

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Дальневосточный федеральный университет»**

(ДВФУ)

|  |
| --- |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**  **Департамент математического и компьютерного моделирования** |

**ДОКЛАД**

о практическом задание по дисциплине «АИСД»

«Дерево интервалов»

НАПРАВЛЕНИЕ ПОДГОТОВКИ

09.03.03 «Прикладная информатика»

«Прикладная информатика в компьютерном дизайне»

|  |  |  |
| --- | --- | --- |
|  |  | Выполнил студент  гр. Б9121-09.03.03пикд  Чурганов Никита Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Доклад защищен:  С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | *(подпись)*  Руководитель практики  Доцент ИМКТ А.С Кленин  *(должность, уч. звание)*  \_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)*  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022г. |
| Рег. № \_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |  |  |

г. Владивосток

2022г

**Оглавление**

[Глоссарий 3](#_Toc125148636)

[Введение 4](#_Toc125148637)

[1 История развития 6](#_Toc125148638)

[2 Описание алгоритма 7](#_Toc125148639)

[3 Построение дерева интервалов 8](#_Toc125148640)

[4 Операции над деревом 10](#_Toc125148641)

[4.1 Добавление 10](#_Toc125148642)

[4.2 Проверка на перекрытие 10](#_Toc125148643)

[4.3 Удаление узла 12](#_Toc125148644)

[5 Реализация 15](#_Toc125148645)

[5.1 Добавление 15](#_Toc125148646)

[5.2 Поиск перекрываемых интервалов 16](#_Toc125148647)

[5.3 Удаление 17](#_Toc125148648)

[6 Тестирование 18](#_Toc125148649)

[7 Исследование 21](#_Toc125148650)

[7.1 Добавления узла в дерево 21](#_Toc125148651)

[7.2 Анализ удаления узла из дерева 21](#_Toc125148652)

[7.2 Анализ удаления узла из дерева 22](#_Toc125148653)

[8 Формальная постановка задачи 23](#_Toc125148654)

[**Список литературы** 24](#_Toc125148655)

## Глоссарий

**Коллекция** в программировании — программный объект, содержащий в себе, тем или иным образом, набор значений одного или различных типов, и позволяющий обращаться к этим значениям. [23]

**Метод** — это блок кода, содержащий ряд инструкций. Программа инициализирует выполнение инструкций, вызывая метод и указывая все аргументы, необходимые для этого метода. [22]

**Лист** — узел, не имеющий узлов-потомков на дереве. [21]

**Поддерево** — часть древообразной структуры данных, которая может быть представлена в виде отдельного дерева. [21]

**Кореневой** **узел** — узел, не имеющий предков (самый верхний). [21]

**Корень** — одна из вершин, по желанию наблюдателя. [21]

**Интервал** — множество всех чисел, удовлетворяющих строгому неравенству a <x <b. [25]

## Введение

#### Суть и назначение

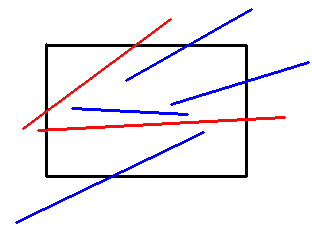
Интервальное дерево – это древовидная структура данных, узел которой представляет интервал и максимальное значение в поддереве, поэтому характеризуются начальным и конечным значениями. В частности, дерево интервалов позволяет эффективно находить все интервалы, которые перекрываются любым заданным интервалом (точкой).

#### Пример использования

Дерево интервалов используется для поиска видимых элементов внутри трехмерной сцены.

Имея набор отрезков и выровненный по оси прямоугольник, необходимо найти точки пересечения отрезков с прямоугольником. Эту проблему можно решить, используя деревья интервалов в сочетании с деревьями диапазонов.[14]

Деревья диапазонов — это эффективная структура данных для поиска набора точек, присутствующих в диапазоне/прямоугольнике. Таким образом, их можно использовать для поиска всех сегментов линии, так что одна из конечных точек каждого сегмента линии присутствует в прямоугольнике. Они соответствуют сегментам синей линии на рисунке 1.

Деревья интервалов можно использовать для поиска тех сегментов, которые пересекаются с прямоугольником, но конечные точки которых находятся за пределами окна. Это красные сегменты на рисунке.

Если рассмотреть ограниченную версию задачи, в которой все отрезки горизонтальны или вертикальны, то в таком случае любой горизонтальный отрезок, пересекающий прямоугольник, должен пересекать левый (правый) вертикальный край прямоугольника. Если представить горизонтальные сегменты как интервалы, а вертикальный край прямоугольника как точку, проблема состоит в том, чтобы найти все интервалы, содержащие эту точку. Таким образом, задача решается, используя интервальные деревья. Точно так же находятся все пересекающиеся вертикальные отрезки.

## 1 История развития

Точной информации в открытых источниках нет, но можно предположить, что дерево интервалов появилось примерно в 1979 г. Это можно обосновать тем, что бинарное дерево поиска появилось в 1960 г. и бинарное дерево поиска более простая структура данных в сравнении с деревом интервалов, так же есть такая структура данных, как дерево диапазонов, которая отдаленно похожа на дерево интервалов и появилось в 1979 г.

## 2 Описание алгоритма

Дерево интервалов позволяет решать следующую задачу. Дано множество отрезков и множество запросов. Каждый запрос характеризуется точкой . Для каждого запроса необходимо определить множество отрезков из , которые содержат в себе . Построение дерева интервалов занимает время , а также памяти. На каждый запрос дерево интервалов позволяет отвечать за , где – размер ответа на запрос.

## 3 Построение дерева интервалов

Изначально дерево не заполнено, поэтому берется произвольный интервал [5;10] и вставляется в дерево. Таким образом, дерево состоит из одного узла, который является корневым узлом (рисунок 1).



Рисунок 1 – Добавление первого узла

Затем берется следующий произвольный интервал , и проверяется следующее условие: так как , то узел идет в левое поддерево (рисунок 2).

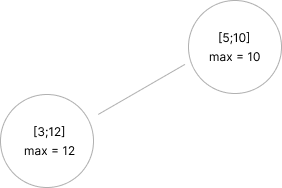


Рисунок 2 – Добавление узла в дерево

Далее проверяется максимум в левом поддереве и в правом при их наличии, если максимум в левом поддереве больше максимума в правом поддереве и максимума в корне, то в корень записывается максимум левого поддерева, если максимум в правом поддереве больше, то в корень записывается максимум правого поддерева, иначе в корень записывается самого корня (рисунок 3).

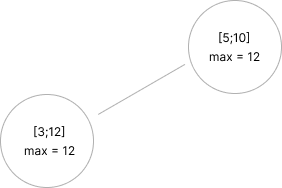


Рисунок 3 – Вычисление максимума в дереве

Аналогично предыдущим действиям добавляются другие узлы

(рисунок 4).

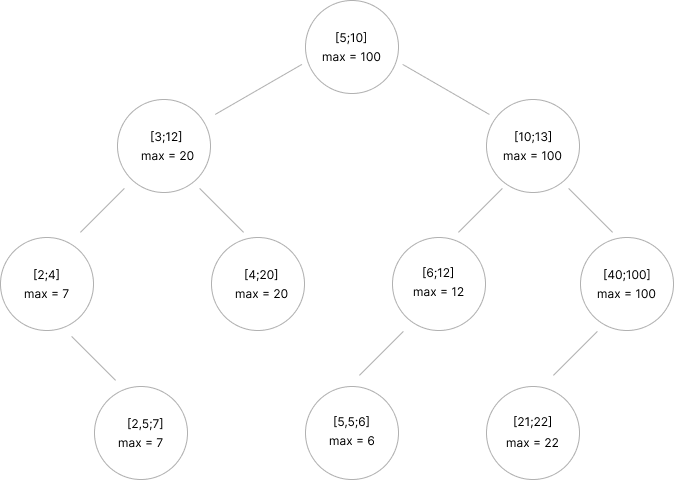


Рисунок 4 – Полученное дерево

## 4 Операции над деревом

1. Добавление
2. Проверка на перекрытие
3. Удаление

## 4.1 Добавление

Для добавления узла в дерево берется интервал, который необходимо добавить, и если нижняя граница интервала меньше нижней границы интервала в корне или равна ей, то узел идет влево, иначе – вправо. Данные действия выполняются до тех пор, пока добавляемый узел не станет листом (рисунок 1 – рисунок 3).

## 4.2 Проверка на перекрытие

Для проверки на перекрытие дерево должно иметь хотя бы один узел. Проверка делается через сравнение границ интервала у узла в дереве и границ заданного интервала, который проверяется на то, какие интервалы он перекрывает.

Условные обозначения, используемые далее:

- интервал , - интервал ;

– произвольный интервал, который проверяется на то, какие интервалы он перекрывает и имеет параметры и ;

– нижняя граница интервала ;

– верхняя граница интервала ;

– интервал из дерева, который имеет параметры и ;

– нижняя граница интервала ;

– верхняя граница интервала .

Варианты, которые входят в проверку:

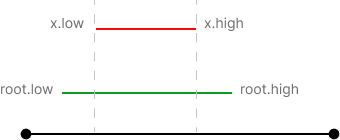
1. Интервал частично перекрывает интервал , при этом не выходит за границы (рисунок 5).

Рисунок 5 – Случай 1

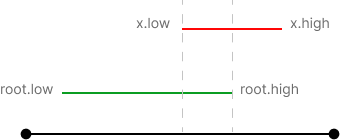
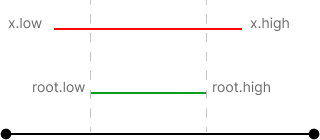
1. Интервал частично перекрывает интервал , при этом выходит за правую границу (рисунок 6).

Рисунок 6 – Случай 2

1. Интервал полностью перекрывает интервал , при этом выходит границы (рисунок 7).

Рисунок 7 – Случай 3

1. Интервал не перекрывает интервал , при этом лежит левее интервала (рисунок 8).

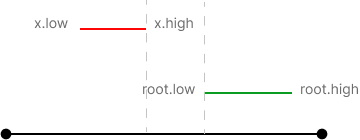


Рисунок 8 – Случай 4

1. Интервал не перекрывает интервал , при этом выходит за левую границы (рисунок 9).

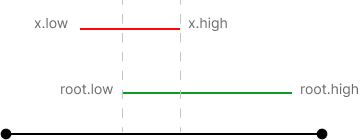


Рисунок 9 – Случай5

1. Интервал не перекрывает интервал , при этом x лежит правее интервала (рисунок 10).

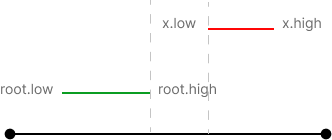


Рисунок 10 – Случай 6

## 4.3 Удаление узла

Для удаления узла рассматривается несколько ситуаций: первая ситуация – узел является листом, вторая ситуация – у узла есть хотя бы один сын.

1. Узел является листом.

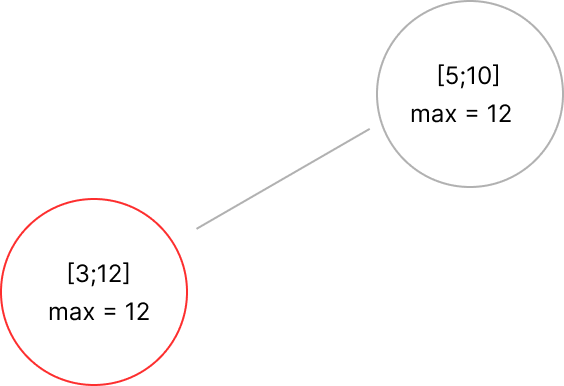


Рисунок 11 – Удаление узла

Удаляемый узел – . Если узла нет обоих сыновей, то лист удаляется и максимум в узле на уровень выше пересчитывается. Таким образом в дереве (рисунок 11) остаетсмя только один узел - (рисунок 12).



Рисунок 12 – Результат удаления

1. Узел не является листом

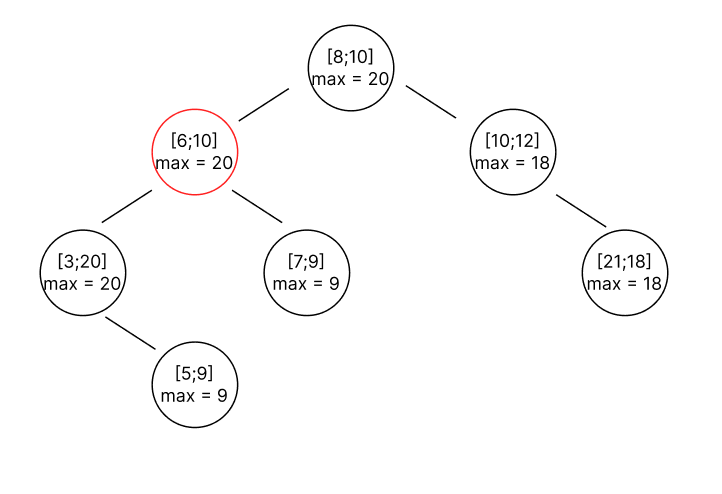


Рисунок 13 – Удаление узла [6;10], max = 20

Удаляемый узел – . Перед тем, как удалить узел необходимо проверить есть ли сыновья у этого узла. В данном случаи у удаляемого есть оба сына. Тогда этому узлу присваиваются значения следующего узла по величение параметра .

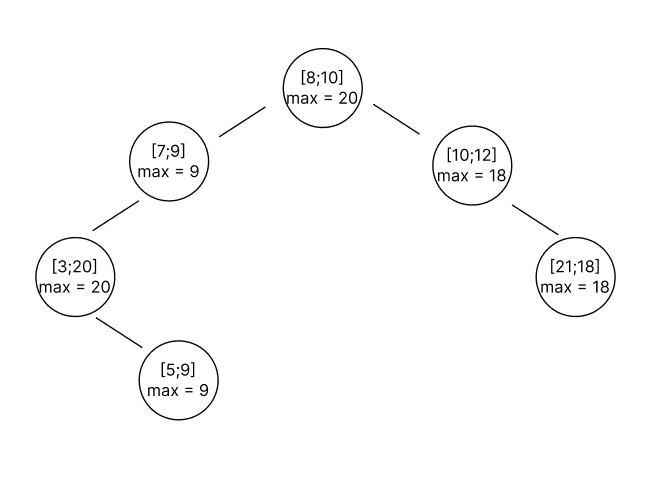


Рисунок 14 – Результат удаления

## 5 Реализация

## 5.1 Добавление

, где – дерево, – добавляемый интервал

1. public Node Insert(Interval x)
2. {
3. if (this == null)
4. return new Node(x, x.high);
5. else
6. {
7. this.\_Insert(this, x);
8. return this;
9. }
10. }
11. Node \_Insert(Node root, Interval x)
12. {
13. if (root.range.Equals(x))
14. {
15. return null;
16. }
17. else
18. {
19. if (root.max < x.high)
20. root.max = x.high;
21. if (root.range.low >= x.low)
22. {
23. if (root.left == null)
24. {
25. root.left = new Node(x, x.high);
26. return root;
27. }
28. else
29. this.\_Insert(root.left, x);
30. }
31. else
32. {
33. if (root.right == null)
34. {
35. root.right = new Node(x, x.high);
36. return root;
37. }
38. else
39. this.\_Insert(root.right, x);
40. }
41. return root;
42. }
43. }

## 5.2 Поиск перекрываемых интервалов

, где – дерево, – интервал

1. public string Search(Interval x)
2. {
4. if (this.range.low >= x.low && this.range.high

<= x.high)

1. {
2. msg += $"{x} полностью перекрывает

{this.range}\n";

3. }
4. else if (this.range.low > x.high ||

this.range.high < x.low)

1. {
2. msg += $"{x}  перекрывает {this.range}\n";
4. }
5. else
6. {
7. msg += $"{x} не перекрывает {this.range}\n";
9. }
11. if (this.left != null)
12. {
13. this.left.Search(x);
14. }
15. if (this.right != null)
16. {
17. this.right.Search(x);
18. }
19. return msg;
21. }

## 5.3 Удаление

, где – дерево, –удаляемый интервал

1. **public** Node Delete(Interval x)
2. {
3. **if** (root == **null**)
4. {
5. **Console.WriteLine("Список интервалов пуст");**
6. **return** root;
7. }
8. **if** (root.range.low == x.low && root.range.high == x.high)
9. {
10. **if (root.left == null && root.right == null)**
11. {
12. root.range.low = 0;
13. root.range.high = 0;
14. root.max = 0;
16. **return** root;
17. }
18. **if** (root.left != **null** || root.right != **null**)
19. {
20. **root\_replace(root);**
21. **return** root;
22. }
24. Console.WriteLine("Узел в дереве");
25. **return root;**
26. }
27. **if** (root.range.low >= x.low && root.left != **null**)
28. {
29. **remove**(root.left, x);
30. **}**

**else** **if** (root.range.low <= x.low && root.right != **null**)

1. {
2. **remove**(root.right, x);
3. }
4. **else**
5. Console.WriteLine("Такого узла нет в дереве");
6. }

## 6 Тестирование

Тестирующая система состоит из трех частей, которые в сумме дают 37 тестов. Первая часть – это набор тестов (1–12) на добавление. Вторая часть – это набор тестов (13–27) на перекрытие. Третья часть – это набор тестов (28–37) на удаление.

Каждая часть состоит из кода, который сначала открывает файл из папки “in” (папка с входные данные, два числа, которые описывают границы интервала), затем считываются входные данные из файла, создаются интервалы на основе входных данных, и выполняются определенная последовательность действий, чтобы выполнить задачу. После того, как задача решена ответ записывает в файл из папки “out” и содержимое этого файла сравнивается с ожидаемым результатом, который храниться в папке “result”. Данная последовательность действий выполняется до тех пор, пока в папке “in” не будут прочитаны все файлы.

Папка с тестами имеет следующий вид:

Тесты (папка)

In – папка с входными данными

1.txt

…

37.txt

Out – папка с выходными данными

1.txt

…

37.txt

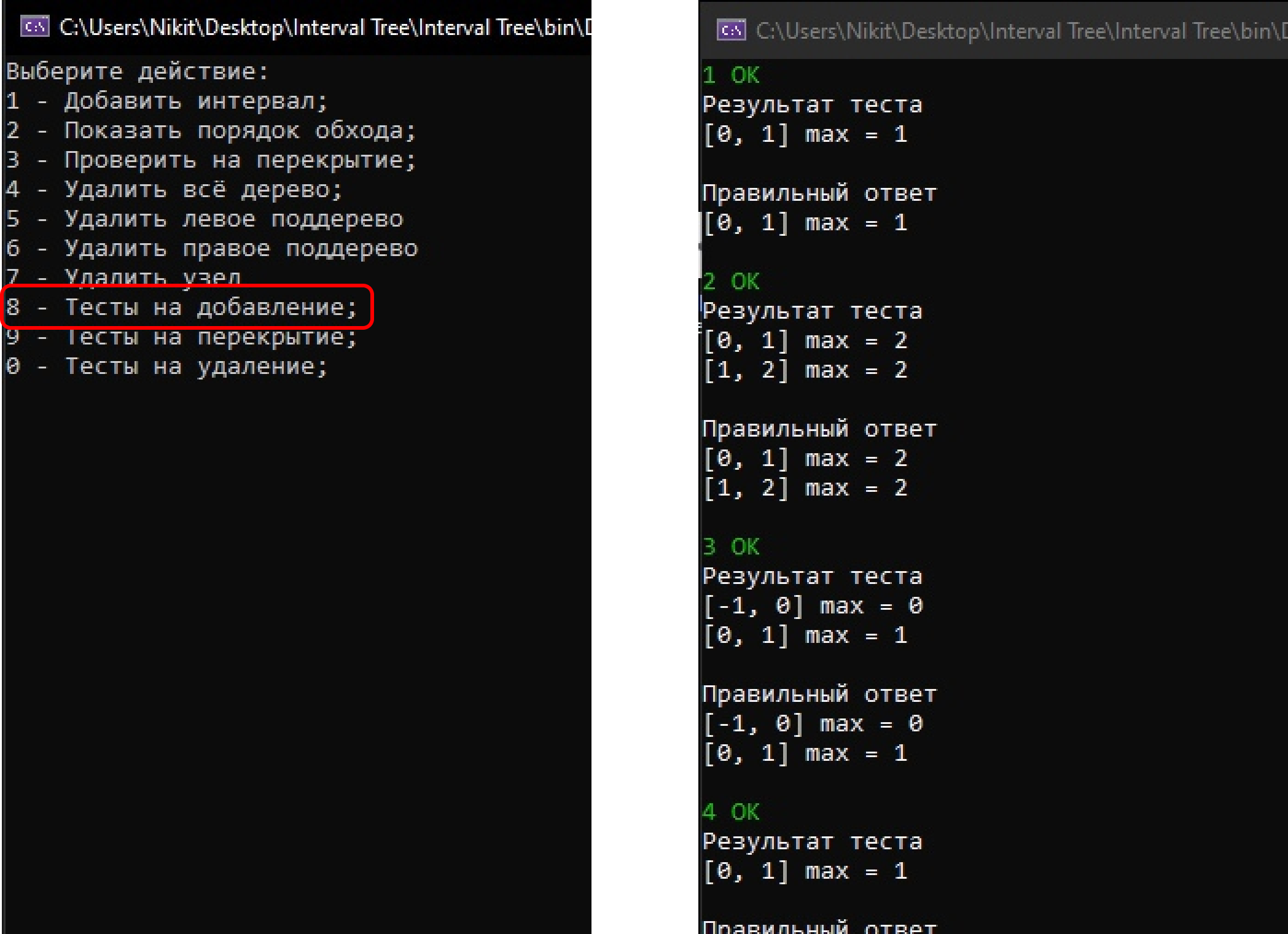
Result – папка с предполагаемыми результатами тестов

1.txt

…

37.txt

Названия файлов содержат номер теста, то есть результат, который был получен после работы программы на основе входных данных из файла 1, лежащего в папке “in”, записывается в файл 1 из папки “out”, затем содержимое из файла 1 из папки “out” сравнивается с содержимым файла 1 из папки “result”.

Пример запуска тестов:

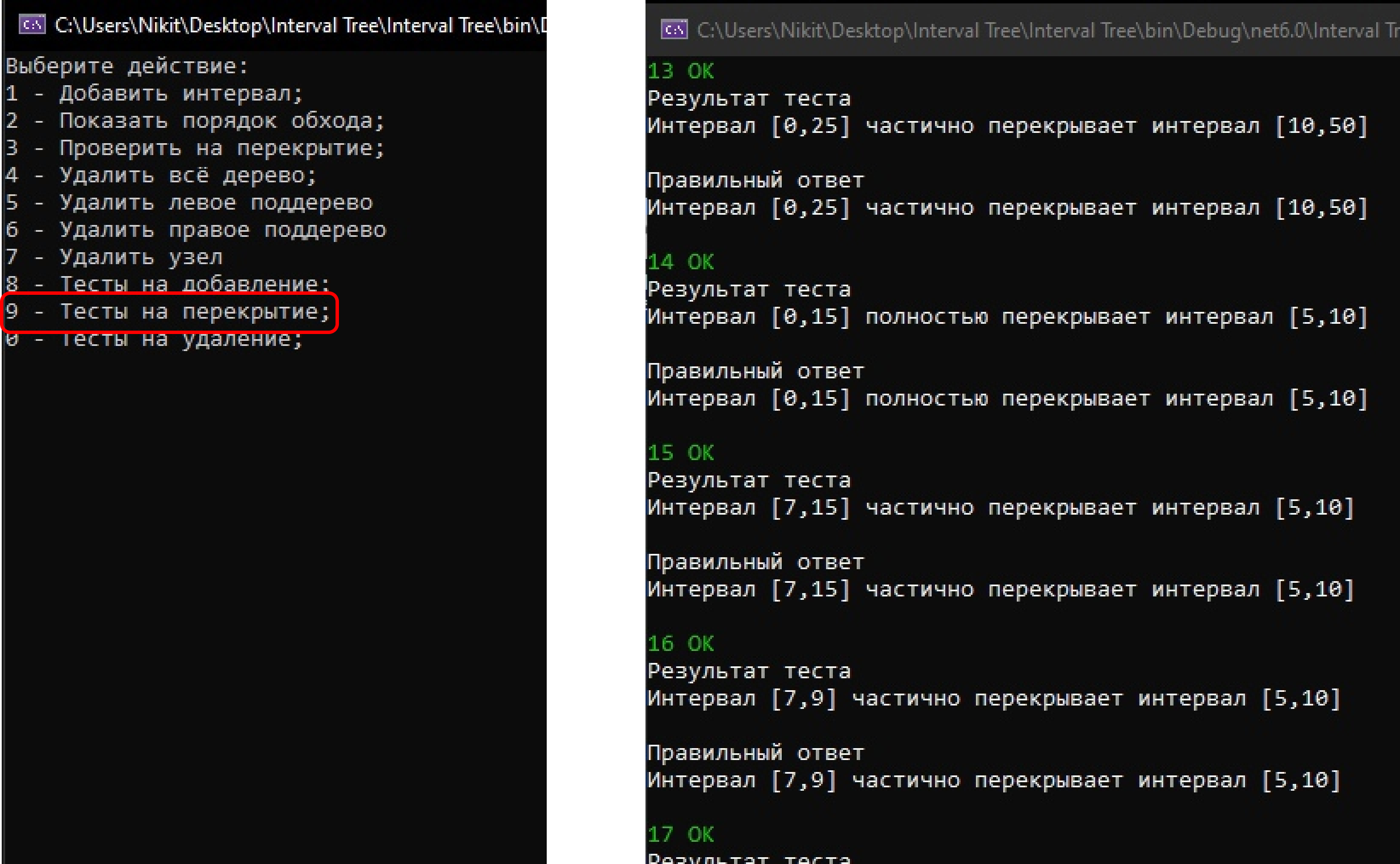
Рисунок 18. Тесты на добавление

Рисунок 19. Тесты на перекрытие

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 20. Тесты на удаление

## 7 Исследование

Исследование времени работы алгоритма состоит из трех частей, где первая часть – это анализ времени работы добавления узла в дерево, вторая часть – анализ времени работы удаления узла из дерева, третья часть – анализ поиска перекрываемых интервалов.

## 7.1 Добавления узла в дерево

Рисунок 21. Время работы добавления

## 7.2 Анализ удаления узла из дерева

Рисунок 22. Время работы удаления (удаляемого узла нет в дереве)

Рисунок 23. Время работы удаления (удаляемый узел есть в дереве)

## 7.2 Анализ удаления узла из дерева

Рисунок 24. Время работы проверки на перекрытие

## 8 Формальная постановка задачи

1. Изучить структуру данных “Дерево интервалов”;

2) Реализовать для дерева интервалов следующие операции:

1. Добавление

2. Удаление

3. Поиск перекрываемых интервалов

3) Выполнить исследование на производительность (время работы);

4) Реализовать систему автоматического тестирования;

5) Результаты проделанной работы выложить в среде GitHub:

1. Отчет

2. Презентация

3. Программный код

4. Тесты

5. Пакет CATS

# Список литературы

Электронные ресурсы

1. Дерево интервалов (interval tree) и пересечение точки с множеством интервалов [Электронный ресурс] // neerc.ifmo. — Режим доступа: [Дерево интервалов (interval tree) и пересечение точки с множеством интервалов — Викиконспекты (ifmo.ru)](https://neerc.ifmo.ru/wiki/index.php?title=%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%B2%D0%B0%D0%BB%D0%BE%D0%B2_(interval_tree)_%D0%B8_%D0%BF%D0%B5%D1%80%D0%B5%D1%81%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D1%82%D0%BE%D1%87%D0%BA%D0%B8_%D1%81_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE%D0%BC_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%B2%D0%B0%D0%BB%D0%BE%D0%B2)
2. Interval Tree. [Электронный ресурс] // geeksforgeeks. — Режим доступа: [Interval Tree - GeeksforGeeks](https://www.geeksforgeeks.org/interval-tree/)
3. Interval Trees: One step beyond BST [Электронный ресурс] // iq.opengenus. — Режим доступа: [Interval Trees: One step beyond BST (opengenus.org)](https://iq.opengenus.org/interval-tree/)
4. Interval tree [Электронный ресурс] // wikipedia. — Режим доступа: [Interval tree - Wikipedia](https://en.wikipedia.org/wiki/Interval_tree#:~:text=%D0%92%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B5%20%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%20%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%B2%D0%B0%D0%BB%D0%BE%D0%B2%2D%D1%8D%D1%82%D0%BE%20%D0%B4%D1%80%D0%B5%D0%B2%D0%BE%D0%B2%D0%B8%D0%B4%D0%BD%D0%B0%D1%8F,%D0%B2%D0%B8%D0%B4%D0%B8%D0%BC%D1%8B%D1%85%20%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2%20%D0%B2%D0%BD%D1%83%D1%82%D1%80%D0%B8%20%D1%82%D1%80%D0%B5%D1%85%D0%BC%D0%B5%D1%80%D0%BD%D0%BE%D0%B9%20%D1%81%D1%86%D0%B5%D0%BD%D1%8B)
5. node-interval-tree [Электронный ресурс] // npm. — Режим доступа: [node-interval-tree - npm (npmjs.com)](https://www.npmjs.com/package/node-interval-tree)
6. 5cript/interval – three [Электронный ресурс] // github. — Режим доступа: [GitHub - 5cript/interval-tree: A C++ header only interval tree implementation.](https://github.com/5cript/interval-tree)
7. INTERVAL TREES [Электронный ресурс] // dgp.toronto. — Режим доступа: [CSC378: Interval Trees (toronto.edu)](https://www.dgp.toronto.edu/public_user/JamesStewart/378notes/22intervals/)
8. Interval tree [Электронный ресурс] // cmu.edu. — Режим доступа: [intervaltrees.pdf (cmu.edu)](https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/intervaltrees.pdf)
9. Windowing queries [Электронный ресурс] // personal.us.es. — Режим доступа: [Lecture 8: Windowing queries (us.es)](https://personal.us.es/almar/cg/08windowing.pdf)
10. Interval tree [Электронный ресурс] // tutorialandexample. — Режим доступа: [Interval Tree - TAE (tutorialandexample.com)](https://www.tutorialandexample.com/interval-tree)
11. Interval tree [Электронный ресурс] // formulasearchengine. — Режим доступа: [Interval tree - formulasearchengine](https://www.formulasearchengine.com/wiki/Interval_tree)
12. Interval tree [Электронный ресурс] // TutorialCup. — Режим доступа: [Interval Tree - Interval Tree in Data Structure Interval Tree (tutorialcup.com)](https://www.tutorialcup.com/interview/tree/interval-tree.htm)
13. Динамические структуры данных [Электронный ресурс] // штегше. — Режим доступа: [Динамические структуры данных - интуит](https://intuit.ru/studies/courses/648/504/lecture/11455#:~:text=%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5%20%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D1%8B%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20%E2%80%93%20%D1%8D%D1%82%D0%BE%20%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D1%8B,%D0%B8%20%D1%85%D0%B0%D1%80%D0%B0%D0%BA%D1%82%D0%B5%D1%80%20%D1%81%D0%B2%D1%8F%D0%B7%D0%B5%D0%B9%20%D0%BC%D0%B5%D0%B6%D0%B4%D1%83%20%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D0%BC%D0%B8.)
14. Дерево диапозонов [Электронный ресурс] // wikibrief. — Режим доступа: [Дерево диапазонов - Range tree (wikibrief.org)](https://ru.wikibrief.org/wiki/Range_tree#:~:text=%D0%92%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B5%2C%20%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%20%D0%B4%D0%B8%D0%B0%D0%BF%D0%B0%D0%B7%D0%BE%D0%BD%D0%BE%D0%B2%20%2D,%D0%9B%D1%83%D0%B8%20%D0%91%D0%B5%D0%BD%D1%82%D0%BB%D0%B8%20%D0%B2%201979%20%D0%B3%D0%BE%D0%B4%D1%83)
15. Segment Tree Range Minimum Query [Электронный ресурс] // youtube. — Режим доступа: [Segment Tree Range Minimum Query - YouTube](https://www.youtube.com/watch?app=desktop&v=ZBHKZF5w4YU&t=61s)
16. Interval tree [Электронный ресурс] // homepages.math. — Режим доступа: [Interval Trees (uic.edu)](https://homepages.math.uic.edu/~jan/mcs481/windowing.pdf)
17. Дерево Интервалов (Отрезков) [Электронный ресурс] // coolsoftware. — Режим доступа: [Дерево Интервалов (Отрезков) | Cool Software Blog](https://blog.coolsoftware.ru/2014/01/blog-post/)
18. Interval tree [Электронный ресурс] // drdobbs. — Режим доступа: [Interval Trees | Dr Dobb's (drdobbs.com)](https://www.drdobbs.com/cpp/interval-trees/184401179)
19. Data Structures: Augmented Interval Tree to search for intervals overlapping [Электронный ресурс] // davismol. — Режим доступа: [Data Structures: Augmented Interval Tree to search for intervals overlapping | Dede Blog (davismol.net)](http://www.davismol.net/2016/02/07/data-structures-augmented-interval-tree-to-search-for-interval-overlapping/)
20. Augmented Interval Tree in C# [Электронный ресурс] // Software Salad. — Режим доступа: [Augmented Interval Tree in C# | Software Salad (wordpress.com)](https://brooknovak.wordpress.com/2013/12/07/augmented-interval-tree-in-c/)
21. Дерево (структура данных) [Электронный ресурс] // wikipedia. — Режим доступа: [Дерево (структура данных) — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85))
22. Методы [Электронный ресурс] // microsoft. — Режим доступа: [Методы. Руководство по программированию на C# | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/methods)
23. Коллекция [Электронный ресурс] // wikipedia. — Режим доступа: [Коллекция (программирование) — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))
24. Интервал [Электронный ресурс] // wikipedia. — Режим доступа: [Интервал — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%B2%D0%B0%D0%BB)
25. Оценка сложности алгоритмов [Электронный ресурс] // tproger. — Режим доступа: [Оценка сложности алгоритмов, или Что такое О(log n) (tproger.ru)](https://tproger.ru/articles/computational-complexity-explained/)
26. Временная сложность алгоритма [Электронный ресурс] // wikipedia. — Режим доступа: [Временная сложность алгоритма — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%92%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B0)
27. Типы данных [Электронный ресурс] // wikipedia. — Режим доступа: [Типы данных в С# —метанит(metanit.com)](https://metanit.com/sharp/tutorial/2.1.php)