

Beispiellösung: Quadratisch, praktisch, grün

Team-ID: ?????

Team-Name: Team-Name

Bearbeiter/-innen dieser Aufgabe:
Vor- und Nachnamen

Inhaltsverzeichnis

1	Lösungsidee	1
2	Umsetzung	2
3	Werkzeuge	2
4	Beispiele	2
5	Quellcode	2

1 Lösungsidee

In dieser Aufgabe wird ein rechteckiges Grundstück und eine Anzahl von Interessenten an einer Kleingartenanlage gegeben. Die Anzahl der Interessenten nennen wir hier k . Es soll eine Aufteilung des Grundstücks berechnet werden, in welcher gleich große und möglichst quadratische Kleingärten entstehen. Dabei muss die Anzahl der Kleingärten mindestens so groß sein wie k und maximal um 10% größer, also $1,1k$.

Wenn Kleingärten so quadratisch wie möglich sein sollen, muss zuerst festgelegt werden, was das heißt. Ein Rechteck ist quadratisch, wenn Höhe und Breite gleich lang sind. Dementsprechend definieren wir möglichst quadratisch so, dass die Differenz von Höhe und Breite eines Rechtecks möglichst klein sein soll. Je größer der Unterschied zwischen den beiden Seitenlängen ist, desto größer ist deren Differenz. Für ein perfektes Quadrat ist diese Differenz 0. Wichtig ist, dass der Betrag von der Subtraktion genommen wird, damit keine negativen Werte entstehen.

Mit diesen Überlegungen kann eine Kostenfunktion f definiert werden, welche die beiden Seitenlängen h und b eines Rechtecks als Parameter bekommt und die Differenz zurückgibt. $f(h, b) = |h - b|$. Die Kosten sollen bei der gesuchten Aufteilung in Kleingärten möglichst klein sein.

Um die beste Aufteilung zu finden, kann man alle möglichen Aufteilungen durchgehen und deren Kosten vergleichen. Dies ist ein Brute-Force-Ansatz über alle möglichen Gitter. Durch die Beschränkungen der Anzahl der Kleingärten wissen wir, dass es maximal $1,1k$ Zeilen geben darf. Sonst wären es alleine durch die Aufteilung in Zeilen zu viele Gärten. Das Gleiche gilt für die Anzahl der Spalten. Somit können alle möglichen Kombinationen an Zeilen- und Spaltenanzahl durchgegangen werden. Bei jeder Möglichkeit wird geprüft, ob die Anzahl der Kleingärten die Beschränkungen einhält und die Kostenfunktion wird mit der bisher besten Aufteilung verglichen.

2 Umsetzung

Die Lösungsidee wird in Python umgesetzt, Ein- und Ausgabe erfolgen über die Konsole.

Die Kostenfunktion f wird als Python-Funktion **abweichungQuadrat** implementiert.

Der Brute-Force-Ansatz wird durch zwei verschachtelte Schleifen umgesetzt, welche über die Anzahl der Zeilen bzw. die Anzahl der Spalten laufen. Für jede Aufteilung wird geprüft, ob die Rechtecke geringere Kosten haben als die bisherige beste Aufteilung. Eine Aufteilung des Grundstücks wird durch die Zeilen- und Spaltenanzahl gespeichert.

3 Werkzeuge

Da der Bereich der ganzen Zahlen mit der Beschränkung von $1, 1k$ verlassen wird, wird das Modul **math** der Python-Standardbibliothek zum Abrunden verwendet.

4 Beispiele

garten0.txt	4 Zeilen, 6 Spalten, 24 Kleingärten
garten1.txt	5 Zeilen, 4 Spalten, 20 Kleingärten
garten2.txt	6 Zeilen, 6 Spalten, 36 Kleingärten
garten3.txt	10 Zeilen, 11 Spalten, 110 Kleingärten
garten4.txt	5 Zeilen, 264 Spalten, 1320 Kleingärten
garten5.txt	120 Zeilen, 308 Spalten, 36960 Kleingärten

5 Quellcode

```
1 import math

3 # Einlesen
k = int(input())
5 h = int(input())
b = int(input())

7
# Kostenfunktion berechnet Differenz
9 def abweichungQuadrat(hoehe, breite):
    return abs(hoehe-breite)

11
# Initialisierung
13 optimaleZeilenanzahl = -1
    optimaleSpaltenanzahl = -1
15 geringsteAbweichung = 1000000000

17 # Iteration ueber alle moeglichen Gitter
for zeilen in range(1, math.floor(k*1.1)+1):
19     for spalten in range(1, math.floor(k*1.1)+1):
        # Anzahl der Kleingarten pruefen
21         if spalten*zeilen>=k and spalten*zeilen<=k*1.1:
            # pruefen, ob Kleingarten quadratisch sind
23             if abweichungQuadrat(h/zeilen, b/spalten)<geringsteAbweichung:
                # optimale Aufteilung ueberschreiben
25                 optimaleZeilenanzahl = zeilen
                    optimaleSpaltenanzahl = spalten
27                 geringsteAbweichung = abweichungQuadrat(h/zeilen, b/spalten)

29 # Ausgeben
print(str(optimaleZeilenanzahl) + "␣Zeilen,␣" +
31       str(optimaleSpaltenanzahl) + "␣Spalten,␣" +
        str(optimaleZeilenanzahl*optimaleSpaltenanzahl) + "␣Kleingarten")
```

j1.py