

**Отчёт по лабораторной работе
«Деревья двоичного поиска»
по дисциплине
«Алгоритмы и структуры данных»**

Вариант 18

*Выполнили студенты
факультета КТИ группы №3305
Лоуцкер Алексей и Григорьева Анна*

*Проверил старший преподаватель
Колинко Павел Георгиевич*

Оглавление

1. Задание.....	3
2. Описание структуры данных.....	4
2. Тестовый пример.....	5
3. Оценки временной сложности операций.....	6
4. Выводы.....	7
5. Список использованной литературы.....	7
6. Приложение: исходные тексты.....	7

1. Задание

Переделать программу, составленную по теме «Хеш-таблицы», под использование деревьев двоичного поиска с автобалансировкой. Дополнить операциями над последовательностями

Вариант 18

Дерево: AVL-дерево с автобалансировкой

Мощность множества: 32

Формула: $E = A \vee (B \wedge C) \setminus D$

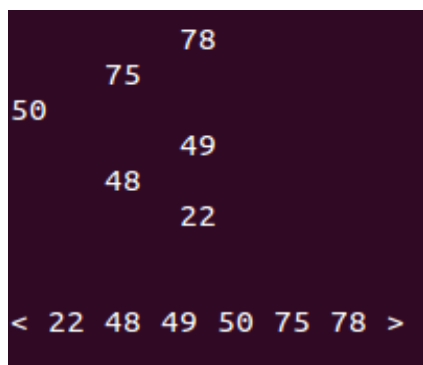
Операции над последовательностями

1. Слияние (merge)
2. Включение (subst)
3. Замена (change)

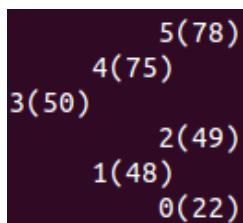
2. Описание структуры данных

Отличительная особенность нашей структуры данных в том, что она может работать в двух режимах и переключаться между ними:

1) Режим value — аналог `std::set`. Данные хранятся в виде бинарного дерева, без повторов, поддерживается быстрый поиск элемента по значению, относительно быстрые вставка/удаление. В этом режиме доступны 'логические' операции: объединение, пересечение, вычитание. Вывод на экран в виде дерева или упорядоченной по возрастанию последовательности



2) Режим sequence. Дерево строится не по значениям элементов, а по их позициям в последовательности.



В этом режиме доступны вставка/удаление элемента на любую позицию, операции: слияние, вставка и замена, а также вывод на экран в виде дерева или последовательности элементов в соответствии с их позицией.

Также возможно переключение между режимами с потерей позиций элементов или повторов

2. Тестовый пример

A:	B:	C:	D:	E:
97	91	92	95	97
94	90	87	93	94
86	83	82	92	86
85	83	80	88	85
83	82	78	80	83
82	77	76	76	82
65	65	68	68	65
60	69	60	66	65
57	64	52	64	57
54	61	52	60	54
52	60	49	54	52
52	58	52	52	50
50	55	48	52	49
49	44	48	49	47
47	43	46	43	46
46	38	42	43	46
42	35	41	40	42
35	32	33	32	35
32	25	31	31	26
26	23	30	30	26
26	19	27	29	24
24	17	26	26	22
22	14	19	25	22
18	13	15	18	18
12	11	11	17	12
12	9	10	11	9
9	5	9	9	6
6	4	5	0	5
0	0	0	0	0

```

=====
< 8 10 12 18 19 21 24 25 28 29 34 38 39 42 47 50 52 53 58 59 64 69 72 73 75 77 79 83 85 86 >
change:
< 0 0 0 0 0 >
==> < 8 10 12 18 19 0 0 0 0 0 34 38 39 42 47 50 52 53 58 59 64 69 72 73 75 77 79 83 85 86 >
=====
merge:
< 0 0 0 1 1 1 97 98 99 >
==> < 0 0 0 0 0 0 0 0 1 1 1 8 10 12 18 19 34 38 39 42 47 50 52 53 58 59 64 69 72 73 75 77 79 83 85 86 97 98 99 >
=====
subst:
< 1 2 3 4 5 6 7 8 9 >
==> < 0 1 2 3 4 5 6 7 8 9 0 0 0 0 0 0 0 0 1 1 1 8 10 12 18 19 34 38 39 42 47 50 52 53 58 59 64 69 72 73 75 77 79 83 85 86 97 98 99 >
=====

```

3. Оценки временной сложности операций

1) Режим value

Функции поворотов и балансировки не содержат ни циклов, ни рекурсии, а значит выполняются за постоянное время, не зависящее от размера AVL-дерева.

Операции вставки и удаления (а также поиска) выполняются за время, пропорциональное высоте дерева, т.к. в процессе выполнения этих операций производится спуск из корня к заданному узлу, и на каждом уровне выполняется некоторое фиксированное число действий. А в силу того, что AVL-дерево является сбалансированным, его высота зависит логарифмически от числа узлов. Таким образом, время выполнения всех трех базовых операций логарифмически зависит от числа узлов дерева.

Двухместные операции получают результат в виде вектора значений, а затем последовательно вставляют элементы в новое дерево

Операция	Сложность
балансировка, повороты	$O(1)$
вставка, удаление, поиск	$O(\log(n))$
пересечение, объединение, вычитание	$O(n \cdot \log(n))$

2) Режим sequence

При вставке/удалении элемента последовательности необходимо не только удалить его из дерева, но и обновить номера остальных элементов, поэтому сложность — линейная

Операции над последовательностями создают из двух последовательностей одну с помощью вставок и удалений, поэтому их сложность пропорциональна квадрату числа элементов.

Преимущество выбранного способа хранения последовательности - быстрый вывод и доступ к элементу по его позиции при любом количестве повторов; не требует дополнительной памяти

Операция	Сложность
балансировка, повороты	$O(1)$
доступ по номеру	$O(\log(n))$
вставка, удаление, вывод на экран	$O(n)$
слияние, подстановка, замена	$O(n^2)$

4. Выводы

Реализованная структура данных, также как и хеш-таблица, поддерживает простейшие и двухместные операции над множеством ключей. За счет строгой внутренней организации AVL-дерево позволяет получить выигрыш в скорости при выполнении двухместных операций, но из-за необходимости регулярно выполнять балансировку, вставка, поиск и удаление требуют больше времени.

Режим последовательности легко справляется с большим количеством повторяющихся элементов, однако, если таких не слишком много, то вариант хранения позиций элементов другим способом будет иметь лучшую среднюю оценку.

5. Список использованной литературы

1) Алгоритмы и структуры данных: методические указания к лабораторным работам, практическим занятиям и курсовому проектированию. Часть 2 / сост. П. Г. Колинко. - СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2015

2) <http://stackoverflow.com>

3) <http://cplusplus.com>

4) <http://habrahabr.ru/post/150732/>

6. Приложение: исходные тексты

Исходный код доступен в репозитории по адресу:

github.com/alout1/avl-tree/sequences