

Санкт-Петербургский политехнический университет Петра Великого  
Институт машиностроения, материалов и транспорта  
Высшая школа автоматизации и робототехники

Курсовая работа  
«Алгоритм Бентли-Оттманна»

Дисциплина: «Объектно-ориентированное программирование»

Студент гр. 3331506/90401

Мымрин А.В.

Преподаватель

Ананьевский М. С.

Санкт-Петербург

2022

## ВВЕДЕНИЕ

Вычислительная геометрия – раздел информатики, изучающий алгоритмы, используемые для решения геометрических задач. В данной работе рассмотрен алгоритм Бентли-Оттманна, который, принимая в качестве входных данных множество плоских отрезков, определяет все точки пересечения этих отрезков за время  $O((n + k) \log(n))$ , где  $k$  – количество точек пересечения. Важно отметить, что алгоритм не предполагает наличия вертикальных отрезков в наборе входных данных, также не должно быть пересечения более двух отрезков в одной точке. Данный алгоритм может применяться в робототехнике, а также при разработке игр.

## ОПИСАНИЕ АЛГОРИТМА

Принцип действия алгоритма заключается в следующем: вдоль набора отрезков слева направо движется вертикальная прямая, которая называется выметающей. Пересекая отрезки при определенной координате  $x$ , выметающая прямая помогает упорядочить те отрезки, которые она пересекает, по координате  $y$ , т.е. по высоте.

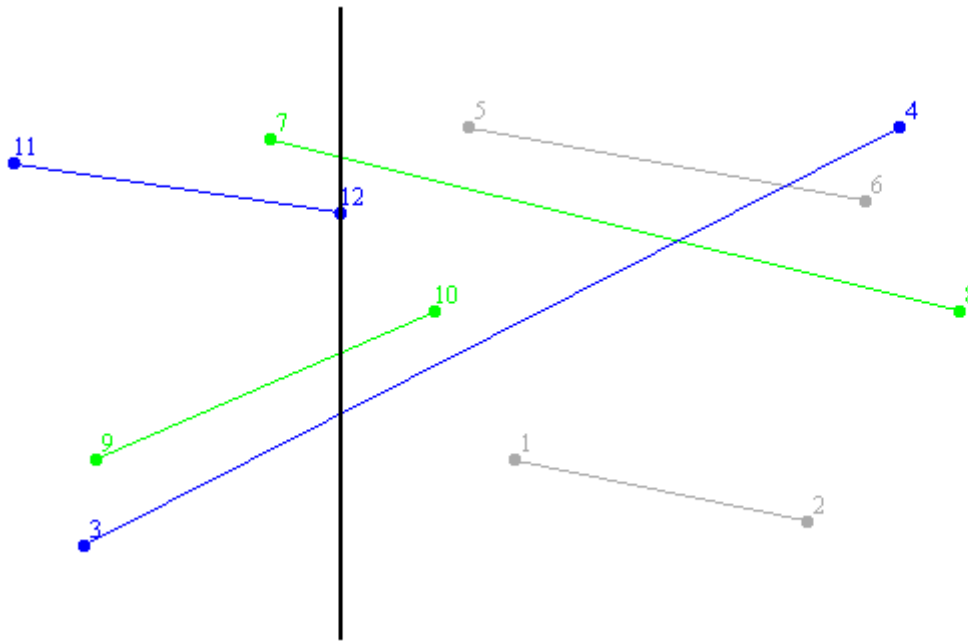


Рисунок 1 – Выметающая прямая

Когда выметающая прямая пересекает начальную точку какого-либо отрезка, он заносится в «очередь» – динамическую структуру данных, в которой по высоте отсортированы все отрезки, которые выметающая прямая пересекает в данный момент. Соответственно, когда прямая достигает конечной точки отрезка, он из очереди удаляется. В случае, когда выметающая прямая находится в точке пересечения двух отрезков, в очереди они меняются местами друг с другом, так как меняется их взаимное расположение.

Так как состояние отрезков относительно выметающей прямой изменяется только в описанных выше случаях, имеет смысл рассматривать состояние очереди только в этих точках. Перед началом работы алгоритма занесем все начальные и конечные точки в другую динамическую структуру

данных – расписание точек-событий. В ней все концы отрезков, а также занесенные туда в будущем точки пересечений отсортированы по координате  $x$ . Сам алгоритм описывает представленный ниже псевдокод. В нем  $Q$  – расписание точек-событий,  $R$  – очередь.

```
FOREACH point  $p$  in  $Q$  (в порядке возрастания  $x$ )
  DO IF  $p$  – левая конечная точка отрезка  $s$ 
    THEN вставить в  $R$ ;
    проверить, пересекает ли  $s$  отрезки непосредственно выше и ниже него, в случае,
    если он пересекает отрезок  $t$ , вставить точку пересечения  $s$  и  $t$  в  $Q$  (в порядке  $x$ )
  ELSE IF  $p$  – правая конечная точка отрезка  $s$ 
    THEN IF точка пересечения пары отрезков непосредственно выше и ниже  $s$  не
    находится в  $Q$  THEN проверить их на пересечение, и если они пересекаются, то
    добавить их точку пересечения в  $Q$  (in  $x$ -order); удалить отрезок из  $R$ 
  ELSE //  $p$  – точка пересечения отрезков  $s$  и  $t$ , сообщить о паре пересечений; поменять
    местами позиции  $s$  и  $t$  в  $R$ 
    (обратите внимание, что они были и остаются смежными);
    проверить верхний отрезок (скажем,  $s$ ) на пересечение с отрезком над ним, нижний
    отрезок ( $t$ ) с отрезком под ним, любые точки пересечений добавить в  $Q$ 
```

Для определения, пересекаются ли два отрезка, используют векторное произведение. Два отрезка пересекаются, если каждый из них пересекает прямую, содержащую другой отрезок.

## ИССЛЕДОВАНИЕ АЛГОРИТМА

Для исследования алгоритма с помощью библиотеки <random> генерировался случайный набор отрезков. Время работы алгоритма измерялось с помощью библиотеки <ctime>. Динамические структуры данных реализованы с помощью библиотеки <map> с использованием красно-черных деревьев.

В таблице 1 представлены экспериментальные данные.

Таблица 1 – экспериментальные данные

n, отрезков	t, мс	k, точек пересечения
100	4	26
1000	84	289
5000	1054	1415
7500	1996	2119
10000	3622	2778
15000	5926	4221
20000	8627	5621
25000	11949	7064
30000	16312	8505
35000	20966	9922

На рисунке 2 изображен график зависимости времени работы алгоритма от количества отрезков.

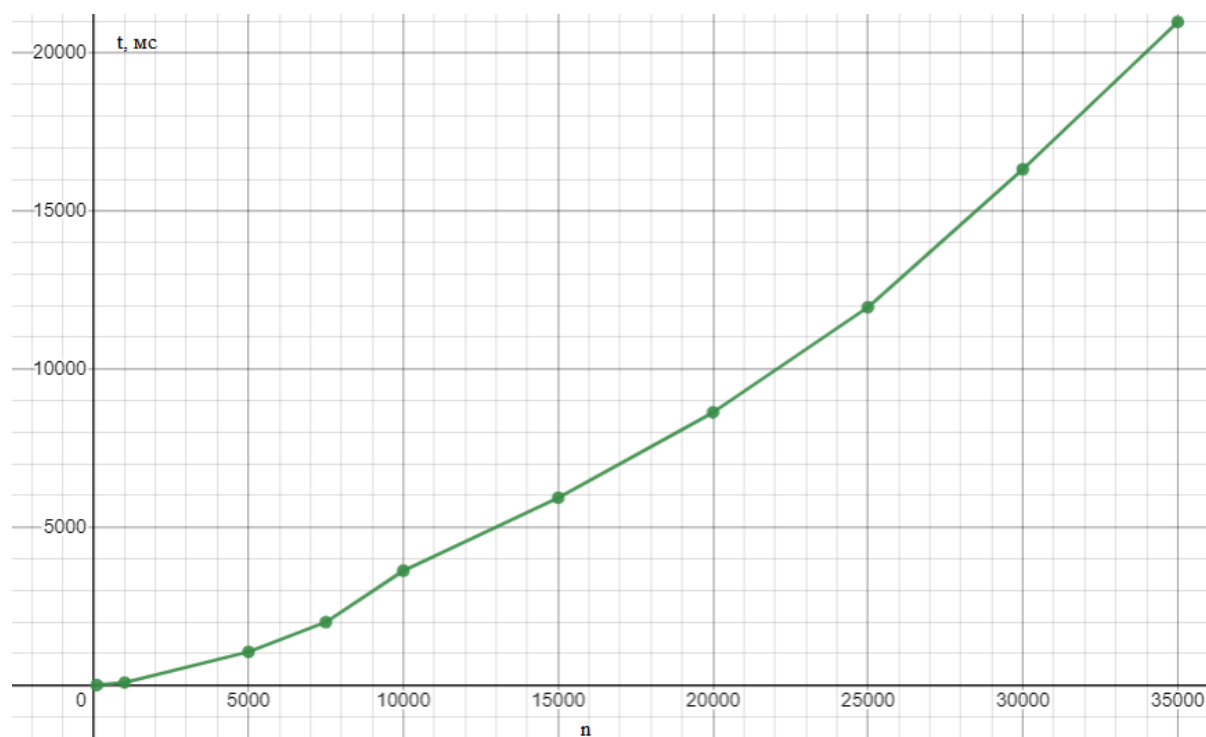


Рисунок 2 – Зависимость времени работы алгоритма от количества отрезков

Как видно из графика, алгоритм имеет сложность  $O(n \log(n))$ .

В случае, когда отрезки многократно друг с другом пересекаются, время работы алгоритма значительно снижается. В таблице 2 приведены экспериментальные данные.

Таблица 2 – экспериментальные данные при многократных пересечениях

n, отрезков	t, мс	k, точек пересечения
10	1	10
25	19	103
50	233	390
75	641	793
100	1565	1301
150	5597	2926
200	10370	4806
250	20105	7130
300	40536	10624
350	85451	14606
400	144170	18455

На рисунке 3 изображен график эксперимента.

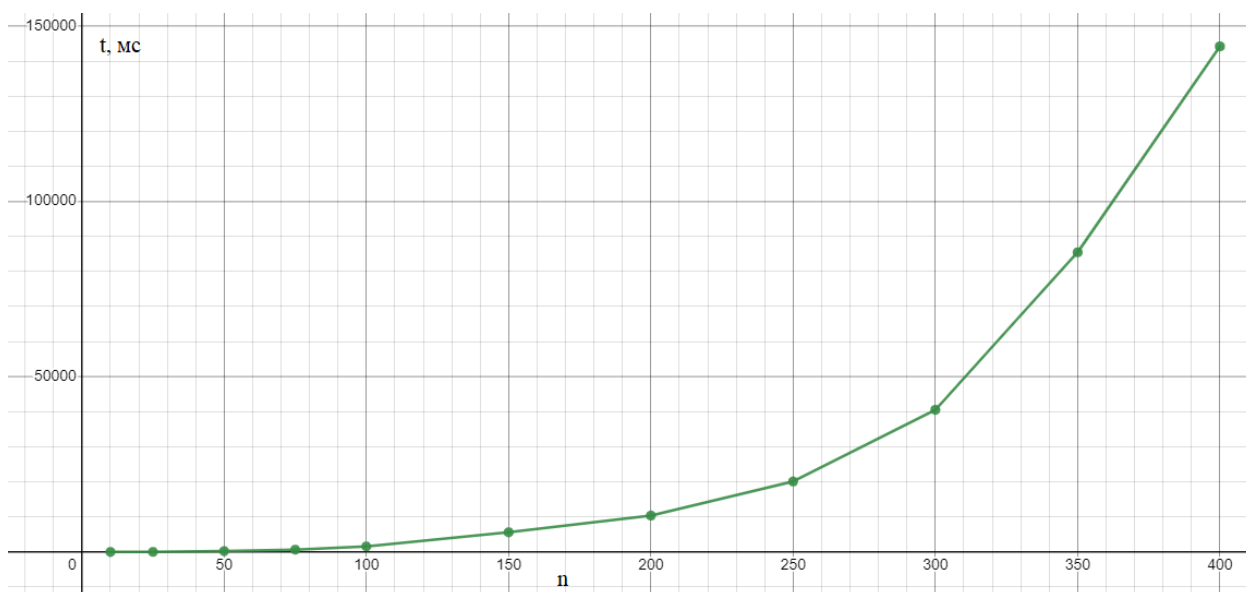


Рисунок 3 – Эксперимент при многократных пересечениях

## ЗАКЛЮЧЕНИЕ

Был рассмотрен алгоритм по поиску всех точек пересечений множества отрезков. Результаты экспериментов показали, что алгоритм становится малоэффективен, если отрезки многократно пересекаются. Скорость алгоритма также зависит от реализации динамических структур данных.

## СПИСОК ЛИТЕРАТУРЫ

1. Jon L. Bentley, Thomas A. Ottmann. Algorithms for Reporting and Counting Geometric Intersections. // IEEE transactions on computers. –1979. –vol. C-28. pp. 643-647.
2. Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. Алгоритмы. Построение и анализ — 2-е изд. — “Вильямс”, 2005. — 1296 с.