Junioraufgabe 1: Zum Winde verweht

Team-ID: 01048

Team: Lorian

Bearbeiter/-innen dieser Aufgabe:
Lorian Linnertz

17. November 2021

Inhaltsverzeichnis

Lösungsidee

Meine Lösungsidee ist, dass man die Distanz eines jeden Windrades zu jedem Dorf berechnet. Dies habe ich gelöst mit dem Satz des Pythagoras. Bedeutet, ich habe die Differenz der x Koordinaten eines Dorfes und die eines Windrades genommen und ins Quadrat genommen. Das gleiche habe ich ebenfalls mit der y Koordinate gemacht und die beiden Wert addiert. Am Ende habe ich die Wurzel von der Summe gezogen. Distanz = ((x2-x1)**2+(y2-y1)**2)**0.5; Im letzten Schritt habe ich dann nur noch das Dorf gesucht mit der Kleinsten Entfernung und diese Entfernung geteilt durch 10 ist die maximal Höhe des Windrades

Umsetzung

Die Kernkomponente des Programms bezieht sich alleine auf die Funktion Windmills_MaxHeight. In der Funktion Iteriert man in einem ersten Schritt durch alle Windräder. In dem zweiten Schritt Itereriert man für jedes Windrad durch alle Dörfer und berechnet die jeweilige Distanz zum Windrad mit dem Satz des Pythagoras um anschliessend mit der Funktion Find_Min() den kleinsten Abstand zu finden, welchen man dann nur noch durch 10 teilt

Beispiele

Landkreis1.txt
48.52319033204638 [1242, -593]
158.98003019247417 [-1223, -1479]
72.41270606737467 [1720, 401]

Landkreis2.txt
115.16179922179056 [359, 20]
201.25093788601333 [2, -773]
138.85074720720806 [315, -213]
209.1200612088663 [-629, -532]
132.0068558825639 [97, -69]
186.16457772626887 [-392, -418]
161.68413651314094 [87, -384]
133.30266313918864 [-597, 612]
133.5359127725572 [-13, -32]
128.76866854945735 [-57, 49]
91.77717581185422 [276, 292]
118.2821203732838 [156, 55]
161.95027014488122 [-423, -93]
142.3905193473217 [202, -219]
177.04174648935205 [-340, -343]

Landkreis3.txt
451.56656209245614 [0, 0]
393.78542380337035 [180, 570]
336.7009949495249 [360, 1140]
280.7385972751164 [540, 1710]
444.619792631862 [360, -120]
385.70631314511826 [540, 450]
327.1054264300732 [720, 1020]
269.021486130755 [900, 1590]
440.8414227361127 [720, -240]
381.25025901630545 [900, 330]
321.72715148087826 [1080, 900]
262.31843244423374 [1260, 1470]
440.31302501743005 [1080, -360]
380.5445571808904 [1260, 210]
320.77836585405817 [1440, 780]
261.0160148343392 [1620, 1350]

Landkreis4.txt
0.0 [-4147, 8575]
383.8062271511498 [-6453, 14307]
262.45491041319843 [-8370, 5831]
233.98739282277583 [13045, -5404]
296.1940242476205 [-8361, 8131]
71.75639344337199 [-6963, -371]
181.4135606838695 [9772, -3239]
235.40008496175187 [-5102, -1726]
343.1133923355368 [13454, 11822]
177.89617758681607 [-7427, 1720]
449.1574000280971 [-7816, 12396]
408.0275725977351 [-11095, 603]
317.9480460704233 [8314, 16301]
221.2855621137538 [15283, -2961]
520.1203418440775 [7082, 18552]
394.71493511140415 [16743, 2687]
433.2531015468902 [17511, -730]
703.8262072415321 [-10767, 12860]
168.41674501070253 [1508, -8030]
201.26889973366477 [-7767, 982]
139.1621356547822 [1277, -11294]
348.9877505013607 [-8724, 3575]
297.9106073975883 [7033, -7766]
110.23107547329838 [2720, -10910]

813.6006145523736 [20589, 7265]
236.18918264814755 [-3214, 15263]
391.94164106407476 [6887, 17263]
125.78346473205451 [-3944, 13584]
241.00663891270716 [6576, 15697]
625.3953069859095 [-12074, 5974]

Quellcode

```
def Find_Min(list):

    minIndex=0              #Definiert die Variable x als int oder float

    minNum = list[minIndex]    #Die Variable minNum,bekommt als Startwert die erste Zahl aus einer gegebenen Liste, diese wird dann immer durch die nächst kleiner Zahl ersetz, solange bis keine kleinere Zahl mehr in der Liste ist.


    for i in range(0,len(list)): #iteriert durch alle Indexe der gegebenen Liste durch

        if (list[i] < minNum):   #Vergleicht ob das Element mit dem Index [i] aus einer gegebenen Liste kliner ist als die bislang kleinste gefundene Zahl

            minNum = list[i]    #Falls eine kleinere Zahl als minNum gefunden wird, so wird minNum durch diese Zahl ersetzt

            minIndex = i        #ebenfalls wird der Index dieser neuen kliensten Zahl aktualisiert


    return [minNum, minIndex]   #gibt eine Liste zurück mit der Form [minNum, minIndex]



def transform_Files(filename):

    file = open(filename, "r").readlines() #speicher die Zeilen der Datei "filename" in eine Liste.

    linesList = []          #Definiert die Variable als Array

    housesCoordinates = []

    windmillCoordinates = []

    for line in file:           # in dieser for Schleife wird das \n aus den Zeilen entfernt und in die neue Variable linesList gespeichert

        linesList.append(line.strip())

    nEnd = int(linesList[0].split(" ")[0])+1                    #int(linesList[0].split(" ")[0])+1 -> Diese Aussage gibt an, in welcher Zeile die letzte N Koordinaten liegen. Dies geschieht in dem die Variable N aus der Datei isoliert wird und anschließend noch eins zu N hinzuaddiert, da die erste Zeile übersprungen wird

    for i in range(1,nEnd):                                     #"range(1,nEnd)" -> dies besagt, dass die Loop von der zweiten Zeile an bis zum letzten N-Koordinatenpaar durch iterieren soll

        coordinates = linesList[i].split()                      #Die Zeile wird bei " " gespalten so dass eine Liste entsteht bei der die x-Koordinate auf [0] ist und die y-Koordinate auf [1]

        for k in range(0,2):                                    #lässte k die Werte 0 und 1 annehmen

            coordinates[k] = int(coordinates[k])                #ändert die Variable coordinates[k] von einem Sting in ein int

        housesCoordinates.append(coordinates)                   #Diese Liste wird in die Liste housesCoordinates eingefügt, so dass eine Matrix entsteht, mit den Verschiedenen Hauskoordinaten

    for i in range(nEnd,len(linesList)):                        #diese for Schleife zählt von n bis zum Ende durch, dies wird dadurch erreicht, dass die range() den Anfangswert N hat und den

        coordinates = linesList[i].split()                      #Die Zeile wird bei " " gespalten so dass eine Liste entsteht bei der die x-Koordinate auf [0] ist und die y-Koordinate auf [1]

        for k in range(0,2):                                    #lässte k die Werte 0 und 1 annehmen
```

```
        coordinates[k] = int(coordinates[k])            #ändert die Variable coordinates[k] von einem Sting in ein int

    windmillCoordinates.append(coordinates)             #Diese Liste wird in die Liste windmillCoordinates eingefügt, so dass eine Matrix entsteht, mit den Verschiedenen Windräderkoordinaten

  return housesCoordinates, windmillCoordinates         #gibt die beiden Matrizen zurück, zuerst die der Häuser und anschließend die der Windräder


def Windmills_MaxHeight(nList, mList): #nList steht für die Liste der Koordinaten der Häuser und mList für die der Windräder; diese Funktion berechnet die maximale Höhe für die jeweiligen Windräder

  hList = []        #in diese Liste kommen die maxmal Höhen der Windräder stehen

  for j in range(0,len(mList)):     #iteriert durch alle Koordinaten der Windräder durch

    elementListofN = [] #in dieser Liste werden die zwischen Ergebnisse abgespeichert

    for i in range(0,len(nList)): #iteriert für jedes Windrad einmal durch alle Koordinaten der Häuser durch

      a = nList[i][0]-mList[j][0]       #berechnet die Differenz der einzelnen x-Koordinaten

      b = nList[i][1]-mList[j][1]       #berechnet die Differenz der einzelnen y-Koordinaten

      c = ((a**2) + (b**2))**0.5        #die beiden Differenzen a und b stehen sich in einem 90 Grad winkel gegenüber womit man den Satz des Pythagoras anwenden kann um die Entfernung des Windrades zum jeweiligen Haus zu berechenen

      elementListofN.append(c)          #fügt die Entfernung zwischen dem jeweiligen Haus und dem Windrad in die Variable elementListofN

    cMin = Find_Min(elementListofN)[0]     #Finde die kleinste Zahl auf der Liste elementListofN mithilfe der Funktion Find_Min()

    hMax = cMin / 10                    #Berechne die maximale Höhe des Windrades unter Anwendung der 10h Regel

    hList.append(hMax)                  #Füge die maximale Höhe für das Windrad j in die Liste hList

    print(hMax,mList[j])

  return hList
```