Лабораторная работа 1. Сбалансированные деревья поиска. Красно-черное дерево.

Постановка задачи

Реализовать структуру данных «красно-черное дерево». Реализовать функции для работы с красно-черным деревом:

```
struct rbtree *rbtree_add(struct rbtree *root,
                          int key, char *value);
struct rbtree *rbtree_lookup(struct rbtree *root, int key);
struct rbtree *rbtree_delete(struct rbtree *root, int key);
struct rbtree *rbtree_min(struct rbtree *root);
struct rbtree *rbtree_max(struct rbtree *root);
void rbtree_free(struct rbtree *root);
void rbtree_print_dfs(struct rbtree *root, int level);
```

Информацию о бинарном дереве поиска и его функциях можно найти в 7-й лекции 2-го семестра. Реализация бинарного дерева поиска та же, что и в 3-й лабораторной работе 2-го семестра, в качестве ключа нужно будет использовать целое число типа *int*.

Если лабораторная работа сдается после 28.09, количество баллов за лабораторную снижается на 2, после 05.10. – на 5.

Экспериментальное исследование

- В качестве ключа использовать элементы типа *int*.
- Для среднего случая для добавления генерировать псевдослучайные числа в диапазоне [10 000, 1 000 000].
- Для худшего случая добавлять в красно-черное дерево и бинарное дерево поиска элементы в порядке возрастания.
- Для каждого случая и количества элементов использовать функцию *rbtree_lookup* и *bstree_lookup* для заполнения таблицы 1. Элемент для поиска — последний добавленный. Рекомендации по более точному измерению времени можно найти в файле *README.pdf*.
- Для каждого случая и количества элементов использовать функцию *rbtree_max* и *bstree_max* для заполнения таблицы 2.
- По результатам экспериментов определите вычислительную сложность вызываемой функции. Объясните результаты.

Таблица 1. Поиск элемента в бинарном дереве поиска и красно-черном дереве

#	Кол-во элементов	Время выполнения функции *bstree_lookup* в среднем случае, с	Время выполнения функции *rbtree_lookup* в среднем случае, с	Время выполнения функции *bstree_lookup* в худшем случае, с	Время выполнения функции *rbtree_lookup* в худшем случае, с
1	20000				
2	40000				
3	60000				
...	...				
10	200000				

Таблица 2. Поиск максимального элемента в бинарном дереве поиска и красно-черном дереве

#	Кол-во элементов	Время выполнения функции *bstree_max* в среднем случае, с	Время выполнения функции *rbtree_max* в среднем случае, с	Время выполнения функции *bstree_max* в худшем случае, с	Время выполнения функции *rbtree_max* в худшем случае, с
1	20000				
2	40000				
3	60000				
...	...				
10	200000				

Контрольные вопросы

- Что такое сбалансированное дерево поиска? Какие сбалансированные деревья поиска вам известны?
- Структура узла красно-черного дерева.
- Объяснить принцип работы красно-черного дерева.
- Вычислительная сложность функций красно-черного дерева.
- Объяснить и продемонстрировать алгоритм добавления узлов в красно-черное дерево.
- Объяснить и продемонстрировать алгоритм удаления узлов из красно-черного дерева.
- Сравнить вычислительную сложность функций красно-черного дерева и бинарного дерева поиска, объяснить результат.
- Доказать утверждение о высоте красно-черного дерева.
- Анализ вычислительной сложности функций красно-черного дерева и бинарного дерева поиска.