Aufgabe 1: Quadratisch, Praktisch, Grün

Team-ID: 00476

Team-Name: HochgradigTalentiertwenigKrips

Bearbeiter/-innen dieser Aufgabe:
Jurek Engelmann, Lennart Peters

November 19, 2024

Contents

1 Lösungsansatz

Der Lösungsansatz des Programms folgt einem optimierten Verfahren, das schrittweise die besten Parameter für die Aufteilung eines Grundstücks in Parzellen berechnet. Ziel ist es, das Grundstück so zu unterteilen, dass die Parzellen eine möglichst gleichmäßige Größe haben und die Anzahl der Parzellen der Anzahl der Interessenten entspricht, ohne eine maximale Obergrenze zu überschreiten. Die maximale Obergröße beträgt 10% mehr bei den Parzellen als es Interessenten gibt.

2 Umsetzung

Die Umsetzung des Programms basiert auf der Anwendung mathematischer Berechnungen und Iterationen, um die optimale Aufteilung eines Grundstücks in Parzellen zu ermitteln. Die Hauptlogik des Programms wurde in mehrere Funktionen unterteilt:

2.1 Berechnung der optimalen Parzellenaufteilung (Funktion `calculate`)

Die Funktion `calculate` berechnet die beste Aufteilung des Grundstücks in Parzellen und optimiert das Seitenverhältnis der Parzellen. Dabei wird die maximale Anzahl der Parzellen mit einem 10% Puffer berechnet:

$$\text{max\_plots} = \lceil 1.1 \times \text{people} \rceil$$

Es folgt eine Schleife, die verschiedene Kombinationen von Reihen (`rows`) und Spalten (`columns`) testet. Für jede Kombination wird die Gesamtzahl der Parzellen (`rows` × `columns`) sowie das Seitenverhältnis (Differenz zwischen Länge und Breite der Parzellen) berechnet:

$$\text{ratio} = \left| \frac{\text{cell\_length}}{\text{cell\_width}} - 1 \right|$$

Die Kombination mit dem besten Seitenverhältnis wird als optimale Lösung ausgewählt.

```
        # Schleife über die Anzahl der Reihen (von 1 bis max_people)
        for rows in range(1, people + 1):
                # Berechne die Anzahl der Spalten
                columns = math.ceil(people / rows)
                total_plots = rows * columns  # Gesamtzahl der Parzellen

            # Überspringe ungültige Aufteilungen
            if total_plots < people or total_plots > max_plots:
                continue

            # Berechne die Dimensionen jeder Parzelle
            cell_length = length / rows
            cell_width = width / columns
            ratio = abs(cell_length - cell_width)  # Differenz der Seitenlängen als Maß für das Verhält

            # Überprüfe, ob das Seitenverhältnis besser ist
            if ratio < best_ratio:
                best_ratio = ratio
                best_rows = rows
            best_columns = columns

```

Listing 1: Berechnung der optimalen Parzellenaufteilung

2.2 Einlesen der Eingabedaten (Funktion `read_file`)

Die Eingabedaten (Anzahl der Interessenten, Länge und Breite des Grundstücks) werden entweder aus einer Datei oder über Kommandozeilenargumente eingelesen. Wird eine Datei verwendet, so liest die Funktion `read_file` die Daten aus der Datei und gibt sie zurück:

$$\text{people}, \text{length}, \text{width}$$

```
        def read_file(task, verbose=False):
            try:
                with open(f"./Beispielaufgaben/garten{task}.txt", 'r') as file:
                    people = int(file.readline().strip())
                    length = float(file.readline().strip())
                    width = float(file.readline().strip())

            if verbose:
                print(f"Datei {task} erfolgreich gelesen mit folgenden Werten: ")
                print(f"Höhe: " + str(length))
                print(f"Breite: " + str(width))
                print(f"Personen: " + str(people))

            return people, length, width
        except Exception as e:
            print(f"Fehler beim Lesen der Datei {task}: {e}")
            return None, None, None

```

Listing 2: Einlesen der Eingabedaten

2.3 Verarbeitung von Kommandozeilen-Argumenten (Funktion `main`)

Die Hauptfunktion (`main`) nutzt das Modul `argparse`, um die Kommandozeilenargumente zu verarbeiten. Es können entweder eine Eingabedatei oder die Werte für Länge, Breite und Anzahl der Interessenten über die Kommandozeile angegeben werden:

```
parser.add_argument("-f", "-file", help="Dateipfad zur Eingabedatei")
```

Je nach den Eingabedaten wird entweder die Datei verarbeitet oder die Werte aus der Kommandozeile verwendet. Druch die Kommandozeile kan man mit `all` alle Datein durchlaufen lassen, oder mit einer direkten Zahl eine bestimmte Datei verwenden.

```
if __name__ == "__main__":
    import argparse

    parser = argparse.ArgumentParser(description="Wählen Sie eine Beispielaufgabe mit ihrer Num
    "oder wählen Sie 'all'.")
    parser.add_argument("-i", "--input", default="all", help="Dateipfad zur Eingabedatei")
    parser.add_argument("-v", "--verbose", action="store_true", help="Ausführliche Ausgaben anz

    args = parser.parse_args()

    if args.input not in ["all", "0", "1", "2", "3", "4", "5"]:
        raise ValueError("Invalid Input! Please choose one of the following: 0, 1, 2, 3, 4, 5, 

    main(args.input, args.verbose)


```

Listing 3: Verarbeitung der Kommandozeilen-Argumente

2.4 Ausgabe der Ergebnisse

Am Ende gibt das Programm die optimale Aufteilung der Parzellen aus. Die Ergebnisse umfassen die Anzahl der Reihen und Spalten, die Dimensionen jeder Parzelle sowie die Gesamtzahl der Parzellen:

Beste Aufteilung: `best_rows` Reihen und `best_columns` Spalten

Jede Parzelle ist cell_length m lang und cell_width m breit.

Die optimale Lösung wird anhand des besten Seitenverhältnisses ermittelt, wobei die Differenz zwischen der Länge und Breite jeder Parzelle minimiert wird.

```
print(f"Beste Aufteilung: {best_rows} Reihen und {best_columns} Spalten")
print(f"Jede Parzelle ist {cell_length:.2f} m lang und {cell_width:.2f} m breit.")
print(f"Gesamtzahl der Parzellen: {total_plots}")

```

Listing 4: Ausgabe der Ergebnisse

3 Beispielaufgaben

3.1 Beispiel 1 (BWinf: Garten_0 Datei)

Eingabe:

- `length` = 42 m
- `width` = 66 m
- `people` = 23

Ausgabe:

```
Beste Aufteilung: 4 Reihen und 6 Spalten
Jede Parzelle ist 10.50 m lang und 11.00 m breit.
Gesamtzahl der Parzellen: 24
```

3.2 Beispiel 2 (BWinf: Garten_1 Datei)

Eingabe:

- `length` = 15 m
- `width` = 12 m
- `people` = 19

Ausgabe:

```
Beste Aufteilung: 5 Reihen und 4 Spalten
Jede Parzelle ist 3.00 m lang und 3.00 m breit.
Gesamtzahl der Parzellen: 20
```

3.3 Beispiel 3 (BWinf: Garten_2 Datei)

Eingabe:

- `length` = 55 m
- `width` = 77 m
- `people` = 36

Ausgabe:

```
Beste Aufteilung: 5 Reihen und 8 Spalten
Jede Parzelle ist 11.00 m lang und 9.62 m breit.
Gesamtzahl der Parzellen: 40
```

3.4 Beispiel 4 (BWinf: Garten_3 Datei)

Eingabe:

- `length` = 15 m
- `width` = 15 m
- `people` = 101

Ausgabe:

```
Beste Aufteilung: 10 Reihen und 11 Spalten
Jede Parzelle ist 1.50 m lang und 1.36 m breit.
Gesamtzahl der Parzellen: 110
```

3.5 Beispiel 5 (BWinf: Garten_4 Datei)

Eingabe:

- `length` = 37 m
- `width` = 2000 m
- `people` = 1200

Code:

```
length = 37
width = 2000
people = 1200
calculate(length, width, people)
```

Ausgabe:

```
Beste Aufteilung: 5 Reihen und 240 Spalten
Jede Parzelle ist 7.40 m lang und 8.33 m breit.
Gesamtzahl der Parzellen: 1200
```

3.6 Beispiel 6 (BWinf: Garten_5 Datei)

Eingabe:

- `length` = 365 m
- `width` = 937 m
- `people` = 35000

Code:

```
length = 365
width = 937
people = 35000
calculate(length, width, people)
```

Ausgabe:

```
Beste Aufteilung: 117 Reihen und 300 Spalten
Jede Parzelle ist 3.12 m lang und 3.12 m breit.
Gesamtzahl der Parzellen: 35100
```

4 Quellcode

In diesem Abschnitt wird der Python-Quellcode zur Berechnung der optimalen Aufteilung eines Gartens auf Basis der angegebenen Länge, Breite und Anzahl der Personen dargestellt. Der Code berechnet die beste Anzahl von Reihen und Spalten, um die Parzellen gleichmäßig zu verteilen, und prüft, ob eine gültige Aufteilung gefunden wird. Im unteren Beispiel zu sehen ist die `solve.py` welche die komplette Berechnung beinhaltet.

```
def calculate_solution(length, width, people, verbose=False):
    # Berechne die maximale Anzahl von Parzellen
    max_plots = math.ceil(1.1 * people)

    best_rows = best_columns = 0
    best_ratio = float('inf')  # Startwert für das Seitenverhältnis

    # Schleife über die Anzahl der Reihen (von 1 bis max_people)
    for rows in range(1, people + 1):
        # Berechne die Anzahl der Spalten
        columns = math.ceil(people / rows)
        total_plots = rows * columns  # Gesamtzahl der Parzellen

        # Überspringe ungültige Aufteilungen
        if total_plots < people or total_plots > max_plots:
            continue

        # Berechne die Dimensionen jeder Parzelle
        cell_length = length / rows
        cell_width = width / columns
        ratio = abs(cell_length - cell_width)  # Differenz der Seitenlängen als Maß für das Verh

        # Überprüfe, ob das Seitenverhältnis besser ist
        if ratio < best_ratio:
            best_ratio = ratio
            best_rows = rows
            best_columns = columns

    # Überprüfen, ob eine gültige Aufteilung gefunden wurde
```

```
            total_plots = best_rows * best_columns
            if total_plots < people or total_plots > max_plots:
                print("[Fehler] Keine gültige Aufteilung gefunden.")
                return None

            # Berechne die endgültigen Dimensionen der Parzellen
            cell_length = length / best_rows
            cell_width = width / best_columns

            # Ausgabe der besten Aufteilung
            print(f"Beste Aufteilung: {best_rows} Reihen und {best_columns} Spalten")
            print(f"Jede Parzelle ist {cell_length:.2f} m lang und {cell_width:.2f} m breit.")
            print(f"Gesamtzahl der Parzellen: {total_plots}")


```

Listing 5: Berechnung der optimalen Aufteilung

```
        def read_file(task, verbose=False):
            try:
                with open(f"./Beispielaufgaben/garten{task}.txt", 'r') as file:
                    people = int(file.readline().strip())
                    length = float(file.readline().strip())
                    width = float(file.readline().strip())

                if verbose:
                    print(f"Datei {task} erfolgreich gelesen mit folgenden Werten: ")
                    print(f"Höhe: " + str(length))
                    print(f"Breite: " + str(width))
                    print(f"Personen: " + str(people))

                return people, length, width
            except Exception as e:
                print(f"Fehler beim Lesen der Datei {task}: {e}")
                return None, None, None


```

Listing 6: Datei einlesen

Im folgenden Abschnitt sehen wir den Code der `main.py`. Diese Datei kümmert sich um die Übergabe der Kommandozeilenargumente und leitet die relevanten Parameter an die Berechnungsfunktionen in der `solve.py` weiter.

```
        def main(task, verbose):
            task = task.strip()
            if task == "all":
                exercises = range(0, 6)
            else:
                exercises = [int(task)]

            for ex in exercises:
                people, length, width = read_file(ex, args.verbose)

                print(f"\n\nFür Beispielaufgabe {ex}:")
                calculate_solution(length, width, people, args.verbose)


        if __name__ == "__main__":
            import argparse

            parser = argparse.ArgumentParser(description="Wählen Sie eine Beispielaufgabe mit ihrer Num
                                                         "oder wählen Sie 'all'.")
            parser.add_argument("-i", "--input", default="all", help="Dateipfad zur Eingabedatei")
            parser.add_argument("-v", "--verbose", action="store_true", help="Ausführliche Ausgaben anz

            args = parser.parse_args()

            if args.input not in ["all", "0", "1", "2", "3", "4", "5"]:
                raise ValueError("Invalid Input! Please choose one of the following: 0, 1, 2, 3, 4, 5,

```

```
main(args.input, args.verbose)
```

Listing 7: Hauptfunktion zur Ausführung

4.1 Erklärung des Quellcodes

Der Quellcode zur Lösung der Aufgabe besteht aus mehreren Funktionen, die die Berechnung der optimalen Aufteilung der Parzellen in einem Garten ermöglichen:

- `calculate(length, width, people)`: Diese Funktion berechnet die beste Aufteilung des Gartens. Sie nimmt die Länge `length`, die Breite `width` und die Anzahl der Personen `people` als Eingabeparameter. Die Funktion berechnet zunächst eine maximale Anzahl an Parzellen `max_people`, die unter Berücksichtigung von 10% Spielraum über der angegebenen Personenanzahl liegt. Anschließend wird in einer Schleife die Anzahl der Reihen (`rows`) getestet, wobei für jede Anzahl an Reihen die entsprechende Anzahl an Spalten (`columns`) berechnet wird, sodass die Gesamtzahl der Parzellen (`total_plots`) mit der Personenanzahl `people` übereinstimmt. Das Ziel ist es, eine möglichst gleichmäßige Aufteilung zu finden, bei der die Länge und Breite der Parzellen möglichst gleich sind. Am Ende gibt die Funktion die beste Aufteilung aus und gibt die Länge und Breite der einzelnen Parzellen sowie die Gesamtzahl der Parzellen zurück.
- `read_file(filename)`: Diese Funktion liest die Eingabedatei ein, die die Werte für `people`, `length` und `width` enthält. Die Datei wird geöffnet, und die Daten werden extrahiert, um sie der `calculate`-Funktion zu übergeben.

Die `calculate`-Funktion stellt den Kern der Berechnungen dar, indem sie die Aufteilung des Gartens optimiert. Dabei werden verschiedene Kombinationen von Reihen und Spalten durchprobiert, um die beste Lösung zu finden. Die `read_file`-Funktion sorgt dafür, dass die notwendigen Daten aus der Datei geladen werden.

Durch die Verwendung von Schleifen und mathematischen Berechnungen wird eine effiziente Lösung gefunden, die eine möglichst gleichmäßige Verteilung der Parzellen gewährleistet, sodass jeder Teilnehmer ausreichend Platz erhält, ohne den verfügbaren Raum zu verschwenden. Der Code gibt schließlich die optimale Aufteilung sowie die entsprechenden Parzellengrößen aus.

5 Was ist „so quadratisch wie möglich“?

„So quadratisch wie möglich“ bedeutet, dass die Parzellen gleichmäßig auf der Fläche verteilt sind, sodass ihre Länge und Breite möglichst gleich sind. Das Verhältnis zwischen Länge und Breite sollte so nah wie möglich an 1 liegen. Denn dann ist Länge und Breite gleich. Um ein möglichst quadratisches Aussehen zu erreichen.