Quadratisch, praktisch, grün

Team-ID: 00852

Team-Name: Team_Magdeburg_an_die_Macht

Bearbeiter/-innen dieser Aufgabe:
Erik Donath

18. November 2024

Inhaltsverzeichnis

1 Lösungsidee

Das Ziel ist eine Gartenfläche in möglichst quadratische Kleingärten zu unterteilen.

1.1 Definition von Quadratisch

Zu erst muss eine Definition des Begriffes Quadratisch beschrieben werden. Quadratisch ist das Verhältnis von der kleinen Seite zur großen Seite. Sollte dieses 1 sein, so ist das gegebene Rechteck zu 100% quadratisch. Der Mathematische Ausdruck lautet wie folgt:

$$s(b,h) = \frac{\min(b,h)}{\max(b,h)}$$

b und h beschreiben die Breite und Höhe des Rechtecks und sind immer positiv. Der Funktionswert von s kann zwischen 0 und 1 liegen.

1.2 Suchstrategie

Die Suche nach der optimalen Aufteilung ist die Suche nach dem maximalen Funktionswert von s. Hierfür suchen wir in einem Intervall von $I = [n; n \cdot 1.1]$. Dabei bezeichnet n die Anzahl an Interessenten. Es werden auch die Begriffe $n_{\min}$ und $n_{\max}$ für die Intervallgrenzen verwendet. Für jede mögliche Anzahl werden alle gültigen Kombinationen von Reihen und Spalten untersucht.

Der Ablauf sieht wie folgt aus:

1. Gehe durch das Intervall mit Schrittweite 1. Dies entspricht der aktuellen Anzahl der Kleingärten.
2. Gehe durch alle möglichen Anzahlen an Reihen von 1 bis zur aktuellen Kleingartenanzahl + 1, die Teiler dieser Anzahl sind.
3. Berechne für jede Reihe die Anzahl an möglichen Spalten.
4. Bestimme die Breite und Höhe eines einzelnen Kleingartens.
5. Merke dir die Höhe und Breite.
6. Nachdem das Intervall untersucht wurde, sortiere die gemerkten Kleingärten nach ihrer Quadratförmigkeit.
7. Das oberste Resultat der sortierten Liste ist das Ergebnis.

Der mathematische Ausdruck lautet:

$$\max_{n,r} \left\{ s\left(\frac{W}{c}, \frac{H}{r}\right) \mid n_{\min} \leq n \leq n_{\max}, c = \frac{n}{r} \right\}$$

Wobei: - n die Kleingartenanzahl ist - r die Anzahl der Reihen ist - c die Anzahl der Spalten ist - W die Gesamtbreite ist - H die Gesamthöhe ist - $n_{\min}$ und $n_{\max}$ die Intervallgrenzen sind

Die Formel beschreibt die Suche nach der Kombination von Kleingartenanzahl und Reihenanzahl, die die höchste Quadratförmigkeit ergibt, unter Berücksichtigung der Randbedingungen.

2 Umsetzung

Die Lösungsidee zur optimalen Aufteilung einer Gartenfläche wurde in der Python-Klasse 'GardenPlotter' wie folgt umgesetzt:

2.1 Berechnung der Quadratförmigkeit

Die Methode `calculate_squareness()` wurde implementiert, um die Ähnlichkeit eines Rechtecks (Kleingartens) zu einem perfekten Quadrat zu quantifizieren. Sie verwendet das Verhältnis der kürzeren zur längeren Seite, was einen Wert zwischen 0 und 1 ergibt.

2.2 Kernalgorithmus zur Findung der besten Aufteilung

Der Hauptalgorithmus ist in der Methode `find_best_division()` implementiert:

1. Es wird über alle möglichen Kleingartenanzahlen von `min_plots` bis `max_plots` iteriert.
2. Für jede Kleingartenanzahl werden alle möglichen Reihen-Spalten-Kombinationen durchlaufen.
3. Die Quadratförmigkeit jedes resultierenden Kleingartens wird berechnet.
4. Die Kombination mit der höchsten Quadratförmigkeit wird als beste Lösung gespeichert.

2.3 Effizienzoptimierungen

- Die Schleife prüft nur Teiler-Kombinationen, um unnötige Berechnungen zu vermeiden.
- Die Berechnung stoppt, sobald eine perfekte quadratische Aufteilung gefunden wird (Quadratförmigkeit = 1).

2.4 Datenstruktur für das Ergebnis

Das Ergebnis wird als Dictionary zurückgegeben, das alle relevanten Informationen zur besten Aufteilung enthält, einschließlich der Anzahl der Reihen und Spalten, der Kleingartenmaße und des Quadratförmigkeitswerts. Diese Implementierung ermöglicht eine effiziente und flexible Berechnung der optimalen Gartenaufteilung, die sich an verschiedene Gartengrößen und Teilnehmerzahlen anpassen lässt.

3 Beispiele

3.1 Beispiel 1: Gartenaufteilung aus garten0.txt

```
Lade Daten von: res/garten0.txt
Interessenten: 23
Höhe: 42
Breite: 66

Beste Aufteilung gefunden:
Anzahl der Kleingärten: 24 (4 Reihen x 6 Spalten)
Größe jedes Kleingartens: 11.00m x 10.50m
Quadratischkeit: 95.5% (100% wäre ein perfektes Quadrat)
```

3.2 Beispiel 2: Gartenaufteilung aus garten1.txt

```
Lade Daten von: res/garten1.txt
Interessenten: 19
Höhe: 15
Breite: 12

Beste Aufteilung gefunden:
Anzahl der Kleingärten: 20 (5 Reihen x 4 Spalten)
Größe jedes Kleingartens: 3.00m x 3.00m
Quadratischkeit: 100.0% (100% wäre ein perfektes Quadrat)
```

3.3 Beispiel 3: Gartenaufteilung aus garten2.txt

```
Lade Daten von: res/garten2.txt
Interessenten: 36
Höhe: 55
Breite: 77

Beste Aufteilung gefunden:
Anzahl der Kleingärten: 36 (6 Reihen x 6 Spalten)
Größe jedes Kleingartens: 12.83m x 9.17m
Quadratischkeit: 71.4% (100% wäre ein perfektes Quadrat)
```

3.4 Beispiel 4: Gartenaufteilung aus garten3.txt

```
Lade Daten von: res/garten3.txt
Interessenten: 101
Höhe: 15
Breite: 15

Beste Aufteilung gefunden:
Anzahl der Kleingärten: 110 (10 Reihen x 11 Spalten)
Größe jedes Kleingartens: 1.36m x 1.50m
Quadratischkeit: 90.9% (100% wäre ein perfektes Quadrat)
```

3.5 Beispiel 5: Gartenaufteilung aus garten4.txt

```
Lade Daten von: res/garten4.txt
Interessenten: 1200
Höhe: 37
Breite: 2000

Beste Aufteilung gefunden:
Anzahl der Kleingärten: 1320 (5 Reihen x 264 Spalten)
Größe jedes Kleingartens: 7.58m x 7.40m
Quadratischkeit: 97.7% (100% wäre ein perfektes Quadrat)
```

3.6 Beispiel 6: Gartenaufteilung aus garten5.txt

```
Lade Daten von: res/garten5.txt
Interessenten: 35000
Höhe: 365
Breite: 937

Beste Aufteilung gefunden:
Anzahl der Kleingärten: 36960 (120 Reihen x 308 Spalten)
Größe jedes Kleingartens: 3.04m x 3.04m
Quadratischkeit: 100.0% (100% wäre ein perfektes Quadrat)
```

4 Quellcode

```python
class GardenPlotter:
    def _init_(self, width, height, interested):
        self.total_width = width
        self.total_height = height
        self.min_plots = interested
        self.max_plots = int(interested * 1.1)

    def calculate_squareness(self, width, height):
        return min(width, height) / max(width, height)

    def find_best_division(self):
        best_squareness = 0
        best_division = None

        for plots in range(self.min_plots, self.max_plots + 1):
            for rows in range(1, plots + 1):
                if plots % rows == 0:
                    cols = plots // rows

                    plot_width = self.total_width / cols
                    plot_height = self.total_height / rows

                    squareness = self.calculate_squareness(plot_width, plot_height)

                    if squareness > best_squareness:
                        best_squareness = squareness
                        best_division = {
                            'rows': rows,
                            'cols': cols,
                            'total_plots': plots,
                            'plot_width': plot_width,
                            'plot_height': plot_height,
                            'squareness': squareness
                        }

        return best_division
```

4.1 Für `calculate_squareness(w,h)` gilt:

Zeitkomplexität: $O(1)$
Mathematischer Ausdruck: $s(w,h) = \frac{min(w,h)}{max(w,h)}$

4.2 Für `find_best_division()` gilt:

Zeitkomplexität: $O(n^2)$
Mathematischer Ausdruck:
1. Menge der möglichen Plotanzahlen:
$P = \{p \in \mathbb{N} | min\_plots \leq p \leq max\_plots\}$ 2. Für jedes $p \in P$, ist die Menge der möglichen Reihenzahlen:
$R_p = \{r \in \mathbb{N} | 1 \leq p \text{ und } pmodr = 0\}$
3. Berechne für jede Kombination (p,r) die Spantenzahl $c = \frac{p}{r}$ und die Plotdimensionen:
$w = \frac{total\_width}{c}, h = \frac{total\_height}{r}$
4. Berechne die Quadratförmigkeit für jede Kompination:
$s(w,h) = \frac{min(w,h)}{max(w,h)}$
5. Finde die optimale Aufteilung:
$best\_division = \max_{p,r} s(p,r) | p \in P, r \in R_p$