Junioraufgabe 1: Quadratisch, praktisch, grün

Team-ID: 00005

Team: BugBusters

Bearbeiter/-innen dieser Aufgabe:

Jonathan Salomo (Teilnahme-ID: 73504)

18. November 2024

Inhaltsverzeichnis

Lösungsidee

Definition von „so quadratisch wie möglich“

Ich habe „so quadratisch wie möglich“ so definiert, dass die Differenz der Kantenlängen x und y der Kleingärten so gering wie möglich sein sollte.

Wenn wir die Aufgabenstellung betrachten fällt auf, dass diese sich in zwei Schritte unterteilen lässt. Der erste hiervon ist die Findung einer Aufteilung in quadratische Kleingärten, welche für maximal so viele Kleingärten sorgt:

$$Interessenten + (Interessenten \div 100 \cdot 10)$$

Wobei nur der ganzzahlige Anteil der Zahl von Interesse ist, da es keine halben Interessenten geben kann.

Im zweiten Schritt nehmen wir dann diese Aufteilung und verändern sie so, dass es genug Kleingerten gibt. Hierbei sorgen wir dafür, dass sich die Form der Kleingärten sich nie mehr als unbedingt nötig vom perfekten Quadrat entfernt.

Möglichst gute quadratische Aufteilung finden

Um eine möglichst gute Quadratische Aufteilung zu finden, will ich als Erstes alle ganzzahligen möglichen gemeinsamen Teiler der Seitenlängen bestimmen, da x und y bei jeder Aufteilung, die aus perfekten Quadraten besteht, gleich und ein Teiler der jeweiligen Seitenlänge sein muss. Somit kann diese Zahl nur ein gemeinsamer Teiler der Seitenlängen sein.

Um all diese Teiler zu finden, wende ich zuerst den euklidischen Algorithmus auf die zwei Seitenlängen an. Dieser berechnet den größten gemeinsamen Teiler iterativ.

Nun berechne ich alle Teiler dieser Zahl, da diese alle gemeinsamen Teiler der zwei ursprünglichen Zahlen sind.

Mein Teiler für eine möglichst gute quadratische Aufteilung ist nun entweder der, der eine perfekte Lösung ist, also zu einer Aufteilung mit bis zu 10% mehr Kleingärten führt, oder der kleinsten, der für nicht zu viele Kleingärten für alle Interessenten sorgt.

Anpassung der Kleingärten an die Anzahl an Interessenten

Wenn es keine perfekt Lösung gibt, schaue ich nun, ob mehr oder weniger Kleingärten benötigt werden. Daraufhin passe ich die Seitenlängen in möglichst kleinen Schritten an, bis die Anzahl der Kleingärten die vorgegebene Bedingung erfüllt.

Hierbei gehe ich so vor, dass ich die längere der beiden Seiten der Parzellen verkleinere, wenn ich die Menge an Kleingärten reduzieren will und die kürzere vergrößere wenn ich die Menge an Kleingärten erhöhen will. Ich erhöhe und reduziere die Längen der Kleingärten immer genau so, dass einer mehr oder weniger in der jeweiligen Länge auf das Grundstück passt.

$$x_{neu} = x_{alt} + [x_{alt} \div ((gx \div x_{alt}) + 1)]$$

Nun verhindere ich immer, dass nach einem Schritt, bei dem x erhört wurde, x gleich wieder reduziert werden kann, da das Programm sonst niemals terminieren würde.

Dieses Vorgehen kann nun aber dafür sorgen, dass das Programm erst x erhöht, dann y reduziert und direkt danach x reduziert und anschließend y erhöht, was wieder dafür sorgt, dass das Programm nicht terminiert.

Nun kann man aber feststellen, dass es niemals sinnvoll ist, x oder y erst zu reduzieren, nur um es danach wieder zu erhöhen, da wir ja immer in den kleinstmöglichen Schritten arbeiten. Deshalb mache ich dieses Verhalten unmöglich.

Dies erreiche ich dadurch, dass ich erst nach dem Kriterium der Länge von x oder y versuche, diese zu verändern, und wenn das zu keinem Ergebnis führt anschließend überprüfe, auf welcher der zwei Seiten mehr Kleingärten in das Grundstück passen, und verändere diese als erste.

Umsetzung

Mein Programm habe ich in Java geschrieben und dabei meine eigene Standart-Klasse zum Einlesen der Datei verwendet.

Zur späteren Verwendung berechne ich als Erstes die obere Grenze der Interessenten. Alle Kommazahlen speichere ich, als double, also mit 64 Bit, um eine hohe Genauigkeit zu erreichen. Diese Speicherung gibt mir eine sehr hohe Genauigkeit, jedoch kann es am Ende durch den iterativen Ansatz einen kleinen Fehler verursachen, der sich daran zeigt, dass die Kleingärten nicht mehr perfekt auf das Grundstück passen. Das Verhalten ist mit der Verwendung des Dezimalsystems auch nicht zu unterbinden, da Perioden und irrationale Zahlen sich nie exakt darstellen lassen. Jedoch entsteht dieser Fehler erst bei so großen Zahlen und ist so klein, dass die Vermessung von topographischen Punkten nach Genauigkeitsklassen in Anlehnung an DIN 18710-1, welche nur eine Genauigkeit von $1{,}5\text{cm} < sX, sY \leq 5{,}0\text{cm}$ fordert, deutlich ungenauer ist. Wenn man die höchsten Ungenauigkeiten der einzelnen Seitenlängen betrachtet, die bei verschiedensten Beispielen auftreten, übersteigen diese, wenn man sie auf die einzelnen Grundstücke runter rechnet, nie diese Angabe. Aus diesem Grund ist es in dieser Anwendung nicht nötig, genauer zu sein, und ich habe mich mit dieser Genauigkeit zufrieden gegeben.

Danach rufe ich in einer anderen Klasse den euklidischen Algorithmus auf die zwei Eingaben auf.

Euklidischer Algorithmus

Hier suche ich mir zu Umsetzung des Algorithmus erst durch eine einfach if-Schleife die größere der beiden zuvor übergebenen Zahlen a. Dann verwende ich eine while-schleife für den eigentlichen Algorithmus, die solange läuft, bis die am Anfang kleinere Zahl $b = 0$ ist. In dieser Schleife setze ich b auf das modulo von $a \bmod b$ und a dann auf den Wert von b am Anfang des Schleifendurchlaufs.

Finden der restlichen gemeinsamen Teiler und des sinnvollsten

Um alle weiteren gemeinsamen Teiler zu finden, schaue ich mir einmal alle Zahlen von 1 bis zur Wurzel des ggT an und speichere diese, so wie den ggT durch diese Zahl, als einen gemeinsamen Teiler. Dies ist effizienter als dies für alle Zahlen von 1 bis zum ggT zu prüfen.

Um die gemeinsamen Teiler zu speichern verwende ich eine ArrayList, da diese mit ermöglicht, keine feste Größe zu haben und stattdessen dynamisch Speicher zuweist. Für die spätere Verarbeitung übertrage ich die Werte aus dieser zur Rückgabe in ein einfaches Array.

Um den sinnvollsten gemeinsamen Teiler nach dem oben erklärten Konzept festzustellen, prüfe ich nun für alle gemeinsamen Teiler, beginnend mit dem größten, mithilfe einer if-Abfrage, ob sie entweder für eine perfekte Lösung sorgen oder ob der nächst kleinere Teiler dafür sorgen würde, dass nicht mehr genug Kleingärten vorhanden wären, um die Interessenten zufriedenzustellen. Sobald diese Abfrage anschlägt speichere ich den aktuellen Teiler für meine vorläufige Aufteilung.

Damit auch der letzte Teiler im Array ohne eine Exception auszulösen aufgerufen werden kann, wird, wann auch immer die if-Abfrage eine Exception auslöst, der aktuelle Wert verwendet. Dies ist sinnvoll, denn wenn es keinen weiteren Teiler gibt kann es auch keinen besseren Teiler geben.

Anpassung zur Lösung

Anschließend passe ich die Seitenlängen der Kleingärten durch eine while-Schleife so lange an, bis die Anzahl der Kleingärten im gewünschten Bereich liegt.

Innerhalb dieser while-Schleife befinden sich zur Umsetzung der ob erklärten Taktik einige verschachtelte if-Abfragen. Ich beginne mit der Abfrage, ob ich die Menge an Kleingärten erhöhen oder senken muss. Danach folgt eine if- elseif-Abfrage mit vier Optionen. Die ersten zwei schauen jeweils, welche Seite der Kleingärten größer beziehungsweise kleiner ist und verändern die passende. Dies wird durch weitere Bedingungen aber nur erlaubt, wenn die beim ersten Durchlauf gespeicherte Richtung verwenden wird, wie ich in der Lösungsidee beschreiben habe. Die nächsten zwei elseif-Abfragen prüfen, welche Grundstückseite größer ist und stellen so sicher, dass immer etwas verändert wird, da das Programm sonst nicht terminieren würde.

Dies wird durch die while-Schleife so lange ausgeführt, bis eine Lösung gefunden wurde. Diese wird dann ausgegeben.

Beispiele

garten0.txt

---

grundstückx: 42m

grundstücky: 66m

kleingartenLängex (vor der weiteren Anpassung): 6.0m

kleingartenLängey (vor der weiteren Anpassung): 6.0m

Gärten (basierend auf den Längen vor der weiteren Anpassung): 77.0

Interesenten: 23

Interesenten_upper: 25.0

Jeder Kleingarten hat eine Breite von 10.5m und eine Länge von 11.0m.

Dies sorgt für eine Abweichung der Kanten vom perfekten Quadrat von: -0.5m (-4.545454545454546 Prozent)

Insgesamt gibt es 24.0 Kleingärten.

Dies sind 1.0 mehr als es Intressenten gibt.(4,34782 Prozent)

---

garten1.txt

---

grundstückx: 15m

grundstücky: 12m

kleingartenLängex (vor der weiteren Anpassung): 3.0m

kleingartenLängey (vor der weiteren Anpassung): 3.0m

Gärten (basierend auf den Längen vor der weiteren Anpassung): 20.0

Interesenten: 19

Interesenten_upper: 20.0

Jeder Kleingarten hat eine Breite von 3.0m und eine Länge von 3.0m.

Dies sorgt für eine Abweichung der Kanten vom perfekten Quadrat von: 0.0m (0.0 Prozent)

Insgesamt gibt es 20.0 Kleingärten.

Dies sind 1.0 mehr als es Interessenten gibt.(5,26316 Prozent)

---

garten2.txt

---

grundstückx: 55m

grundstücky: 77m

kleingartenLängex (vor der weiteren Anpassung): 11.0m

kleingartenLängey (vor der weiteren Anpassung): 11.0m

Gärten (basierend auf den Längen vor der weiteren Anpassung): 35.0

Interesenten: 36

Interesenten_upper: 39.0

Jeder Kleingarten hat eine Breite von 9.166666666666666m und eine Länge von 12.833333333333334m.

Dies sorgt für eine Abweichung der Kanten vom perfekten Quadrat von: -3.666666666666668m (-28.57142857142858 Prozent)

Insgesamt gibt es 36.0 Kleingärten.

Dies sind 0.0 mehr als es Interessenten gibt.(-0,00000 Prozent)

---

garten3.txt

---

grundstückx: 15m

grundstücky: 15m

kleingartenLängex (vor der weiteren Anpassung): 1.0m

kleingartenLängey (vor der weiteren Anpassung): 1.0m

Gärten (basierend auf den Längen vor der weiteren Anpassung): 225.0

Interesenten: 101

Interesenten_upper: 111.0

Jeder Kleingarten hat eine Breite von 1.3636363636363633m und eine Länge von 1.4999999999999996m.

Dies sorgt für eine Abweichung der Kanten vom perfekten Quadrat von: -0.13636363636363624m (-9.090909090909086 Prozent)

Insgesamt gibt es 110.0 Kleingärten.

Dies sind 9.0 mehr als es Interessenten gibt.(8,91089 Prozent)

Die Aufteilung der x Achse ist um 0,000000000000000242230m ungenau.

Es bleiben auf der X Achse also am Nach der Aufteilung der Kleingärten 0,000000000000000242230m an Land übrig.

Dies ist auf die Kleingärten runtergerechnet eine Messungenauigkeit von : 0,000000000000002202095cm

Die Aufteilung der y Achse ist um 0,000000000000000532907m ungenau.

Es bleiben auf der Y Achse also am Nach der Aufteilung der Kleingärten 0,000000000000000484461m an Land übrig.

Dies ist auf die Kleingärten runtergerechnet eine Messungenauigkeit von :
0,000000000000005329071cm

---

garten4.txt

---

grundstückx: 37m

grundstücky: 2000m

kleingartenLängex (vor der weiteren Anpassung): 1.0m

kleingartenLängey (vor der weiteren Anpassung): 1.0m

Gärten (basierend auf den Längen vor der weiteren Anpassung): 74000.0

Interesenten: 1200

Interesenten_upper: 1320.0

Jeder Kleingarten hat eine Breite von 9.249999999999975m und eine Länge von 6.666666666666588m.

Dies sorgt für eine Abweichung der Kanten vom perfekten Quadrat von: 2.583333333333387m (27.927927927928586 Prozent)

Insgesamt gibt es 1200.0 Kleingärten.

Dies sind 0.0 mehr als es Interessenten gibt.(0,00000 Prozent)

Die Aufteilung der x Achse ist um 0,0000000000009858780m ungenau.

Es bleiben auf der X Achse also am Nach der Aufteilung der Kleingärten 0,0000000000009858780m an Land übrig.

Dies ist auf die Kleingärten runtergerechnet eine Messungenauigkeit von : 0,000000000000246469511cm

Die Aufteilung der y Achse ist um 0,000000000002349527980m ungenau.

Es bleiben auf der Y Achse also am Nach der Aufteilung der Kleingärten 0,000000000003259970072m an Land übrig.

Dies ist auf die Kleingärten runtergerechnet eine Messungenauigkeit von : 0,000000000000783175993cm

---

garten5.txt

---

grundstückx: 365m

grundstücky: 937m

kleingartenLängex (vor der weiteren Anpassung): 1.0m

kleingartenLängey (vor der weiteren Anpassung): 1.0m

Gärten (basierend auf den Längen vor der weiteren Anpassung): 342005.0

Interesenten: 35000

Interesenten_upper: 38500.0

Jeder Kleingarten hat eine Breite von 2.9918032786885287m und eine Länge von 2.9746031746031827m.

Dies sorgt für eine Abweichung der Kanten vom perfekten Quadrat von: 0.017200104085346002m (0.5749075886060847 Prozent)

Insgesamt gibt es 38429.0 Kleingärten.

Dies sind 3429.0 mehr als es Interessenten gibt.(9,80000 Prozent)

Die Aufteilung der x Achse ist um 2,991803278688018400000m ungenau.

Es bleiben auf der X Achse also am Nach der Aufteilung der Kleingärten 2,991803278688018400000m an Land übrig.

Dies ist auf die Kleingärten runtergerechnet eine Messungenauigkeit von : 2,452297769416412000000cm

Die Aufteilung der y Achse ist um 2,974603174600646500000m ungenau.

Es bleiben auf der Y Achse also am Nach der Aufteilung der Kleingärten 2,991803278685978000000m an Land übrig.

Dies ist auf die Kleingärten runtergerechnet eine Messungenauigkeit von : 0,944318468127191900000cm

---

Quellcode

QuadratischPraktischGruen

```
public class QuadratischPraktischGruen {
    public static void main(String[] args) throws Exception {

        double anzahlInteresentenUpper = Math.floor(anzahlInteresenten + (anzahlInteresenten /
100.0f) * 10);

        EuklidischerAlgorithmus euklidischerAlgorithmus = new EuklidischerAlgorithmus();
        int[] kleinstegemeinsameTeilerGrundstück =
euklidischerAlgorithmus.gemeinsameTeilerFinden(grundstückx, grundstücky);

        for(int i = 0; i < kleinstegemeinsameTeilerGrundstück.length; i++){
            try {
                if (anzahlInteresentenUpper -
((grundstückx/kleinstegemeinsameTeilerGrundstück[i])*(grundstücky/
kleinstegemeinsameTeilerGrundstück[i])) < (anzahlInteresenten / 100.0f) * 10 ||
anzahlInteresentenUpper - ((grundstückx/kleinstegemeinsameTeilerGrundstück[i]+1)*(grundstücky/
kleinstegemeinsameTeilerGrundstück[i]+1)) < 0){
                    sinfollsterKleinstegemeinsamerTeilerGrundstück =
kleinstegemeinsameTeilerGrundstück[i];
                }
            } catch (Exception e) {
                sinfollsterKleinstegemeinsamerTeilerGrundstück =
kleinstegemeinsameTeilerGrundstück[i];
            }
        }

        double kleingartenLängex = sinfollsterKleinstegemeinsamerTeilerGrundstück;
        double kleingartenLängey = sinfollsterKleinstegemeinsamerTeilerGrundstück;

        while((anzahlInteresentenUpper -
((grundstückx/kleingartenLängex)*(grundstücky/kleingartenLängey)) < 0 || anzahlInteresenten -
((grundstückx/kleingartenLängex)*(grundstücky/kleingartenLängey)) > 0)) {
            if (anzahlInteresenten -
((grundstückx/kleingartenLängex)*(grundstücky/kleingartenLängey)) > 0) {
                if (kleingartenLängex >= kleingartenLängey && lastStep != 4 && xStep != 1|| lastStep
== 3) {
                    kleingartenLängex = kleingartenLängex -
(kleingartenLängex/((grundstückx/kleingartenLängex)+1));
                    lastStep = 1;
                    xStep = -1;
                } else if (kleingartenLängex <= kleingartenLängey && lastStep != 3 && yStep != 1||
lastStep == 4) {
```

```
                    kleingartenLängey = kleingartenLängey -
(kleingartenLängey/((grundstücky/kleingartenLängey)+1));
                    lastStep = 2;
                    yStep = -1;
                } else if (grundstückx/kleingartenLängex > grundstücky/kleingartenLängey){
                    kleingartenLängex = kleingartenLängex -
(kleingartenLängex/((grundstückx/kleingartenLängex)+1));
                    lastStep = 1;
                    xStep = -1;
                } else if (grundstückx/kleingartenLängex < grundstücky/kleingartenLängey){
                    kleingartenLängey = kleingartenLängey -
(kleingartenLängey/((grundstücky/kleingartenLängey)+1));
                    lastStep = 2;
                    yStep = -1;
                }
            } else if (anzahlInteresentenUpper -
((grundstückx/kleingartenLängex)*(grundstücky/kleingartenLängey)) < 0) {
                if (kleingartenLängex >= kleingartenLängey && lastStep != 2 && yStep != -1||
lastStep == 1) {
                    kleingartenLängey = kleingartenLängey +
(kleingartenLängey/((grundstücky/kleingartenLängey)-1));
                    lastStep = 3;
                    yStep = 1;
                } else if (kleingartenLängex <= kleingartenLängey && lastStep != 1 && xStep != -1||
lastStep == 2) {
                    kleingartenLängex = kleingartenLängex +
(kleingartenLängex/((grundstückx/kleingartenLängex)-1));
                    lastStep = 4;
                    xStep = 1;
                } else if (grundstückx/kleingartenLängex > grundstücky/kleingartenLängey){
                    kleingartenLängex = kleingartenLängex +
(kleingartenLängex/((grundstückx/kleingartenLängex)-1));
                    lastStep = 4;
                    xStep = 1;
                } else if (grundstückx/kleingartenLängex < grundstücky/kleingartenLängey){
                    kleingartenLängey = kleingartenLängey +
(kleingartenLängey/((grundstücky/kleingartenLängey)-1));
                    lastStep = 3;
                    yStep = 1;
                }
            }
        }

        System.out.println("Jeder Kleingarten hat eine Breite von " + kleingartenLängex +"m und eine
Länge von " + kleingartenLängey + "m.");
```

```
        System.out.println("Dies sorgt für eine Abweichung der Kanten vom perfekten Quadrat von: " +
(kleingartenLängex - kleingartenLängey) + "m (" + abweichung(kleingartenLängex, kleingartenLängey) +
" Prozent)");
        System.out.println("Insgesamt gibt es " +
Math.floor((grundstückx/kleingartenLängex)*(grundstücky/kleingartenLängey)) + " Kleingärten.");
        System.out.println("Dies sind " +
Math.floor((((grundstückx/kleingartenLängex)*(grundstücky/kleingartenLängey))- anzahlInteresenten))
+ " mehr als es Interessenten gibt.(" + String.format("%.5f",
((grundstückx/kleingartenLängex)*(grundstücky/kleingartenLängey)/(anzahlInteresenten/100.0f))-100) +
" Prozent)");

        kontröler(grundstückx, grundstücky, kleingartenLängex, kleingartenLängey);
    }

    private static double abweichung(double kleingartenLängex, double kleingartenLängey){
        double proAbweichung = 0;

        proAbweichung = ((kleingartenLängex - kleingartenLängey) / Math.max(kleingartenLängey,
kleingartenLängex)) * 100;

        return proAbweichung;
    }
    private static void kontröler(int grundstückx, int grundstücky, double kleingartenLängex, double
kleingartenLängey){
            if ((grundstückx/kleingartenLängex) - (int) (grundstückx/kleingartenLängex) > 0){
                System.err.println("Die Aufteilung der x Achse ist um " + String.format("%.20f",
(((grundstückx / kleingartenLängex) - (int) (grundstückx / kleingartenLängex))*kleingartenLängex)) +
"m ungenau.");
                System.err.println("Es bleiben auf der X Achse also am Nach der Aufteilung der
Kleingärten " + String.format("%.20f", (((grundstückx / kleingartenLängex) - (int) (grundstückx /
kleingartenLängex))*kleingartenLängex)) + "m an Land übrig.");
                System.err.println("Dies ist auf die Kleingärten runtergerechnet eine
Messungenauigkeit von : " +
                String.format("%.20f",
                ((((grundstückx / kleingartenLängex) - (int) (grundstückx / kleingartenLängex)) *
kleingartenLängex) /
                (grundstückx / kleingartenLängex)) * 100) + "cm");
            }
            if ((grundstücky/kleingartenLängey) - (int) (grundstücky/kleingartenLängey) > 0){
                System.err.println("Die Aufteilung der y Achse ist um " + String.format("%.20f",
(((grundstücky / kleingartenLängey) - (int) (grundstücky / kleingartenLängey))*kleingartenLängey)) +
"m ungenau.");
                System.err.println("Es bleiben auf der Y Achse also am Nach der Aufteilung der
Kleingärten " + String.format("%.20f", (((grundstücky / kleingartenLängey) - (int) (grundstücky /
kleingartenLängey))*kleingartenLängex)) + "m an Land übrig.");
                System.err.println("Dies ist auf die Kleingärten runtergerechnet eine
Messungenauigkeit von : " +
```

```
                String.format("%.20f",
                ((((grundstücky / kleingartenLängey) - (int) (grundstücky / kleingartenLängey)) *
kleingartenLängey) /
                (grundstücky / kleingartenLängey)) * 100) + "cm");
            }
            if ((grundstücky/kleingartenLängey) - (int) (grundstücky/kleingartenLängey) > 0 ||
(grundstückx/kleingartenLängex) - (int) (grundstückx/kleingartenLängex) > 0){
            }
    }
}
```

EuklidischerAlgorythmus

```
public class EuklidischerAlgorithmus {

    private int euklidischerAlgorithmus(int zahly ,int zahlx){
        int ggT = 0;
        int a = 0;
        int b = 0;

        if (zahlx < zahly){
            a = zahly;
            b = zahlx;
        } else{
            a = zahlx;
            b = zahly;
        }

        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        ggT = a;
        return ggT;
    }

    public int[] gemeinsameTeilerFinden(int zahly, int zahlx) {
        ArrayList<Integer> teiler = new ArrayList<>();

        int ggT = this.euklidischerAlgorithmus(zahly, zahlx);

        for (int i = 1; i <= Math.sqrt(ggT); i++) {
            if (ggT % i == 0) {
                teiler.add(i);
```

```
                if (i != ggT / i) {
                    teiler.add(ggT / i);
                }
            }
        }

        Arrays.sort(returnTeiler);

        return returnTeiler;
    }
}
```