***Шевченко Алексей 22ПИ-3***

**Лабораторная** **работа №2 по Алгоритмам и структурам данных**

* **ДАНО:** Даны прямоугольники на плоскости с углами в целочисленных координатах ([1..109],[1..109]).
* **ЗАДАЧА:** Требуется как можно быстрее выдавать ответ на вопрос «Скольким прямоугольникам принадлежит точка (x,y)?» И подготовка данных должна занимать мало времени. (Рассматривать только нижние границы включены => (x1<= x) && (x<x2) && (y1<=y) && (y<y2)).

Рассматриваются три разных алгоритма решения задачи: перебор, на карте, на персистентном дереве отрезков.

* **ЦЕЛЬ:** Реализовать эти три алгоритма. Определить, при каком объеме начальных данных и точек наиболее эффективен каждый из алгоритмов.

1. ***Алгоритм перебора:***

* Суть: Прямой перебор прямоугольников для каждой данной точки и определение того, принадлежит ли точка прямоугольнику.
* Сложность: Предварительная подготовка данных отсутствует – O(1). Поиск (определение факта принадлежности точки прямоугольнику) – O(N \* M), где N – количество прямоугольников, M – количество проверяемых точек.

1. ***Алгоритм на карте:***

* Суть: Создание сжатой карты, где каждая ячейка представляет собой прямоугольник на карте, и дальнейший подсчет количества прямоугольников, пересекающихся с каждой из данных точкой. Для каждой точки определяется положение на карте, получаемое при нахождении индексов координат точки. В ячейке с найденными индексами содержится ответ на задачу для проверяемой точки.
* Сложность: Предварительная подготовка данных – O(N^3). Поиск (определение факта принадлежности точки прямоугольнику) – O(M \* log(N)), где N – количество прямоугольников, M – количество проверяемых точек.

1. ***Алгоритм на дереве:***

* Суть: Сначала данные прямоугольники преобразуются в сжатые координаты, чтобы уменьшить размер данных и упростить обработку. Затем для каждого прямоугольника создаются события добавления и удаления в дерево отрезков, основанные на его координатах. События сортируются по координате x для обеспечения последовательного построения дерева отрезков. Далее строится персистентное дерево отрезков, где каждый узел представляет собой сумму значений в соответствующем интервале по координате y. Для каждой точки на карте определяется ее позиция в сжатых координатах, после чего считается сумма значений в соответствующем участке дерева отрезков. Результатом является список, в котором для каждой точки указано количество прямоугольников, содержащих эту точку – ответ на поставленную задачу.
* Сложность: Предварительная подготовка данных – O(N\*log(N)). Поиск (определение факта принадлежности точки прямоугольнику) – O(M \* log(N)), где N – количество прямоугольников, M – количество проверяемых точек.

**Реализация алгоритмов:** [**Github**](https://github.com/alexshevvv/ALG_LAB_2.git)

**Структура репозитория:**

Проект состоит из четырёх папок – brute\_force (для алгоритма перебором), map\_alg (для алгоритма на карте), persistent\_segment\_tree (для алгоритма на персистентном дереве отрезков), result\_graphics (для визуализации результатов измерений на графиках).

Содержимое папок:

1. brute\_force:
   1. Файл brute\_force\_alg.py – реализация самого алгоритма.
   2. Файл brute\_force\_generate.py – генерация точек и прямоугольников для данного алгоритма.
   3. Файл brute\_force\_result.py – код запуска тестов, замеров и получение результатов.
2. map\_alg:
   1. Файл map\_alg.py – реализация самого алгоритма.
   2. Файл map\_alg\_generate.py – генерация точек и прямоугольников для данного алгоритма.
   3. Файл map\_alg\_result.py – код запуска тестов, замеров и получение результатов.
3. persistent\_segment\_tree:
   1. Файл persistent\_segtree\_alg.py – реализация самого алгоритма.
   2. Файл persistent\_segtree\_generate.py – генерация точек и прямоугольников для данного алгоритма.
   3. Файл persistent\_segtree\_result.py – код запуска тестов, замеров и получение результатов.
4. result\_graphics:
   1. Файл data\_prep.py – визуализация данных по времени предварительной подготовки данных для алгоритмов на карте и на дереве.
   2. Файл with\_data\_prep.py – визуализация данных по времени работы всех трёх алгоритмов с учётом предварительной подготовки данных.
   3. Файл without\_data\_prep.py – визуализация данных по времени работы всех трёх алгоритмов без учёта предварительной подготовки данных.

**На чём запускался код:** PyCharm 2024.1, интерпретатор: Python 3.9.

**Логин для контеста:** **aashevchenko\_10@edu.hse.ru**

**Результаты запусков:**

В соответствии с предложенными рекомендациями из ТЗ для тестирования были использованы сгенерированные наборы данных:

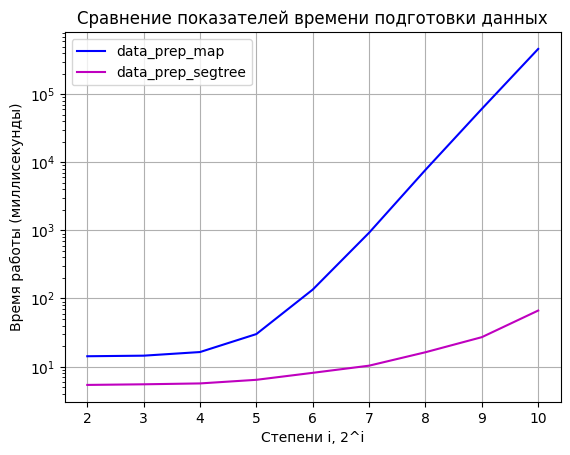
* набор вложенных друг в друга прямоугольников, имеющих координаты с шагом больше 1: {(10\*i, 10\*i), (10\*(2\*N-i), 10\*(2\*N-i))}.
* неслучайный набор координат точек распределенных равномерно по ненулевому пересечению прямоугольников - хэш-функции от i с разным базисом для x и y: (p\*i)^31%(20\*N), p - большое простое, разное для x и y. В качества базиса для x и для y были выбраны большие простые числа: basis\_x = 3089, basis\_y = 6089 соответственно.
* Количество прямоугольников, N= 2^i; 1 <= i <= 10.
* Количество проверяемых точек, M = 100000.
* Для каждого показателя степени i проводилось по 10 замеров, из которых потом бралось медианное значение.

Для анализа зависимости времени работы алгоритмов от размеров входных данных были проведены следующие сравнения:

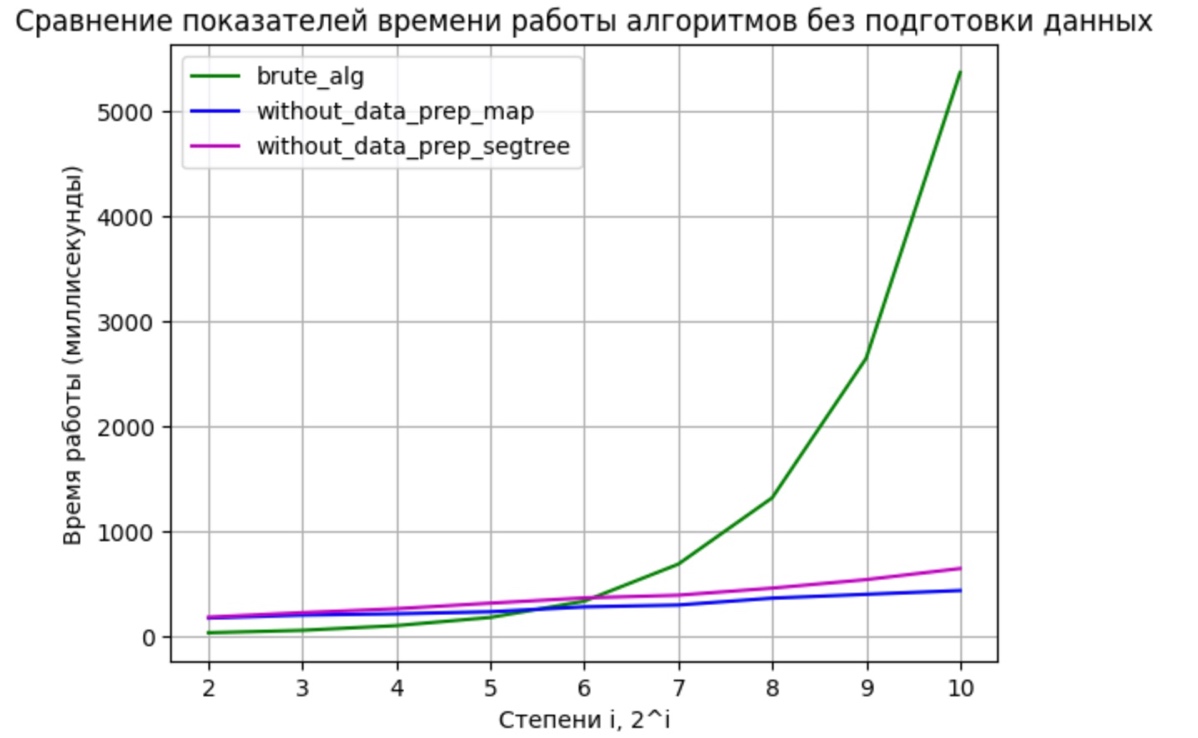
1. Время, необходимое для предварительной обработки данных для алгоритмов на карте и на дереве
2. Время работы всех трёх алгоритмов без учёта предварительной подготовки данных.
3. Время работы всех трёх алгоритмов с учётом предварительной подготовки данных.

Результаты времени представлены в миллисекундах.

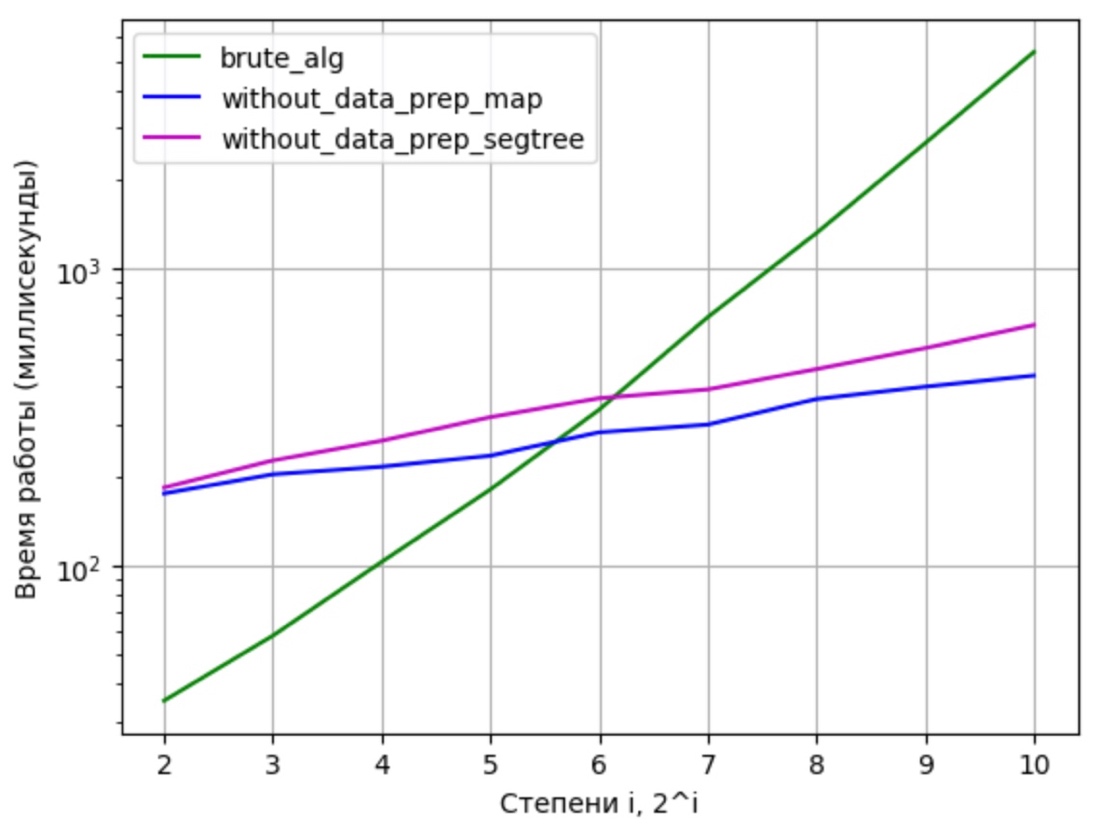
1. Время, необходимое для предварительной обработки данных для алгоритмов на карте и на дереве:



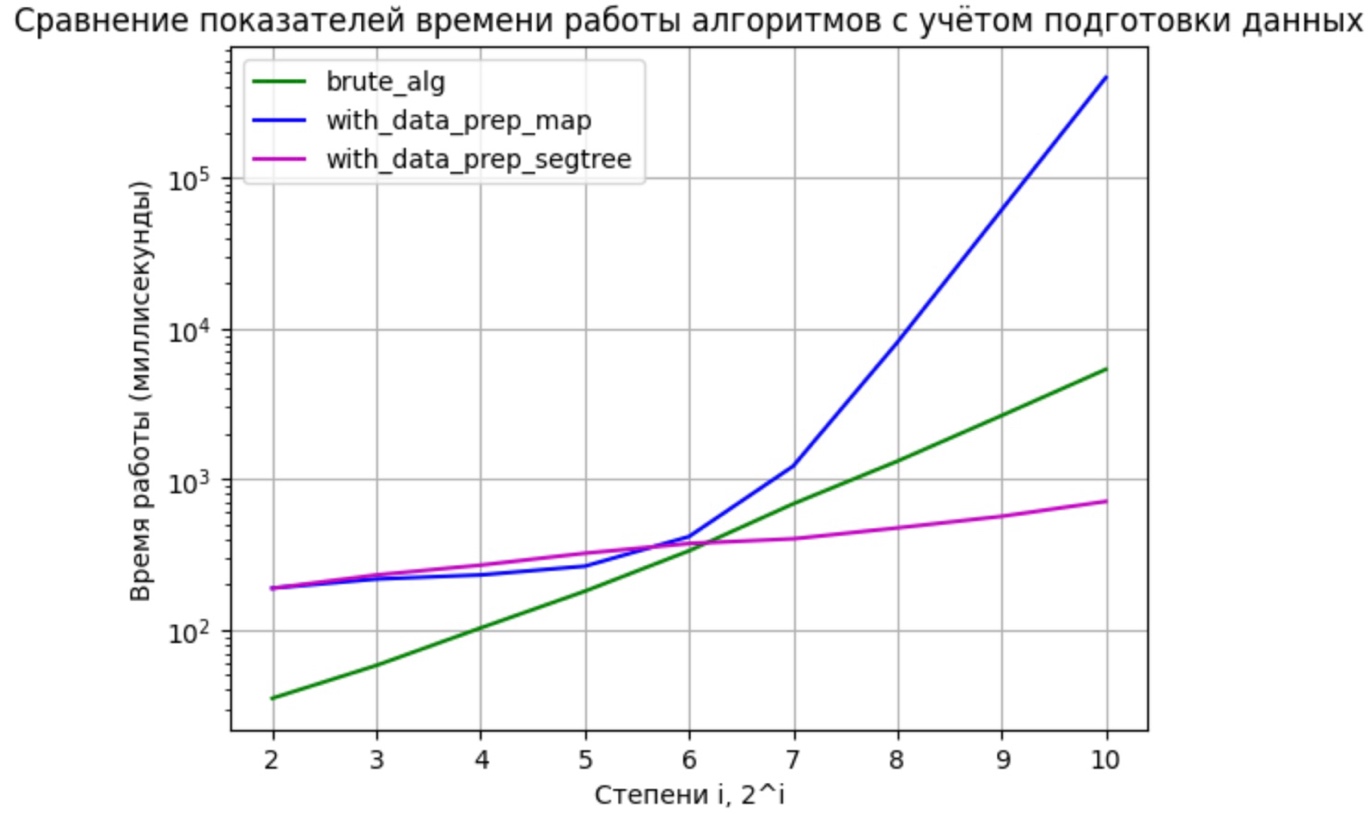
1. Время работы всех трёх алгоритмов без учёта предварительной подготовки данных:



Тот же график, но большего масштаба:



1. Время работы всех трёх алгоритмов с учётом предварительной подготовки данных:



**Данные для графиков (Результаты времени представлены в миллисекундах):**

**Brute force:**

Cырые данные:

1. i = 1: [35.8682, 34.0258, 34.7568, 35.6596, 34.5055, 34.5602, 35.2449, 37.3529, 35.3254, 34.6951]
2. i = 2: [60.2333, 58.9373, 57.5554, 63.2568, 57.6679, 58.342, 55.582, 57.3638, 56.853, 56.4132]
3. i = 3: [96.4065, 99.2139, 104.6593, 99.7119, 108.8754, 107.7073, 103.9757, 109.2945, 101.6657, 99.7715]
4. i = 4: [185.0486, 179.716, 171.9642, 179.8458, 183.2365, 179.8887, 184.0288, 175.6928, 193.1628, 174.7224]
5. i = 5: [351.5918, 344.2473, 336.5678, 328.7196, 331.8338, 329.8929, 336.5458, 330.5773, 330.1658, 332.1873]
6. i = 6: [700.5062, 798.5182, 670.1494, 658.8301, 656.6789, 665.2405, 688.4238, 656.6514, 661.8931, 718.0937]
7. i = 7: [1349.0109, 1326.5368, 1309.2146, 1304.1895, 1324.9171, 1325.5913, 1326.4016, 1311.093, 1310.879, 1302.3997]
8. i = 8: [2776.4111, 2634.546, 2665.9909, 2653.5988, 2648.478, 2650.5995, 2617.2894, 2586.7607, 2589.414, 2643.1818]
9. i = 9: [5307.7336, 5197.502, 5300.058, 5606.9957, 5508.2168, 5426.4766, 5486.6973, 5296.2476, 5333.6941, 5183.9644]
10. i = 10: [10831.2471, 10807.405, 10705.2631, 10765.759, 10983.1409, 10898.3299, 10680.0923, 10723.6313, 10898.071, 11238.7706]

Oкруглённые средние значения данных:

1. i = 1: 35.1994
2. i = 2: 58.2205
3. i = 3: 103.1282
4. i = 4: 180.7307
5. i = 5: 335.2329
6. i = 6: 687.4985
7. i = 7: 1319.0234
8. i = 8: 2646.6270
9. i = 9: 5364.7586
10. i = 10: 10853.1710

**Map algorithm:**

Сырые данные:

1. i = 1:

Map preparation time: [13.421, 13.368, 15.56, 13.107, 13.892, 14.041, 13.173, 15.196, 14.873, 15.131]

Map algorithm time: [169.212, 180.501, 181.511, 172.436, 176.876, 175.954, 171.167, 177.589, 171.708, 175.783]

1. i = 2:

Map preparation time: [14.777, 16.065, 14.676, 15.372, 14.916, 13.968, 13.677, 13.589, 13.864, 13.638]

Map algorithm time: [188.753, 222.491, 279.98, 206.866, 188.933, 190.147, 195.679, 186.347, 186.777, 187.058]

1. i = 3:

Map preparation time: [15.71, 14.974, 14.797, 14.518, 17.671, 14.89, 23.262, 14.896, 18.024, 14.645]

Map algorithm time: [207.374, 211.336, 193.705, 193.522, 221.942, 268.811, 239.719, 197.984, 225.95, 196.17]

1. i = 4:

Map preparation time: [33.105, 29.697, 28.225, 28.816, 28.791, 28.579, 32.285, 33.338, 28.823, 28.497]

Map algorithm time: [241.331, 241.773, 229.74, 231.476, 230.22, 227.837, 235.086, 236.965, 238.449, 236.459]

1. i = 5:

Map preparation time: [135.77, 150.842, 132.807, 136.085, 133.29, 130.771, 134.888, 131.642, 132.686, 132.035]

Map algorithm time: [289.362, 284.907, 281.888, 280.243, 276.762, 286.296, 275.708, 280.285, 281.433, 278.093]

1. i = 6:

Map preparation time: [976.326, 913.437, 919.76, 911.989, 913.764, 909.5, 923.398, 925.267, 915.586, 933.916]

Map algorithm time: [298.127, 302.914, 304.618, 291.489, 289.014, 305.253, 304.167, 303.58, 293.461, 296.365]

1. i = 7:

Map preparation time: [7749.877, 7733.158, 7788.892, 7788.806, 7700.37, 7740.115, 7756.204, 7741.24, 7722.435, 7726.741]

Map algorithm time: [355.399, 364.746, 357.205, 363.544, 370.325, 364.092, 396.306, 348.633, 358.799, 365.391]

1. i = 8:

Map preparation time: [62699.707, 60706.639, 61114.853, 60937.141, 60506.385, 60674.994, 60702.133, 60781.008, 60531.041, 60519.016]

Map algorithm time: [384.266, 417.151, 394.588, 387.604, 406.273, 412.151, 399.581, 411.167, 403.016, 393.538]

1. i = 9:

Map preparation time: [475745.605, 467604.97, 463678.389, 462330.089, 459061.943, 460341.179, 460547.083, 461145.508, 459249.008, 459157.774]

Map algorithm time: [418.849, 454.694, 430.101, 422.488, 442.838, 449.245, 435.543, 448.172, 439.287, 428.956]

1. i = 10:

Map preparation time: [3609481.905, 3547718.907, 3517927.937, 3507698.385, 3482902.962, 3492608.525, 3494170.719, 3498710.969, 3484322.224, 3483630.031]

Map algorithm time: [502.619, 545.633, 516.121, 506.986, 531.406, 539.094, 522.652, 537.806, 527.144, 514.747]

Округлённые средние значения данных:

1. i = 1: Map preparation time: 14.1762; Map algorithm time: 175.2737; Total: 189.4499
2. i = 2: Map preparation time: 14.4542; Map algorithm time: 203.3031; Total: 217.7573
3. i = 3: Map preparation time: 16.3387; Map algorithm time: 215.6513; Total: 231.990
4. i = 4: Map preparation time: 30.0156; Map algorithm time: 234.9336; Total: 264.9492
5. i = 5: Map preparation time: 135.0816; Map algorithm time: 281.4977; Total: 416.5793
6. i = 6: Map preparation time: 924.2943; Map algorithm time: 298.8988; Total: 1223.1931
7. i = 7: Map preparation time: 7744.7838; Map algorithm time: 364.4440; Total: 8109.2278
8. i = 8: Map preparation time: 60917.2917; Map algorithm time: 400.9335; Total: 61318.2252
9. i = 9: Map preparation time: 462886.1548; Map algorithm time: 437.0173; Total: 463323.1721
10. i = 10: Map preparation time: 3511917.2564; Map algorithm time: 524.4208; Total: 3512441.6772

**Persistent Segment Tree:**

Сырые данные:

1. i = 1:

Tree preparation time: [0.097, 6.267, 5.679, 5.53, 6.017, 5.896, 6.176, 6.766, 5.758, 5.563]

Persistent segment tree algorithm time: [268.691, 185.998, 170.843, 172.845, 173.311, 173.296, 175.543, 172.368, 172.285, 170.271]

1. i = 2:

Tree preparation time: [0.113, 5.545, 5.926, 6.89, 7.086, 5.616, 5.501, 6.575, 6.043, 5.596]

Persistent segment tree algorithm time: [224.191, 239.694, 247.835, 230.781, 218.212, 217.992, 218.39, 222.333, 218.991, 225.062]

1. i = 3:

Tree preparation time: [0.271, 6.342, 5.777, 6.398, 6.932, 5.757, 6.442, 6.469, 6.268, 5.883]

Persistent segment tree algorithm time: [271.758, 272.471, 256.33, 257.038, 257.092, 279.304, 256.928, 265.053, 260.789, 260.366]

1. i = 4:

Tree preparation time: [0.538, 10.127, 6.557, 6.767, 6.479, 6.194, 7.312, 7.135, 6.402, 6.373]

Persistent segment tree algorithm time: [331.923, 345.17, 305.218, 316.816, 312.83, 308.439, 312.775, 303.882, 308.73, 320.631]

1. i = 5:

Tree preparation time: [1.262, 9.149, 9.176, 9.98, 8.579, 9.434, 8.411, 8.401, 8.127, 8.392]

Persistent segment tree algorithm time: [356.524, 378.754, 397.223, 381.162, 356.224, 364.966, 360.942, 359.614, 359.728, 351.963]

1. i = 6:

Tree preparation time: [3.257, 13.147, 10.36, 10.038, 11.758, 11.6, 10.336, 10.748, 11.135, 10.689]

Persistent segment tree algorithm time: [390.159, 422.224, 387.474, 386.129, 388.469, 392.165, 385.54, 394.563, 388.871, 389.118]

1. i = 7:

Tree preparation time: [6.35, 16.802, 14.757, 16.688, 14.658, 14.408, 15.275, 33.5, 14.889, 14.979]

Persistent segment tree algorithm time: [459.593, 467.529, 455.121, 456.845, 475.678, 460.314, 456.211, 456.808, 454.629, 452.432]

1. i = 8:

Tree preparation time: [16.534, 24.102, 26.08, 24.776, 28.22, 27.397, 46.59, 26.504, 23.78, 26.313]

Persistent segment tree algorithm time: [556.049, 529.727, 557.442, 541.301, 551.596, 538.846, 538.08, 533.7, 531.027, 528.189]

1. i = 9:

Tree preparation time: [38.228, 75.608, 78.159, 51.667, 74.337, 44.783, 75.21, 76.9, 75.105, 74.856]

Persistent segment tree algorithm time: [706.656, 744.149, 624.922, 617.669, 623.013, 603.814, 629.613, 633.78, 668.797, 610.033]

1. i = 10:

Tree preparation time: [87.24, 143.446, 140.868, 133.542, 134.775, 137.763, 138.338, 138.12, 138.957, 178.398]

Persistent segment tree algorithm time: [681.071, 676.949, 665.553, 673.988, 678.771, 687.21, 675.166, 683.977, 697.272, 680.256]

Округлённые средние значения данных:

1. i = 1: Tree preparation time: 5.3749; Persistent segment tree algorithm time: 183.5451; Total: 188.92;
2. i = 2: Tree preparation time: 5.4891; Persistent segment tree algorithm time: 226.3481; Total: 231.8372;
3. i = 3: Tree preparation time: 5.6539; Persistent segment tree algorithm time: 263.7129; Total: 269.3668;
4. i = 4: Tree preparation time: 6.3884; Persistent segment tree algorithm time: 316.6414; Total: 323.0298;
5. i = 5: Tree preparation time: 8.0911; Persistent segment tree algorithm time: 366.7100; Total: 374.8011;
6. i = 6: Tree preparation time: 10.3068; Persistent segment tree algorithm time: 392.4712; Total: 402.778;
7. i = 7: Tree preparation time: 16.2306; Persistent segment tree algorithm time: 459.5159; Total: 475.7465;
8. i = 8: Tree preparation time: 27.0296; Persistent segment tree algorithm time: 540.5957; Total: 567.6253;
9. i = 9: Tree preparation time: 66.4853; Persistent segment tree algorithm time: 646.2446; Total: 712.7299;
10. i = 10: Tree preparation time: 137.1447; Persistent segment tree algorithm time: 680.0213; Total: 817.166;

**Данные, по которым строились графики:**

1. Время, необходимое для предварительной обработки данных для алгоритмов на карте и на дереве:

data\_prep\_map = [14.1762, 14.4542, 16.3387, 30.0156, 135.0816, 924.2943, 7744.7838, 60917.2917, 462886.1548, 3511917.2564]

data\_prep\_segtree = [5.3749, 5.4891, 5.6539, 6.3884, 8.0911, 10.3068, 16.2306, 27.0296, 66.4853, 137.1447]

1. Время работы всех трёх алгоритмов без учёта предварительной подготовки данных:

brute\_alg = [35.1994, 58.2205, 103.1282, 180.7307, 335.2329, 687.4985, 1319.0234, 2646.6270, 5364.7586, 10853.1710]

without\_data\_prep\_map = [175.2737, 203.3031, 215.6513, 234.9336, 281.4977, 298.8988, 364.4440, 400.9335, 437.0173, 524.4208]

without\_data\_prep\_segtree = [183.5451, 226.3481, 263.7129, 316.6414, 366.7100, 392.4712, 459.5159, 540.5957, 646.2446, 680.0213]

1. Время работы всех трёх алгоритмов с учётом предварительной подготовки данных:

brute\_alg = [35.1994, 58.2205, 103.1282, 180.7307, 335.2329, 687.4985, 1319.0234, 2646.6270, 5364.7586, 10853.1710]

with\_data\_prep\_map = [189.4499, 217.7573, 231.990, 264.9492, 416.5793, 1223.1931, 8109.2278, 61318.2252, 463323.1721, 3512441.6772]

with\_data\_prep\_segtree = [188.92, 231.8372, 269.3668, 323.0298, 374.8011, 402.778, 475.7465, 567.6253, 712.7299, 817.166]

**ВЫВОДЫ:**

* Исходя из данных и графика по подготовке данных для карты и персистентного дерева отрезков, можно отметить, что предварительная подготовка данных для метода персистентного дерева отрезков занимает меньше времени, чем для метода картой. Действительно, так как сложность по времени предварительной подготовки данных для карты – O(N^3), а для персистентного дерева отрезков – O(N\*log(N)).
* Говоря о графике времени работы без учёта предварительной подготовки данных, видно, что при небольших объемах данных (при степени 2-ки < 5, кол-во прямоугольников < 32) время работы алгоритма перебора сравнительно меньше, чем показатели двух других алгоритмов, время работ методов карты и дерева сравнимы. Однако, с увеличением размера входных данных (при степени 2-ки ≥ 5, кол-во прямоугольников ≥ 32) время работы алгоритма полного перебора стремительно возрастает, в то время как временные показатели работ двух других алгоритмов сохраняют стабильную тенденцию. Несмотря на одинаковую асимптотику поиска O(M \* logN) у методов картой и дерева, алгоритм, использующий карту, более эффективен. Это объяснимо отсутствием затрат на построение структуры данных: для алгоритма на карте не требуется дополнительного времени на построение структуры данных, так как он просто создает двумерный массив и заполняет его значениями на основе координат прямоугольников. Также такой результат может следовать из того, что алгоритм на персистентном дереве отрезков сначала выполняет операцию бинарного поиска два раза для нахождения индексов x и y координат точки в массивах сжатых координат, а затем производит спуск по дереву. Это приводит к тому, что константа перед логарифмом оказывается больше, чем у алгоритма, использующего карту.
* Говоря о графике времени работы с учётом предварительной подготовки данных, видно, что при небольших объемах данных (при степени 2-ки < 6, кол-во прямоугольников < 64) время работы алгоритма перебора сравнительно меньше, чем показатели двух других алгоритмов. На небольших размерах данных алгоритм картой справляется чуть быстрее, чем алгоритм персистентного дерева отрезков. Однако, с увеличением размера входных данных (при степени 2-ки ≥ 6, кол-во прямоугольников ≥ 64) время работы алгоритма полного перебора и карты стремительно возрастает, в то время как временные показатели работы персистентного дерева отрезков сохраняют стабильную тенденцию. На больших входных данных дольше всех работает алгоритм карты – у этого алгоритма дольше всех работает предварительная подготовка данных: O(N^3). Лучше и быстрее всех на входных данных больших размеров работает алгоритм персистентного дерева отрезков. Это может быть объяснимо тем, что, несмотря на необходимость реализации специальной структуры данных для этого алгоритма, он обеспечивает высокую производительность и оптимальное использование ресурсов.
* В заключение, можно отметить, что на входных данных небольших размеров лучше всего справляется алгоритм полного перебора без затрат дополнительной памяти и времени на предварительную подготовку данных. Алгоритм картой справляется быстрей персистентного дерева отрезков на входных данных небольших размеров, но хуже остальных алгоритмов на больших входных данных. Алгоритм персистентного дерева отрезков на небольших данных справляется дольше карты и полного перебора, но быстрее всех на входных данных больших размеров.
* Дополнительно: Не стоит отрицать, что выбор языка программирования, встроенный компилятор, как и среда разработки (в данном случае Python и PyCharm) могут влиять на временные показания. Замеры происходили без параллельно работающих программ.