## Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Российский химико-технологический университет имени Д.И. Менделеева» Кафедра информационных компьютерных технологий

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3**

Выполнил студент группы.............КС-30 Сидоров Сергей Александрович

Ссылка на репозиторий: https://github.com/MUCTR-IKT-CPP/Sidorov.S.A\_KS-30\_2sem/tree/main/lab3

Приняли: Пысин Максим Дмитриевич

.............................................................................................................Краснов Дмитрий Олегович

Дата сдачи: 08.05.2021

## Москва, 2021

**Оглавление**

[Описание задачи. 3](#_TOC_250003)

[Описание метода/модели. 4](#_TOC_250002)

[Выполнение задачи. 4](#_TOC_250001)

[Заключение. 4](#_TOC_250000)

# Описание задачи.

Цель работы была в изучении и реализации бинарного дерева. Были реализованы:

* класс бинарного дерева
* операция вставки элемента в дерево
* операция удаления элемента из дерева
* операция поиска значения в дереве

Для демонстрации работы класса нужно было реализовать работу тестов. Работали они по следующему принципу: проводилось 20 запусков для массива, состоящего из N элементов с переменной N от 50000 до 500000 с шагом 50000. В каждом из запусков раз были произведены генерация и заполнение массива. Массив заполнялся числами типа double в диапазоне от -1 до 1. 5 из 20 раз после генерации и заполнения массива производилась сортировка этого массива. Для каждого из запусков были проделаны следующие тесты:

* 100 операций поиска случайного числа в дереве с замером времени на эту операцию
* 100 операций вставки случайного числа в дерево с замером времени на эту операцию
* 100 операций удаления случайного числа из дерева с замером времени на эту операцию

# Описание метода/модели.

Деревом называют структуру данных, которая имеет древовидный вид, то есть характеризуется наличием набора связанных узлов. Бинарное дерево — это конечное множество элементов, связанных с двумя разными бинарными деревьями — правым и левым поддеревьями. Его способ представления показан на рис.1, где A – корень дерева; B – корень левого поддерева и предок D; С – корень правого поддерева; D – потомок родительского узла B.

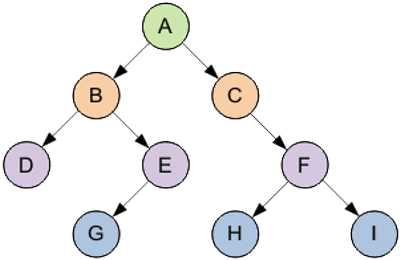


Рис. 1. Способ представления дерева.

Так же есть следующая терминология для работы с бинарными деревьями:

— узел (вершина) — это каждый элемент бинарного дерева;

— ветви — связи между узлами;

— глубина (высота) — наибольший уровень какого-нибудь элемента;

— лист (терминальный узел) — термин применяется, если элемент не имеет потомков;

— внутренние узлы — узлы ветвления;

— степень внутреннего узла — число его потомков (наибольшая степень всех существующих узлов — это степень всего бинарного дерева);

— длина пути к x — количество ветвей, которые необходимо пройти от корня до текущего узла x. Длина пути самого корня равна нулю, а узел на уровне i обладает длиной пути, которая равна i.

Так же для построения правильного дерева должно соблюдаться следующее свойство: если x – узел бинарного дерева с ключом k, то все узлы в левом поддереве должны иметь ключи, меньшие k, а в правом поддереве большие k.

**Выполнение задачи.**

* 1. Создаём класс бинарного дерева и реализуем там конструктор и геттеры с сеттерами.
  2. Реализуем функции для добавления, поиска и удаления элементов из дерева, а также функцию удаления самого дерева. В коде присутствуют функции поиска минимального и максимального узла, следующего за узлом узла, печати дерева в трёх разных порядках: в отсортированном порядке; вершина, левое поддерево, правое поддерево; левое поддерево, правое поддерево, вершина. Но многие из них были либо использованы косвенно через другие функции, либо не были использованы вообще.
  3. Затем в main делаем циклю для 20 запусков тестирования.
  4. Для каждых тестов засекаем время.
  5. В конце запуска удаляем динамический массив и дерево.
  6. Строим графики.

Прилагаю скриншоты кода:



Рис. 2. Скриншот кода 1. Класс Дерева.

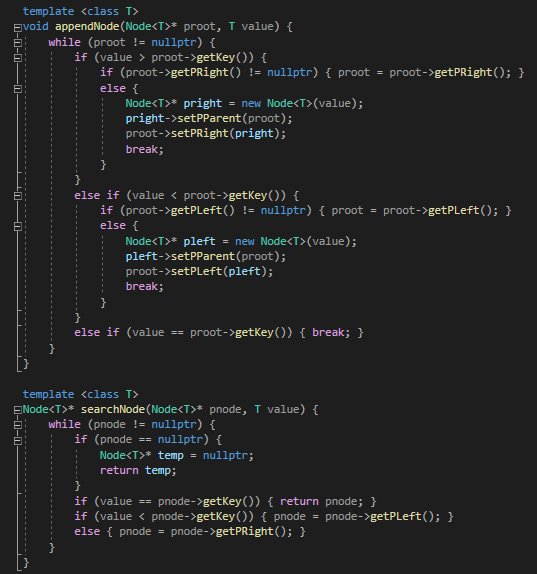


Рис. 3. Скриншот кода 2. Функции добавления и поиска элемента в дереве.



Рис. 4. Скриншот кода 3. Функции поиска максимального, минимального и следующего элемента в дереве.



Рис. 5. Скриншот кода 4. Функция удаления элемента из дерева.



Рис. 6. Скриншот кода 5. Функции удаления и печати дерева.

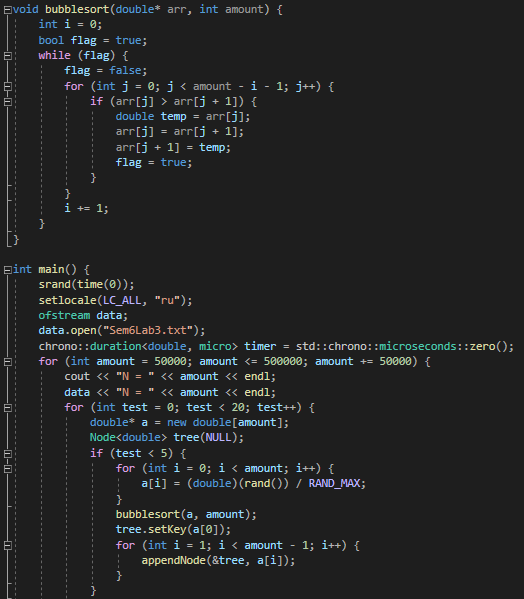


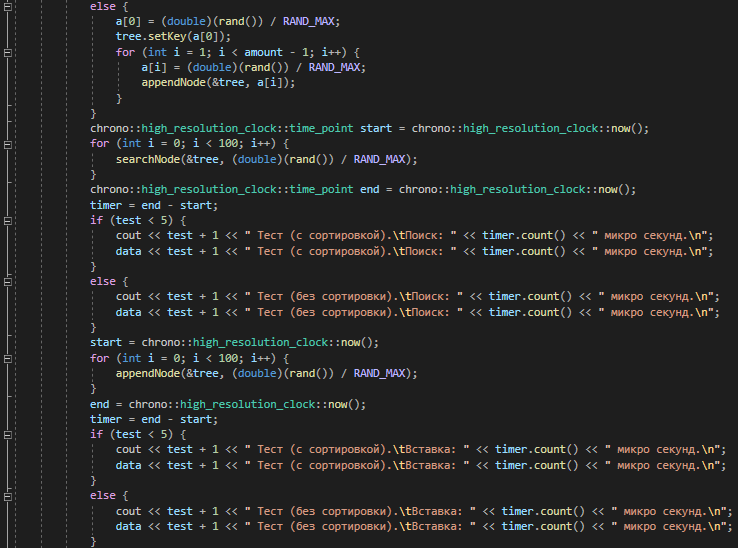
Рис. 7. Скриншот кода 6. Функция сортировки для массива и начало main. 

Рис. 8. Скриншот кода 7. Продолжение main.

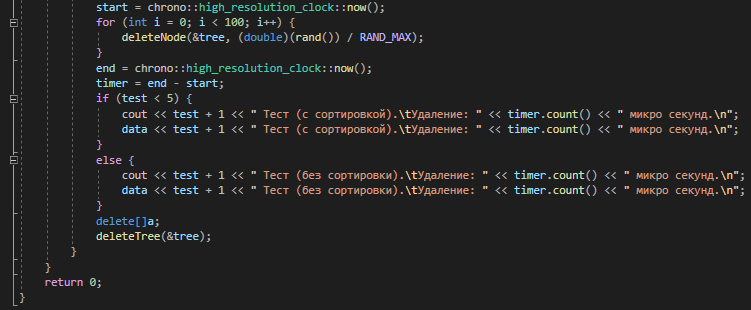


Рис. 9. Скриншот кода 8. Конец main.

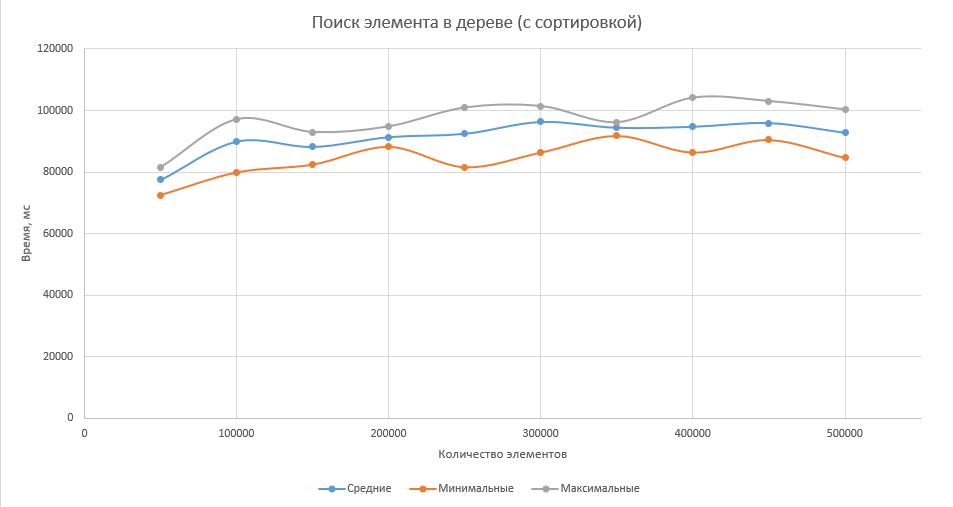


Рис. 10. График 1.

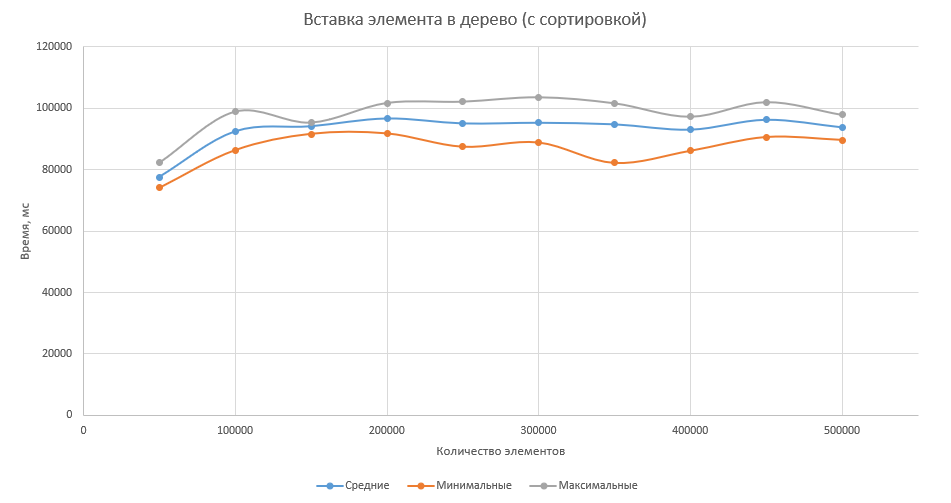


Рис. 11. График 2.

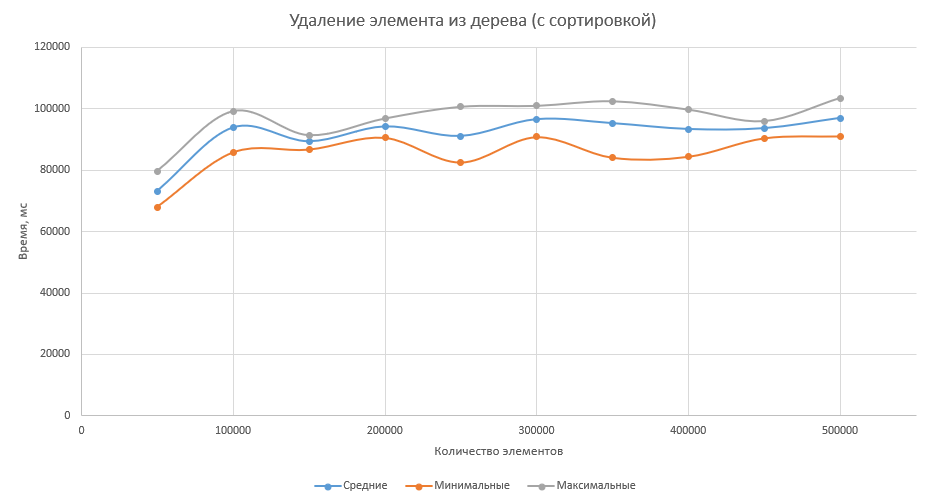


Рис. 12. График 3.

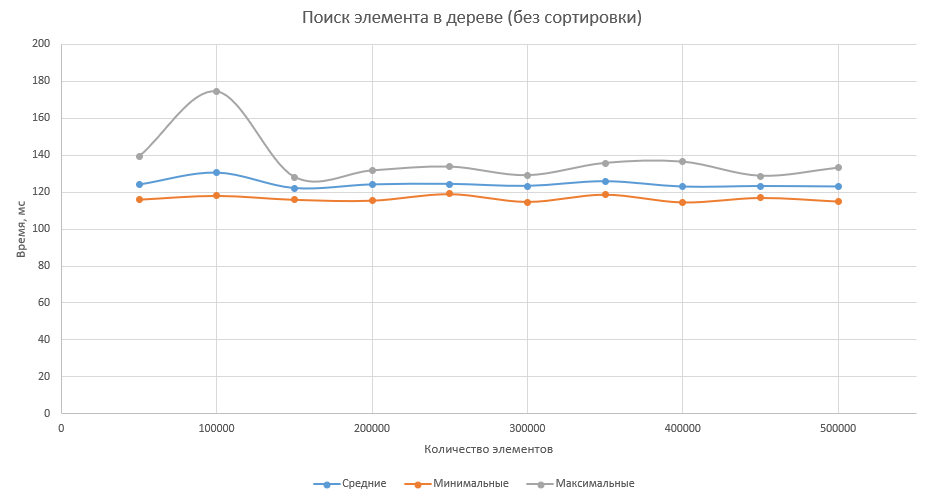


Рис. 13. График 4.

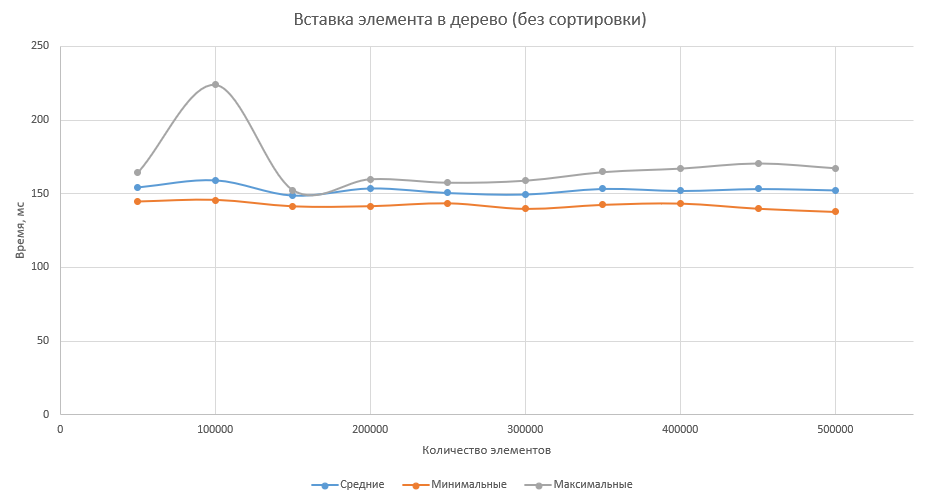


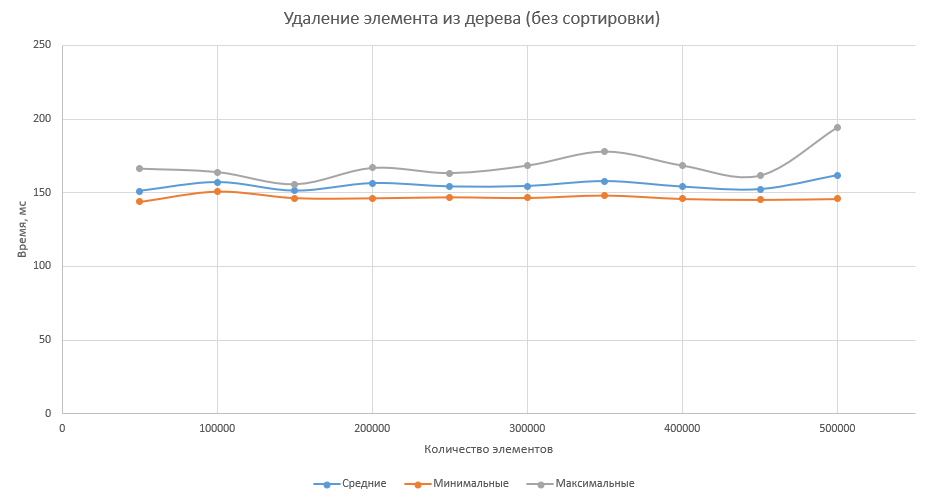
Рис. 14. График 5. 

Рис. 15. График 6.

# Заключение.

В конце работы можно проанализировать графики и сделать выводы. При использовании сортировки время, требуемое для выполнения всех операций значительно больше, чем без сортировки. Если же сравнить 1, 2 и 3 график, то их время не сильно отличается. Аналогично и с 4, 5 и 6 графиком.