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
_____
(подпись)
Руководитель практики
Доцент ИМКТ А.С Кленин
(должность, уч. звание)
_____
(подпись)
«_____» _____ 2022г.

Доклад защищен:
С оценкой _____

Рег. № _____

«____» _____ 2022 г.

г. Владивосток
2022г

Оглавление

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, узлы которой представляют интервалы и максимальное значение в поддереве поэтому характеризуются начальным и конечным значениями. В частности, дерево интервалов позволяет эффективно находить все интервалы, которые перекрываются с любым заданным интервалом или точкой.

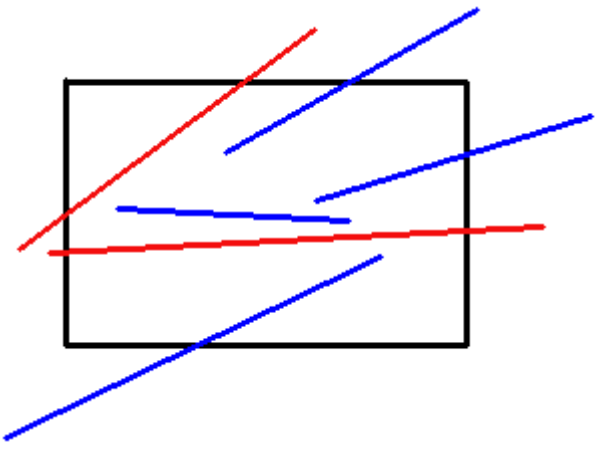
Перспектива использования:

Дерево интервалов используется для поиска видимых элементов внутри трехмерной сцены.

Имея набор отрезков линии и выровненный по оси прямоугольник, необходимо найти точки пересечения отрезков линии 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 ниже.

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. В этом случае любой горизонтальный отрезок, пересекающий прямоугольник, должен пересекать левый (и 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

Задача алгоритма состоит в том, чтобы посетить каждый интервал и проверить, пересекает ли он заданную точку или интервал, что требует O(n) время, где n – количество интервалов в коллекции. Поскольку запрос может возвращать все интервалы, например, если запрос представляет собой большой интервал, пересекающий все интервалы в коллекции. Интервальные деревья имеют время запроса O(log n + m), m – размер ответа на запрос, и начальное время создания O(n log n), ограничивая потребление памяти до O(n). После создания интервальные деревья могут быть динамическими [13], что позволяет эффективно вставлять и удалять интервал в O(log n) время. [1][26][27].

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 [3;12], и проверяется следующее условие: если 3 <= 5, то узел идет в левое поддерево, иначе – в правое поддерево (рисунок 2).

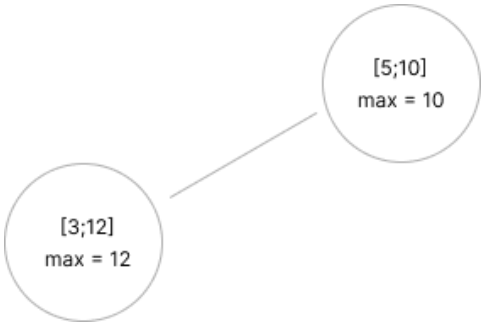


Рисунок 2 – Добавление узла в дерево

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).

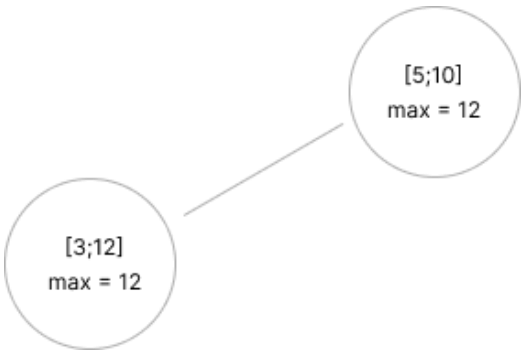


Рисунок 3 – Вычисление максимума в дереве

Аналогично предыдущим действиям добавляются другие узлы (рисунок 4).

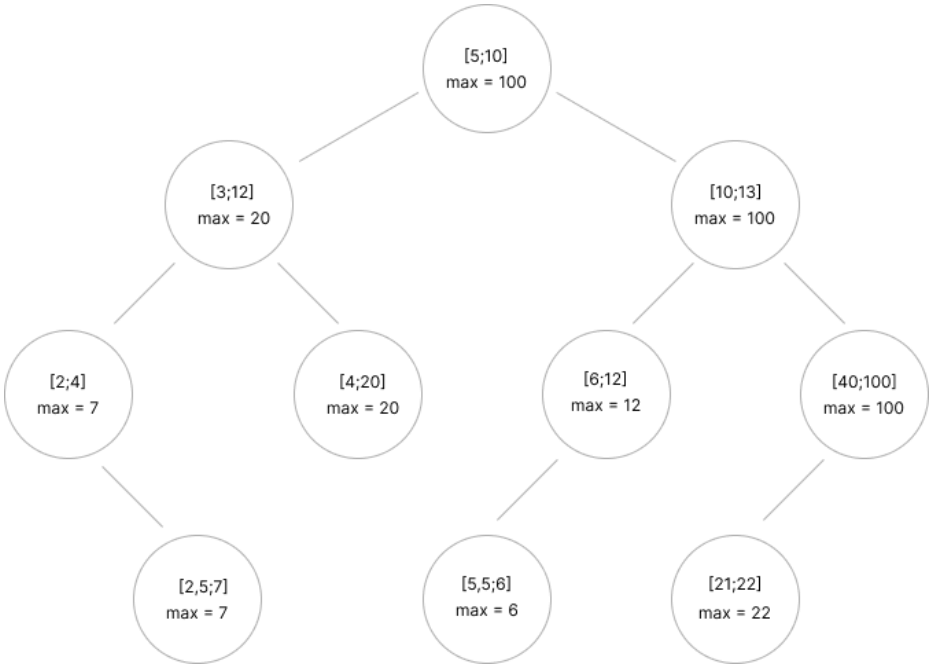


Рисунок 4 – Полученное дерево

4 Операции над деревом

1. Добавление
2. Проверка на перекрытие
3. Удаление

4.1 Добавление

Для добавления узла в дерево берется произвольный интервал, 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 x, —— - интервал root;

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

x.low – нижняя граница интервала x;

x.high – верхняя граница интервала x;

root – интервал из дерева, который имеет параметры low и high;

root.low – нижняя граница интервала root;

root.high – верхняя граница интервала root.

Варианты, которые входят в проверку:

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

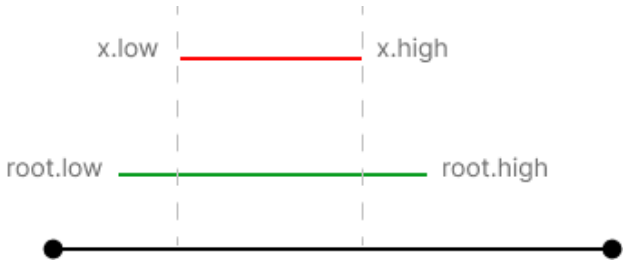


Рисунок 5 – Случай 1

2. Интервал x частично перекрывает интервал root, при этом x выходит за правую границы root (рисунок 6).

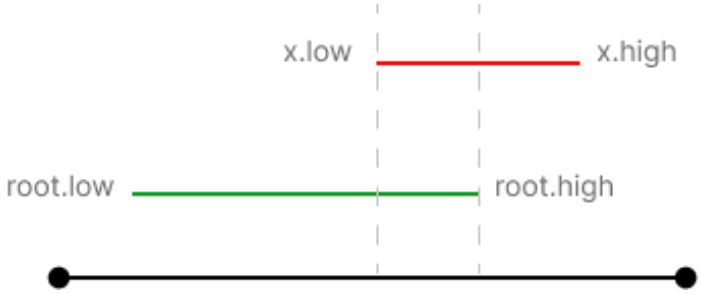


Рисунок 6 – Случай 2

3. Интервал x полностью перекрывает интервал root, при этом x выходит за границы root (рисунок 7).

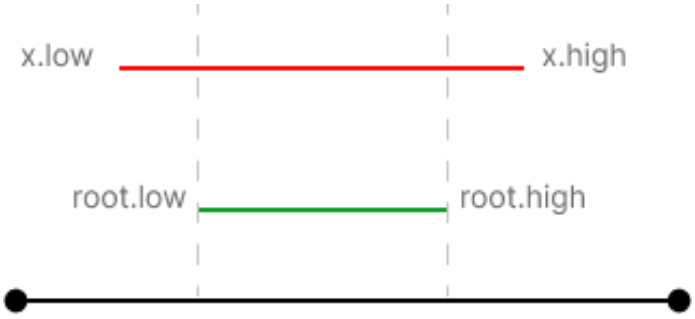


Рисунок 7 – Случай 3

4. Интервал x не перекрывает интервал root, при этом x лежит правее интервала root (рисунок 8).

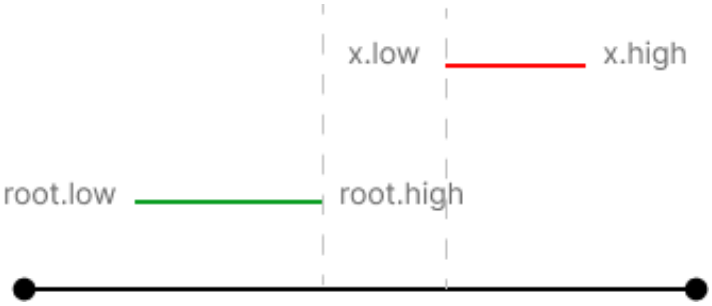


Рисунок 8 – Случай 4

5. Интервал x не перекрывает интервал root, при этом x лежит левее интервала root (рисунок 9).

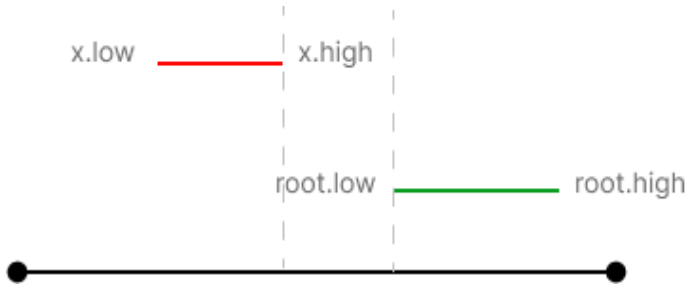


Рисунок 9 – Случай 5

6. Интервал x частично перекрывает интервал root, при этом x выходит за левую границы root (рисунок 10).

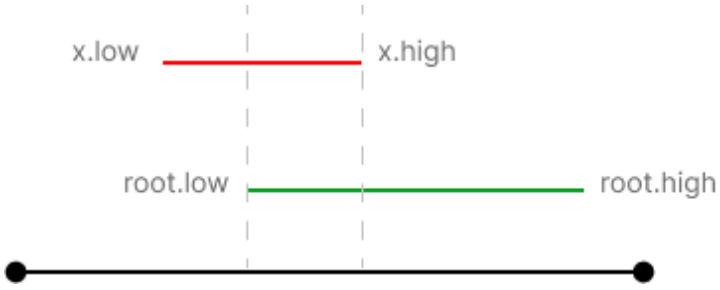


Рисунок 10 – Случай 6

4.3 Удаление узла

Для удаления узла рассматривается несколько ситуаций:

1. Узел является листом.

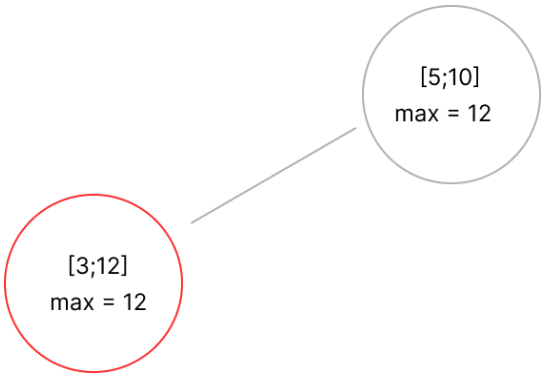


Рисунок 11 – Удаление узла

Узел [3,12] max = 12 является листом (рисунок 11). Таким образом Данный узел удаляется, а в узле [5,10] max = 12 пересчитывается максимум,

так как узел [3,12] max = 12 удален, а других узлов с таким максимумом нет (рисунок 12).

Рисунок 12 – Результат удаления

2. Если узел не является листом (рисунок 13).

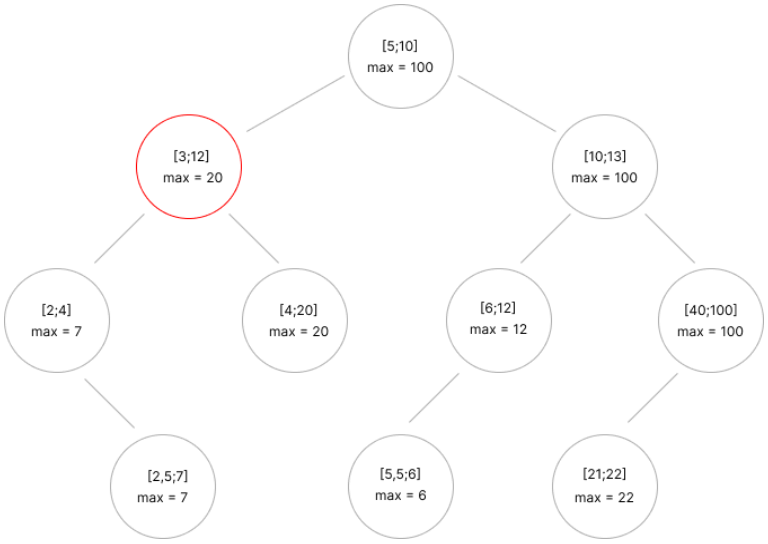


Рисунок 13 – Удаление узла [3;12], max = 20

Для удаления узла [3,12] max = 20 (рис. 13) требуется проверить есть ли у правое поддерево, если есть, то удаляем узел принимает значения узла, который находится правее (в данном случае таким узлом является узел [4,20] max = 20) (рисунок 14).

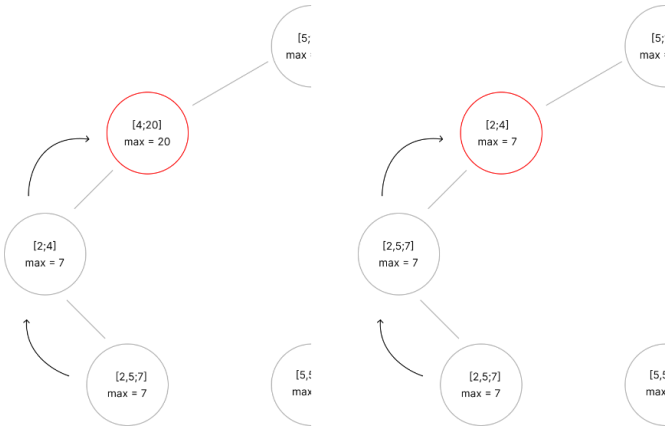


Рисунок 14 – Изменение значений в удаленном узле

Таким образом, узел [4,20] max = 20 переместился на место удаляемого узла. Далее поддерево, в котором был узел [4,20] max = 20 удаляется, так как в нем был один лист ([4,20] max = 29) (рисунок 15).

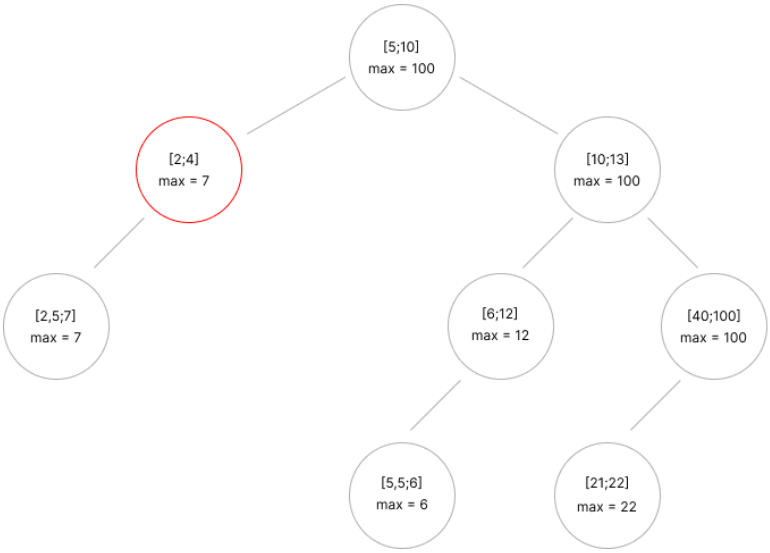


Рисунок 15 – Результат удаления

3. Если у узла нет правого поддерева, но есть левое (риснок 15).

В данном случае узел принимает значения узла, который лежит левее, а узел, лежащий левее принимает значения следующего узла и так далее (рисунок 16).

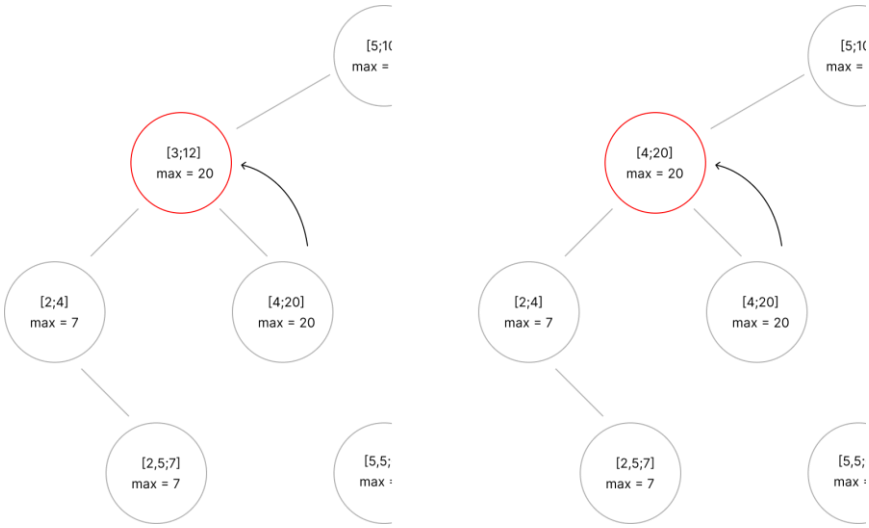


Рисунок 16 – Смена значений

Далее лист в левом поддереве ([2,5;7] max = 7) удаляется (рисунок 17).

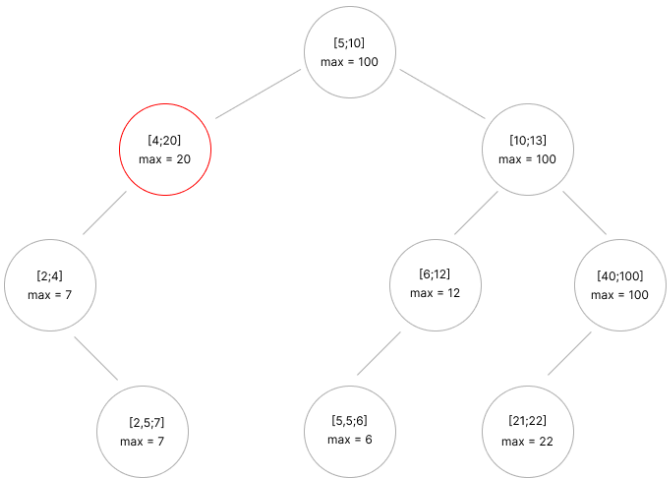


Рисунок 17 – Результат удаления

5 Реализация

5.1 Добавление

Insert (*null*, [*x.low*, *x.high*]), если дерево пустое и нужно добавить в него первый элемент, где *x.low* – нижняя граница интервала, *x.high* – верхняя граница интервала, *null* – обозначение того, что дерево не имеет ни одного узла.

Insert (*root*, [*x.low*, *x.high*]), если в дереве уже есть хотя бы один узел, где *x.low* – нижняя граница интервала, *x.high* – верхняя граница интервала, *root* – коллекция, в которой хранится дерево.

5.2 Проверка на перекрытие

isOverlapping(*root*, [*x.low*, *x.high*]), где *x.low* – нижняя граница интервала, *x.high* – верхняя граница интервала, *root* – коллекция, в которой хранится дерево.

5.3 Удаление

Remove (root, [*x.low*, *x.high*]), где *x.low* – нижняя граница интервала, *x.high* – верхняя граница интервала, *root* – коллекция, в которой хранится дерево.

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:

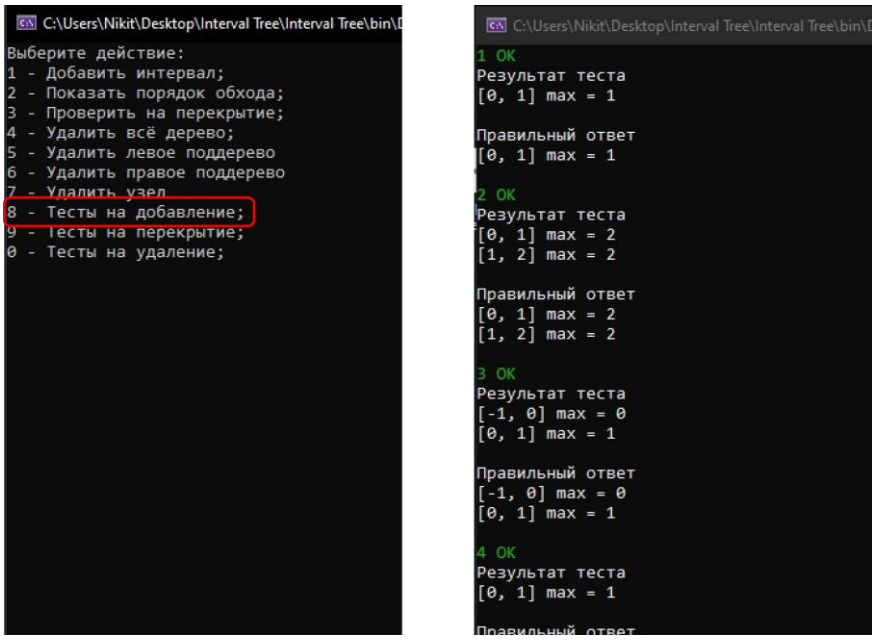


Рисунок 18. Тесты на добавление

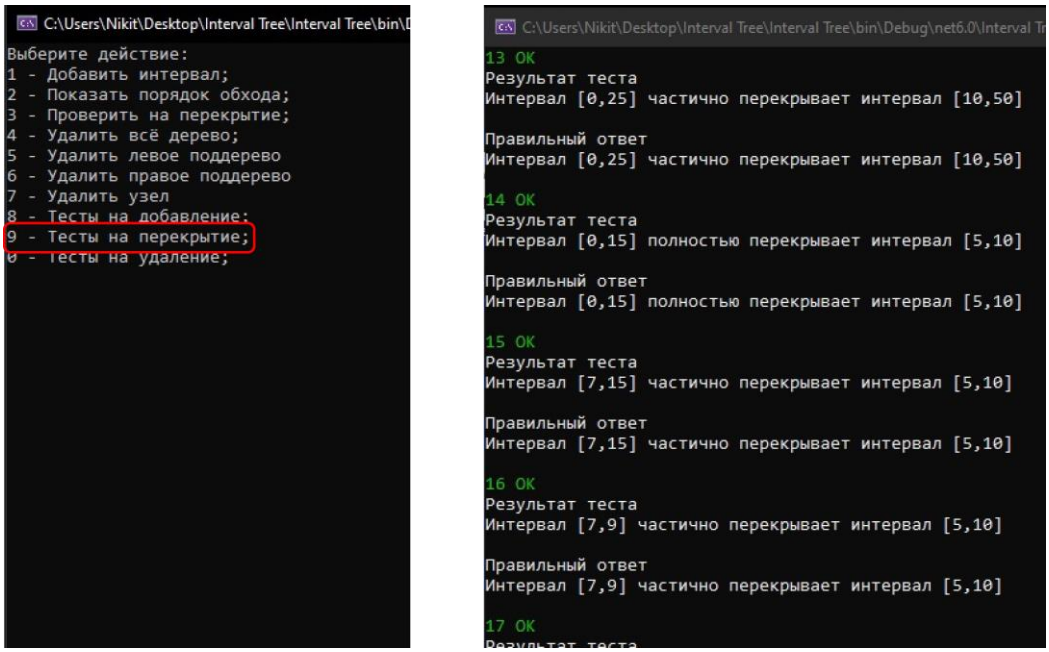


Рисунок 19. Тесты на перекрытие

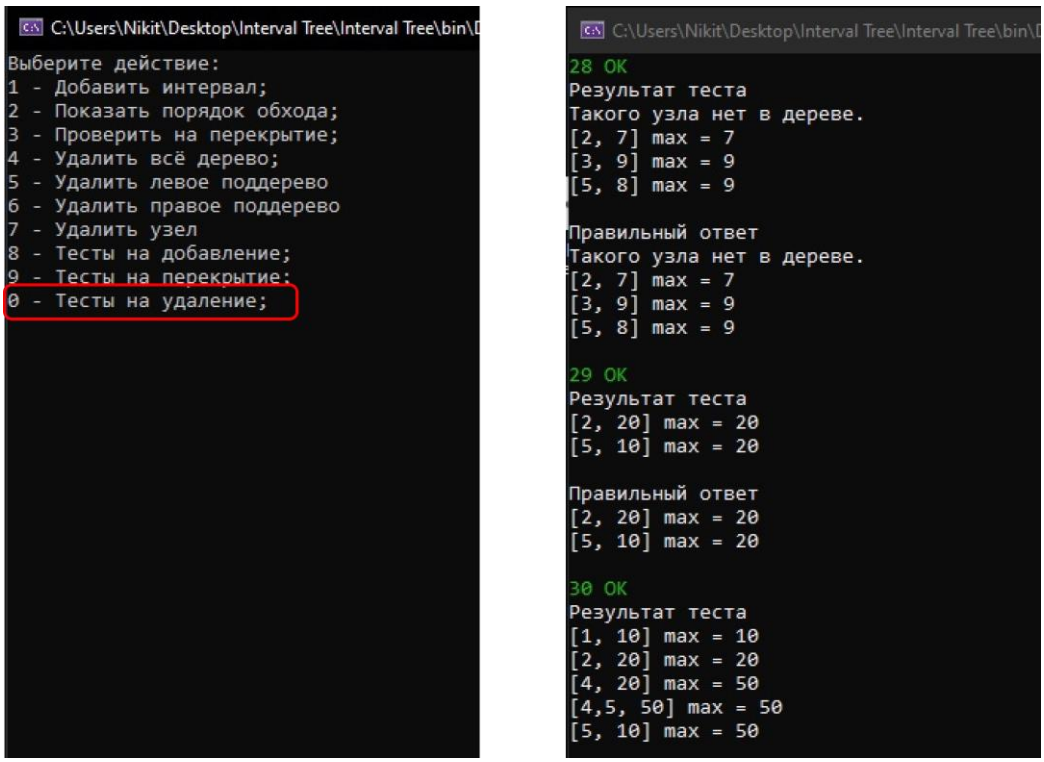


Рисунок 20. Тесты на удаление

7 Формальная постановка задачи

1) Изучить структуру данных, дерево интервалов;

2) Реализовать для структуры данных следующие операции:

    1. Добавление
    2. Удаление
    3. Проверка на перекрытие

Ограничения:

На вход принимаются значения типа double. [27]

3) Выполнить исследование на производительность;

4) Результаты выложить GitHub;

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)
2. Interval Tree. [Электронный ресурс] // geeksforgeeks. — Режим доступа: Interval Tree - GeeksforGeeks
3. Interval Trees: One step beyond BST [Электронный ресурс] // iq.opengenus. — Режим доступа: Interval Trees: One step beyond BST (opengenus.org)
4. Interval tree [Электронный ресурс] // wikipedia. — Режим доступа: Interval tree - Wikipedia
5. node-interval-tree [Электронный ресурс] // npm. — Режим доступа: node-interval-tree - npm (npmjs.com)
6. 5cript/interval – three [Электронный ресурс] // github. — Режим доступа: GitHub - 5cript/interval-tree: A C++ header only interval tree implementation.
7. INTERVAL TREES [Электронный ресурс] // dgp.toronto. — Режим доступа: CSC378: Interval Trees (toronto.edu)
8. Interval tree [Электронный ресурс] // cmu.edu. — Режим доступа: intervaltrees.pdf (cmu.edu)
9. Windowing queries [Электронный ресурс] // personal.us.es. — Режим доступа: Lecture 8: Windowing queries (us.es)
10. Interval tree [Электронный ресурс] // tutorialandexample. — Режим доступа: Interval Tree - TAE (tutorialandexample.com)
11. Interval tree [Электронный ресурс] // formulasearchengine. — Режим доступа: Interval tree - formulasearchengine

12. Interval tree [Электронный ресурс] // TutorialCup. — Режим доступа: Interval Tree - Interval Tree in Data Structure Interval Tree (tutorialcup.com)

13. Динамические структуры данных [Электронный ресурс] // штегше. — Режим доступа: Динамические структуры данных - интуит

14. Дерево диапозонов [Электронный ресурс] // wikibrief. — Режим доступа: Дерево диапазонов - Range tree (wikibrief.org)

15. Segment Tree Range Minimum Query [Электронный ресурс] // youtube. — Режим доступа: Segment Tree Range Minimum Query - YouTube

16. Interval tree [Электронный ресурс] // homepages.math. — Режим доступа: Interval Trees (uic.edu)

17. Дерево Интервалов (Отрезков) [Электронный ресурс] // coolsoftware. — Режим доступа: Дерево Интервалов (Отрезков) | Cool Software Blog

18. Interval tree [Электронный ресурс] // drdobbs. — Режим доступа: Interval Trees | Dr Dobb's (drdobbs.com)

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)

20. Augmented Interval Tree in C# [Электронный ресурс] // Software Salad. — Режим доступа: Augmented Interval Tree in C# | Software Salad (wordpress.com)

21. Дерево (структура данных) [Электронный ресурс] // wikipedia. — Режим доступа: Дерево (структура данных) — Википедия (wikipedia.org)

22. Методы [Электронный ресурс] // microsoft. — Режим доступа: Методы. Руководство по программированию на C# | Microsoft Learn

23. Коллекция [Электронный ресурс] // wikipedia. — Режим доступа: Коллекция (программирование) — Википедия (wikipedia.org)

24. Интервал [Электронный ресурс] // wikipedia. — Режим доступа: Интервал — Википедия (wikipedia.org)

25. Оценка сложности алгоритмов [Электронный ресурс] // tproger. — Режим доступа: Оценка сложности алгоритмов, или Что такое O(log n) (tproger.ru)

26. Временная сложность алгоритма [Электронный ресурс] // wikipedia. — Режим доступа: Временная сложность алгоритма — Википедия (wikipedia.org)

27. Типы данных [Электронный ресурс] // wikipedia. — Режим доступа: Типы данных в C# —метанит(metanit.com)