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.

СОДЕРЖАНИЕ

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['_E
LEMENTS_'])[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:

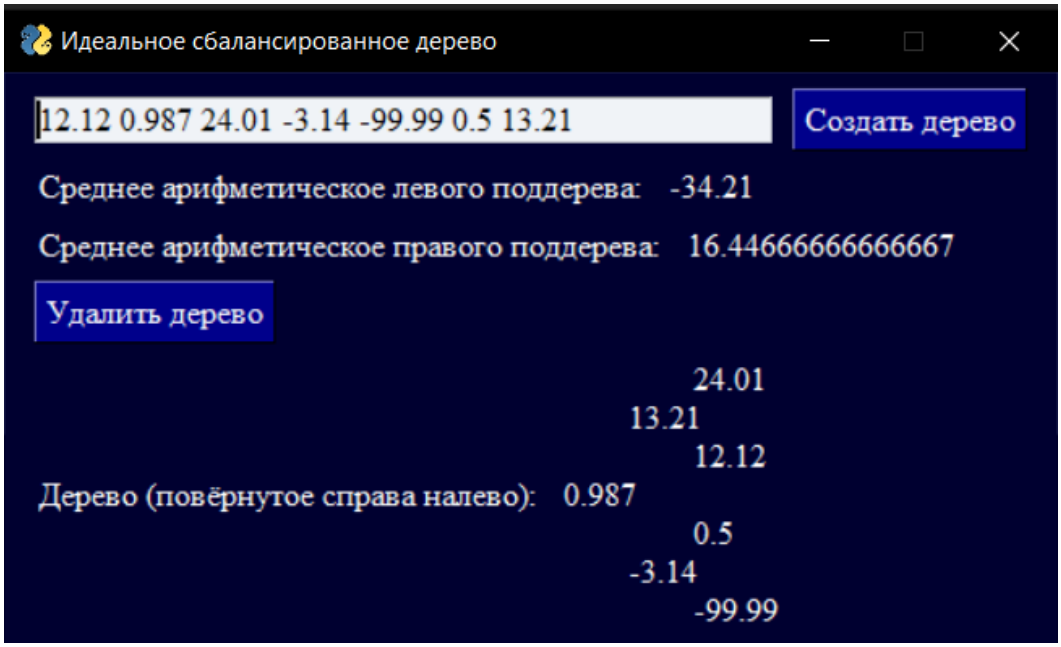

Рисунок 1 – Ввод элементов.

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:

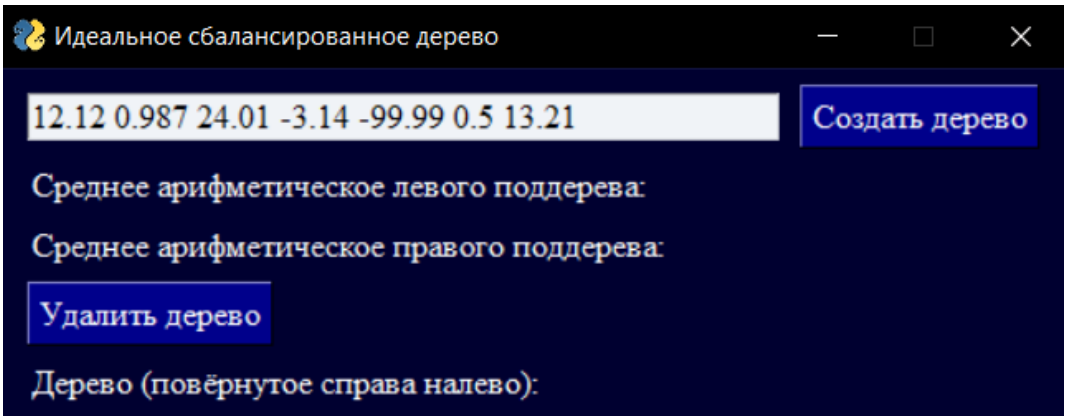

Рисунок 2 – Результаты тестирования.

Ответы на вопросы

1. Что определяет степень дерева?

Степень дерева – максимальная степень его узлов.

Степень узла – число непосредственных потомков.

2. Какова степень сильноветвящегося дерева?

Степень сильноветвящегося дерева больше второй.

3. Что определяет путь в дереве?

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

4. Как рассчитать длину пути в дереве?

Длина пути в дереве равно сумме длин всех его ребер.

5. Какова степень бинарного дерева?

Степень бинарного дерева не больше второй.

6. 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).

7. Дайте определение бинарного дерева.

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

8. Дайте определение алгоритму обхода.

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

9. Приведите рекуррентную зависимость для вычисления высоты дерева:

$$Hight(T) = \begin{cases} -1 \text{ если } T \text{ пусто} \\ 1 + \max(Hight(LeftTree(T)), Hight(RightTree(T))) \text{ иначе} \end{cases}$$

10. 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

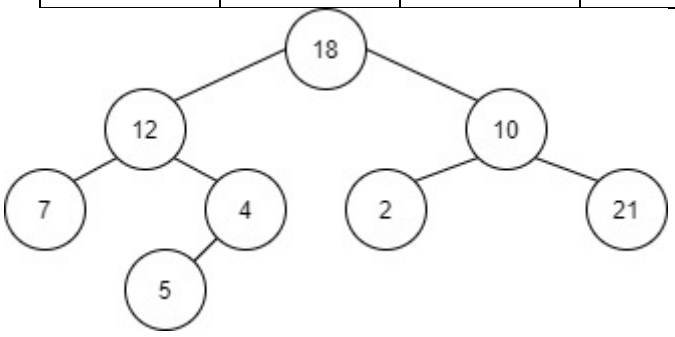

Рисунок 6 – Дерево по таблице 3

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

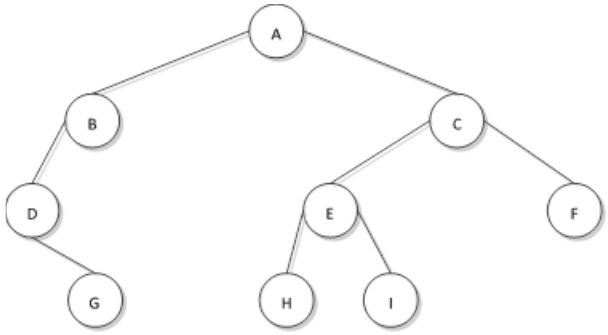

Рисунок 7 – Дерево по 11 вопросу

Прямой: ABDGCEHIF.

Обратный: GDBHIEFCA.

Симметричный: GDBAHEICF.

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

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

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

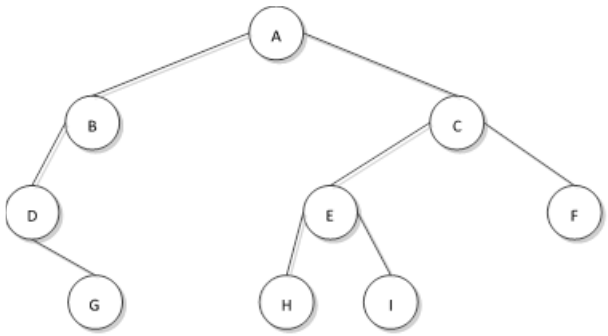

Рисунок 8 – Дерево по 13 вопросу

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

Q: A

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

Q: BC Вывод A

3) Из очереди извлекается узел Вдля обработки и в очередь помещаются его сыновья.

Q: C D Вывод B

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

Q:DEF Вывод C

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

Q: EFG Вывод D

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

Q: FG HI Вывод E

7) 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

8) 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

9) 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

10) 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.

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

Используется структура «стек».

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

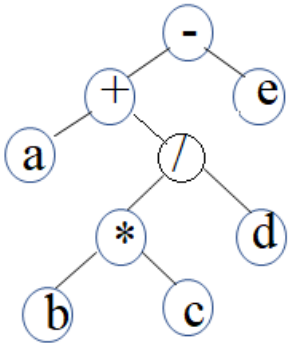

Рисунок 9 – Дерево по 15 вопросу

Прямой: – + a / * b c d e.

Обратный: abc * d / + e –.

Симметричный: a + b * c / d – e.

16. Для каждого заданного арифметического выражения постройте бинарное дерево выражений:

1) a + b – c * d + e

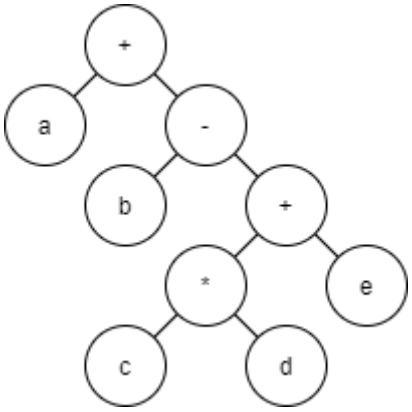

Рисунок 10 – Дерево по 16 вопросу. Выражение 1

2) / a – b * c d

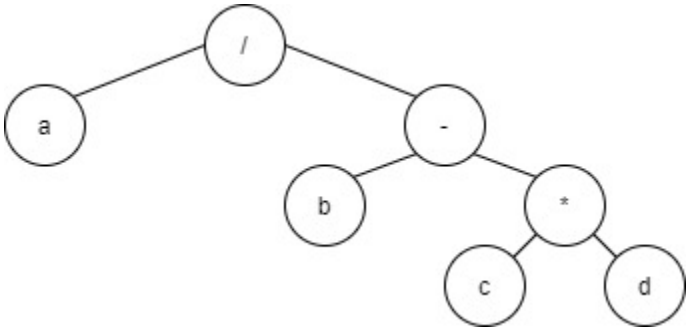

Рисунок 11 – Дерево по вопросу 16. Выражение 2

3) a b c d / – *

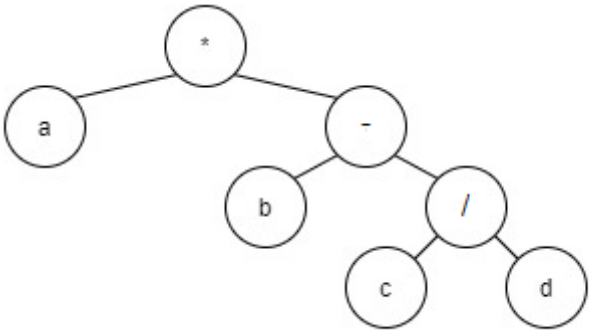

Рисунок 12 – Дерево по вопросу 16. Выражение 3

4) * – / + a b c d e

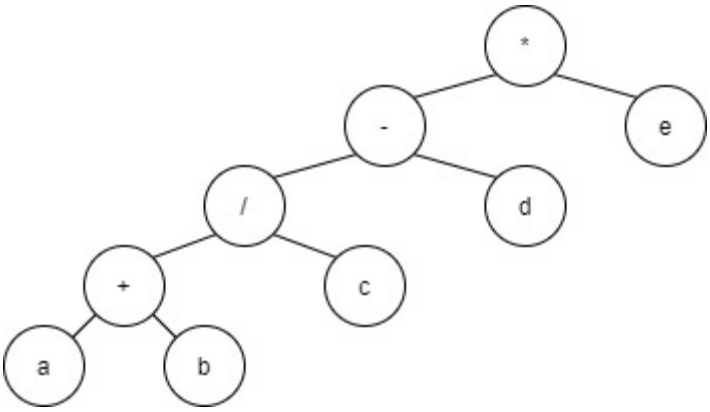

Рисунок 13 – Дерево по вопросу 16. Выражение 4

17. 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».

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

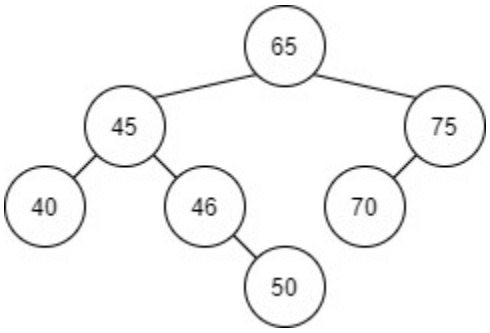

Рисунок 14 – Дерево по 18 вопросу

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

50 45 35 15 40 46 65 75 70

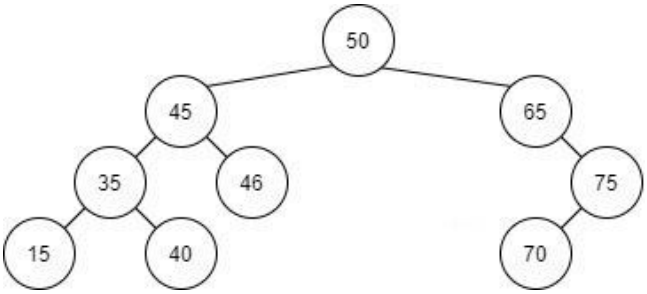

Рисунок 15 – Дерево по 19 вопросу

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

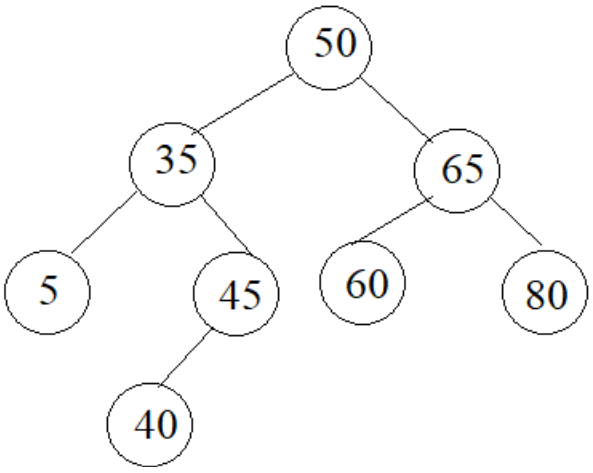

Рисунок 16 – Исходное бинарное дерево поиска по 20 вопросу

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

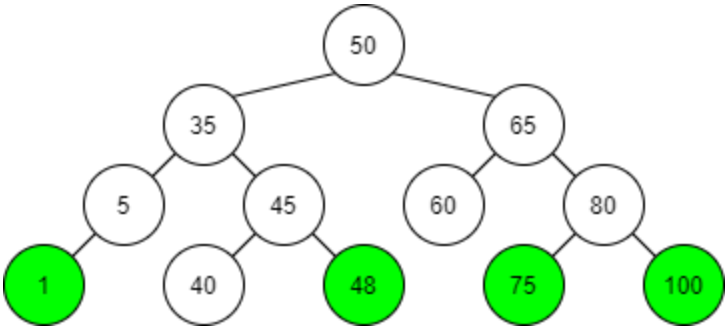

Рисунок 17 – Дерево поиска по 20 вопросу

2) После удаления узлов 5, 35

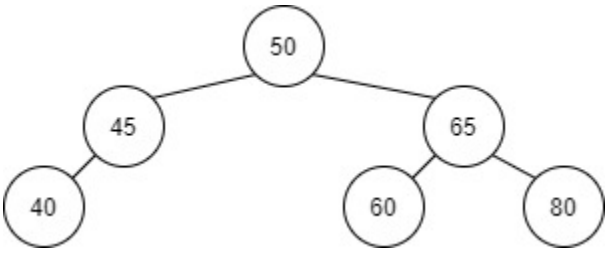

Рисунок 18 – Дерево поиска по 20 вопросу

3) После удаления узла 45

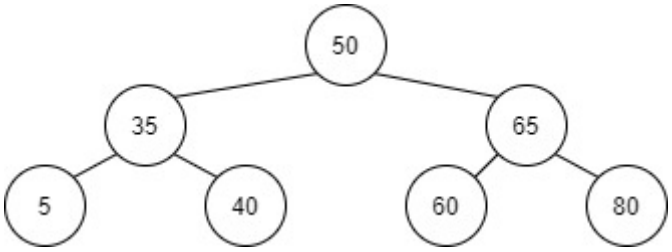

Рисунок 19 – Дерево поиска по 20 вопросу

4) После удаления узла 50

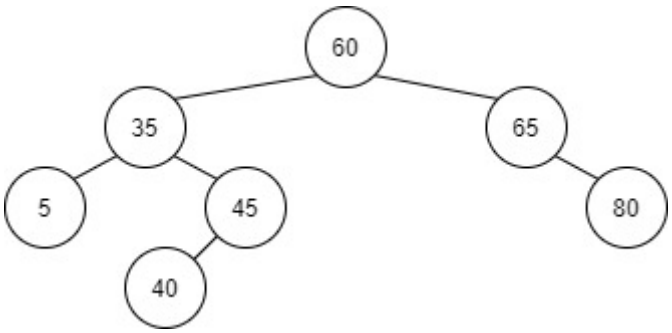

Рисунок 20 – Дерево поиска по 20 вопросу

5) После удаления узлов 5, 35

Ответ показан на рисунке 18

6) После удаления узла 65 и вставки его снова

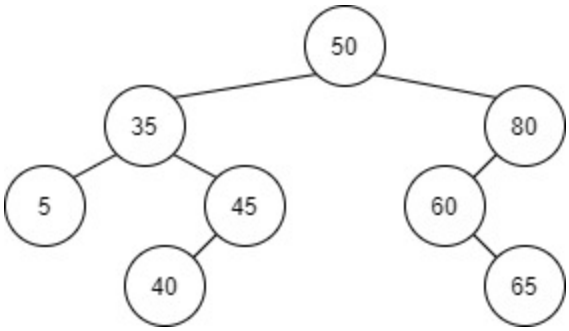

Рисунок 21 – Дерево поиска по 20 вопросу

7) 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.