Aufgabe 3: Die Siedler

Teilnahme-ID: 72940

Bearbeiter dieser Aufgabe:
Erik Donath

15. April 2024

Inhaltsverzeichnis

Lösungsidee

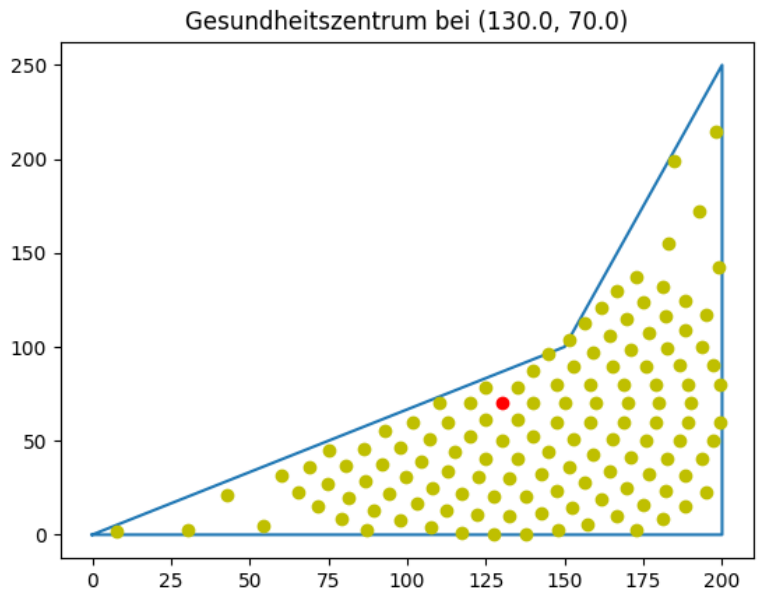
Als erstes betrachtet man alle eck Punkte des Polygons und sucht die kleinsten x und y Werte sowie die größten x und y Werte. Dann geht man Schritt für Schritt im Abstand von 10 von dem min x Wert zu dem max x Wert. Für jeden x Wert geht man nun von min y zu max y im Abstand von 10 hoch. Dadurch bekommen wird eine Raster form mit Punkten. Nun streicht man alle Punkte, die nicht im Polygon liegen und behält somit nur die Punkte, die im Polygon liegen. Nun geht man durch alle Punkte und für jeden Punkt zählt man, wie viele Punkte sich im Radius von 85 befinden und Notiert sich diese Anzahl. Danach findet man den Punkt mit dem größten Notierten Wert. Das ist unser Gesundheitszentrum. Nun zieht man um das Gesundheitszentrum kreise mit je 10 Abstand bis zum Radius 85. Für jeden Kreis rechnet man nun den Umfang (u=2r*π) aus. Diesen Teilt man dann Ganzzahlig durch 10 (u // 10) um die Anzahl an Punkten auf dem Kreis zu finden. Dann geht zählt man von 0 zu der Anzahl an Punkten und für jeden Wert Rechnet man den Winkel α in Bogenmaß aus. Die Formel lautet: $\alpha_i = \frac{10i}{u} * 2\pi = \frac{10i}{2r\pi} * 2\pi = \frac{10i}{r}$. Nun Platziert man einen Punkt bei $P_i = (r\cos\alpha_i, r\sin\alpha_i) + M$. wobei M für das Gesundheitszentrum steht. Sollte der Punkt außerhalb des Polygons liegen wird er nicht platziert. Nach dem Radius 85 ändert sich der Abstand auf 20 und wenn keine Punkte mehr platziert werden können ist man fertig.

Umsetzung

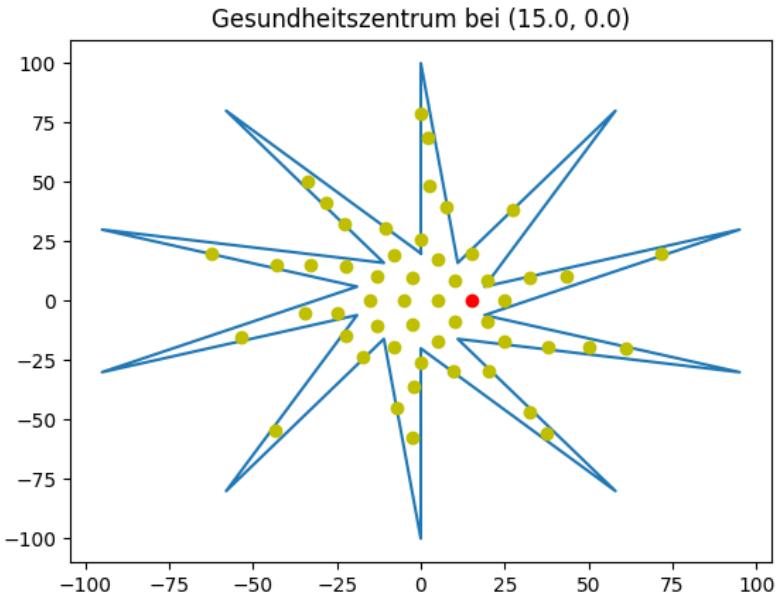
Für das Projekt wird Matplotlib zusammen mit shapely genutzt. Als erstes wird das Polygon mit den gegebenen eck Punkten erstellt. Dann wird die Bounds berechnet und die min und max punkte abgelesen. Im nächsten Schritt läuft eine For-Schleife von min x zu max x mit Abstand 10. In dieser For-Schleife läuft eine Weitere For-Schleife von min y zu max y im Abstand 10. Dann wird überprüft ob der Punkt sich im Polygon befindet und wenn es so ist wird er in eine Liste points gespeichert.

Als nächstes wir eine weiter Dictionary counts erstellt welches als Key einen Punkt hat und als Value ein Integer hat. Dann wird Doppelt über die points Liste iteriert, erster Wert point und zweiter other. Als nächstes wird überprüft ob die Distanz zwischen point und other kleiner gleich 85 ist. Wenn ja wird der Wert in counts an Stelle point um 1 erhöht. Danach wird counts nach den Values sortiert. Dann werden alles Keys aus count in einer separaten Liste gespeichert. lastPoint ist der letzte Punkt in der Liste und Somit der Punkt der am meisten Punkte im Radius 85 haben muss. Somit ist das unser Gesundheitszentrum. Nun läuft eine For-Schleife von 10 bis 85 im Abstand von 10 und für jeden Wert i ruft er die Funktion GenerateCirclePointsInPolygon auf mit i als Radius und lastPoint als Zentrum. Die Zurück gegebene Liste wird nun in Matplotlib gerendert. Nach der For-Schleife folgt eine While-True schleife. Diese ruft wieder GenerateCirclePointsInPolygon mit größer werdenden Werten für den Radius auf. Sollte die Liste lehr sein, die zurück gegeben wurde wird die While-Schleife abgebrochen. Andernfalls werden die Punkte in Matplot gerendert.

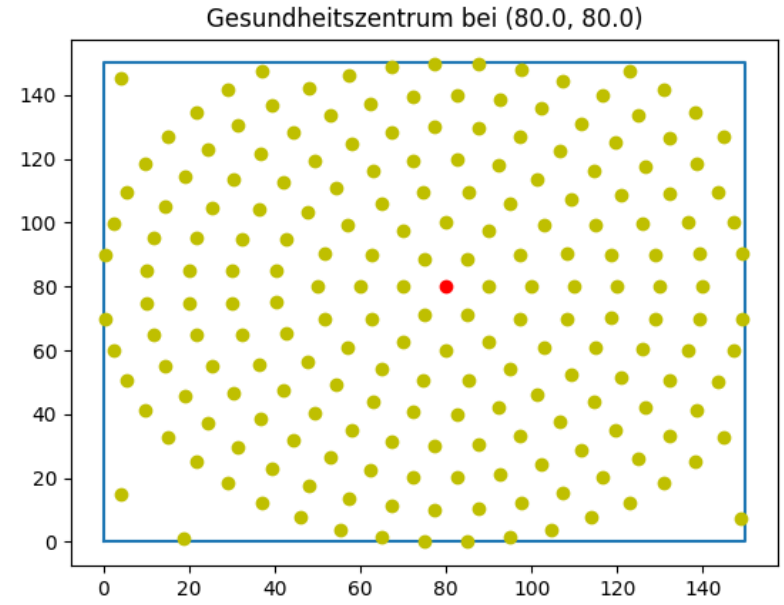
Beispiele

Eingabe	Ausgabe
4 0.000000 0.000000 200.000000 0.000000 200.000000 250.000000 150.000000 100.000000	

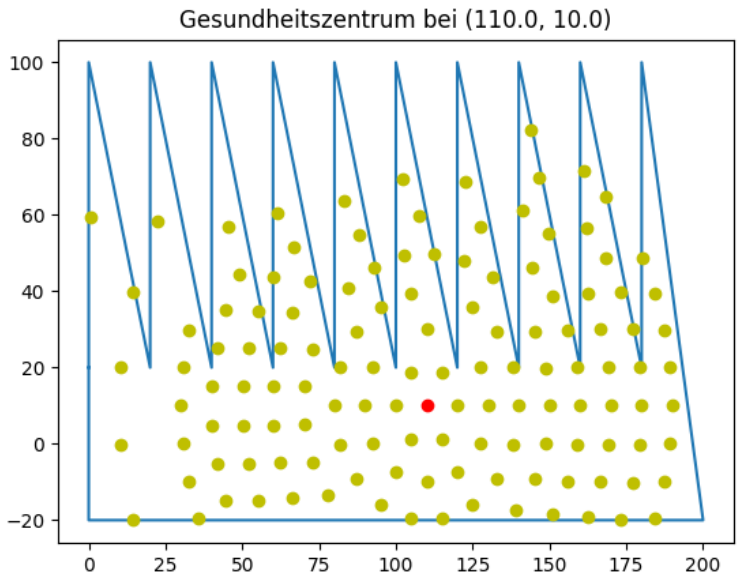
```
20
0.000000 20.000000
0.000000 100.000000
11.000000 16.000000
58.000000 80.000000
19.000000 6.000000
95.000000 30.000000
19.000000 -6.000000
95.000000 -30.000000
11.000000 -16.000000
58.000000 -80.000000
0.000000 -20.000000
0.000000 -100.000000
-11.000000 -16.000000
-58.000000 -80.000000
-19.000000 -6.000000
-95.000000 -30.000000
-19.000000 6.000000
-95.000000 30.000000
-11.000000 16.000000
-58.000000 80.000000
```




```
4
0.000000 0.000000
150.000000 0.000000
150.000000 150.000000
0.000000 150.000000
```



```
22
0.000000 20.000000
0.000000 100.000000
20.000000 20.000000
20.000000 100.000000
40.000000 20.000000
40.000000 100.000000
60.000000 20.000000
60.000000 100.000000
80.000000 20.000000
80.000000 100.000000
100.000000 20.000000
100.000000 100.000000
120.000000 20.000000
120.000000 100.000000
140.000000 20.000000
140.000000 100.000000
160.000000 20.000000
160.000000 100.000000
180.000000 20.000000
180.000000 100.000000
200.000000 -20.000000
0.000000 -20.000000
```



Quellcode

```python
import matplotlib
import matplotlib.pyplot as plt
import shapely.geometry as geo
import numpy as np
import math


def GenerateCirclePointsInPolygon(polygon: geo.Polygon, center: geo.Point, cra-
dius: int, distance: int) -> list[geo.Point]:
    u = 2 * math.pi * cradius
    number_of_points = int(u // distance)
    distance = u / number_of_points

    cpoints: list[geo.Point] = []
    for i in range(number_of_points):
        at = i * distance
        radians = (at / u) * 2 * math.pi
        cpoint = geo.Point(math.cos(radians) * cradius + center.x, math.sin(ra-
dians) * cradius + center.y)

        if polygon.contains(cpoint):
            cpoints.append(cpoint)
    return cpoints


def renderPoints(rpoints: list[geo.Point], arg: str = "yo") -> None:
    xes: list[float] = []
    yes: list[float] = []

    for point in rpoints:
        xes.append(point.x)
        yes.append(point.y)
```

```
    plt.plot(xes, yes, arg)


if __name__ == '__main__':
    print(f"Matplot Version '{matplotlib.__version__}'")

    path = input("Enter a file path: ")
    vertices: list[geo.Point] =S []
    with open(path, "r") as txt_file:
        count = int(txt_file.readline())
        for i in range(count):
            line = txt_file.readline()
            segments = line.split(" ")
            vertices.append(geo.Point(float(segments[0]), float(segments[1])))

    # Optionen
    # vertices: list[tuple[int, int]] = [(0, 0), (200, 200), (300, 200), (100,
300), (300, 320), (-50, 400)]
    renderGrid: bool = False
    gridDistances = 10
    radius = 85

    # Erstellen des Polygons
    polygon = geo.Polygon(vertices)
    plt.plot(*polygon.exterior.xy)

    # Finden der min und max punkte
    bound = polygon.bounds
    minPoint = geo.Point(bound[0], bound[1])
    maxPoint = geo.Point(bound[2], bound[3])

    # Generieren von Punkten
    print("Generating points")
    points: list[geo.Point] = []
    for x in np.arange(minPoint.x, maxPoint.x, gridDistances):
        for y in np.arange(minPoint.y, maxPoint.y, gridDistances):
            if polygon.contains(geo.Point(x, y)):
                points.append(geo.Point(x, y))
                if renderGrid:
                    plt.plot(x, y, 'bo')

    # Zählen der Punkte im Radius
    print(f"Counting points in distance {radius} [O(n²)={len(points) *
len(points)}]")
    counts: dict[geo.Point, int] = {}
    for point in points:
        for other in points:
            if other.distance(point) <= radius:
                counts[point] = (counts[point] if point in counts else 1) + 1

    # Finden des besten Punktes durch Sortierung des Arrays
    print("Finding best point")
    counts = dict(sorted(counts.items(), key=lambda item: item[1]))
    if len(counts) == 0:
        print("=> No point found")
        plt.title("No point found")
        plt.show()
        exit(0)
    print(counts.values())
```

```
    lastPoint = list(counts.keys())[-1]

    # Zeichnet die Dörfer in dem Kreis im Abstand von 10
    print("Generating Villages in distance")
    cradius = 0
    for cradius in range(10, radius, 10):
        cpoints = GenerateCirclePointsInPolygon(polygon, lastPoint, cradius, 10)
        renderPoints(cpoints, "yo")

    print("Generating Villages outside distance")
    cradius += 20
    while True:
        cpoints = GenerateCirclePointsInPolygon(polygon, lastPoint, cradius, 20)
        if len(cpoints) == 0:
            break
        renderPoints(cpoints, "yo")
        cradius += 20

    # Debug
    print(f"Center: {lastPoint}")
    print(f"Min {minPoint} | Max {maxPoint}")
    plt.plot(lastPoint.x, lastPoint.y, "ro")
    plt.title(f"Gesundheitszentrum bei ({lastPoint.x}, {lastPoint.y})")

    plt.show()
```