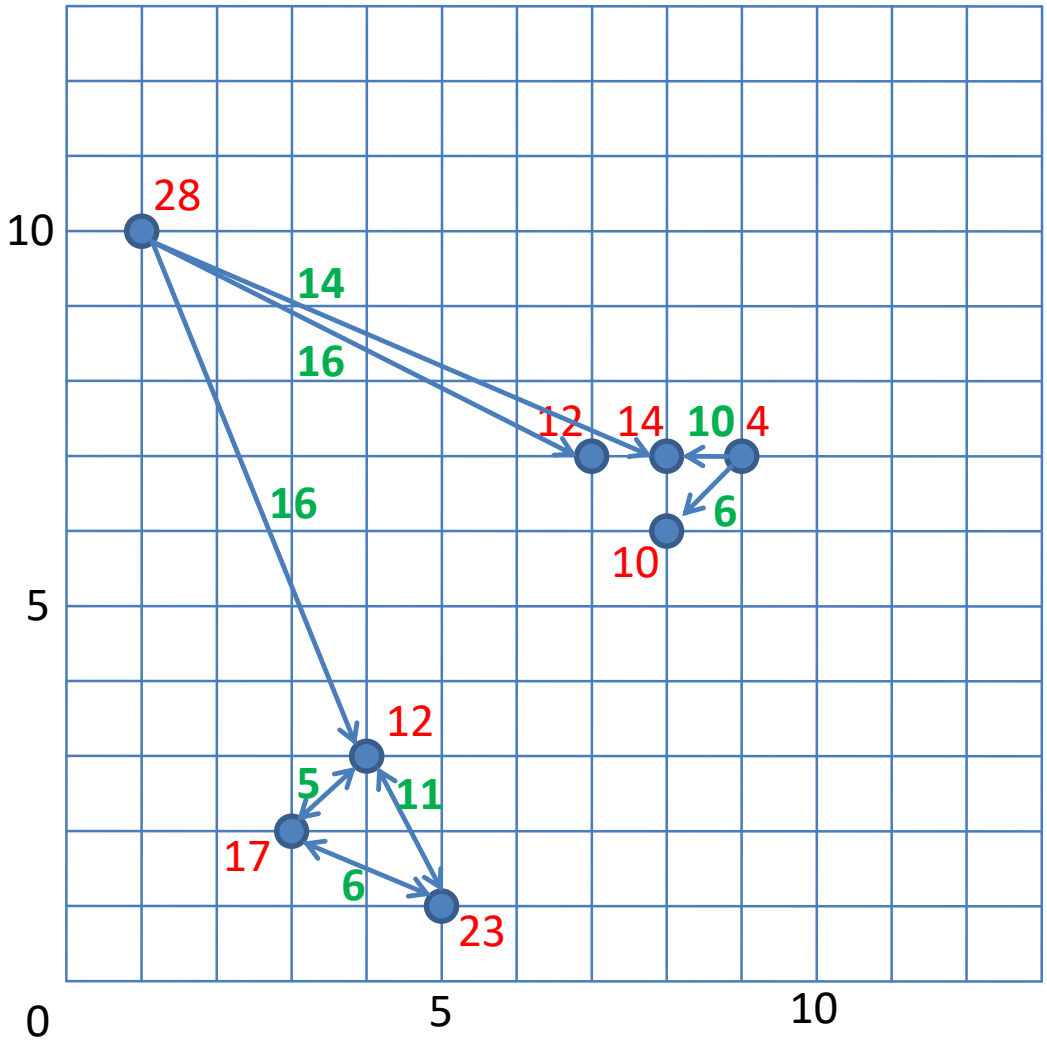

28
12
14
4
10
12
17
23
10
5
0
5
10

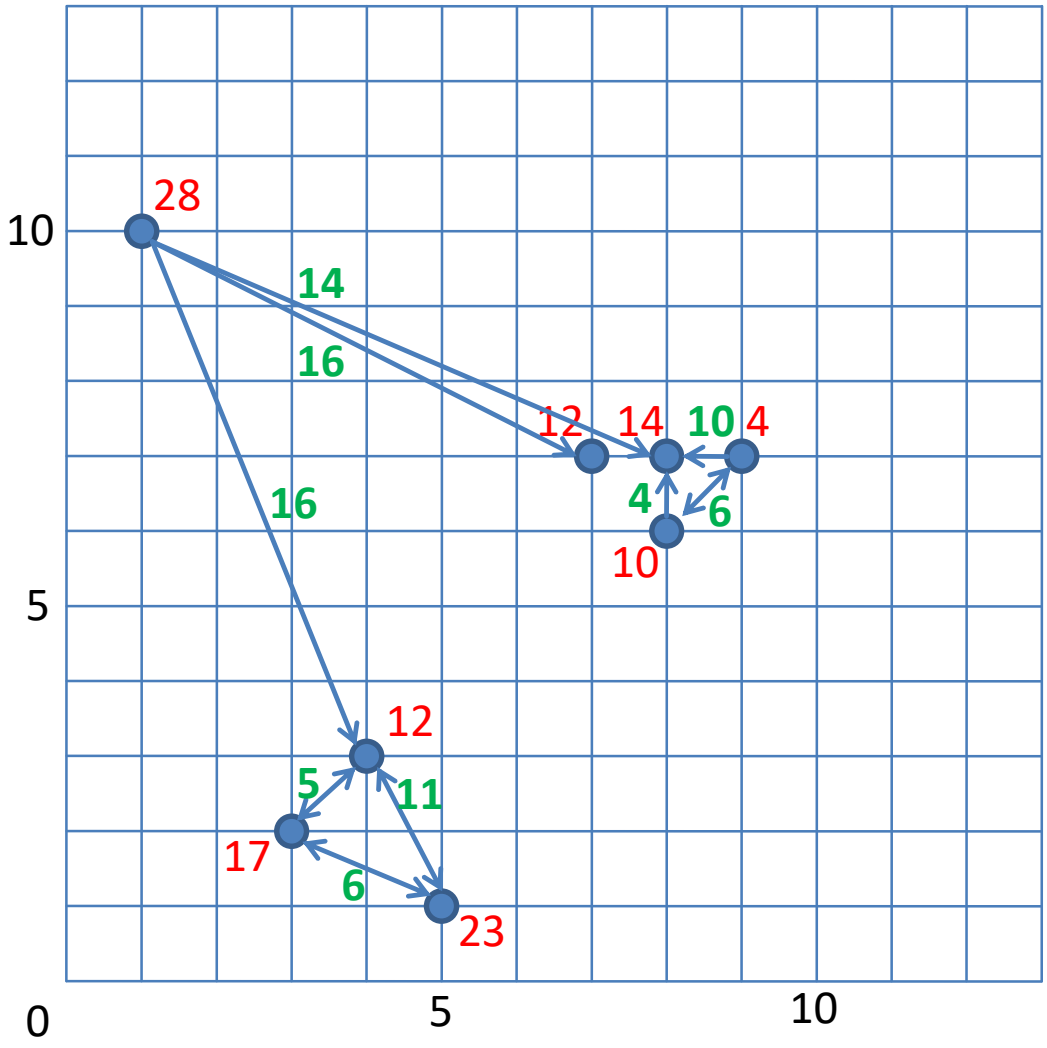

28
12
14
4
10
12
11
17
6
23
10
5
0
5
10

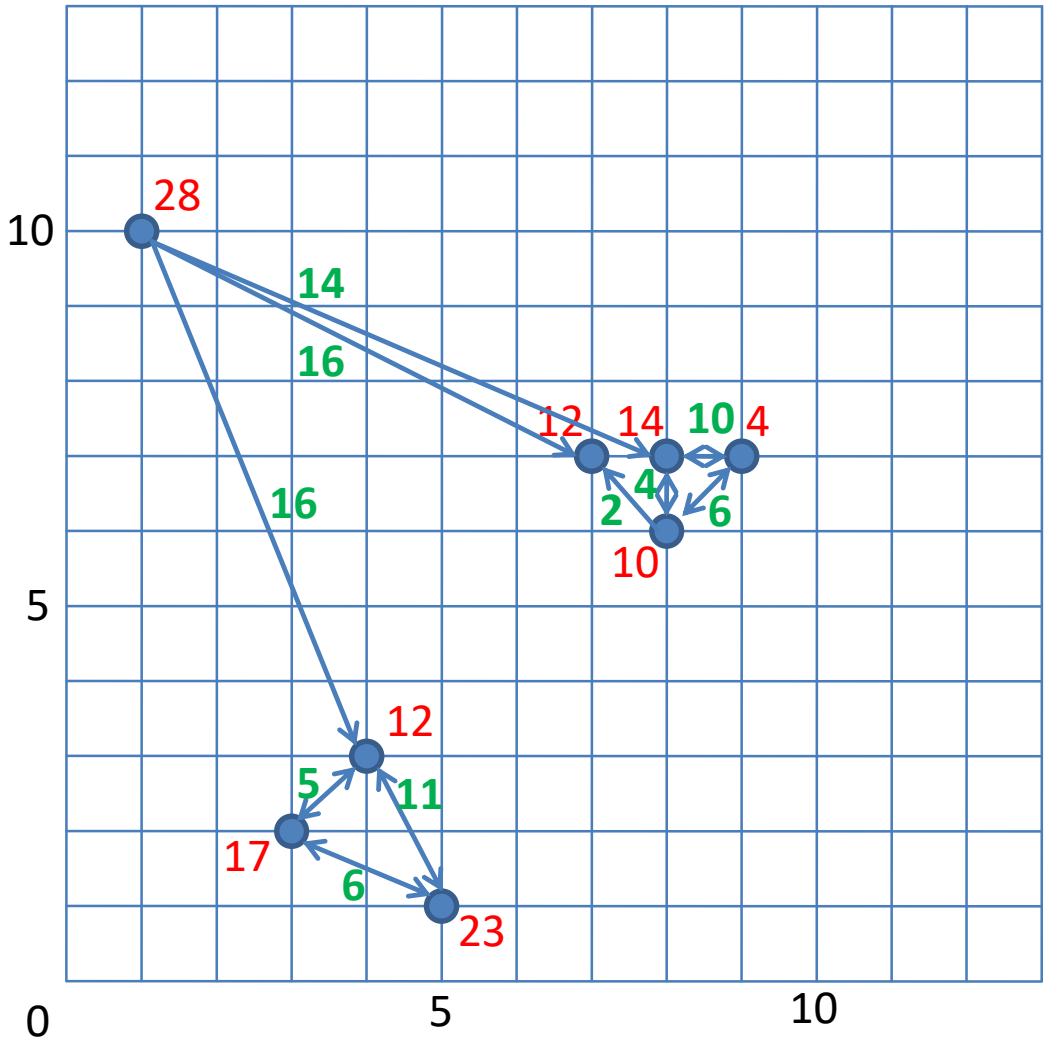

28
12
14
4
10
12
5
11
17
6
23
10
5
0
5
10


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


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

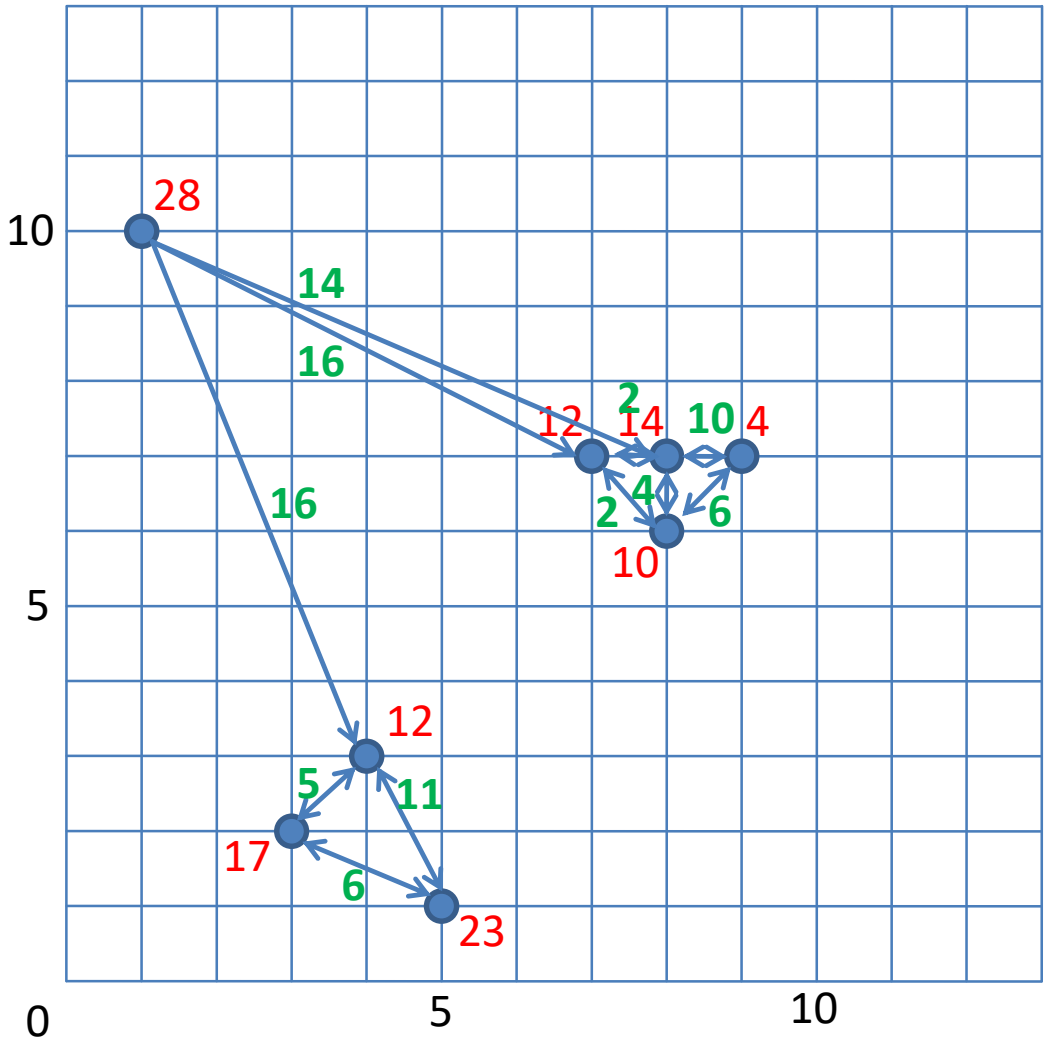

28
14
16
12
14
10
4
6
10
16
12
5
11
17
6
23
10
5
0
5
10

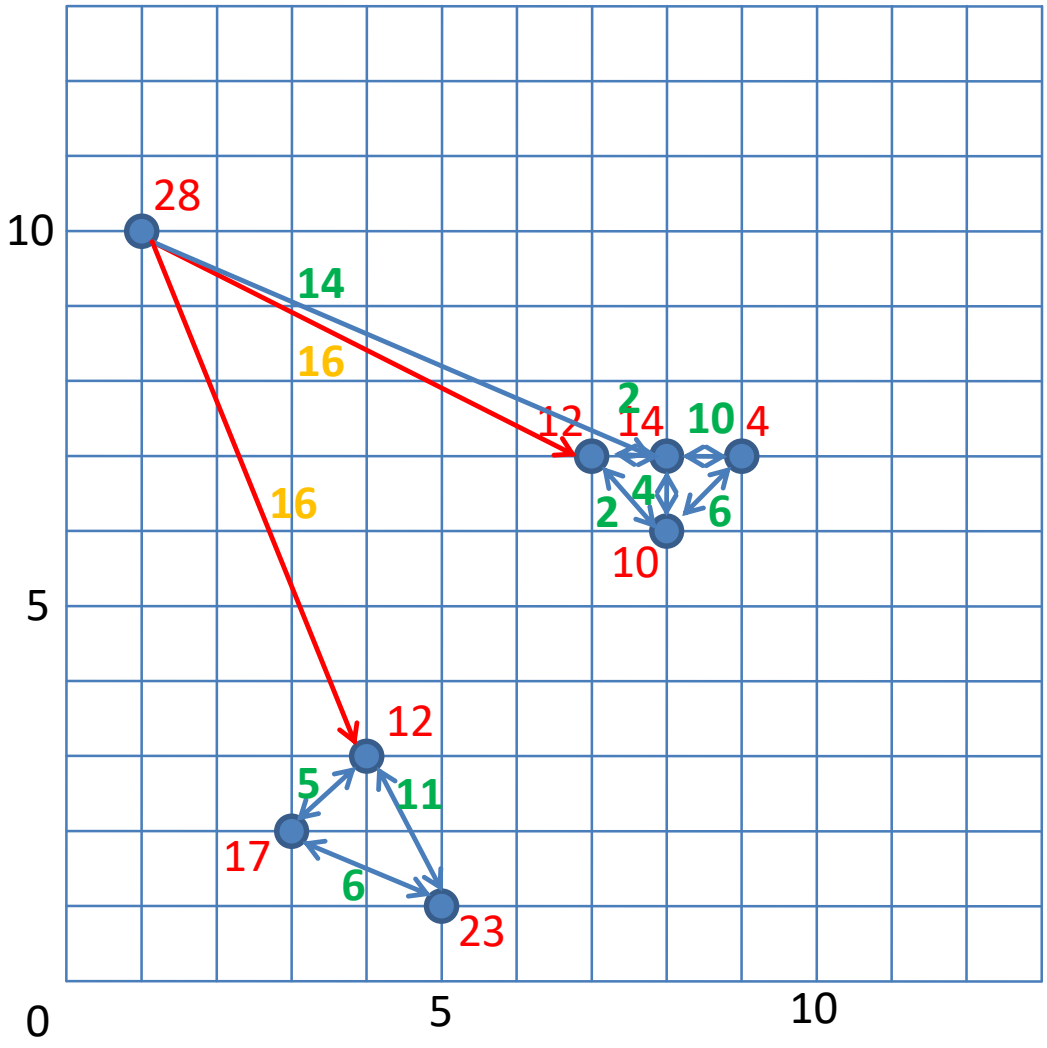

28
14
16
12
14
10
4
4
6
16
10
12
5
11
17
6
23
10
5
0
5
10

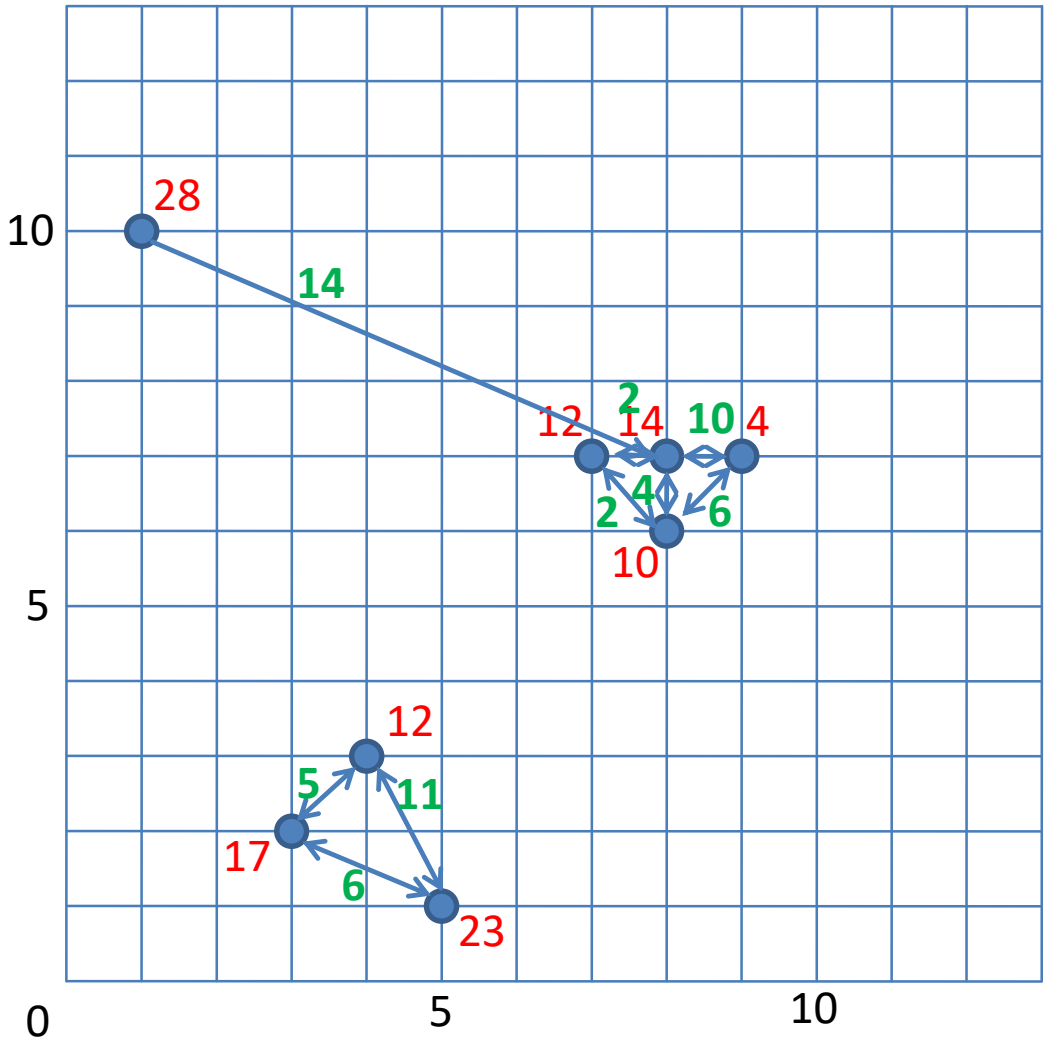

28
14
16
16
12
14
10
4
4
2
6
10
12
5
11
17
6
23
10
5
0
5
10

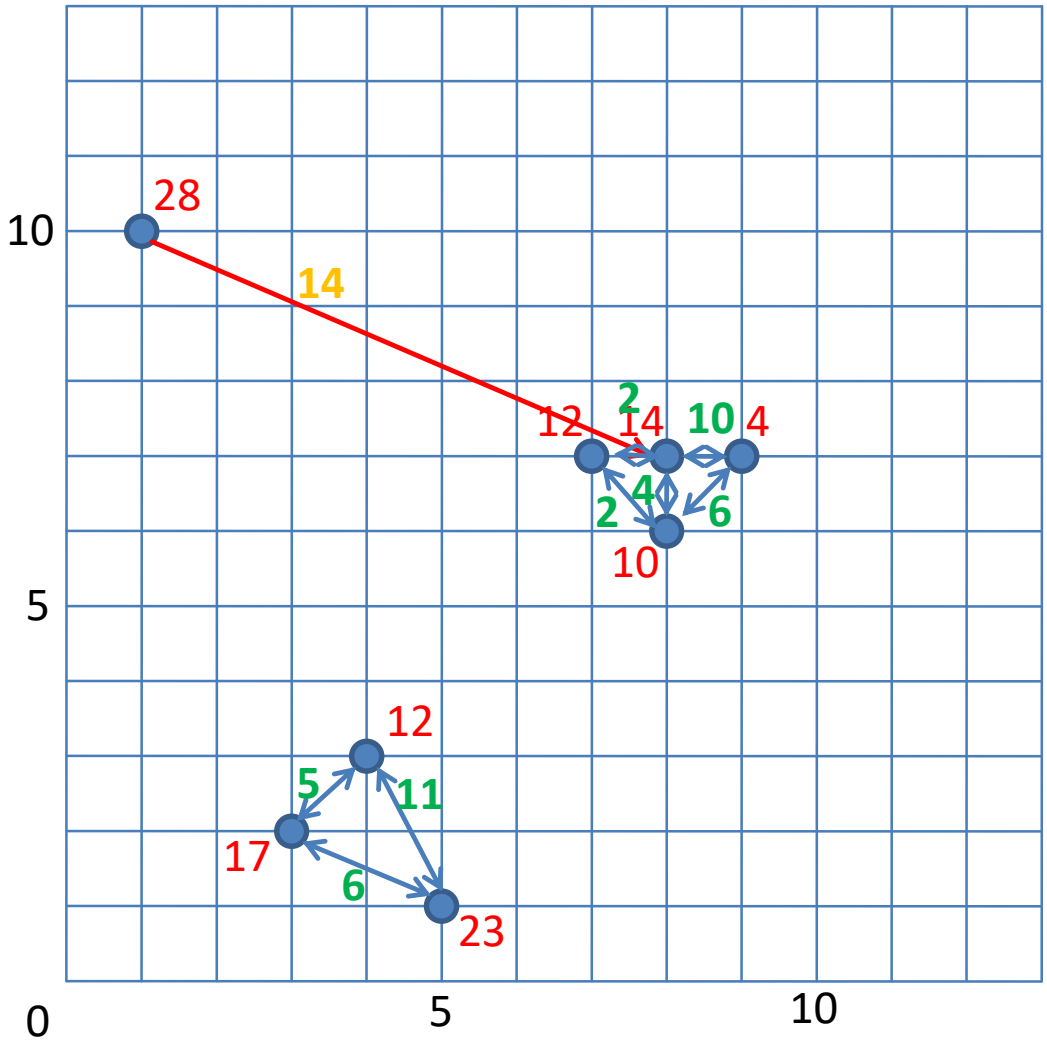
Construction of neighborhood graph finished

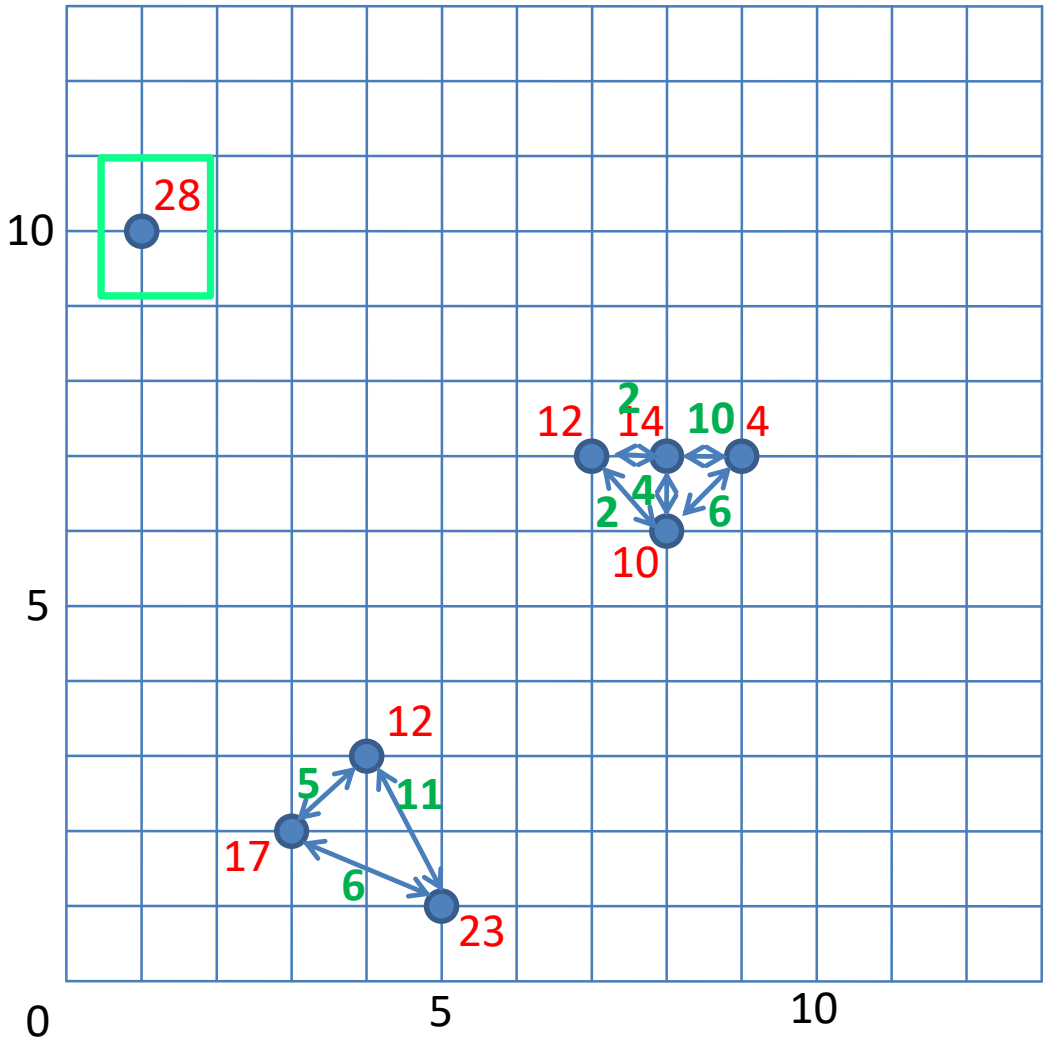
Now delete the edges with highest weights (=score difference) in a descending order

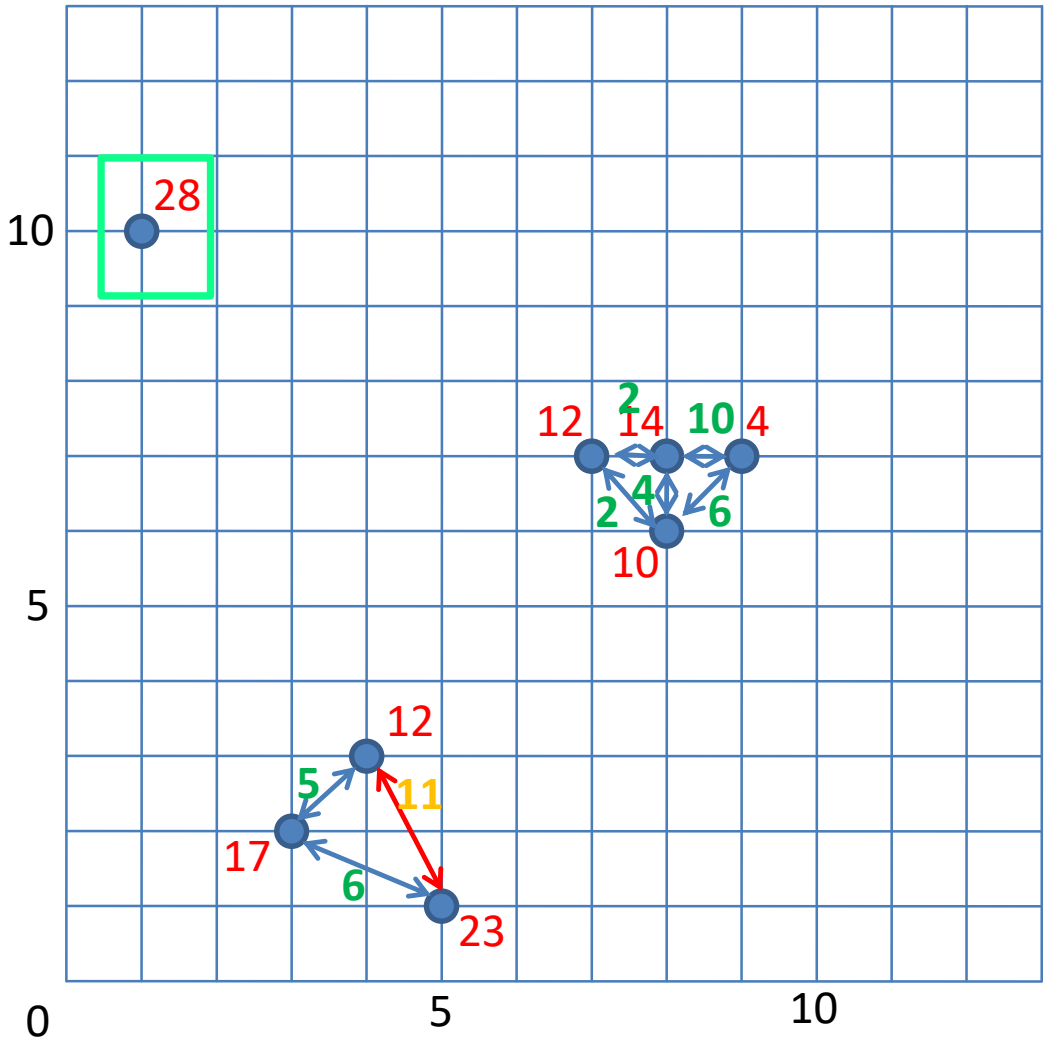


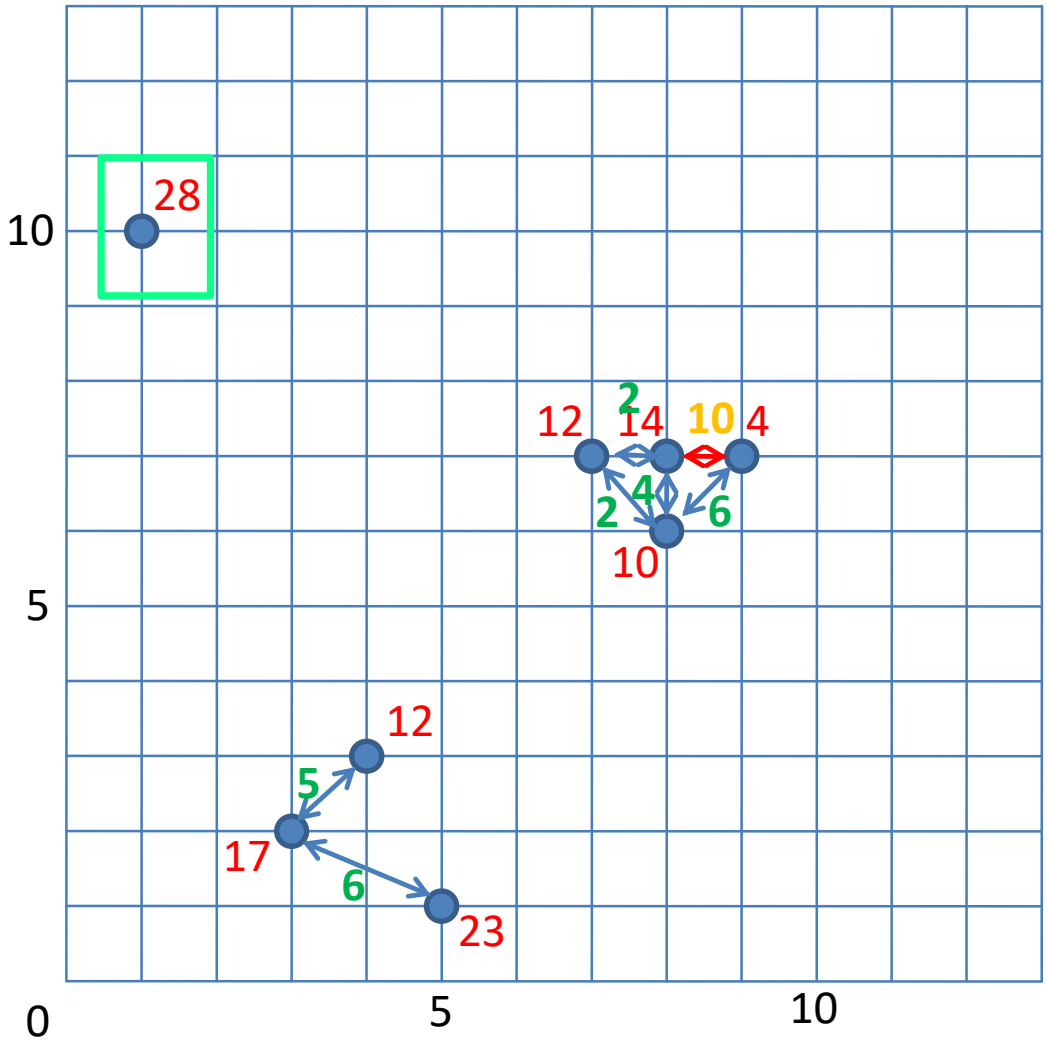

10
5
0
5
10
28
14
16
16
12
14
2
10
4
4
2
6
10
12
5
11
17
6
23

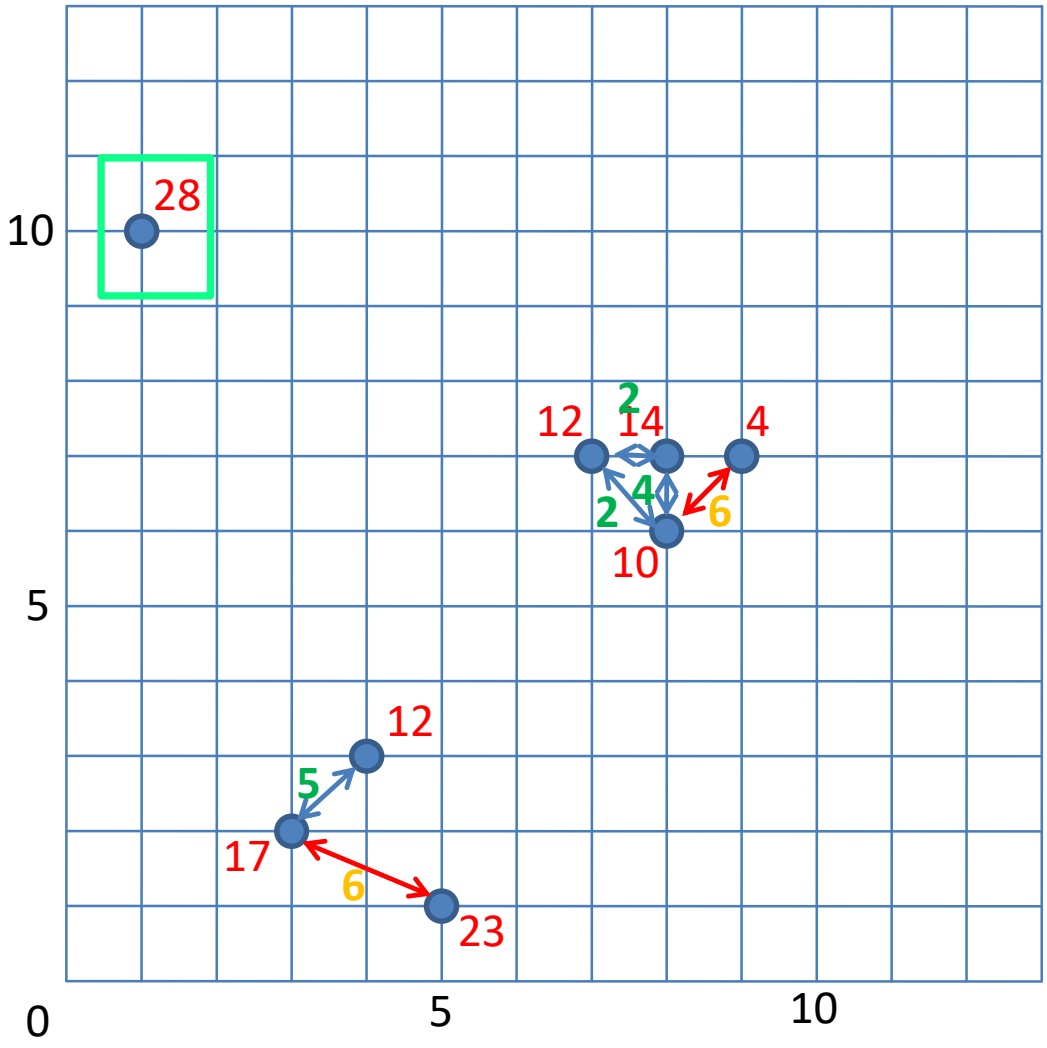

28
14
12
14
2
10
4
2
4
6
10
12
5
11
17
6
23
10
5
0
5
10


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


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

The three strongest outliers are determined