МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по индивидуальному заданию №1**

**«ДЕРЕВЬЯ»**

**по курсу**

**«ОЦЕНКА СЛОЖНОСТИ АЛГОРИТМОВ»**

Работу выполнил

Студент 39 группы

Зайцев А.С.

Преподаватель:

Нигодин Е.А.

**Итеративная процедура, подсчитывающая количество вершин с двумя детьми.**

Метод возвращает искомое число.

Оценка сложности:

Пространственная сложность совпадает с количеством вершин в дереве – это V.

Теперь определим временную сложность, итеративный метод работу по алгоритму BFS.

Первая операция — это внесение дерева в стек: 1

Следующим идет цикл while, который работает до тех пор, пока стек не пуст. Размер стека зависит от размера дерева и потому максимум может достичь размера n/2. Но цикл пройдёт по всем имеющимся вершинам. Из-за этого лучший, средний и худшего случая не будет, алгоритм всегда будет проходить все имеющиеся вершины.

Оператор сравнения: 1.

В случае положительности работает оператор присваивания:1.

Добавляем в очередь вершины, если они есть, оператор сравнения и присваивания: 2

Итого сложность будет

**Рекурсивная процедура, подсчитывающая количество вершин с двумя детьми.**

На вход поступает узел. Если узел пустой, возвращаем 0. Иначе рекурсивно вызываем функцию для его детей. Тогда точное значение сложности имеем вид:

**Экспериментальная сложность алгоритма**

Таблица №1

| Количество | Рекурсия | Итерация |
| --- | --- | --- |
| 10 | 0.0000069 | 0.0000174 |
| 100 | 0.000007 | 0.0000833 |
| 1000 | 0.0000171 | 0.0007389 |
| 10000 | 0.0002144 | 0.0061134 |
| 100000 | 0.0088632 | 0.0597321 |
| 1000000 | 0.107301 | 0.561517 |

Результаты выполнения представлены в виде графика на рисунке 1.

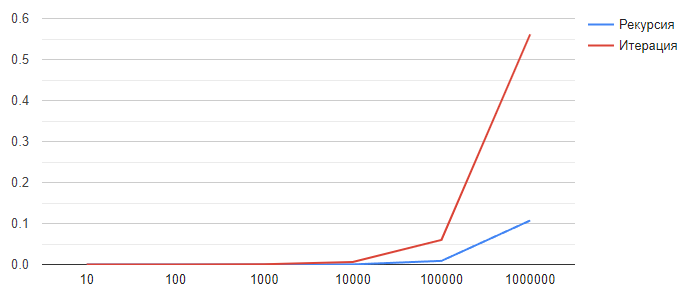


Рисунок 1 – Эксперимент

**Результаты работы программы:**

Результат работы программы представлен на рисунке 2.

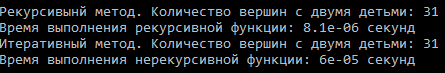


Рисунок 2 – Работа программы

**Текст программы:**

#include <iostream>

#include <chrono>

#include <random>

#include <stack>

struct Node {

int data;

Node\* left;

Node\* right;

Node(int value) : data(value), left(NULL), right(NULL) {}

};

int i = 1;

void fillTree(Node\* node, int data)

{

if (data < node->data)

{

if (node->left == NULL)

{

Node\* nod = new Node(data);

nod->data = data;

nod->left = NULL;

nod->right = NULL;

node->left = nod;

}

else

fillTree(node->left, data);

}

else

{

if (node->right == NULL)

{

Node\* nod = new Node(data);

nod->data = data;

nod->left = NULL;

nod->right = NULL;

node->right= nod;

}

else

fillTree(node->right, data);

}

}

void preorderPrint(Node\* root)

{

if (root == NULL)

{

return;

}

std::cout << root->data << " ";

preorderPrint(root->left);

preorderPrint(root->right);

}

int countNodes(Node\* root) {

if (root == nullptr) {

return 0;

}

if (root->right && root->left) {

return 1 + countNodes(root->left) + countNodes(root->right);

}

return countNodes(root->left) + countNodes(root->right);

}

int iterativeCount(Node\* root)

{

int count = 0;

std::stack<Node\*> s;

s.push(root);

while (s.empty() == false)

{

Node\* temp = s.top();

s.pop();

if(temp->left&&temp->right){

count++;

}

if (temp->right)

s.push(temp->right);

if (temp->left)

s.push(temp->left);

}

return count;

}

int main() {

setlocale(LC\_ALL, "Russian");

Node\* root = new Node(5);

int n = 100;

for (int i = 0; i < n; i++) {

fillTree(root,rand()%n+1);

}

auto startRecursive = std::chrono::high\_resolution\_clock::now();

int recNodesCount = countNodes(root);

auto endRecursive = std::chrono::high\_resolution\_clock::now();

//preorderPrint(root);

std::cout << std::endl;

std::cout << "Рекурсивынй метод. Количество вершин с двумя детьми: " << recNodesCount << std::endl;

std::chrono::duration<double> elapsedRecursive = endRecursive - startRecursive;

std::cout << "Время выполнения рекурсивной функции: " << elapsedRecursive.count() << " секунд" << std::endl;

auto startNonRecursive = std::chrono::high\_resolution\_clock::now();

int iterNodesCount = iterativeCount(root);

auto endNonRecursive = std::chrono::high\_resolution\_clock::now();

std::cout << "Итеративный метод. Количество вершин с двумя детьми: " << iterNodesCount << std::endl;

std::chrono::duration<double> elapsedNonRecursive = endNonRecursive - startNonRecursive;

std::cout << "Время выполнения нерекурсивной функции: " << elapsedNonRecursive.count() << " секунд" << std::endl;

return 0;

}