

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

|  |
| --- |
| **Институт информационных технологий (ИТ)** |
| **Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)** |

**ОТЧЁТ ПО ПРАКТИЧЕСКОМУ ЗАДАНИЮ №3**

**по дисциплине «Структуры и алгоритмы обработки данных»**

Тема: **«**Нелинейные структуры данных. Бинарное дерево.»

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет представлен к рассмотрению:  Студент группы ИНБО-01-20 | «12» октября 2021 г. |  | Салов В.Д. |
| (подпись) | | | |
| Преподаватель | «12» октября 2021 г. |  | Сорокин А.В. |
|  | (подпись) | | |

Москва, 2021 г.

CОДЕРЖАНИЕ

[Цель работы 3](#_Toc85234589)

[Постановка задачи 3](#_Toc85234590)

[Подход к решению. 3](#_Toc85234591)

[Алгоритмы операций на псевдокоде. 5](#_Toc85234592)

[Код приложения. 6](#_Toc85234593)

[Ответы на вопросы 9](#_Toc85234594)

[Вывод 18](#_Toc85234595)

[Список информационных источников 19](#_Toc85234596)

# Цель работы

Получение умений и навыков разработки и реализаций операций над структурой данных бинарное дерево.

# Постановка задачи

Разработать программу, которая создает идеально сбалансированное дерево из n узлов и выполняет операции.

**Вариант 5.**

|  |  |
| --- | --- |
| Значение информационной части | Операции варианта |
| Вещественное число | Вычислить среднее арифметическое чисел левого поддерева, а также и правого, по отдельности. Удалить двоичное дерево |

**Дано:**

Идеально сбалансированное бинарное дерево из n узлов.

Структура узла дерева, включающая в себя информационную часть узла, указатель на левое и указатель на правое поддерево. Информационная часть узла – вещественное число.

**Результат.**

Отображение на экране дерева, повернутого справа налево.

Реализованные операции варианта.

## Подход к решению.

1. В ходе работы над задачей был разработан класс идеально сбалансированного бинарного дерева, реализующий, согласно варианту, методы создания бинарного дерева из массива элементов, вычисления среднего арифметического левого и правого поддеревьев, а также удаления дерева.
2. В ходе работы был разработан класс узла бинарного дерева. Он содержит информационную часть и ссылки на левое и правое поддерево. В узле реализованы методы распределения элементов входного массива на свою информационную часть и на правое и левое поддеревья, а также метод вывода дерева на экран.
3. В ходе работы был разработан класс приложения с графическим пользовательским интерфейсом для тестирования работоспособности программы.
4. В ходе решения были разработаны методы обработки бинарного дерева:
   1. Вычисление среднего арифметического – метод, возвращающий массив из двух элементов: среднего арифметического левого поддерева и среднего арифметического правого поддерева.
   2. Удаление дерева – метод, задающий стартовому узлу значение ‘None’, тем самым удаляя всё дерево.
   3. Горизонтальный вывод дерева – последовательный вывод дерева, начиная с правого элемента, в процессе которого отступ от левого края выставляется сообразно глубине элемента.

## Алгоритмы операций на псевдокоде.

**Метод вычисления среднего арифметического для поддеревьев:**

функция calculate\_average(elements := массив[вещественный тип]):  
 elements.сортировать()  
 middle := длина(elements) // 2  
 sum\_left = sum\_right := 0.0  
 count\_left = count\_right := 0.0  
 для каждого i в elements от 0 до middle:  
 sum\_left := sum\_left + i  
 count\_left := count\_left + 1  
 для каждого i в elements от middle + 1 до конец\_массива(elements):  
 sum\_right := sum\_right + i  
 count\_right := count\_right + 1  
 average\_left := sum\_left / count\_left  
 average\_right := sum\_right / count\_right  
 возврат [average\_left, average\_right]

**Метод удаления дерева:**процедура delete():  
 стартовый\_узел := пустой

**Метод вывода дерева (повёрнутого справа налево):**

процедура add\_to\_hor\_list(список, глубина):  
 если правый\_узел не пустой:  
 правый\_узел.add\_to\_hor\_list(список, глубина + 1)  
 список.добавить\_элемент(' ' \* глубина + перевод\_в\_строковый\_формат(значение\_узла))  
 если левый узел не пустой:  
 левый\_узел.add\_to\_hor\_list(список, глубина + 1)

## Код приложения.

Класс узла дерева:

class Node:  
 def \_\_init\_\_(self, members: list):  
 if len(members) == 0:  
 pass  
 elif len(members) == 1:  
 self.info = members[0]  
 self.right = None  
 self.left = None  
 elif len(members) == 2:  
 self.info = members[1]  
 self.left = Node([members[0]])  
 self.right = None  
 else:  
 middle = len(members) // 2  
 self.left = Node(members[0: middle])  
 self.info = members[middle]  
 self.right = Node(members[middle + 1:])  
  
 def add\_to\_hor\_list(self, ls: list, depth: int) -> None:  
 if self.right is not None:  
 self.right.add\_to\_hor\_list(ls, depth + 1)  
 ls.append(' ' \* depth + str(self.info))  
 if self.left is not None:  
 self.left.add\_to\_hor\_list(ls, depth + 1)

Класс дерева:

class Tree:  
 def \_\_init\_\_(self, elements: str):  
 elements = [float(x) for x in elements.split()]  
 elements.sort()  
 self.\_\_start = Node(elements)  
  
 def \_\_str\_\_(self):  
 if self.\_\_start == None: return ''  
 ls = []  
 self.\_\_start.add\_to\_hor\_list(ls, 0)  
 return '\n'.join(ls)  
  
 def calculate\_average(self, elements: str) -> int:  
 elements = [float(x) for x in elements.split()]  
 elements.sort()  
 middle = len(elements) // 2  
 sum\_left = sum\_right = 0.0  
 count\_left = count\_right = 0.0  
 for i in elements[0: middle]:  
 sum\_left += i  
 count\_left += 1  
 for i in elements[middle + 1:]:  
 sum\_right += i  
 count\_right += 1  
 average\_left = sum\_left / count\_left  
 average\_right = sum\_right / count\_right  
 return [average\_left, average\_right]  
  
 def delete(self) -> None:  
 self.\_\_start = None

Класс для тестирования:

import PySimpleGUI as PSG  
from Tree import Tree  
  
class App:  
 def \_\_init\_\_(self):  
 default\_tree = '12.12 0.987 24.01 -3.14 -99.99 0.5 13.21'  
 self.tree = Tree(default\_tree)  
 self.calculate\_average\_left = PSG.Text(font=("Times New Roman", 12), text\_color='white', background\_color='#000030')  
 self.calculate\_average\_right = PSG.Text(font=("Times New Roman", 12), text\_color='white', background\_color='#000030')  
 self.show\_hor = PSG.Text(str(self.tree), font=("Times New Roman", 12), text\_color='white', background\_color='#000030')  
 self.layout = [  
 [PSG.In(key='\_ELEMENTS\_', default\_text=default\_tree, font=("Times New Roman", 12), text\_color='black'), PSG.Button(key='\_CREATE\_', button\_text='Создать дерево', font=("Times New Roman", 12), button\_color=('white', 'darkblue'))],  
 [PSG.Text('Среднее арифметическое левого поддерева:', font=("Times New Roman", 12), text\_color='white', background\_color='#000030'), self.calculate\_average\_left],  
 [PSG.Text('Среднее арифметическое правого поддерева:', font=("Times New Roman", 12), text\_color='white', background\_color='#000030'), self.calculate\_average\_right],  
 [PSG.Button(key='\_DELETE\_', button\_text='Удалить дерево', font=("Times New Roman", 12), button\_color=('white', 'darkblue'))],  
 [PSG.Text('Дерево (повёрнутое справа налево):', font=("Times New Roman", 12), text\_color='white', background\_color='#000030'), self.show\_hor],  
 ]  
  
 def run(self):  
 app = PSG.Window('Идеальное сбалансированное дерево', self.layout, background\_color='#000030')  
 while True:  
 event, values = app.read()  
  
 if event == PSG.WIN\_CLOSED:  
 break  
 elif event == '\_CREATE\_':  
 self.tree = Tree(values['\_ELEMENTS\_'])  
 self.show\_hor.update(str(self.tree))  
 self.calculate\_average\_left.update(str(self.tree.calculate\_average(values['\_ELEMENTS\_'])[0]))  
 self.calculate\_average\_right.update(str(self.tree.calculate\_average(values['\_ELEMENTS\_'])[1]))  
 elif self.tree is not None:  
 if event == '\_DELETE\_':  
 self.tree.delete()  
 self.show\_hor.update(str(self.tree))  
 self.calculate\_average\_left.update('')  
 self.calculate\_average\_right.update('')  
  
application = App()  
application.run()

**Скриншоты результатов.**

Было запущено тестирование и создано дерево из элементов:

12.12, 0.987, 24.01, -3.14, -99.99, 0.5, 13.21

Результат:

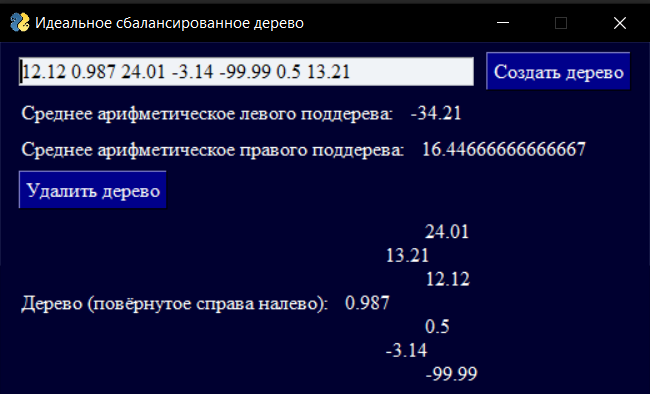


Рисунок 1 – Ввод элементов.

В результате работы программы было выведено дерево из данных элементов (повёрнутое справа налево) и среднее арифметическое для левого и правого поддеревьев.

Продолжая тестирование программы, удалим созданное дерево.

Результат:

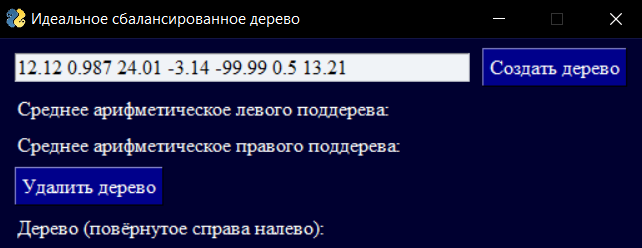


Рисунок 2 – Результаты тестирования.

# Ответы на вопросы

1. Что определяет степень дерева?

Степень дерева – максимальная степень его узлов.

Степень узла – число непосредственных потомков.

1. Какова степень сильноветвящегося дерева?

Степень сильноветвящегося дерева больше второй.

1. Что определяет путь в дереве?

Путь в дереве – последовательность узлов от корня к указанному узлу.

1. Как рассчитать длину пути в дереве?

Длина пути в дереве равно сумме длин всех его ребер.

1. Какова степень бинарного дерева?

Степень бинарного дерева не больше второй.

1. Может ли дерево быть пустым?

Дерево – совокупность узлов, один из которых корень, и отношений, образующих иерархическую структуру, поэтому дерево может быть:

* пустым;
* состоящим из одного узла, который является корнем своего поддерева;
* связанным с несколькими узлами – корнями деревьев (поддеревьев данного дерева).

1. Дайте определение бинарного дерева.

Бинарное (двоичное) дерево – дерево, степень которого не больше двух.

1. Дайте определение алгоритму обхода.

Обход дерева – алгоритм, обеспечивающий посещение каждого узла дерева с целью выполнить операцию с данными текущего узла.

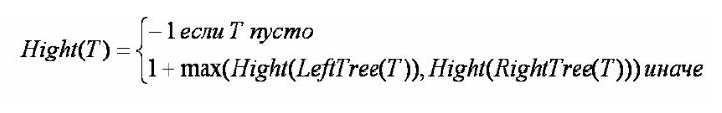
1. Приведите рекуррентную зависимость для вычисления высоты дерева:
2. Изобразите бинарное дерево, корень которого имеет индекс 6, и которое представлено в памяти таблицей вида:

Таблица 3

|  |  |  |  |
| --- | --- | --- | --- |
| *Индекс* | *key* | *left* | *right* |
| 1 | 12 | 7 | 3 |
| 2 | 15 | 8 | NULL |
| 3 | 4 | 10 | NULL |
| 4 | 10 | 5 | 9 |
| 5 | 2 | NULL | NULL |
| 6 | 18 | 1 | 4 |
| 7 | 7 | NULL | NULL |
| 8 | 14 | 6 | 2 |
| 9 | 21 | NULL | NULL |
| 10 | 5 | NULL | NULL |

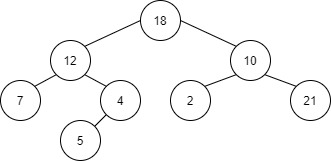


Рисунок 6 – Дерево по таблице 3

1. Укажите путь обхода дерева по алгоритму: прямой; обратный; симметричный.



Рисунок 7 – Дерево по 11 вопросу

Прямой: ABDGCEHIF.

Обратный: GDBHIEFCA.

Симметричный: GDBAHEICF.

1. Какая структура используется в алгоритме обхода дерева методом в «ширину»?

В алгоритме обхода дерева методом в «ширину» используется структура «очередь».

1. Выведите путь при обходе дерева в «ширину». Продемонстрируйте использование структуры при обходе дерева.



Рисунок 8 – Дерево по 13 вопросу

1. В очередь Q помещается первый узел с меткой A.

Q: A

1. Из очереди извлекается узел A для обработки и в очередь помещаются его сыновья.

Q: BC Вывод A

1. Из очереди извлекается узел Bдля обработки и в очередь помещаются его сыновья.

Q: C D Вывод B

1. Из очереди извлекается узел C для обработки и в очередь помещаются его сыновья.

Q:DEF Вывод C

1. Из очереди извлекается узел D для обработки и в очередь помещаются его сыновья.

Q: EFG Вывод D

1. Из очереди извлекается узел E для обработки и в очередь помещаются его сыновья.

Q: FG HI Вывод E

1. Из очереди извлекается узел F для обработки и в очередь должны помещаться его сыновья, но их нет, значит ничего не помещается.

Q: G HI Вывод F

1. Из очереди извлекается узел G для обработки и в очередь должны помещаться его сыновья, но их нет, значит ничего не помещается.

Q: HI Вывод G

1. Из очереди извлекается узел H для обработки и в очередь должны помещаться его сыновья, но их нет, значит ничего не помещается.

Q: I Вывод H

1. Из очереди извлекается узел I для обработки и в очередь должны помещаться его сыновья, но их нет, значит ничего не помещается.

Q: очередь пуста, дерево пройдено Вывод I

Путь обхода в «ширину»: ABCDEFGHI.

1. Какая структура используется в не рекурсивном обходе дерева методов в «глубину»?

Используется структура «стек».

1. Выполните прямой, симметричный, обратный методы обхода дерева выражений.



Рисунок 9 – Дерево по 15 вопросу

Прямой: – + a / \* b c d e.

Обратный: abc \* d / + e –.

Симметричный: a + b \* c / d – e.

1. Для каждого заданного арифметического выражения постройте бинарное дерево выражений:
2. a + b – c \* d + e

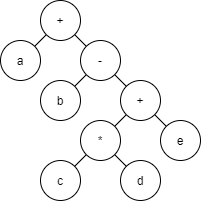


Рисунок 10 – Дерево по 16 вопросу. Выражение 1

1. / a – b \* c d

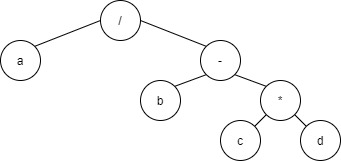


Рисунок 11 – Дерево по вопросу 16. Выражение 2

1. a b c d / – \*

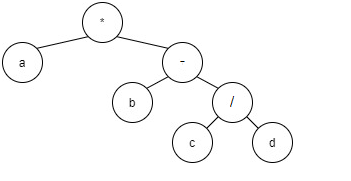


Рисунок 12 – Дерево по вопросу 16. Выражение 3

1. \* – / + a b c d e

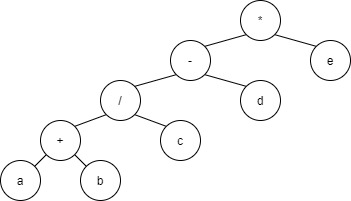


Рисунок 13 – Дерево по вопросу 16. Выражение 4

1. В каком порядке будет проходиться бинарное дерево, если алгоритм обхода в ширину будет запоминать узлы не в очереди, а стеке?

В таком случае будет использован метод обхода в «глубину».

1. Постройте бинарное дерево поиска, которое в результате симметричного обхода дало бы следующую последовательность узлов: 40 45 46 50 65 70 75

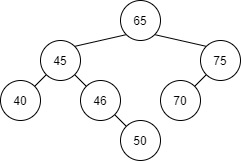


Рисунок 14 – Дерево по 18 вопросу

1. Приведенная ниже последовательность получена путем прямого обхода бинарного дерева поиска. Постройте это дерево.

50 45 35 15 40 46 65 75 70

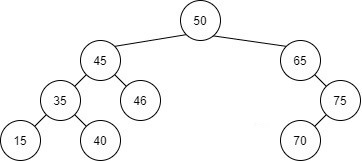


Рисунок 15 – Дерево по 19 вопросу

1. Дано бинарное дерево поиска, представленное на рисунке 21. Выполните действия **над исходным** дерево и покажите дерево:



Рисунок 16 – Исходное бинарное дерево поиска по 20 вопросу

1. После включения узлов 1, 48, 75, 100

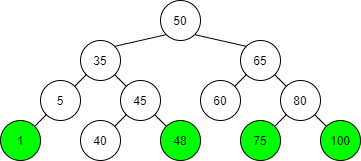


Рисунок 17 – Дерево поиска по 20 вопросу

1. После удаления узлов 5, 35

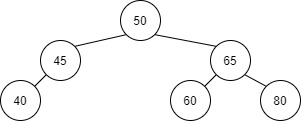


Рисунок 18 – Дерево поиска по 20 вопросу

1. После удаления узла 45

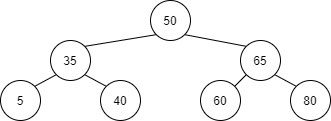


Рисунок 19 – Дерево поиска по 20 вопросу

1. После удаления узла 50

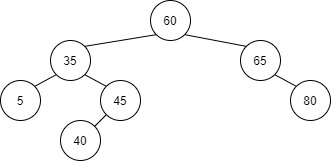


Рисунок 20 – Дерево поиска по 20 вопросу

1. После удаления узлов 5, 35

Ответ показан на рисунке 18

1. После удаления узла 65 и вставки его снова

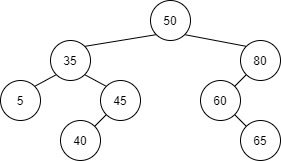


Рисунок 21 – Дерево поиска по 20 вопросу

1. После удаления узлов 5, 35

Ответ на рисунке 18

# Вывод

В ходе работы были приобретены умения и навыки разработки и реализации операций над структурой данных «бинарное дерево».

# Список информационных источников

1. Лекции по дисциплине «Структуры и алгоритмы обработки данных» /

Л. А. Скворцова, МИРЭА – Российский технологический университет, 2021.