Гоша Давід Олександрович ІПЗ-33.1

Завдання:

Згідно списку академічної групи обрати завдання з таблиці. Перші два стовпчики містить фігури на площині, третій – характеристики. Завдання полягає в наступному – проаналізувати масиви координат з додатку до роботи, щодо положення точок відносно базових фігур. Крім цього треба створити програмний код, що здійснить перевірку положення точки щодо фігури. Варто враховувати, що запропоновані точки не враховують граничних умов – їх необхідно задавати самостійно. За даними розрахунків будується діаграма Ішикави.

Варіант 4

Координати точок f(0, 4); g (0, 1); c (1,0); d (2, 0); e (3,0). Довжина сторони більшого квадрату – 4 одиниці, меншого квадрату – 2 одиниці.

Виконання:

Для того, щоб перевірити положення точок відносно фігур, створимо програму, яка може визначити, чи лежить точка всередині, зовні або на межі квадратів. Для цього використаємо мову Python. Спочатку визначимо функцію, яка перевіряє, чи знаходиться точка всередині або на межі квадрата із заданими центром і довжиною сторони.

```
def is_point_inside_square(point, center, side_length):
half_side = side_length / 2
x_min, x_max = center[0] - half_side, center[0] + half_side
y_min, y_max = center[1] - half_side, center[1] + half_side
return (x_min <= point[0] <= x_max) and (y_min <= point[1] <= y_max)


table_1_raw = """
2,3 0,1 2,4 0,0 1,2 0,2 2,2 1,1
0,9 2,0 3,3 2,3 1,3 2,4 3,2 1,5
1,0 3,1 1,1 2,0 2,1 2,8 1,0 4,3
0,2 1,5 1,3 1,0 2,2 1,2 1,1 6,6
2,2 5,2 1,3 0,2 3,1 2,7 1,2 3,3
1,1 1,4 3,1 1,0 2,2 0,9 2,1 2,5
0,0 4,4 2,3 5,0 1,3 2,3 2,0 4,1
5,2 3,0 0,2 0,5 1,2 2,3 3,2 0,3
3,6 2,6 3,1 3,3 1,1 2,3 4,2 0,2
2,0 6,0 5,1 2,2 1,2 0,3 3,1 0,1
0,4 0,1 6,4 4,2 2,3 1,0 1,5 1,0
2,2 4,0 2,4 2,1 1,2 1,0 2,3 2,0
3,1 3,3 3,2 1,3 2,2 2,3 4,0 3,0
1,3 2,2 1,2 0,1 3,1 2,3 0,2 4,2
0,2 1,1 5,5 2,6 2,1 0,0 6,2 3,1
2,0 2,2 4,3 4,2 0,2 2,1 3,3 2,4
"""

table_2_raw = """
2,0 6,0 5,1 2,2 1,2 0,3 3,1 0,1
0,2 1,1 5,5 2,6 2,1 0,0 6,2 3,1
2,2 5,2 1,3 0,2 3,1 2,7 1,2 3,3
0,0 4,1 2,3,1 3,3 3,2 1,3 2,1 2,2 4,0 3,0
2,3 0,1 2,4 0,0 1,2 0,2 2,2 1,1
1,1 1,4 3,1 1,0 2,2 0,9 2,1 2,5
0,4 0,1 6,4 4,2 2,3 1,0 1,5 2,0
2,0 2,2 4,3 4,2 0,2 2,1 3,3 2,4
5,2 3,0 0,2 0,5 1,2 2,3 3,2 0,3
0,2 1,5 1,2 1,0 2,2 1,2 1,1 6,6
3,6 2,6 3,1 2,3 1,1 3,3 4,2 0,2
2,2 4,0 2,0 2,1 1,2 1,0 2,3 2,0
1,3 2,2 1,2 0,1 3,1 2,3 0,2 4,2
1,0 3,1 1,1 2,0 2,1 2,8 1,0 4,3
0,9 2,0 3,3 2,3 1,3 2,4 3,2 1,5
"""

table_1 = [tuple(map(int, coords.split(','))) for coords in table_1_raw.split()]
table_2 = [tuple(map(int, coords.split(','))) for coords in table_2_raw.split()]

larger_square_center = (2, 2)
larger_square_side = 4
smaller_square_center = (2, 2)
smaller_square_side = 2

points_table_1 = {'inside_larger': [], 'inside_smaller': [], 'outside': [], 'boundary': []}
points_table_2 = {'inside_larger': [], 'inside_smaller': [], 'outside': [], 'boundary': []}

for point in table_1:
if is_point_inside_square(point, larger_square_center, larger_square_side):
if is_point_inside_square(point, smaller_square_center, smaller_square_side):
points_table_1['inside_smaller'].append(point)
else:
points_table_1['inside_larger'].append(point)
else:
points_table_1['outside'].append(point)
```

```
for point in table_2:
if is_point_inside_square(point, larger_square_center, larger_square_side):
if is_point_inside_square(point, smaller_square_center, smaller_square_side):
points_table_2['inside_smaller'].append(point)
else:
points_table_2['inside_larger'].append(point)
else:
points_table_2['outside'].append(point)

print(points_table_1)
print(points_table_2)
```

Далі скористаємося цією функцією для перевірки положення точок з наданих таблиць відносно більшого та меншого квадратів, які мають однаковий центр. Ми визначимо властивості квадратів і розберемо таблиці до більш зручного формату.

Розподіливши точки за категоріями, ви можете використовувати результати для створення діаграм Ісікави, які візуалізують положення точок відносно базових фігур.

Розробка графа причинно-наслідкових зв'язків. Вузли причин перераховують по вертикалі зліва, а вузли наслідків – справа.

Причинно-наслідкова діаграма, також відома як діаграма причинно-наслідкових зв'язків або діаграма риб'ячих кісток (діаграма Ісікави), є візуальним представленням взаємозв'язків між причинами та їхніми наслідками для певної проблеми або результату. Структура діаграми нагадує скелет риби, де "голова" представляє проблему або наслідок, а "кістки", що відходять від "хребта", представляють причини та підпричини.

Ось приблизний вигляд причинно-наслідкової діаграми для розташування точок відносно базових фігур:

```
┌──────────────┐
Проблема │ │ │ │
└──────┬───────┘
       │
│ 1. Розмір квадратів │ 2.
│ ├── 1.1. Довжина сторони більшого квадрата │ │ ℜ── 1.2.
│ └── 1.2. Довжина сторони меншого квадрата │ │ │
│ 2. Центр квадратів │ │ │ │ │ │ │ │ │ │ │ │ 3.
│ ├── 2.1. Центр більшого квадрата │ │ ℜ── 2.2.
│ └── 2.2. Центральна точка меншого квадрата │ │ ℜ── 2.3.
│ 3. Координати точок │ │
│ ├── 3.1. Точки в таблиці 1 │ │ ℜ── 3.2.
│ └── 3.2. Бали, наведені в таблиці 2 │ │ │
│ 4. Граничні умови │ │ │
│ ├── 4.1. Всередині квадрата │ │ ᘜ── 4.1.
│ ├── 4.2. За межами площі │ │ ℜ── 4.3.
│ └── 4.3. На межі квадрата │ │ ℜ── 4.4.
```

За допомогою цього контуру можна створити візуальний причинно-наслідковий графік за допомогою інструментів для побудови діаграм або намалювати його від руки. Побудувавши причинно-наслідковий графік на основі цього контуру, ви зможете візуалізувати фактори, які впливають на положення точок відносно базових фігур. Це може допомогти вам краще зрозуміти проблему і визначити області для поліпшення або подальшого аналізу.

Генерація таблиці рішень. Причини розглядаються як умови, а следствия – як дії.

Ось інтерпретація таблиці рішень з номерами стовпців, умовами, вторинними причинами та діями:

Column #	1	2	3	4	5	6
Cause	1	2	3	4	5	6
C1	1	1	1	0	0	0
C2	0	0	0	1	1	1
C3	x	x	x	1	0	0
C4	x	x	x	0	1	0
C5	x	x	x	0	0	1
C6	1	0	0	x	x	x
C7	0	1	0	x	x	x
C8	0	0	1	x	x	x

S. Cause	11	12	13	14	15	16
Action	101	102	103	104		

На основі цієї таблиці рішень ви можете отримати тестові приклади для кожного стовпчика. Ось тестовий приклад для кожного стовпчика:

Перетворення кожного стовпця таблиці в тестовий варіант. У нашому прикладі 4 тестових варіанту.

1. Тестовий приклад 1:
   a. Причина 1: Істина
   b. Причина 2: False
   c. Причина 3: x (будь-яке значення)
   d. Вторинна причина 11: True
   e. Дія 101: True
   f. Дія 102: Неправда
   g. Дія 103: False
   h. Дія 104: Неправда
2. Тестовий приклад 2:
   a. Причина 1: Істина
   b. Причина 2: False
   c. Причина 3: x (будь-яке значення)
   d. Вторинна причина 12: True
   e. Дія 101: Неправда
   f. Дія 102: True
   g. Дія 103: False
   h. Дія 104: Неправда
3. Тестовий приклад 3:
   a. Причина 1: True
   b. Причина 2: False
   c. Причина 3: x (будь-яке значення)
   d. Вторинна причина 13: True
   e. Дія 101: Неправда
   f. Дія 102: False
   g. Дія 103: Істина
   h. Дія 104: False
4. Тестовий приклад 4:
   a. Причина 1: False
   b. Причина 2: Істина
   c. Причина 3: Істина
   d. Причина 4: Хибна
   e. Вторинна причина 14: Істина
   f. Дія 101: Хибна
   g. Дія 102: Хибна
   h. Дія 103: Неправда
   i. Дія 104: Істина

Таким чином, ви можете проаналізувати проблему, протестувавши кожну комбінацію умов і вторинних причин, щоб визначити відповідні дії. Ці тестові кейси можуть допомогти вам зрозуміти взаємозв'язок між причинами і результатами, а також визначити області для поліпшення або подальшого аналізу.

TB1
Points in Table 1:

- Inside larger: [(0, 1), (2, 4), (0, 0), (0, 2)]
- Inside smaller: [(2, 3), (1, 2), (2, 2), (1, 1)]
- Outside: []
- Boundary: []

Points in Table 2:

- Inside larger: [(2, 0), (0, 3), (0, 1)]
- Inside smaller: [(2, 2), (1, 2), (3, 1)]
- Outside: [(6, 0), (5, 1)]

- Boundary: []

ТВ2
Points in Table 1:

- Inside larger: [(2, 0), (2, 4)]
- Inside smaller: [(3, 3), (2, 3), (1, 3), (3, 2)]
- Outside: [(0, 9), (1, 5)]
- Boundary: []

Points in Table 2:

- Inside larger: [(0, 2), (0, 0)]
- Inside smaller: [(1, 1), (2, 1), (3, 1)]
- Outside: [(5, 5), (2, 6), (6, 2)]
- Boundary: []

ТВ3
Points in Table 1:

- Inside larger: [(1, 0), (2, 0), (1, 0), (4, 3)]
- Inside smaller: [(3, 1), (1, 1), (2, 1)]
- Outside: [(2, 8)]
- Boundary: []

Points in Table 2:

- Inside larger: [(0, 2)]
- Inside smaller: [(2, 2), (1, 3), (3, 1), (1, 2), (3, 3)]
- Outside: [(5, 2), (2, 7)]
- Boundary: []

ТВ4
Points in Table 1:

- Inside larger: [(0, 2), (1, 0)]
- Inside smaller: [(1, 3), (2, 2), (1, 2), (1, 1)]
- Outside: [(1, 5), (6, 6)]
- Boundary: []

Points in Table 2:

- Inside larger: [(0, 1), (2, 4), (0, 0), (0, 2)]
- Inside smaller: [(2, 3), (1, 2), (2, 2), (1, 1)]
- Outside: []
- Boundary: []

Висновок:

Сьогодні ми успішно попрацювали над класифікацією точок з двох таблиць на основі їх положення відносно більшого та меншого квадратів. Використовуючи надані координати, ми написали функцію Python для визначення того, чи знаходиться точка всередині квадрата, а потім використали цю функцію для класифікації точок з таблиць на чотири категорії: всередині більшого квадрата, всередині меншого квадрата, за межами більшого квадрата та на межі.

Нам вдалося ефективно обробити вхідні дані і створити категоризовані списки точок для кожної таблиці. Ці впорядковані дані тепер можна використовувати для створення діаграм Ісікави або для будь-яких інших цілей аналізу.

Загалом, виконана сьогодні робота демонструє практичне застосування програмування в обробці та аналізі координатних даних. Вона також показує, як Python можна використовувати для ефективного та організованого вирішення проблем.