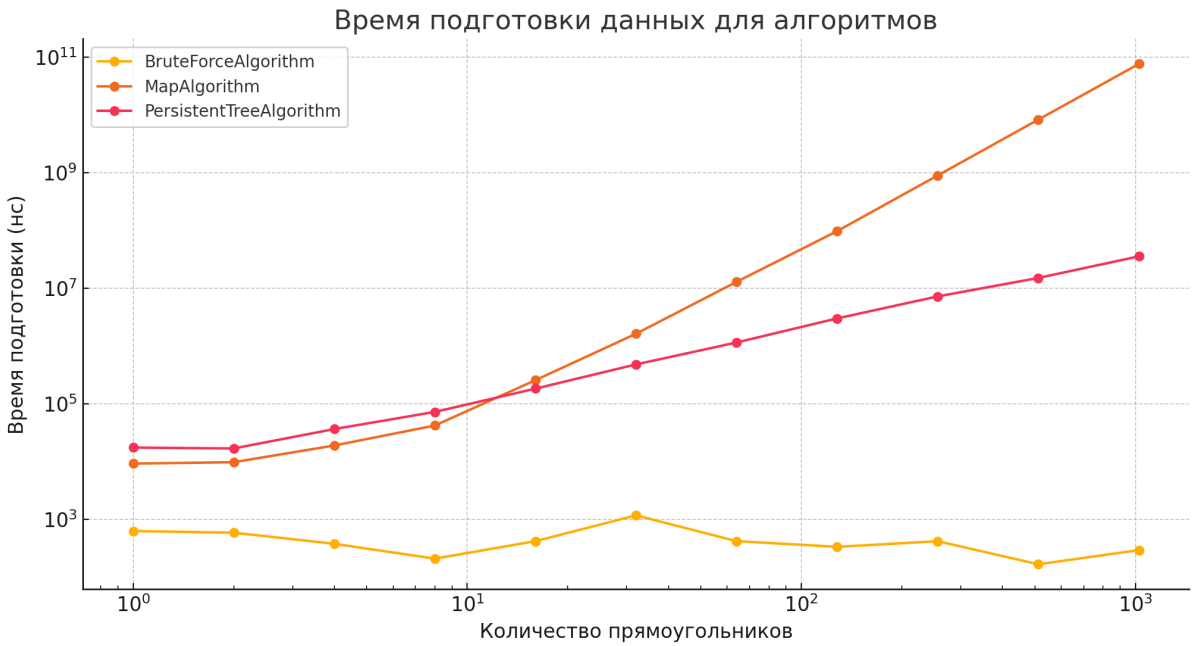

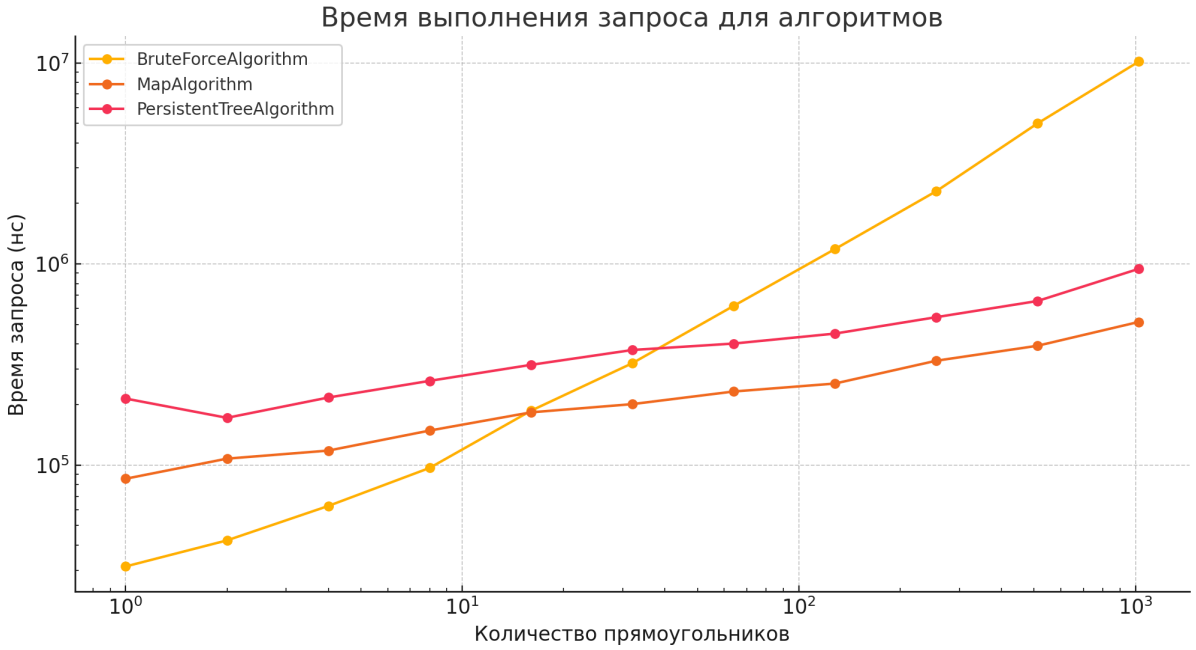

Анализ результатов

Время выполнения запросов

1. **BruteForceAlgorithm**:
   - Наихудшее время выполнения запросов среди всех трех алгоритмов, особенно при большом количестве запросов.
2. **MapAlgorithm**:
   - Хорошие результаты для небольшого количества запросов, но производительность снижается при увеличении числа запросов.

3. **PersistentTreeAlgorithm**:
    - Медленный рост времени выполнения запросов, что делает его наиболее эффективным для задач с большим количеством запросов.

BruteForceAlgorithm:предпроцессинг O(1),поиск O(MN)

Реализация:

1. Инициализация списка прямоугольников.
2. Проверка каждой точки на принадлежность каждому прямоугольнику.
3. Возврат количества прямоугольников, которым принадлежит точка.

MapAlgorithm

- **Препроцессинг**: O(N^3)
    1. Сжатие координат всех угловых точек прямоугольников по осям x и y.
    2. Построение карты (матрицы) размером количество сжатых точек по оси x на количество сжатых точек по оси y.
    3. Заполнение карты: обход всех прямоугольников и увеличение на единицу значения в ячейке карты, соответствующей сжатым координатам проекции каждого прямоугольника.
- **Поиск**: O(M * logN)
    1. Бинарный поиск сжатых координат точки запроса по осям x и y.
    2. Обращение к ячейке карты, соответствующей найденным сжатым координатам.

PersistentTreeAlgorithm

- **Препроцессинг**: O(N * logN)
    1. Сжатие координат всех угловых точек прямоугольников по осям x и y.
    2. Создание структуры Event, содержащей начало или конец существования прямоугольника и сжатые координаты.
    3. Построение дерева отрезков на сжатых координатах.
    4. Добавление в дерево отрезков персистентных узлов с использованием событий из структуры Event.
- **Поиск**: O(M * logN)
    1. Бинарный поиск сжатых координат точки запроса по осям x и y.
    2. Нахождение нужного корня дерева отрезков по сжатым координатам точки запроса.
    3. Обход дерева отрезков до нужного листа с использованием сжатых координат точки запроса.

Заключение

При увеличении количества прямоугольников время выполнения алгоритма на карте (MapAlgorithm) резко увеличивается из-за высокой асимптотической сложности построения карты - O(N^3). Хотя построение персистентного дерева (PersistentTreeAlgorithm) на небольших наборах данных занимает примерно столько же времени, что и построение карты, алгоритм на дереве демонстрирует значительно более медленный рост времени выполнения с ростом количества данных. Это означает, что алгоритм на дереве может эффективно выполнять предварительную обработку даже для больших наборов прямоугольников, в отличие от алгоритма на карте.

Алгоритм полного перебора лучше всего подходит для небольших наборов данных, так как он не требует предварительной подготовки данных. MapAlgorithm обеспечивает удовлетворительную производительность для небольшого количества прямоугольников. PersistentTreeAlgorithm демонстрирует наилучшую производительность при работе с большими наборами данных.