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 № 6

Выполнил студент группы КС-30 Суханова Евгения Валерьевна
Ссылка на репозиторий:
https://github.com/MUCTR-IKT-CPP/EVSuhanova_30/blob/main/Algoritms/laba6.cpp

Приняли: Пысин Максим Дмитриевич
Краснов Дмитрий Олегович

Дата сдачи: 03.04.2023

Оглавление

Описание задачи.

В рамках лабораторной работы необходимо изучить и реализовать бинарное дерево поиска и его самобалансирующийся вариант в лице AVL дерева.

1. Для проверки анализа работы структуры данных требуется провести 10 серий тестов.
2. В каждой серии тестов требуется выполнять 20 циклов генерации и операций. При этом первые 10 работают с массивом заполненным случайным образом, во второй половине случаев, массив заполняется в порядке возрастания значений индекса, т.е. является отсортированным по умолчанию.
3. Требуется создать массив состоящий из 2^(10 + i) элементов, где i это номер серии.
4. Массив должен быть помещен в оба вариант двоичных деревьев. При этому замеряется время затраченное на всю операцию вставки всего массива.
5. После заполнения массива, требуется выполнить 1000 операций поиска по обоим вариантам дерева, случайного числа в диапазоне генерируемых значений, замерев время на все 1000 попыток и вычислив время 1 операции поиска.
6. Провести 1000 операций поиска по массиву, замерить требуемое время на все 1000 операций и найти время на 1 операцию.
7. После, требуется выполнить 1000 операций удаления значений из двоичных деревьев, и замерить время затраченное на все операции, после чего вычислить время на 1 операцию.
8. После выполнения всех серий тестов, требуется построить графики зависимости времени затрачиваемого на операции вставки, поиска, удаления от количества элементов. При этом требуется разделить графики для отсортированного набора данных и заполненных со случайным распределением. Так же, для операции поиска, требуется также нанести для сравнения график времени поиска для обычного массива.

Описание метода/модели.

Бинарное дерево — это иерархическая структура данных, в которой каждый узел имеет значение (оно же является в данном случае и ключом) и ссылки на левого и правого потомка.

Бинарное дерево поиска — это бинарное дерево, обладающее дополнительными свойствами: значение левого потомка меньше значения родителя, а значение правого потомка больше значения родителя для каждого узла дерева. То есть, данные в бинарном дереве поиска хранятся в отсортированном виде. При каждой операции вставки нового или удаления существующего узла отсортированный порядок дерева сохраняется. При поиске элемента сравнивается искомое значение с корнем. Если искомое больше корня, то поиск продолжается в правом потомке корня, если меньше, то в левом, если равно, то значение найдено и поиск прекращается.

Сбалансированное бинарное дерево поиска — это бинарное дерево поиска с логарифмической высотой. Данное определение скорее идейное, чем строгое. Строгое определение оперирует разницей глубины самого глубокого и самого неглубокого листа (в AVL-деревьях) или отношением глубины самого глубокого и самого неглубокого листа (в красно-черных деревьях). В сбалансированном бинарном дереве поиска операции поиска, вставки и удаления выполняются за логарифмическое время (так как путь к любому листу от корня не более логарифма). В вырожденном случае несбалансированного бинарного дерева поиска, например, когда в пустое дерево вставлялась отсортированная последовательность, дерево превратится в линейный список, и операции поиска, вставки и удаления будут выполняться за линейное время. Поэтому балансировка дерева крайне важна. Технически балансировка осуществляется поворотами частей дерева при вставке нового элемента, если вставка данного элемента нарушила условие сбалансированности.

Выполнение задачи.

Описание всех функций, с помощью которых будем проводить различные операции над бинарными деревьями:

1. Для начала зададим структуру одного узла дерева (где key - значение узла, height - высота, на которой находится узел, left - левый потомок, right - правый потомок).

```
struct node // структура для представления узлов дерева
{
    int key;
    unsigned char height;
    node* left;
    node* right;
    node(int k) { key = k; left = right = 0; height = 1; }
};
```

2. Добавление узла в дерево. Если узел не конечный, то мы определяем в какую сторону необходимо двигаться для правильной постановки нового узла. Если узел конечный, то ставится новый узел. Для AVL-дерева вызывается балансировка в конце рекурсии.

```
// Добавление узлов в дерево
node* addnode(int k, node* p, bool AVL) {
    if (!p) return new node(k);
    if (k <= p->key)
        p->left = addnode(k, p->left, AVL);
    else
        p->right = addnode(k, p->right, AVL);
    if (AVL) return balance(p);
    else return p;
}
```

3. Поиск узла в бинарном дереве. Определяем меньше ли необходимое значение текущего узла. Если да, то двигаемся в левую сторону дерева, иначе в правую сторону.

```
void reverse(node* p, int k) {
    if (p->key == k) cout << "Найден узел со значением " << k << endl;
    else {
        if (k < p->key) reverse(p->left, k);        //Рекурсивная функц
        else reverse(p->right, k);                  //Рекурсивная функц
    }
}
```

4. Вывод дерева в консоль. Выводим с самого левого узла к самому правому с помощью рекурсий.

```
// Вывод бинарного дерева
void print_Tree(node* p, int level) {
    if (p) {
        print_Tree(p->right, level + 1);
        for (int i = 0; i < level; i++) cout << "    ";
        cout << p->key << endl;
        print_Tree(p->left, level + 1);
    }
}
```

5. Удаление узла из дерева. Проверяем есть ли потомки у данного узла. Определяем в каком потомке находится узел для удаления. С помощью рекурсивной перестановки удаляем нужный узел и поднимаем на уровень правого потомка.

```
node* remove(node* p, int value) {  // удалени
    if (!p) return 0;
    if (value < p->key)
        p->left = remove(p->left, value);
    else if (value > p->key)
        p->right = remove(p->right, value);
    else {
        node* q = p->left;
        node* r = p->right;
        delete p;
        if (!r) return q;
        node* min = findmin(p: r);
        min->right = removemin(p: r);
        min->left = q;
        return balance(p: min);
    }
    return balance(p);
}
```

6. Функция поиска узла с минимальным значением. Данная функция находит минимальный по значению узел в правом потомке узла.

```
node* findmin(node* p) // поиск узла с мин
{
    return p->left ? findmin(p->left) : p;
}
```

7. Удаление узла с минимальным значением. Данная функция удаляет найденный в ранее описанной функции узел и ставит на его место правый узел. При необходимости вызывается балансировка дерева.

```
node* removemin(node* p) // удален
{
    if (p->left == 0)
        return p->right;
    p->left = removemin(p->left);
    return balance(p);
}
```

8. Балансировка. Изначально мы определяем глубину каждой ветки относительно заданного узла. Далее, если у нас в правом дереве больше потомков, то идет та же проверка для правого узла. Если левый потомок больше, то происходит правый поворот вокруг этого правого узла. После чего, будет левый поворот вокруг изначального узла. Аналогично будет происходить, если у изначального узла левых потомков больше.

```
node* balance(node* p) // балансировка узла p
{
    fixheight(p);
    if (bfactor(p) == 2)
    {
        if (bfactor(p->right) < 0)
            p->right = rotateright(p->right);
        return rotateleft(q: p);
    }
    if (bfactor(p) == -2)
    {
        if (bfactor(p->left) > 0)
            p->left = rotateleft(q: p->left);
        return rotateright(p);
    }
    return p; // балансировка не нужна
}
```

9. Функция для фиксации максимальной глубины у бинарного дерева. Мы определяем глубину по правому и по левому потомкам и определяем максимальное значение.

```
void fixheight(node* p) {
    unsigned char hl = height(p->left);
    unsigned char hr = height(p->right);
    p->height = (hl > hr ? hl : hr) + 1;
}
```

10. Функция определения разницы между глубиной правого и левого потомков.

```
int bfactor(node* p) {
    return height(p->right) - height(p->left);
}
```

11. Функция определения глубины потомка.

```
unsigned char height(node* p) {
    return p ? p->height : 0;
}
```

12. Правый поворот вокруг узла. Переставляем местами левого и правого потомков. Фиксируем их глубину. Возвращаем нового правого потомка.

```
node* rotateright(node* p) {
    node* q = p->left;
    p->left = q->right;
    q->right = p;
    fixheight(p);
    fixheight(p: q);
    return q;
}
```

13. Левый поворот вокруг узла. Переставляем местами левого и правого потомков. Фиксируем их глубину. Возвращаем нового левого потомка.

```
node* rotateleft(node* q) // л
{
    node* p = q->right;
    q->right = p->left;
    p->left = q;
    fixheight(p: q);
    fixheight(p);
    return p;
}
```

14. Освобождение памяти из под дерева. Рекурсивно удаляем каждый узел у дерева.

```
//Освобождение памяти дерева
void freemem(node* tree) {
    if (tree != NULL) {
        freemem(tree->left);
        freemem(tree->right);
        delete tree;
    }
}
```

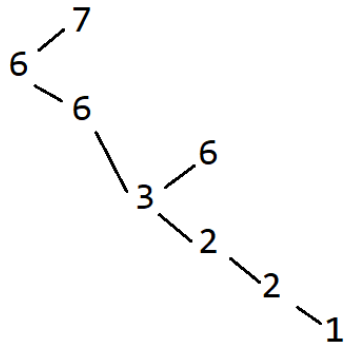
Результаты работы алгоритма

Для демонстрации работоспособности алгоритма в отчете были проведены тесты на меньших значениях деревьев.

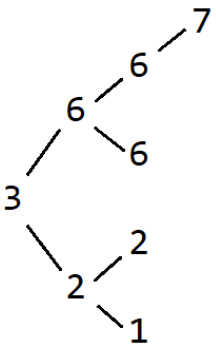
Таким образом, была проведена всего одна серия для двух циклов. В первом массив заполнялся случайно, а во втором по порядку. Также на каждую операцию было проведено по 3 теста.

С помощью функции вставки мы создали обычное и сбалансированное бинарное дерево из массива случайных чисел:

Обычное бинарное дерево



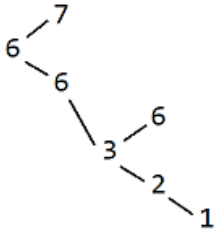
Сбалансированное бинарное дерево



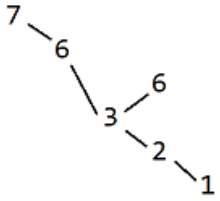
Далее приведен пример того, как алгоритм удаляет узлы из дерева:

Удаление из обычного бинарного дерева

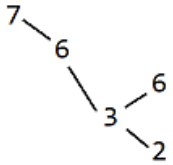
Удален узел 2



Удален узел 6

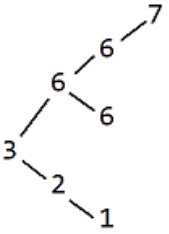


Удален узел 1

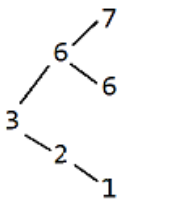


Удаление из сбалансированого бинарного дерева

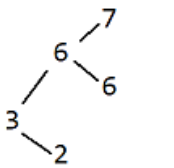
Удален узел 2



Удален узел 6

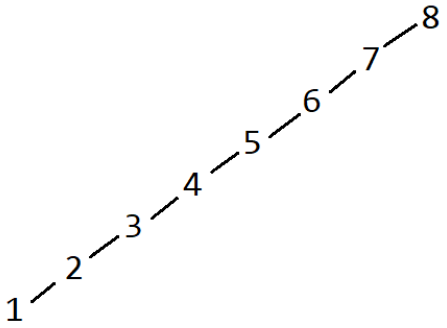


Удален узел 1

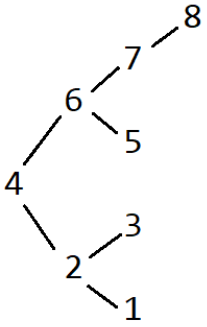


С помощью функции вставки мы создали обычное и сбалансированное бинарное дерево из отсортированного массива:

Обычное бинарное дерево

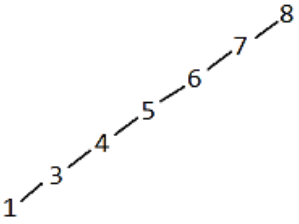


Сбалансированное бинарное дерево

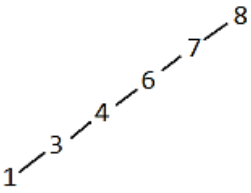


Далее приведен пример того, как алгоритм удаляет узлы из дерева:

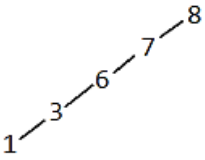
Удаление из обычного бинарного дерева
Удален узел 2



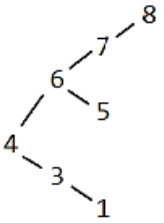
Удален узел 5



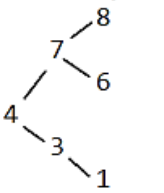
Удален узел 4



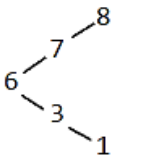
Удаление из сбалансированого бинарного дерева
Удален узел 2



Удален узел 5



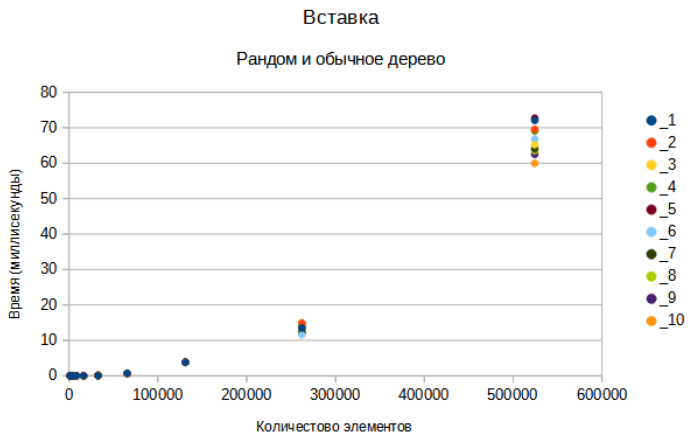
Удален узел 4

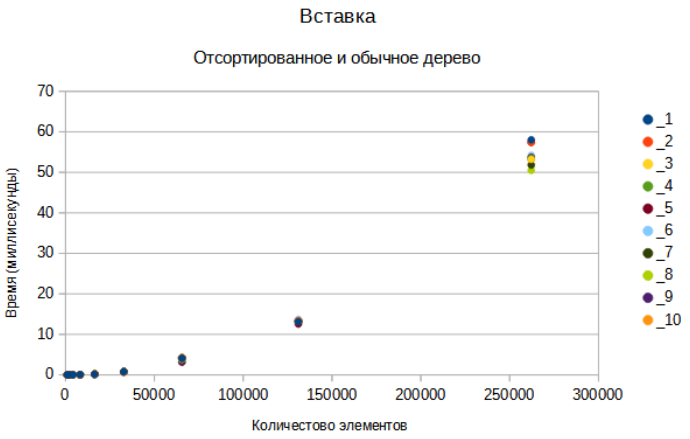


Полученные результаты

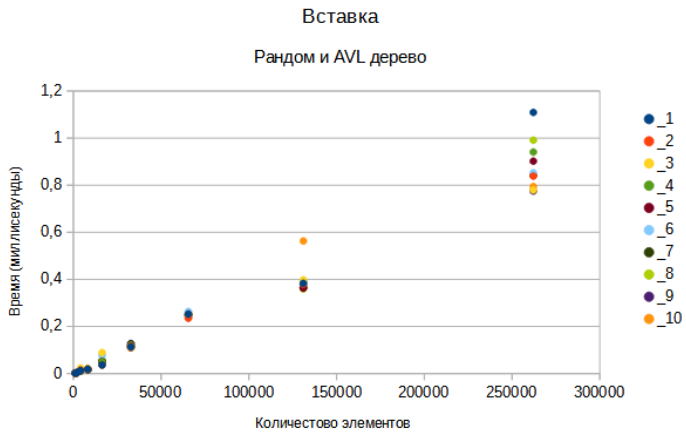
Вставка:

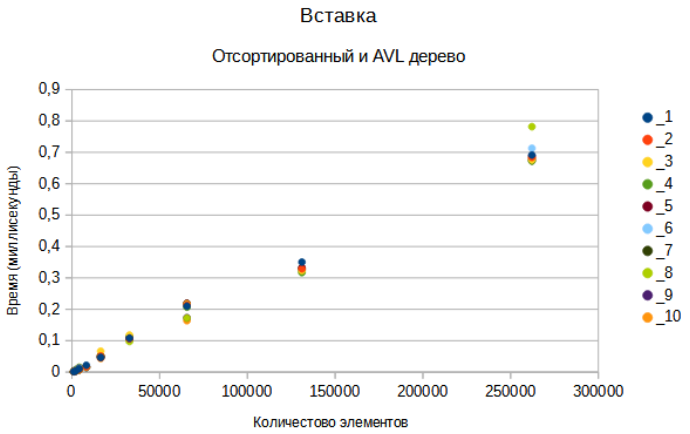
		Вставка в обычное бинарное дерево									
		1024	2048	4096	8192	16384	32768	65536	131072	262144	524288
Рандомный	1	0,000527	0,0007373	0,0018135	0,0073785	0,0190288	0,0661122	0,747958	3,84482	13,5437	72,0649
	2	0,0003364	0,0007383	0,0017228	0,0056077	0,016487	0,0627791	0,590608	3,92264	14,9229	69,555
	3	0,0004173	0,0007991	0,00187	0,0210423	0,0211657	0,119202	0,598997	3,88872	14,7776	65,2009
	4	0,0003521	0,0007646	0,0018173	0,013349	0,0268267	0,08966	0,620023	3,79924	14,6009	69,0593
	5	0,0003758	0,0007011	0,0017553	0,0112838	0,019203	0,108029	0,562213	3,93452	13,6473	72,723
	6	0,0003841	0,0009388	0,0016452	0,0113283	0,0174025	0,0978801	0,56787	3,81613	11,6096	66,8004
	7	0,0003481	0,0008268	0,001761	0,0150409	0,016095	0,103849	0,591617	3,81377	12,1622	64,08
	8	0,0003707	0,0008369	0,0018227	0,0102806	0,0153785	0,109808	0,582039	3,89343	12,7258	63,5381
	9	0,0003805	0,0009446	0,0018195	0,0139342	0,016404	0,12854	0,633802	3,81375	12,519	62,4912
	10	0,0004301	0,0007771	0,0018382	0,0110015	0,0166812	0,10753	0,608227	3,91793	14,3284	59,965
Отсортированный	11	0,0009188	0,0017215	0,0042024	0,0164358	0,0510042	0,142121	0,762952	4,10125	13,0984	58,0281
	12	0,0008702	0,0018521	0,0041139	0,0141618	0,032507	0,176528	0,682323	4,15984	13,3462	57,3666
	13	0,0008604	0,0018075	0,0043138	0,0118749	0,0309174	0,173146	0,707144	4,26572	13,2312	53,2183
	14	0,0008463	0,0020013	0,0055715	0,0110578	0,0312385	0,168884	0,789327	4,16673	13,246	53,4837
	15	0,0009911	0,0018165	0,0060297	0,009605	0,0259621	0,147708	0,778375	4,1827	12,5582	53,689
	16	0,0008388	0,0021504	0,0066896	0,0190942	0,0242238	0,166192	0,757528	4,18332	13,6585	54,1891
	17	0,0012544	0,0020463	0,0128555	0,0119061	0,0239401	0,18245	0,805796	4,24736	13,3366	51,8197
	18	0,0008493	0,0022967	0,0062181	0,0301435	0,0241334	0,150288	0,762476	3,82205	13,1996	50,51
	19	0,0008814	0,0028208	0,0042412	0,0139137	0,0290482	0,135453	0,732352	3,14604	13,1239	57,4418
	20	0,0008849	0,0027357	0,0044637	0,009246	0,0239348	0,178156	0,759721	3,09535	13,5343	53,1601



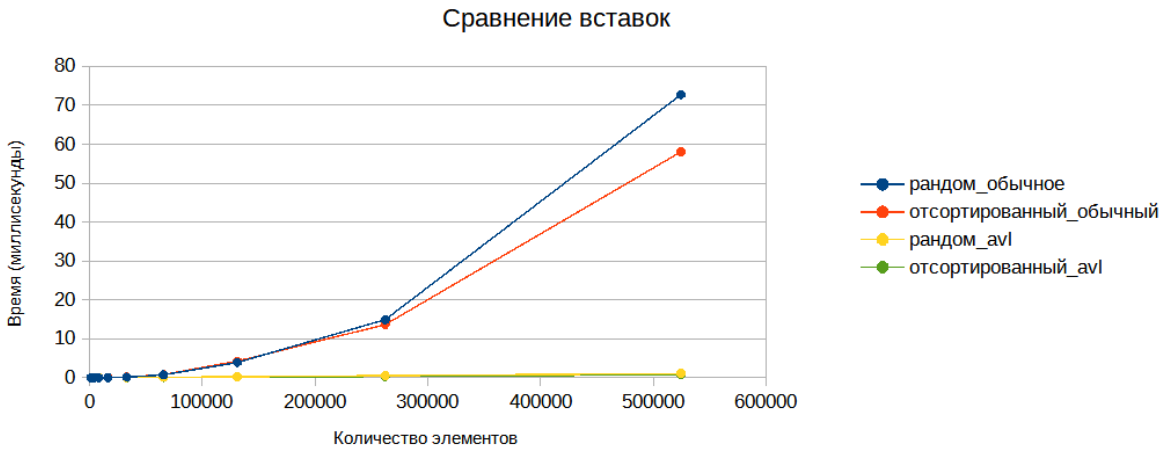


		Вставка в AVL бинарное дерево									
		1024	2048	4096	8192	16384	32768	65536	131072	262144	524288
Рандомный	1	0,0007026	0,0014343	0,0032038	0,0127322	0,0186598	0,0362669	0,113331	0,252187	0,383078	1,1093
	2	0,0007455	0,0013934	0,0032469	0,0138282	0,0182636	0,0355653	0,116502	0,234769	0,378683	0,838638
	3	0,0007014	0,0013984	0,0033236	0,0234184	0,0222179	0,0888254	0,112235	0,245211	0,397861	0,779187
	4	0,0006716	0,0014471	0,0032758	0,0133461	0,0172449	0,0501042	0,120026	0,237934	0,391918	0,940738
	5	0,0007596	0,0014885	0,00318	0,0151305	0,0164371	0,050814	0,110422	0,249927	0,362888	0,901904
	6	0,0006654	0,0015701	0,0034594	0,0142386	0,015998	0,0771272	0,117248	0,263271	0,377243	0,852677
	7	0,0006564	0,0014402	0,0037586	0,0101197	0,0158821	0,0525122	0,125209	0,247252	0,364456	0,841698
	8	0,000672	0,0015666	0,0040182	0,0119448	0,0162709	0,0547139	0,11605	0,250543	0,359435	0,991079
	9	0,000683	0,0018578	0,0034192	0,015379	0,0176008	0,0533232	0,127102	0,248373	0,365414	0,774857
	10	0,0008605	0,0014665	0,0043105	0,0115573	0,0176363	0,0504355	0,109989	0,261055	0,562875	0,794466
Отсортированный	11	0,0006036	0,0013061	0,0032555	0,0106264	0,0217163	0,0471591	0,108423	0,210176	0,350405	0,691318
	12	0,0005807	0,0014466	0,0034152	0,0079647	0,016805	0,0514046	0,107248	0,213512	0,328988	0,683941
	13	0,0005804	0,0014408	0,0032623	0,0078227	0,0164137	0,0665735	0,118094	0,216956	0,323163	0,675867
	14	0,000581	0,001402	0,005447	0,0069211	0,0162135	0,0492418	0,113757	0,206773	0,317063	0,672179
	15	0,0005766	0,0013405	0,0036004	0,0077356	0,0159342	0,0475976	0,106528	0,218998	0,332698	0,678857
	16	0,0005743	0,0013168	0,0066971	0,0127654	0,0148034	0,048337	0,106295	0,212657	0,324854	0,712964
	17	0,0006157	0,0015167	0,0075426	0,0142914	0,0154773	0,051288	0,108484	0,218672	0,331115	0,673049
	18	0,0005772	0,0019135	0,0053388	0,0152195	0,0151339	0,0518583	0,0975666	0,172075	0,320054	0,781725
	19	0,0005781	0,0017471	0,0033607	0,0077299	0,0148748	0,0467219	0,108897	0,173081	0,326522	0,688841
	20	0,0005753	0,0016424	0,0055367	0,0075562	0,0150025	0,043473	0,102098	0,163696	0,325955	0,682807



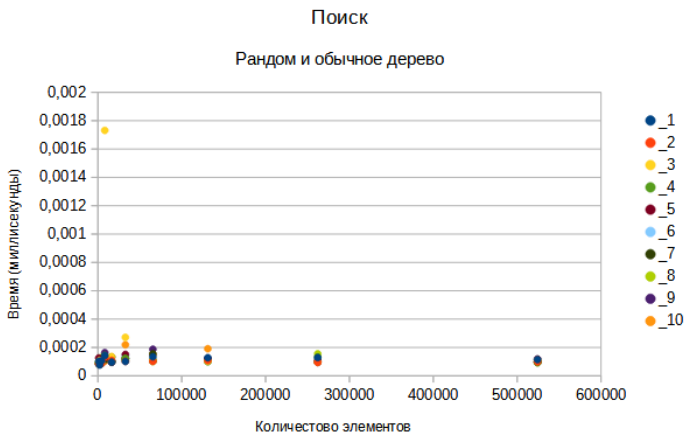


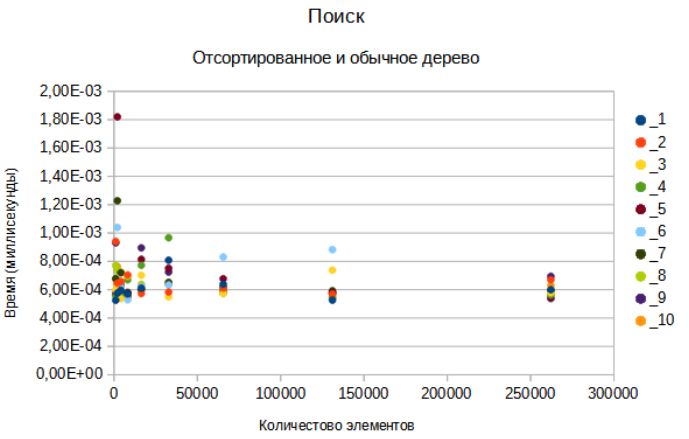
	Сравнение всех вариантов вставок (худшие случаи)									
	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288
Рандом обычное	0,000527	0,0009446	0,00187	0,0210423	0,0268267	0,12854	0,747958	3,93452	14,9229	72,723
Отсортированное обычное	0,0012544	0,0028208	0,0128555	0,0301435	0,0510042	0,18245	0,805796	4,26572	13,6585	58,0281
Рандом AVL	0,0008605	0,0018578	0,0043105	0,0234184	0,0222179	0,0888254	0,127102	0,263271	0,562875	1,1093
Отсортированное AVL	0,0006157	0,0019135	0,0075426	0,0152195	0,0217163	0,0665735	0,118094	0,218998	0,350405	0,781725



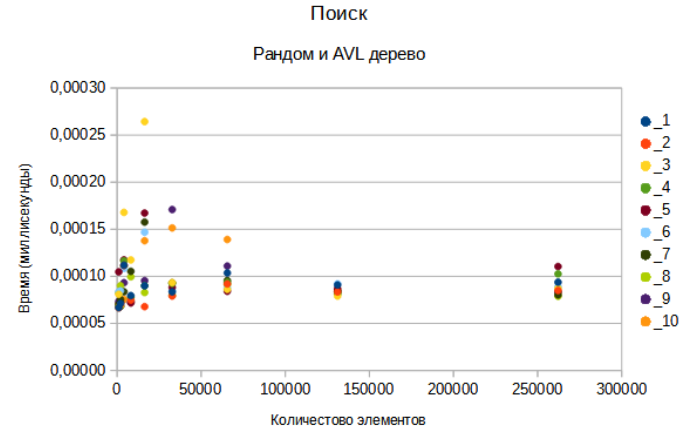
Поиск:

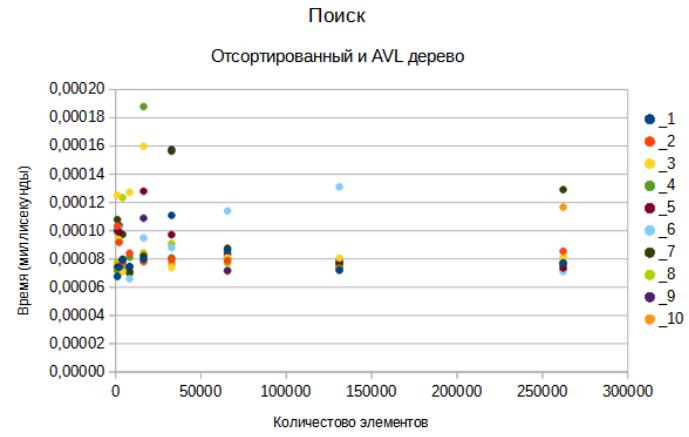
		Поиск в обычном бинарном дереве									
		1024	2048	4096	8192	16384	32768	65536	131072	262144	524288
Рандомный	1	1,02E-04	7,65E-05	1,06E-04	1,43E-04	9,68E-05	1,01E-04	1,34E-04	1,28E-04	1,29E-04	1,14E-04
	2	8,49E-05	8,61E-05	8,58E-05	1,31E-04	1,09E-04	1,03E-04	1,02E-04	1,08E-04	9,50E-05	9,80E-05
	3	9,10E-05	1,08E-04	8,67E-05	1,73E-03	1,36E-04	2,72E-04	1,08E-04	1,03E-04	9,60E-05	1,10E-04
	4	9,03E-05	9,30E-05	8,72E-05	1,44E-04	1,02E-04	1,24E-04	1,12E-04	9,94E-05	1,32E-04	9,20E-05
	5	1,26E-04	8,85E-05	8,52E-05	1,30E-04	1,01E-04	1,51E-04	1,06E-04	1,23E-04	1,07E-04	1,18E-04
	6	8,85E-05	1,01E-04	1,04E-04	1,37E-04	1,05E-04	1,44E-04	1,05E-04	1,03E-04	1,01E-04	1,18E-04
	7	8,25E-05	9,08E-05	9,78E-05	1,14E-04	1,32E-04	1,23E-04	1,56E-04	1,12E-04	1,00E-04	9,76E-05
	8	9,81E-05	1,03E-04	1,29E-04	1,32E-04	1,01E-04	1,10E-04	1,01E-04	1,10E-04	1,55E-04	9,46E-05
	9	8,76E-05	1,14E-04	1,07E-04	1,64E-04	9,53E-05	1,20E-04	1,87E-04	1,19E-04	9,48E-05	9,76E-05
	10	8,49E-05	8,91E-05	1,08E-04	1,04E-04	9,58E-05	2,19E-04	1,50E-04	1,91E-04	9,71E-05	1,00E-04
Отсортированный	11	5,61E-04	5,26E-04	5,76E-04	5,96E-04	5,72E-04	6,11E-04	8,08E-04	6,39E-04	5,26E-04	6,00E-04
	12	5,95E-04	9,41E-04	6,46E-04	6,58E-04	7,04E-04	5,73E-04	5,83E-04	6,12E-04	5,69E-04	6,70E-04
	13	5,35E-04	6,00E-04	6,38E-04	5,38E-04	6,96E-04	7,02E-04	5,50E-04	5,72E-04	7,38E-04	5,78E-04
	14	5,32E-04	5,56E-04	6,01E-04	5,61E-04	6,70E-04	7,72E-04	9,67E-04	6,03E-04	5,41E-04	5,62E-04
	15	5,30E-04	5,72E-04	1,82E-03	5,70E-04	5,73E-04	8,14E-04	7,52E-04	6,78E-04	5,79E-04	5,39E-04
	16	5,31E-04	5,66E-04	1,04E-03	6,20E-04	5,29E-04	6,21E-04	6,34E-04	8,31E-04	8,83E-04	5,37E-04
	17	5,31E-04	6,79E-04	1,23E-03	7,20E-04	5,81E-04	6,00E-04	6,52E-04	6,21E-04	5,93E-04	5,95E-04
	18	6,82E-04	7,71E-04	7,35E-04	7,23E-04	5,67E-04	6,38E-04	6,42E-04	5,76E-04	5,73E-04	5,76E-04
	19	5,30E-04	9,31E-04	5,69E-04	6,29E-04	5,53E-04	8,95E-04	7,25E-04	5,81E-04	5,77E-04	6,95E-04
	20	5,29E-04	6,74E-04	7,64E-04	6,36E-04	5,76E-04	6,07E-04	6,39E-04	5,87E-04	5,53E-04	6,29E-04



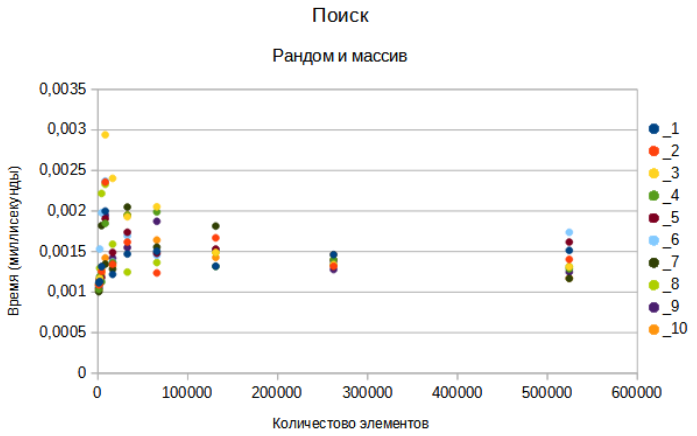


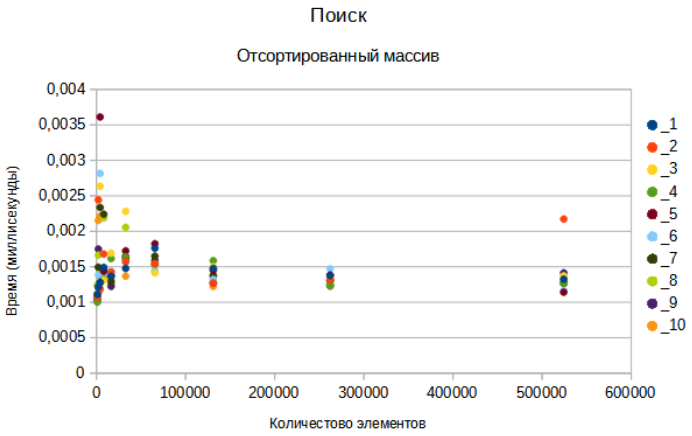
		Поиск в AVL бинарном дереве									
		1024	2048	4096	8192	16384	32768	65536	131072	262144	524288
Рандомный	1	8,95E-05	6,67E-05	7,12E-05	1,12E-04	7,95E-05	8,99E-05	8,35E-05	1,04E-04	9,08E-05	9,38E-05
	2	7,05E-05	6,66E-05	7,02E-05	1,12E-04	7,47E-05	6,78E-05	7,91E-05	9,19E-05	8,34E-05	8,49E-05
	3	6,76E-05	8,09E-05	6,91E-05	1,68E-04	1,17E-04	2,64E-04	9,33E-05	8,61E-05	7,90E-05	8,72E-05
	4	7,02E-05	6,78E-05	6,80E-05	1,17E-04	7,76E-05	9,00E-05	9,25E-05	9,38E-05	8,17E-05	1,03E-04
	5	9,35E-05	1,05E-04	6,80E-05	1,18E-04	7,17E-05	1,67E-04	8,82E-05	8,41E-05	8,68E-05	1,10E-04
	6	7,00E-05	8,36E-05	8,46E-05	1,08E-04	7,35E-05	1,47E-04	8,49E-05	8,77E-05	9,20E-05	8,91E-05
	7	7,26E-05	7,03E-05	7,54E-05	8,36E-05	1,05E-04	1,58E-04	9,30E-05	9,56E-05	8,38E-05	8,03E-05
	8	6,98E-05	8,26E-05	9,00E-05	1,10E-04	9,95E-05	8,28E-05	7,98E-05	9,12E-05	8,14E-05	7,88E-05
	9	6,86E-05	7,33E-05	7,25E-05	9,32E-05	7,29E-05	9,54E-05	1,71E-04	1,11E-04	8,40E-05	8,35E-05
	10	6,97E-05	7,13E-05	8,46E-05	7,54E-05	7,28E-05	1,38E-04	1,51E-04	1,39E-04	8,55E-05	8,26E-05
Отсортированный	11	7,02E-05	6,77E-05	7,41E-05	7,94E-05	7,47E-05	7,99E-05	1,11E-04	8,60E-05	7,20E-05	7,73E-05
	12	7,41E-05	1,03E-04	9,18E-05	7,80E-05	8,41E-05	7,81E-05	7,98E-05	7,88E-05	7,25E-05	8,55E-05
	13	8,54E-05	1,25E-04	9,36E-05	7,10E-05	1,27E-04	1,60E-04	7,39E-05	8,12E-05	8,06E-05	8,13E-05
	14	6,66E-05	7,19E-05	7,09E-05	7,28E-05	8,13E-05	1,88E-04	8,09E-05	8,05E-05	7,37E-05	7,73E-05
	15	6,73E-05	7,42E-05	9,91E-05	7,98E-05	7,45E-05	1,28E-04	9,72E-05	8,33E-05	7,89E-05	7,37E-05
	16	6,78E-05	6,75E-05	1,02E-04	7,50E-05	6,61E-05	9,49E-05	8,81E-05	1,14E-04	1,31E-04	7,10E-05
	17	6,69E-05	1,08E-04	1,04E-04	9,74E-05	7,03E-05	8,21E-05	1,56E-04	8,76E-05	7,72E-05	1,29E-04
	18	6,85E-05	7,75E-05	9,40E-05	1,24E-04	7,02E-05	8,42E-05	9,09E-05	7,70E-05	7,48E-05	7,25E-05
	19	6,71E-05	1,00E-04	7,30E-05	7,59E-05	7,07E-05	1,09E-04	1,58E-04	7,19E-05	7,59E-05	7,59E-05
	20	6,70E-05	9,89E-05	7,67E-05	7,96E-05	7,13E-05	8,26E-05	7,67E-05	7,13E-05	7,62E-05	1,17E-04



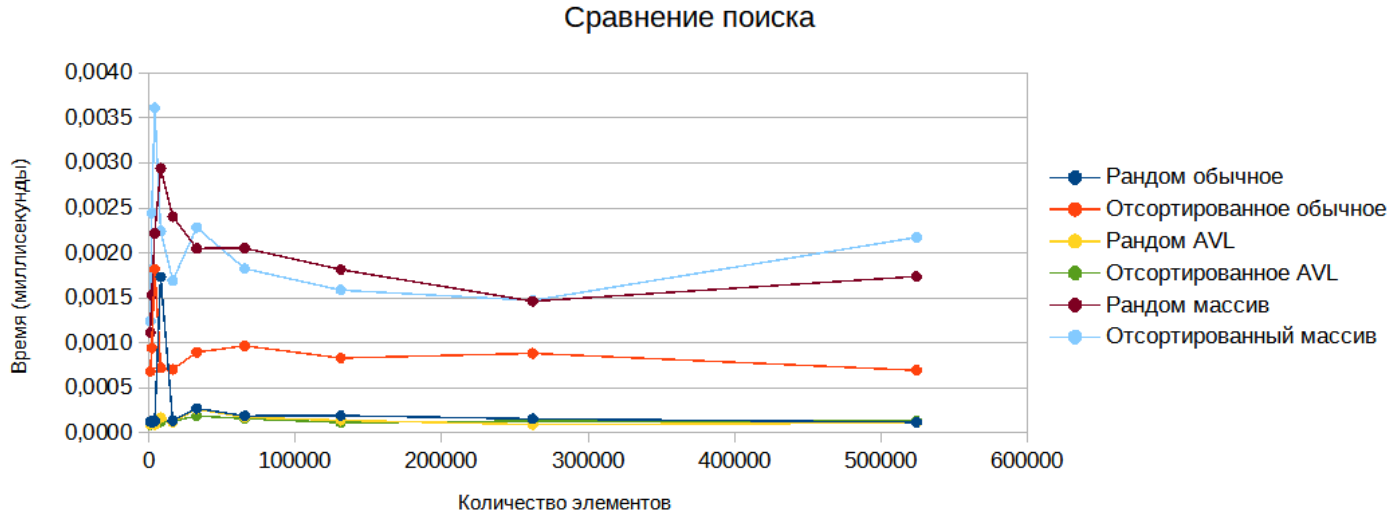


		Поиск в массиве									
		1024	2048	4096	8192	16384	32768	65536	131072	262144	524288
Рандомный	1	0,0011142	0,0011312	0,0013131	0,001999	0,0012183	0,0014698	0,0015038	0,0013197	0,0014625	0,0015143
	2	0,0010889	0,0010977	0,0012565	0,0023576	0,0013436	0,001617	0,0012364	0,001669	0,0013192	0,0014035
	3	0,0010927	0,0011696	0,0012796	0,00294	0,0024031	0,0019295	0,0020523	0,0014895	0,0013394	0,0013099
	4	0,0010329	0,0011904	0,0011256	0,0018473	0,0013653	0,001949	0,0019893	0,0013174	0,0014001	0,0012861
	5	0,0010416	0,0011597	0,0011803	0,0019047	0,0014888	0,0017382	0,0014793	0,0015306	0,0013923	0,0016172
	6	0,0010689	0,0015307	0,0019793	0,0023708	0,0013736	0,0016947	0,0014548	0,0015223	0,0013588	0,0017373
	7	0,0010048	0,0011418	0,0018196	0,0013457	0,0012834	0,002049	0,0015545	0,0018139	0,0013393	0,0011679
	8	0,0010197	0,0012953	0,0022167	0,0023303	0,0015911	0,001247	0,0013646	0,0014902	0,0013382	0,0013157
	9	0,0010584	0,0011623	0,0013112	0,0019467	0,001419	0,0015491	0,0018732	0,0015181	0,0012786	0,0012474
	10	0,0010877	0,0011399	0,0012288	0,0014213	0,0013192	0,0019519	0,0016422	0,0014276	0,001462	0,0011716
Отсортированный	11	0,0011115	0,0012124	0,0012837	0,0014913	0,0013668	0,0014764	0,0017611	0,0014763	0,0013762	0,0013243
	12	0,001073	0,0024429	0,0011856	0,001679	0,0014238	0,0015744	0,0015454	0,0012717	0,0013164	0,0021727
	13	0,0011036	0,0011471	0,0026345	0,0013106	0,0016914	0,002281	0,0014132	0,0015022	0,0013394	0,0013728
	14	0,001001	0,0012184	0,0012674	0,001322	0,0016156	0,0016422	0,0015356	0,0015856	0,0012441	0,0012676
	15	0,0010901	0,0011401	0,0036113	0,0014376	0,0013732	0,0017246	0,0018248	0,0014517	0,0013826	0,0011413
	16	0,001099	0,0013854	0,0028156	0,0013847	0,0014371	0,0015564	0,0014468	0,0013228	0,0014715	0,001164
	17	0,0011022	0,0014934	0,0023355	0,002241	0,001291	0,00162	0,0016494	0,001373	0,0013047	0,0012644
	18	0,0012424	0,0016642	0,0014335	0,0021895	0,0013264	0,0020556	0,0015415	0,0013848	0,0012367	0,0013315
	19	0,0010307	0,0017495	0,0013829	0,0014509	0,0012244	0,001648	0,001589	0,0012731	0,0012271	0,0014139
	20	0,0010621	0,0021534	0,0022136	0,0013822	0,0012537	0,0013666	0,0015247	0,001223	0,0013021	0,0012836



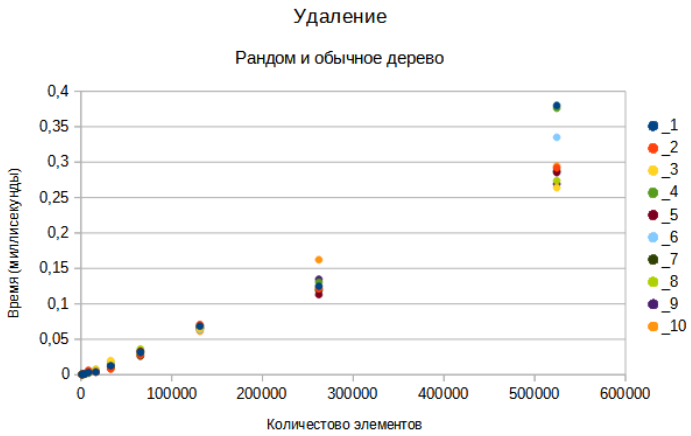


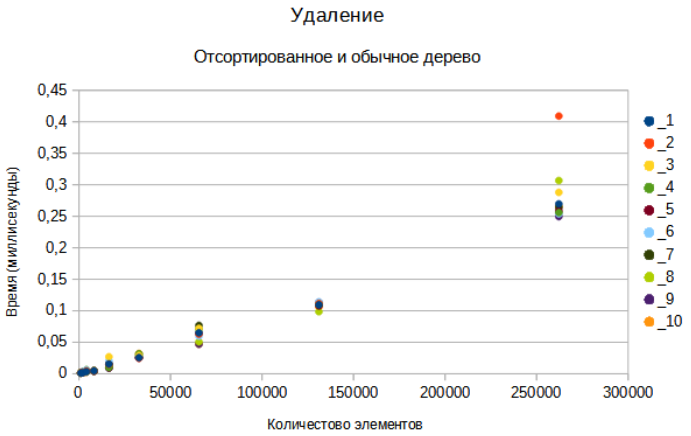
	Сравнение всех вариантов поисков (худшие случаи)									
	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288
Рандом обычное	1,26E-04	1,14E-04	1,29E-04	1,73E-03	1,36E-04	2,72E-04	1,87E-04	1,91E-04	1,55E-04	1,18E-04
Отсортированное обычное	6,82E-04	9,41E-04	1,82E-03	7,23E-04	7,04E-04	8,95E-04	9,67E-04	8,31E-04	8,83E-04	6,95E-04
Рандом AVL	9,35E-05	1,05E-04	9,00E-05	1,68E-04	1,17E-04	2,64E-04	1,71E-04	1,39E-04	9,20E-05	1,10E-04
Отсортированное AVL	8,54E-05	1,25E-04	1,04E-04	1,24E-04	1,27E-04	1,88E-04	1,58E-04	1,14E-04	1,31E-04	1,29E-04
Рандом массив	1,11E-03	1,53E-03	2,22E-03	2,94E-03	2,40E-03	2,05E-03	2,05E-03	1,81E-03	1,46E-03	1,74E-03
Отсортированный массив	1,24E-03	2,44E-03	3,61E-03	2,24E-03	1,69E-03	2,28E-03	1,82E-03	1,59E-03	1,47E-03	2,17E-03



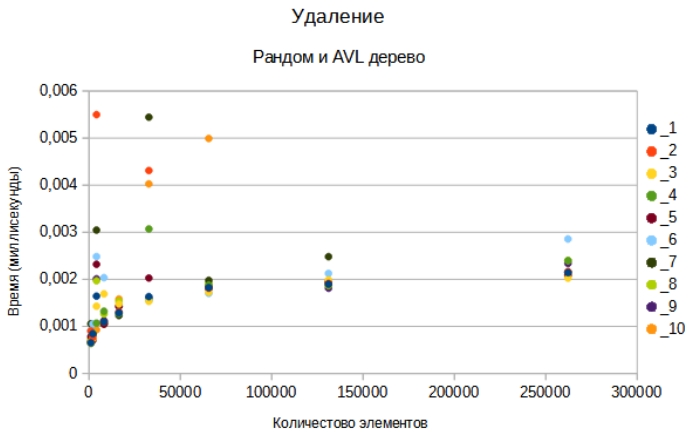
Удаление:

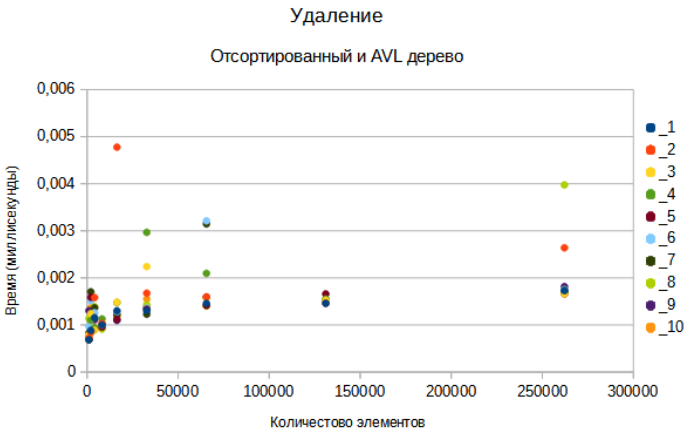
		Удаление из обычного бинарного дерева									
		1024	2048	4096	8192	16384	32768	65536	131072	262144	524288
Рандомный	1	0,0004276	0,0005479	0,0010217	0,0031215	0,0038834	0,0126889	0,0312941	0,0684481	0,125026	0,380061
	2	0,0004197	0,0007729	0,0009068	0,0062278	0,004035	0,0081119	0,0275502	0,0707397	0,121143	0,291726
	3	0,0004467	0,0005255	0,0008917	0,0044946	0,0067658	0,0199645	0,0297411	0,0651611	0,124666	0,263968
	4	0,0003863	0,0005332	0,0008979	0,0026919	0,0042512	0,0114963	0,0309426	0,0700967	0,130646	0,376175
	5	0,0004166	0,0006806	0,0008413	0,0042509	0,004116	0,0104529	0,0329582	0,0661883	0,113116	0,285234
	6	0,000394	0,0008453	0,0013131	0,005075	0,0042605	0,0118255	0,0317258	0,0635487	0,122284	0,335018
	7	0,0005046	0,0005669	0,001263	0,0045664	0,0046168	0,0111103	0,0262787	0,0654054	0,118821	0,287815
	8	0,0004317	0,0006453	0,0012604	0,0042478	0,0076596	0,0181023	0,036252	0,0642221	0,119928	0,273701
	9	0,0004076	0,0005956	0,0011582	0,0043344	0,0047791	0,0146581	0,0335941	0,0679509	0,134733	0,269022
	10	0,0003951	0,0005613	0,0010938	0,0019453	0,0038543	0,0084763	0,0280422	0,0614899	0,162272	0,294366
Отсортированный	11	0,0006988	0,0010526	0,0016321	0,0032634	0,0047077	0,0155342	0,0256406	0,0650792	0,109607	0,269603
	12	0,0006594	0,001563	0,0016598	0,0044536	0,0045472	0,015021	0,024549	0,0628873	0,111337	0,409071
	13	0,0006537	0,0010413	0,0022883	0,0029615	0,004974	0,0269559	0,0290255	0,0721202	0,110621	0,287956
	14	0,0007656	0,00116	0,0022783	0,0031795	0,0046723	0,0097809	0,0316115	0,0709594	0,110302	0,25593
	15	0,000652	0,0010221	0,0021803	0,0028362	0,0043831	0,0093114	0,0253978	0,064241	0,107655	0,25869
	16	0,0009731	0,0014621	0,0035751	0,0064075	0,0055235	0,02021	0,0292673	0,0597391	0,114004	0,253846
	17	0,0006511	0,0014735	0,0023196	0,0037598	0,0043198	0,0155398	0,0266598	0,076772	0,109693	0,264058
	18	0,000761	0,0020917	0,0014853	0,0065539	0,005003	0,0115377	0,030012	0,0506023	0,0985705	0,306727
	19	0,0007908	0,001477	0,0017754	0,0045566	0,0043882	0,0087822	0,0249294	0,0469913	0,113404	0,249517
	20	0,0006431	0,0013972	0,0028388	0,0030743	0,0045047	0,0144113	0,0321188	0,0470537	0,106021	0,265681



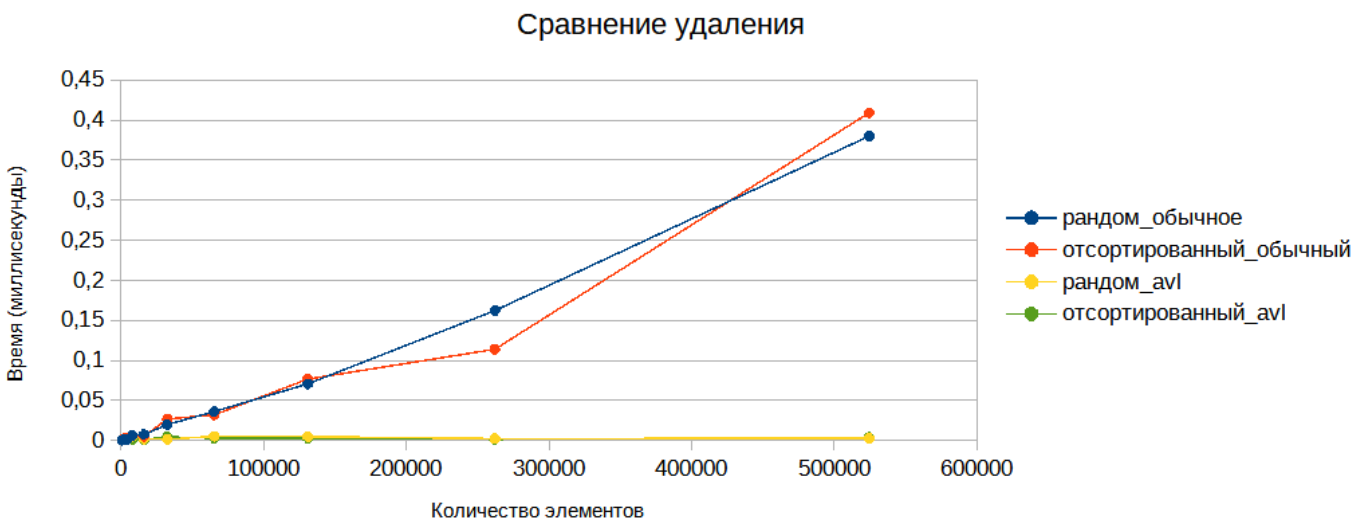


		Удаление из AVL бинарного дерева									
		1024	2048	4096	8192	16384	32768	65536	131072	262144	524288
Рандомный	1	0,0005735	0,0006505	0,0008438	0,0016421	0,0010968	0,0012896	0,0016331	0,0018218	0,0018993	0,0021414
	2	0,0006304	0,0009014	0,000765	0,0054986	0,0010956	0,0013104	0,0043104	0,00181	0,0019139	0,0021645
	3	0,0006119	0,0006471	0,0007349	0,0014284	0,0016906	0,0014733	0,0015327	0,0017267	0,0019878	0,0020246
	4	0,0005747	0,0006376	0,0008245	0,001069	0,0013228	0,0012771	0,0030705	0,0018818	0,0018821	0,0023969
	5	0,0005815	0,0007835	0,0007028	0,0023187	0,0010481	0,0014288	0,0020273	0,0018486	0,0019389	0,0020903
	6	0,0005787	0,000911	0,001052	0,0024831	0,0020354	0,0012798	0,0015585	0,0017005	0,0021248	0,0028566
	7	0,0006414	0,0010557	0,000823	0,0030455	0,0010538	0,0012296	0,0054446	0,0019792	0,0024814	0,0021541
	8	0,0005741	0,000726	0,0008742	0,0019688	0,0012615	0,0015359	0,001573	0,0019271	0,0018669	0,0023978
	9	0,0005726	0,0006786	0,0008427	0,00201	0,0011226	0,0014254	0,0015982	0,0018016	0,0018113	0,0023376
	10	0,0006244	0,0006661	0,0009225	0,0009274	0,0010544	0,0015814	0,0040276	0,004992	0,0019543	0,0020764
Отсортированный	11	0,0005614	0,0006861	0,0008846	0,0011557	0,0009996	0,0013013	0,0013048	0,0014578	0,0014593	0,0017288
	12	0,0005752	0,000723	0,0008485	0,0015821	0,0010341	0,0047773	0,0016745	0,0015969	0,0014583	0,00264
	13	0,0005918	0,0007935	0,0012394	0,0008884	0,0009993	0,0014807	0,0022405	0,0015689	0,001527	0,0016626
	14	0,0005425	0,0006883	0,0010995	0,0009409	0,0011271	0,001476	0,0029672	0,002096	0,0015503	0,0017326
	15	0,0005371	0,000685	0,0015878	0,0009084	0,0009539	0,0011218	0,0013327	0,0014162	0,0016588	0,0016606
	16	0,0007498	0,0009671	0,0015163	0,0012601	0,0009755	0,0012801	0,0013604	0,0032122	0,0014963	0,0017522
	17	0,0005353	0,0008279	0,0017041	0,0013632	0,0009904	0,00122	0,001228	0,0031463	0,0015225	0,0017184
	18	0,0006318	0,0011382	0,0008677	0,0013882	0,0009123	0,0011965	0,0014295	0,0014034	0,0015428	0,0039743
	19	0,0005773	0,0013023	0,0008404	0,0011092	0,0010006	0,0010972	0,0013745	0,0015892	0,0015129	0,0018158
	20	0,0005404	0,0007387	0,0013643	0,0009355	0,0010183	0,0012197	0,0015506	0,001405	0,0015362	0,0017117





	Сравнение всех вариантов удаления (худшие случаи)									
	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288
Рандом обычное	0,0005046	0,0008453	0,0013131	0,0062278	0,0076596	0,0199645	0,0362520	0,0707397	0,1622720	0,3800610
Отсортированное обычное	0,0009731	0,0020917	0,0035751	0,0065539	0,0055235	0,0269559	0,0321188	0,0767720	0,1140040	0,4090710
Рандом AVL	0,0006414	0,0010557	0,0010520	0,0054986	0,0020354	0,0015814	0,0054446	0,0049920	0,0024814	0,0028566
Отсортированное AVL	0,0007498	0,0013023	0,0017041	0,0015821	0,0011271	0,0047773	0,0029672	0,0032122	0,0016588	0,0039743



Заключение.

Из полученных результатов можно сделать вывод, что все операции в сбалансированном дереве происходят несколько быстрее, чем в обычном бинарном дереве, так как балансировка позволяет равномерно расставить все элементы, что сильно ускоряет каждую операцию в дереве. Если говорить конкретно про поиск в данных бинарных деревьях, то из графиков заметно, что поиск в обычном бинарном дереве, составленном из отсортированной последовательности чисел, будет наихудшим вариантов поиска в бинарном дереве. Это связано с тем, что дерево представляет из себя своего рода массив, так как при заполнении числа будут вставать каждый раз в правых потомков. Поэтому время поиска по такому бинарному дереву будет приближаться к времени поиска по массиву. Однако поиск по бинарному дереву все же быстрее, чем поиск по обычному массиву. Это связано с тем, что поиск по массиву происходит путем последовательного сравнения каждого элемента с искомым значением, что сильно затормаживает процесс.