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**

о выполнении лабораторной работы № 2

по дисциплине «Оценка сложности алгоритмов»

Выполнил:

Воробьев А.Д.

Проверил: преподаватель

Лапина О.Н.

Краснодар

2024

**Рекурсивная процедура подсчета вершин в бинарном упорядоченном дереве(рекурсия)**

Метод «obh\_recurs» возвращает количество вершин в этом дереве. Рекурсивно вызывается для левого и правого поддерева, прибавляя 1 за каждую вершину.

Оценка сложности:

Пространственная сложность связана с размером стека вызовов, который будет пропорционален глубине рекурсии, следовательно, равна глубине дерева

Временная сложность составляет

Итого сложность будет

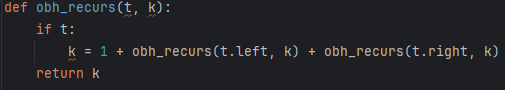


Рисунок 1. Рекурсивный обход

**Рекурсивная процедура подсчета вершин в бинарном упорядоченном дереве (Итеративная версия)**

Метод «iterativeCount» аналогичен рекурсивно, представлен в виде итерации (соответствует DFS), и каждая вершина посещается ровно один раз, следовательно временная сложность будет

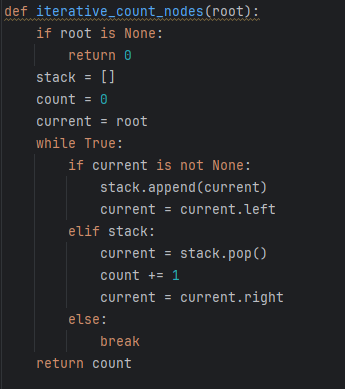


Рисунок 2. Итеративный обход

**Экспериментальная сложность алгоритма**

Таблица №1

|  |  |  |
| --- | --- | --- |
| Количество | Рекурсия | Итерация |
| 50 | 0.000017 | 0.000007 |
| 500 | 0.000088 | 0.000062 |
| 1000 | 0.000152 | 0.000127 |
| 2000 | 0.000254 | 0.000240 |
| 3000 | 0.000367 | 0.000373 |
| 5000 | 0.000564 | 0.000540 |

На 5000 – произошло переполнение стека. Результаты на рисунке 1.

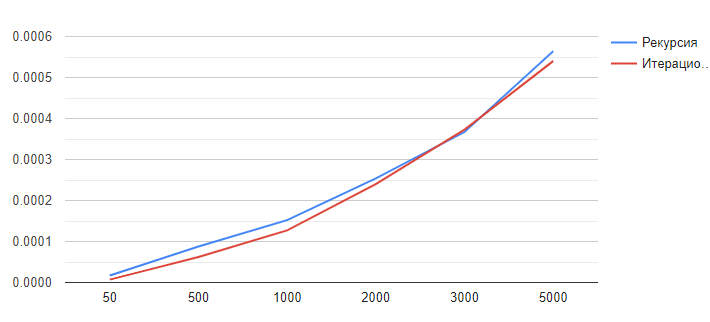


Рисунок 3 – Эксперимент

**Найти сложность алгоритма с косвенной рекурсией.**

Сложность косвенной рекурсии в том, что в данном случае нужно учитывать вызовы функций, которые могут следовать не прямо, а через несколько других функций. Так как сложность обычной рекурсии

то для косвенной рекурсии в данном случае, будет

Следовательно сложность составит

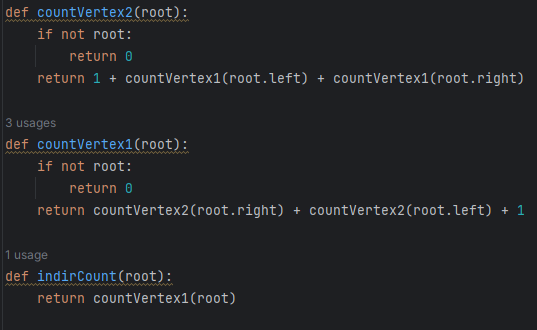


Рисунок 4. Косвенная рекурсия

**Результаты работы программы:**

На рисунке 2 представлен результат работы программы – работа с деревом и вычисления.

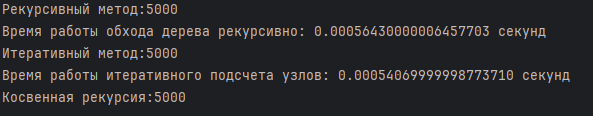


Рисунок 5 – Работа программы

**Листинг программы:**

import timeit  
  
class Tree:  
 def \_\_init\_\_(self, val=0, left=None, right=None):  
 self.val = val  
 self.left = left  
 self.right = right  
  
def completionTree(root, e):  
 if not root.left:  
 root.left = Tree(e)  
 elif not root.right:  
 root.right = Tree(e)  
 else:  
 if iterative\_count\_nodes(root.left) <= iterative\_count\_nodes(root.right):  
 completionTree(root.left, e)  
 else:  
 completionTree(root.right, e)  
  
def obh\_recurs(t, k):  
 if t:  
 k = 1 + obh\_recurs(t.left, k) + obh\_recurs(t.right, k)  
 return k  
  
def iterative\_count\_nodes(root):  
 if root is None:  
 return 0  
 stack = []  
 count = 0  
 current = root  
 while True:  
 if current is not None:  
 stack.append(current)  
 current = current.left  
 elif stack:  
 current = stack.pop()  
 count += 1  
 current = current.right  
 else:  
 break  
 return count  
  
def countVertex2(root):  
 if not root:  
 return 0  
 return 1 + countVertex1(root.left) + countVertex1(root.right)  
  
def countVertex1(root):  
 if not root:  
 return 0  
 return countVertex2(root.right) + countVertex2(root.left) + 1  
  
def indirCount(root):  
 return countVertex1(root)  
  
  
t = Tree(0)  
root = t  
for i in range(1, 5000):  
 completionTree(root, i)  
  
k = 0  
start = timeit.default\_timer()  
print('Рекурсивный метод:' + str(obh\_recurs(root, k)))  
finish = timeit.default\_timer()  
print('Время работы обхода дерева рекурсивно: {:.20f} секунд'.format(finish - start))  
  
start = timeit.default\_timer()  
print('Итеративный метод:' + str(iterative\_count\_nodes(root)))  
finish = timeit.default\_timer()  
print('Время работы итеративного подсчета узлов: {:.20f} секунд'.format(finish - start))  
  
print('Косвенная рекурсия:' + str(indirCount(root)))