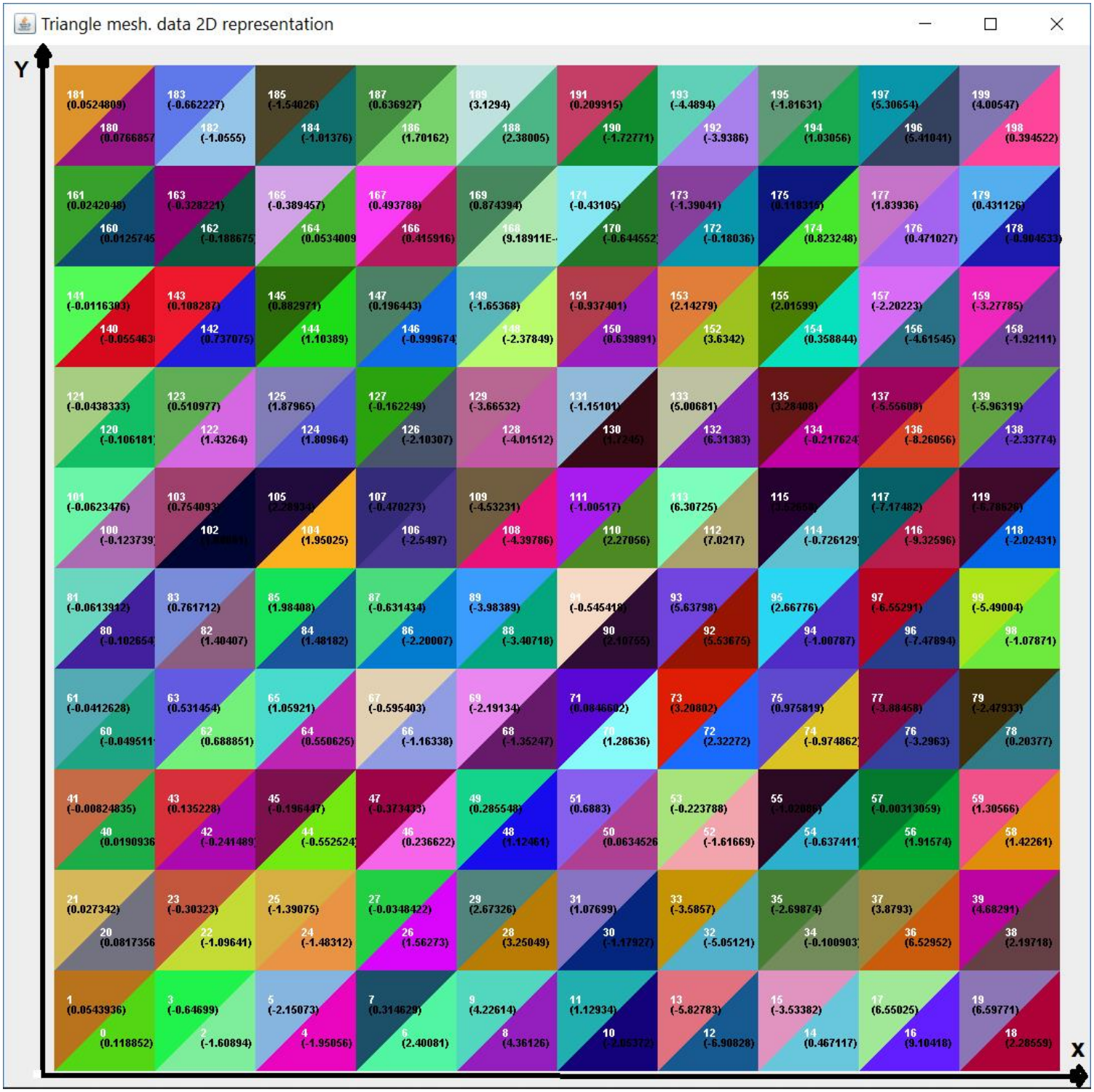

Nachbarelemente:

```
int[] x1 = { 1, 0, 3, 2, 5, 4, 7, 6, 9, 8, 11, 10, 13, 12, 15, 14, 17, 16, 19, 18 };
int[] x2 = { 21, 20, 23, 22, 25, 24, 27, 26, 29, 28, 31, 30, 33, 32, 35, 34, 37, 36, 39, 38 };
int[] x3 = { 41, 40, 43, 42, 45, 44, 47, 46, 49, 48, 51, 50, 53, 52, 55, 54, 57, 56, 59, 58 };
int[] x4 = { 61, 60, 63, 62, 65, 64, 67, 66, 69, 68, 71, 70, 73, 72, 75, 74, 77, 76, 79, 78 };
int[] x5 = { 81, 80, 83, 82, 85, 84, 87, 86, 89, 88, 91, 90, 93, 92, 95, 94, 97, 96, 99, 98 };
int[] x6 = { 101, 100, 103, 102, 105, 104, 107, 106, 109, 108, 111, 110, 113, 112, 115, 114, 117, 116, 119, 118 };
int[] x7 = { 121, 120, 123, 122, 125, 124, 127, 126, 129, 128, 131, 130, 133, 132, 135, 134, 137, 136, 139, 138 };
int[] x8 = { 141, 140, 143, 142, 145, 144, 147, 146, 149, 148, 151, 150, 153, 152, 155, 154, 157, 156, 159, 158 };
int[] x9 = { 161, 160, 163, 162, 165, 164, 167, 166, 169, 168, 171, 170, 173, 172, 175, 174, 177, 176, 179, 178 };
int[] x10 = { 181, 180, 183, 182, 185, 184, 187, 186, 189, 188, 191, 190, 193, 192, 195, 194, 197, 196, 199, 198 };
int[] y1 = { 0, 1, 20, 21, 40, 41, 60, 61, 80, 81, 100, 101, 120, 121, 140, 141, 160, 161, 180, 181 };
int[] y2 = { 2, 3, 22, 23, 42, 43, 62, 63, 82, 83, 102, 103, 122, 123, 142, 143, 162, 163, 182, 183 };
int[] y3 = { 4, 5, 24, 25, 44, 45, 64, 65, 84, 85, 104, 105, 124, 125, 144, 145, 164, 165, 184, 185 };
int[] y4 = { 6, 7, 26, 27, 46, 47, 66, 67, 86, 87, 106, 107, 126, 127, 146, 147, 166, 167, 186, 187 };
int[] y5 = { 8, 9, 28, 29, 48, 49, 68, 69, 88, 89, 108, 109, 128, 129, 148, 149, 168, 169, 188, 189 };
int[] y6 = { 10, 11, 30, 31, 50, 51, 70, 71, 90, 91, 110, 111, 130, 131, 150, 151, 170, 171, 190, 191 };
int[] y7 = { 12, 13, 32, 33, 52, 53, 72, 73, 92, 93, 112, 113, 132, 133, 152, 153, 172, 173, 192, 193 };
int[] y8 = { 14, 15, 34, 35, 54, 55, 74, 75, 94, 95, 114, 115, 134, 135, 154, 155, 174, 175, 194, 195 };
int[] y9 = { 16, 17, 36, 37, 56, 57, 76, 77, 96, 97, 116, 117, 136, 137, 156, 157, 176, 177, 196, 197 };
int[] y10 = { 18, 19, 38, 39, 58, 59, 78, 79, 98, 99, 118, 119, 138, 139, 158, 159, 178, 179, 198, 199 };
```

diese Eingaben wurden mit der main Funktion in der Datei „`GenerateNeighbors`“ erzeugt.
Um die mit Nodes zu berechnen, sind Antworten auf folgende Testdatenfragen notwendig:

1. Sind Dreiecks immer rechteckig?
2. Ist eine Dreiecks Seite immer parallel zu Achse „X“ und andere zu Achse „Y“?
3. Sind die Nachbar Dreiecks Seiten nicht einmal 1 Nanometer voneinander entfernt?
4. Ist das überhaupt richtige Vorgehensweise?

Mit der Funktion „`calculateHotspot`“ ermitteln wir für jede Nachbarliste Hotspots.

```java
/**
 * Calculates hotspots for given neighbors element list
 * @param list - neighbors elements
 * @return calculated hotspots
 */
private Map<Integer, TriangleElement> calculateHotspot(List<TriangleElement> list) {
    Map<Integer, TriangleElement> result = new HashMap<Integer, TriangleElement>();
    TriangleElement last = null;
    TriangleElement secondToLast = null;
    for (TriangleElement el : list) {
        if (secondToLast == null) {
            secondToLast = last;
            last = el;
            continue;
        }
        if (secondToLast.getValue() < last.getValue() && (last.getValue() > el.getValue())) {
            result.put(last.getId(), last);
        }
        secondToLast = last;
        last = el;
    }
    return result;
}
```

Ergebnis:

```
X1) 0, 16, 8
X2) 20, 36, 28
X3) 48, 56, 43, 45
X4) 65, 73
X5) 85, 93
X6) 112, 105
X7) 132, 125
X8) 144, 152
X9) 177, 162, 167, 169
X10) 180, 196, 189
Y1) 81, 20, 180, 101, 121, 61
Y2) 82, 3, 102, 122, 62, 142
Y3) 85, 105, 125
Y4) 67, 87, 26, 186, 107, 46, 127
Y5)
Y6) 130, 70, 150, 90, 11, 171, 110, 31
Y7) 112, 132
Y8) 34, 194, 115, 54, 135, 75, 155, 95
Y9) 97, 196, 117
Y10) 98, 19, 179, 118, 39, 138, 158


Calculated HotSpots(Element Id's) before filtering
0, 3, 8, 11, 16, 19, 20, 26, 28, 31, 34, 36, 39, 43, 45, 46, 48, 54, 56, 61, 62, 65, 67, 70, 73, 75,
81, 82, 85, 87, 90, 93, 95, 97, 98, 101, 102, 105, 107, 110, 112, 115, 117, 118, 121, 122, 125, 127,
130, 132, 135, 138, 142, 144, 150, 152, 155, 158, 162, 167, 169, 171, 177, 179, 180, 186, 189, 194,
196
```

```java
/**
 * Filtering same hotspots.
 *
 * the first two hotspots are not the elements with the two first maximum
 * values because the element where the second maximum value is achieved
 * will probably be the neighbor element of the first one. In this case it
 * would belong to the first hotspot from the engineering point of view.
 */
public void filterSameHotspots() {
    int[][] neighborsElementIdsLists = neighborsIds();
    for (int[] neighborsElementIds : neighborsElementIdsLists) {
        TriangleElement last = null;
        for (int id : neighborsElementIds) {
            TriangleElement newLast = hotSpots.get(id);
            // same hotspot
            if ((last != null) && (newLast != null)) {
                if (newLast.getValue() > last.getValue()) {
                    hotSpots.remove(last.getId());
                } else {
                    hotSpots.remove(newLast.getId());
                    continue;
                }
            }
            last = newLast;
        }
    }
}
```

```
Calculated HotSpots(Element Id's) after filtering
0, 8, 16, 20, 26, 28, 34, 36, 43, 45, 46, 48, 54, 56, 61, 65, 67, 73, 75, 81, 85, 87, 95, 97, 98,
101, 105, 107, 110, 112, 117, 118, 121, 125, 127, 130, 132, 138, 142, 150, 152, 158, 162, 167, 169,
171, 179, 180, 189, 194, 196
```

Calculated HotSpots(Element Id's) after filtering sorted by value
16(9.10418), 112(7.0217), 36(6.52952), 132(6.31383), 196(5.41041), 8(4.36126), 152(3.6342), 28(3.25049), 73(3.20802), 189(3.1294), 95(2.66776), 105(2.28934), 110(2.27056), 85(1.98408), 56(1.91574), 125(1.87965), 130(1.7245), 26(1.56273), 48(1.12461), 65(1.05921), 194(1.03056), 75(0.975819), 169(0.874394), 142(0.737075), 150(0.639891), 167(0.493788), 179(0.431126), 46(0.236622), 43(0.135228), 0(0.118852), 20(0.0817356), 180(0.0766857), 61(-0.0412628), 121(-0.0438333), 81(-0.0613912), 101(-0.0623476), 34(-0.100903), 127(-0.162249), 162(-0.188675), 45(-0.196447), 171(-0.43105), 107(-0.470273), 67(-0.595403), 87(-0.631434), 54(-0.637411), 98(-1.07871), 158(-1.92111), 118(-2.02431), 138(-2.33774), 97(-6.55291), 117(-7.17482)

Bei nachbar-Hotspots mit Werten zum Beispiel –9,2363 und –7,27336 wird dann –7,27336 für Hotspot übernommen....
bei 9,2363 und 7,27336 –> 9,2363

P.S.: Grafik bekommt man wenn man de.homedev.mesh.Triangle2DFrameMain.java startet
und Berechnungen -> de.homedev.mesh.CalculateMain.java
Aufgabenbeschreibung, alle Eingabe-/Ausgabedateien und Dokumentation befinden sich im Verzeichnis <project>\etc