­­Бюджетное учреждение высшего образования   
Ханты-Мансийского автономного округа   
«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**Отчет**

по лабораторной работе № 6

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы 609-21,

Шумилов И.Д.

Принял: старший преподаватель кафедры АиКС

Назаров Е.В.

2024 г.

**Цель работы:** изучить основные принципы построения сбалансированных и цифровых деревьев, их свойства и назначение, закрепить навыки структурного программирования.

**Общее задание на работу:**

1. Реализовать один из вариантов задания в соответствии с таблицей 2 и тестовое приложение. При необходимости реализовать вспомогательные структуры данных и функции, например, функции вывода на дерева на экран, удаления одного узла и т.п.
2. Составить отчет, в котором привести листинги реализованных функций, свойства и отличительные особенности дерева, для которого были реализованы функции и выводе по работе.

**Индивидуальное задание на работу:**

Таблица 1 – Вариант 4 индивидуального ндивидуального задания

|  |  |
| --- | --- |
| № | Задание |
| 4 | Вставка ключа в скошенное дерево |

**Свойства и отличительные особенности скошенных деревьев.**

Скошенные деревья – это сбалансированное двоичное дерево поиска, которое относится к классу саморегулирующихся деревьев. Балансировка при этом достигается за счет операций расширения (splay), частью которых является вращения, которые выполняются при каждом обращении к дереву таким образом, чтобы узел, к которому обратились становился новым корнем дерева.

Когда говорят о скошенных деревья повороты принято раделять на 3 типа: Zig, Zig-Zig, Zig-Zag. Zig – это одинарный поворот либо влево, либо вправо. Zig-Zig – это двойной поворот дважды в одну сторону, то есть либо дважды влево, либо дважды вправо. Zig-Zag – это двойной поворот, но первый поворот делается в одну сторону, а второй в другую. Таким образом, чередуя данные повороты в определенном порядке можно вывести любой узел в корень, при этом не нарушая правил бинарных деревьев поиска.

При использовании на практике скошенные деревья очень эффективные благодаря своим свойствам. Чаще всего используются для систем кэширования или индексации баз данных.

Учетная стоимость на одну операцию с деревом (вставки, поиска, удаления) составляет O(log n).

**Код программы**

Листинг 1. Класс узла.

class Node {

public:

Node() : key(0), size(1), left(nullptr), right(nullptr), parent(nullptr) {};

Node(int key) : size(1), left(nullptr), right(nullptr), parent(nullptr) { this->key = key; };

int key;

int size;

Node\* left;

Node\* right;

Node\* parent;

};

Листинг 2. Функция вставки в скошенное дерево.

Node\* insert(Node\* root, int key) {

if (root == nullptr)

return new Node(key);

root = splay(root, key);

if (root->key == key)

return root;

Node\* node = new Node(key);

if (key < root->key) {

node->right = root;

node->left = root->left;

root->left = nullptr;

} else {

node->left = root;

node->right = root->right;

root->right = nullptr;

}

return node;

}

Листинг 3. Функция расширения скошенного дерева.

Node\* splay(Node\* root, int key) {

if (!root || root->key == key)

return root;

if (key < root->key) {

if (!root->left)

return root;

if (key < root->left->key) {

root->left->left = splay(root->left->left, key);

root = rightRotate(root);

} else if (key > root->left->key) {

root->left->right = splay(root->left->right, key);

if (root->left->right) root->left = leftRotate(root);

}

return (!root->left) ? root : rightRotate(root);

} else {

if (!root->right)

return root;

if (key < root->right->key) {

root->right->left = splay(root->right->left, key);

if (root->right->left) root->right = rightRotate(root);

} else if (key > root->right->key) {

root->right->left = splay(root->right->right, key);

root = leftRotate(root);

}

return (!root->right) ? root : leftRotate(root);

}

}

Листинг 4. Функция вывода дерева на экран.

void print(Node\* root) {

if (root) {

int lk = 0, rk = 0;

if (root->left)

lk = root->left->key;

if (root->right)

rk = root->right->key;

if (root->left || root->right) std::cout << "-------node-------" << std::endl;

else std::cout << "-------leaf-------" << std::endl;

std::cout << "this\tleft\tright" << std::endl;

std::cout << root->key << "\t" << lk << "\t" << rk << "\n\n" << std::endl;

print(root->left);

print(root->right);

}

}

**Пример выполнения реализованных функций.**

Вставка происходила в следующем порядке: 50, 100, 150, 200, 500, 175. С помощью функции расширения при вставке узел с ключом 175, несмотря на то что был вставлен самым последним является корнем дерева. И зная свойства дерева можно смело утверждать, что любой узел, вставленный последним будет в корне дерева.

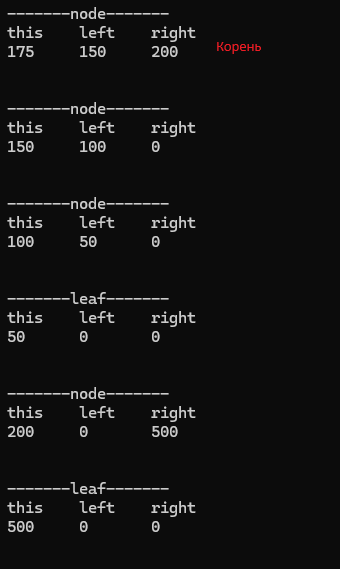


Рис. 1 – Пример работоспособности вставки.

**Вывод:** были изучены основные принципы построения сбалансированных и цифровых деревьев, их свойства и назначение, закреплены навыки структурного программирования.