Table of Content:

General Info:

- The goals of the program are to deal with Spatial Data Management and Range Queries.
- Written in Jupyter Notebook / Python 3.6
- To run the program, please prepare three csv files that include a set of cleaned polygons coordinations, a set of window queries coordinations and a set of distance queries coordinations with the query distances.

Problem-Solving Targets:

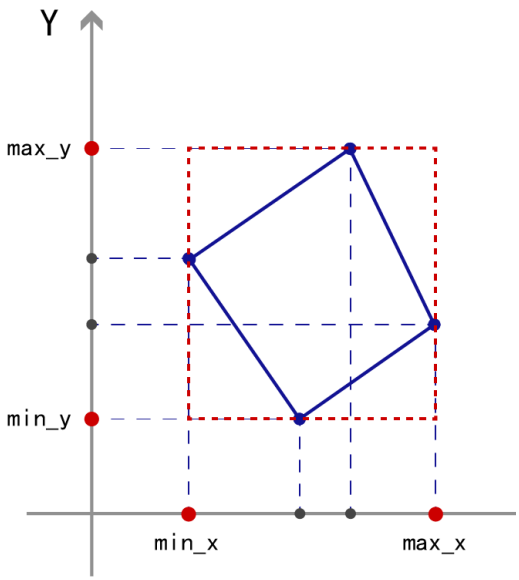
1. Calculate the individual MBR of the polygons in the dataset, as well as the global MBR of all the individual MBRs.
2. Carry out window range queries - find the MBRs intersecting with the window rectangles of the range query.
3. Carry out distance range queries - find the MBRs intersecting with the circle generated by the distance query.

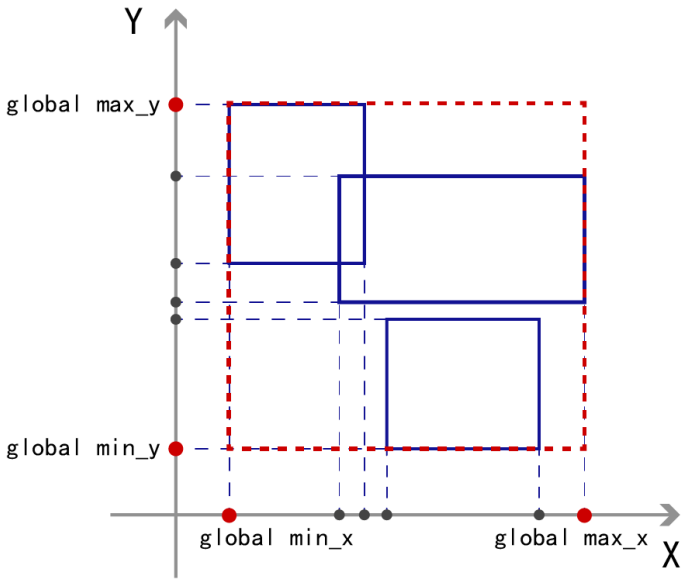
Algorithm:

Algorithm1 - Calculate individual and global MBR: to find the minimum x, y, and maximum x, y in the set of coordinations of the polygons or MBRs

```
def MBR(coordination):
    min_x, min_y = np.min(coordination, axis=0)
    max_x, max_y = np.max(coordination, axis=0)
    return [min_x, max_x, min_y, max_y]
```

```
def globalMBR(coordset):
    xmin_MBR = []
    xmax_MBR = []
    ymin_MBR = []
    ymax_MBR = []

    for i in coordset:
        xmin_MBR.append(i[0])
        xmax_MBR.append(i[1])
        ymin_MBR.append(i[2])
        ymax_MBR.append(i[3])

    return [min(xmin_MBR), max(xmax_MBR), min(ymin_MBR), max(ymax_MBR)]
```

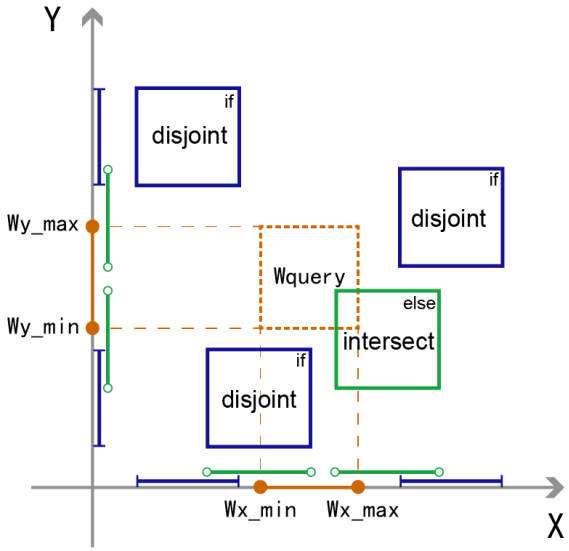




Algorithm2 - To find the relationship between the rectangles, the four points of coordination can be projected to the X and Y axis, which transfers the two-dimensional range query to one-dimensional range query. Instead of finding the overlap between MBRs' X and Y range overlap, the program excludes the MBRs whose X or Y ranges disjoint with the window query's ranges.

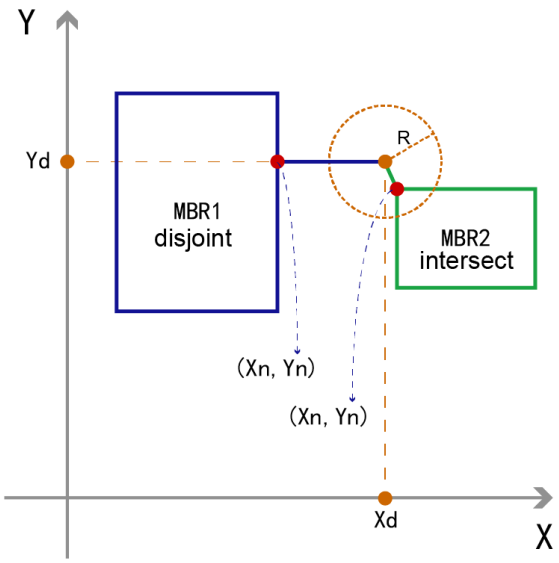
```
def Windowquery(Wquery, recMBR):
    Wx_min = Wquery[0]
    Wx_max = Wquery[1]
    Wy_min = Wquery[2]
    Wy_max = Wquery[3]

    polygon_number = 0
    intersection_rec = []

    for i in polygon_MBR:
        xmin = i[0]
        xmax = i[1]
        ymin = i[2]
        ymax = i[3]

        if xmax< Wx_min or xmin > Wx_max or ymax < Wy_min or ymin > Wy_max:
            polygon_number+= 1
        else:
            intersection_rec.append(polygon_number)
            polygon_number+= 1

    return intersection_rec
```

Only when the MBR's ranges overlap with the window query's ranges on both X and Y axis, the MBR can be considered as intersecting with the window query, otherwise, their relationship is disjointing.



Algorithm3 - To determine the relationship between the MBRs and the query circle, the method is to find the MBRs' closest points to the center of the circle. If the distance between the closest point and the center of the query circle is shorter than the radius of the circle, the MBR intersects with the circle.

```
def Distquery(Dquery, recMBR):
    R = Dquery[2]
    Xd = Dquery[0]
    Yd = Dquery[1]
    Q = Dquery[:2]

    polygon_number = 0
    intersection_circle = []

    for i in recMBR:
        X1 = i[0]
        Y1 = i[2]
        X2 = i[1]
        Y2 = i[3]

        Xn = max(X1, min(Xd, X2))
        Yn = max(Y1, min(Yd, Y2))

        Distx = Xn - Xd
        Disty = Yn - Yd

        if (Distx**2 + Disty**2) <= R**2:
            intersection_circle.append(polygon_number)
            polygon_number+= 1
        else:
            polygon_number+= 1

    return Q, R, intersection_circle
```



The distance between the closest point (Xn, Yn) and the circle center (Xd, Yd) is calculated by the Pythagorean theorem.

Program Application:

Take the datasets provided for the Assignment 1 as examples, the printout results of the program are shown in the screenshots below:

1. Loading the file contents in memory and computing MBRs

```
first polygon: [[-87.866893, 32.825274], [-87.863009, 32.815742], [-87.863257, 32.815652], [-87.867142, 32.82521],
[-87.866893, 32.825274]]
first polygon's MBR: [-87.867142, -87.863009, 32.815652, 32.825274]
the global MBR: [-179.147236, 178.591597, -14.548699, 71.359879]
```

2. Filter step of rectangular range queries

```
query:[-88.0, -87.5, 32.5, 33.0] has 19 results in the filter step: [0, 1, 2, 3, 40, 43, 746, 749, 760, 1371, 1372,
1375, 1376, 2611, 2612, 2613, 2854, 2880, 2882]
query:[-88.0, -87.0, 32.0, 32.5] has 71 results in the filter step: [742, 743, 744, 745, 747, 748, 750, 752, 753, 7
54, 755, 756, 757, 758, 759, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 897, 901, 902, 903, 911, 9
13, 1351, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1363, 1364, 1365, 1366, 1367, 1368, 1373, 1374, 2304, 230
6, 2307, 2310, 2311, 2312, 2314, 2315, 2319, 2320, 2322, 2325, 2326, 2327, 2333, 2334, 2340, 2576, 2734, 2786, 2927
]
query:[-87.5, -87.0, 33.0, 34.0] has 76 results in the filter step: [446, 495, 496, 500, 644, 645, 652, 657, 658, 6
59, 660, 661, 662, 664, 665, 667, 671, 672, 675, 677, 688, 690, 692, 693, 695, 698, 700, 701, 702, 703, 704, 705, 7
06, 707, 709, 711, 713, 715, 718, 719, 721, 722, 725, 726, 727, 728, 729, 731, 732, 737, 738, 740, 741, 1016, 1017,
1020, 1022, 1023, 1024, 1029, 1030, 1031, 1032, 1033, 1037, 1129, 1130, 2578, 2584, 2603, 2604, 2633, 2830, 2831, 2
923, 2928]
query:[-89.0, -88.5, 32.5, 33.0] has 88 results in the filter step: [64199, 64203, 64204, 64207, 64211, 64215, 6421
6, 64218, 64219, 64221, 64226, 64227, 64230, 64231, 64701, 64731, 64736, 64742, 64744, 64747, 64749, 64756, 64759,
64767, 64770, 64774, 65269, 65271, 65280, 65282, 65286, 65305, 65314, 65569, 65570, 65571, 65572, 65574, 65575, 655
76, 65577, 65578, 65579, 65580, 65581, 65582, 65584, 65586, 65587, 65588, 65589, 65590, 65594, 65595, 65596, 65597,
65600, 65601, 65602, 65603, 65605, 65606, 65609, 65612, 65613, 65614, 65615, 65616, 65617, 65618, 65619, 65620, 661
24, 66126, 66127, 66128, 66132, 66133, 66134, 66135, 66136, 66137, 66138, 66139, 66140, 66141, 66142, 66549]
query:[-89.0, -88.5, 33.0, 34.0] has 108 results in the filter step: [64049, 64050, 64051, 64052, 64053, 64054, 640
55, 64056, 64057, 64058, 64059, 64060, 64061, 64062, 64063, 64064, 64065, 64066, 64067, 64069, 64070, 64071, 64072,
64074, 64075, 64076, 64077, 64201, 64205, 64206, 64208, 64209, 64213, 64217, 64220, 64222, 64223, 64224, 64225, 642
28, 64229, 64635, 64640, 64669, 64670, 64671, 64672, 64673, 64679, 64688, 64689, 64728, 64729, 64730, 64733, 64734,
64737, 64739, 64741, 64743, 64745, 64746, 64748, 64750, 64751, 64752, 64758, 64760, 64761, 64763, 64766, 64768, 647
71, 64775, 64776, 65321, 65322, 65325, 65384, 65385, 65386, 65388, 65389, 65390, 65391, 65392, 65393, 65395, 65396,
65400, 65401, 65405, 65406, 66454, 66455, 66481, 66486, 66585, 66586, 66592, 66593, 66594, 66655, 66656, 66668, 666
81, 66717, 66735]
query:[-89.5, -89.0, 32.5, 33.0] has 13 results in the filter step: [64048, 66125, 66129, 66130, 66131, 66135, 6614
3, 66144, 66565, 66587, 66588, 66704, 66713]
```

3. Filter step of distance range queries

```
query:[-88.0, 32.0] with dist 0.2 has 10 results in the filter step: [745, 751, 758, 762, 1537, 1541, 1542, 1545, 1
546, 1555]
query:[-88.0, 32.0] with dist 0.4 has 75 results in the filter step: [293, 294, 295, 742, 744, 745, 748, 750, 751,
753, 756, 758, 759, 761, 762, 764, 765, 766, 767, 769, 772, 1514, 1515, 1516, 1517, 1518, 1520, 1523, 1524, 1525, 1
526, 1528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 15
48, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1563, 1573, 1577, 1578, 1579, 2328, 2329, 233
0, 2560, 2734, 2761, 2762, 2881, 2915]
query:[-88.0, 32.0] with dist 0.6 has 165 results in the filter step: [284, 285, 286, 287, 288, 289, 290, 291, 292,
293, 294, 295, 321, 325, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 338, 351, 352, 520, 521, 522, 523, 742,
743, 744, 745, 746, 747, 748, 749, 750, 751, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766,
767, 768, 769, 770, 771, 772, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1
528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 15
47, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 156
6, 1567, 1568, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 2313, 2314, 2315, 2316, 2318, 2322
, 2323, 2327, 2328, 2329, 2330, 2332, 2333, 2334, 2337, 2339, 2341, 2560, 2664, 2734, 2761, 2762, 2854, 2881, 2886,
2889, 2915, 2929, 65290, 65294, 65308, 65311, 65316, 65748, 65768, 65769, 65770, 65773]
query:[-90.0, 35.0] with dist 0.1 has 3 results in the filter step: [66436, 66677, 103846]
query:[-90.0, 35.0] with dist 0.3 has 51 results in the filter step: [9302, 9498, 9499, 9500, 9501, 64860, 66436, 6
6437, 66438, 66439, 66440, 66441, 66442, 66443, 66444, 66445, 66492, 66493, 66677, 103839, 103840, 103841, 103842,
103843, 103844, 103845, 103846, 106471, 106472, 106502, 106503, 106527, 106530, 106531, 106532, 106533, 106534, 106
559, 106564, 106565, 106571, 106576, 106577, 106578, 106579, 106580, 106581, 106583, 106584, 106585, 106586]
query:[-90.0, 35.0] with dist 0.5 has 130 results in the filter step: [6502, 9302, 9498, 9499, 9500, 9501, 64170, 6
4172, 64174, 64175, 64177, 64180, 64183, 64188, 64190, 64193, 64194, 64196, 64198, 64851, 64852, 64853, 64854, 6485
5, 64856, 64857, 64858, 64859, 64860, 64861, 64862, 64863, 64864, 64865, 64866, 64867, 64868, 64869, 64870, 64871,
64872, 64873, 64874, 64875, 64876, 64877, 65118, 65119, 65125, 65416, 65417, 65418, 65419, 65420, 65421, 65424, 654
31, 65432, 65433, 65438, 65439, 65441, 65444, 66436, 66437, 66438, 66439, 66440, 66441, 66442, 66443, 66444, 66445,
66492, 66493, 66625, 66626, 66627, 66633, 66677, 103839, 103840, 103841, 103842, 103843, 103844, 103845, 103846, 10
3847, 103848, 106041, 106054, 106065, 106068, 106162, 106163, 106164, 106165, 106166, 106167, 106169, 106174, 10619
1, 106193, 106471, 106472, 106502, 106503, 106527, 106528, 106529, 106530, 106531, 106532, 106533, 106534, 106559,
106564, 106565, 106571, 106576, 106577, 106578, 106579, 106580, 106581, 106583, 106584, 106585, 106586]
```