УДК 004.43

Афанасьєва Д. С., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,
Вєтров О. С., старший викладач кафедри прикладної математики та кібербезпеки

РЕАЛІЗАЦІЯ БІНАРНОГО ДЕРЕВА МОВОЮ ПРОГРАМУВАННЯ PYTHON З ВИКОРИСТАННЯМ РЕКУРСІЇ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі програмування бінарні дерева є важливими структурами даних, які знаходять широке застосування у різних областях. Особливо важливим є їх використання у великих проєктах. Одним із варіантів побудови бінарного дерева є застосування рекурсії (принцип у програмуванні, коли функція у своїй реалізації викликає сама себе), яка дає змогу ефективно опрацьовувати та зберігати дані. Така реалізація матиме багато переваг, а саме: простота і зрозумілість, ефективність у використанні пам'яті та застосування у різних типах алгоритмів. У представленій роботі буде розглянуто реалізацію бінарного дерева мовою програмування Python із використанням підходу, що базується на рекурсії.

Бінарне дерево – певна абстрактна структура, яка складається з вершин (вузлів) та ребер, які пов'язують ці вершини між собою. Вузли дерева можуть мати лише по два нащадки кожен, а саме: лівого та правого [1]. Ребра ж поєднують між собою вершини-родичі.

Основні елементи бінарних дерев [2]:

1. Кореневий елемент (root) – перший вузол дерева, від якого відходять усі інші елементи.

2. Внутрішні вузли (internal nodes) – це ті вузли, які мають хоча б одного нащадка.

3. Листя (leaves) – вузли, які не мають жодного нащадка.

4. Лівий та правий нащадки – два варіанти нащадків певного вузла.

Бінарні дерева активно використовуються в алгоритмах пошуку даних, адже вони можуть бути побудовані так, що отримувати доступ до даних буде доволі швидко. Також їх застосування можливе в алгоритмах сортування, де використовується бінарне дерево пошуку, використання якого допомагає реалізовувати великі об'єми даних.

Розглянемо, як саме можна реалізувати бінарне дерево мовою програмування Python завдяки застосуванню рекурсії (рис. 1).

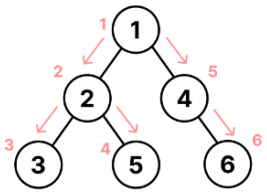

Рисунок 1. Приклад бінарного дерева для реалізації [3]

У цьому прикладі обхід бінарного дерева буде реалізованим за принципом прямого обходу, коли першим виводиться кореневий вузол, далі рекурсивний обхід лівого піддерева й рекурсивний обхід правого піддерева. Спочатку необхідно реалізувати клас Node, який представляє вузол бінарного дерева. У конструкторі __init__ вузол ініціалізується з ключем value та двома дочірніми вузлами (left_node та right_node), які на початку встановлені в None (рис. 2).

```
class Node:
    def __init__(self, value):
        self.left_node = None
        self.right_node = None
        self.data = value
```

Рисунок 2. Створення класу Node [4]

Додаємо функцію insert_node, що вставляє новий вузол зі значенням value в бінарне дерево. Якщо вузол node є порожнім, то створюється новий вузол зі значенням value. Якщо value менше, ніж значенню поточного вузла, то функція рекурсивно викликається для вузла лівого піддерева. У випадку, якщо value більше або рівне значенню поточного вузла, функція викликається для вузла правого піддерева. У будь-якому випадку функція повертає оновлений вузол node (рис. 3).

```
def insert_node(node, value):
    if node is None:
        return Node(value)
    if value < node.data:
        if node.left_node is None:
            node.left_node = Node(value)
        else:
            insert_node(node.left_node, value)
    else:
        if node.right_node is None:
            node.right_node = Node(value)
        else:
            insert_node(node.right_node, value)
    return node
```

Рисунок 3. Функція додавання вузлів до дерева [4]

Далі необхідно створити функцію для побудови бінарного дерева з використанням масиву вузлів nodes. Якщо список nodes порожній, функція повертає None. Інакше, вона створює кореневий вузол з першого значення списку, а потім викликає функцію insert_node для додавання всіх інших значень у дерево (рис. 4).

```
def build_binary_tree(nodes):
    if not nodes:
        return None
    root = Node(nodes[0])
    for val in nodes[1:]:
        insert_node(root, val)
    return root
```

Рисунок 4. Побудова бінарного дерева

Фінальним етапом є реалізація функції preorder_traversal для виведення значення кожного вузла в порядку прямого обходу. Вибудоване бінарне дерево створюється з допомогою списку nodes, а потім виконується прямий обхід від кореневого вузла tree_root, виводячи значення вузлів (рис. 5).

Після запуску програми отримуємо результат та перевіряємо правильність виконання програми (рис. 6).

```
def preorder_traversal(node):
    if node:
        print(node.data, end=" ")
        preorder_traversal(node.left_node)
        preorder_traversal(node.right_node)


nodes = [1, 2, 4, 5, 3, 6]
tree_root = build_binary_tree(nodes)
print("Прямий обхід дерева:")
preorder_traversal(tree_root)
```

Рисунок 5. Прямий обхід дерева та виведення результатів [4]

```
Прямий обхід дерева:
1 2 4 3 5 6
Process finished with exit code 0
```

Рисунок 6. Результат виконання програми

Отже, було розглянуто питання реалізації бінарного дерева мовою програмування Python з використанням рекурсивного підходу. Така реалізація дасть змогу легко створювати та опрацьовувати бінарні дерева. Використання рекурсивних методів спрощує реалізацію операцій над деревом та надає чітку

структуру для подальших досліджень у галузі алгоритмів обробки даних для широкого застосування у різних областях інформатики та програмування.

Список використаних джерел

1. Binary Tree. URL: https://www.programiz.com/dsa/binary-tree
2. Binary Trees. URL: https://www.andrew.cmu.edu/course/15-121/lectures/Trees/trees.html
3. More recursion examples. URL: https://programming-23.mooc.fi/part-11/4-more-recursion-examples
4. Binary Tree Implementation and Visualization in Python. URL: https://levelup.gitconnected.com/binary-tree-implementation-and-visualization-in-python-2f4782887ca2

УДК 517.2

Афанасьєва Д. С., здобувачка 2 курсу спеціальності 122 Комп'ютерні науки,
Фриз І. В., канд. фіз.-мат. наук, старший викладач кафедри інформаційних технологій

ЗАСТОСУВАННЯ ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ ТА ЇХ СИСТЕМ У СФЕРІ КОМП'ЮТЕРНИХ НАУК

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному інформаційному суспільстві використання диференціальних рівнянь та їх систем у різноманітних сферах стає все більш актуальним та розповсюдженим. Диференціальні рівняння виступають не лише як математична теорія, але й як потужний інструмент для аналізу, моделювання та розв'язування різноманітних задач.

У процесі розв'язування багатьох задач із різних предметних областей часто виникає ситуація, коли важко чи навіть неможливо встановити функціональну залежність між шуканими величинами та вхідними змінними. Проте може існувати інший тип зв'язку, який пов'язує незалежні змінні, функції та їх похідні, тобто приходимо до диференціальних рівнянь. Нагадаємо, що диференціальним рівнянням n-го порядку називається рівняння виду:

$$F\left(x, y, y', y'', \ldots y^{(n)}\right) = 0,$$

де x, y – змінні,

$y', y'', y^{(n)}$ – похідні відповідних порядків [1].