## Протокол

## Лабораторна робота №5. Варіант 14 (2)

### Завдання

Побудувати двійкове дерево пошуку, в вершинах якого знаходяться слова з текстового файлу. Вивести його на екран у вигляді дерева. Визначити кількість вершин дерева, що містять слова, які починаються на зазначену букву. Видалити з дерева ці вершини.

### Реалізація

class Node:

    def \_\_init\_\_(self, key):

        self.left = None

        self.right = None

        self.val = key

        self.count = 1

Ця функція ініціалізує новий вузол для двійкового дерева пошуку. Кожен вузол має значення ключа, лічильник входжень цього ключа, і посилання на ліве та праве піддерева.

def insert(node, key):

    if node is None:

        return Node(key)

    if key == node.val:

        node.count += 1

    elif key < node.val:

        node.left = insert(node.left, key)

    else:

        node.right = insert(node.right, key)

    return node

Функція вставляє ключ у двійкове дерево пошуку. Якщо місце вставки порожнє, створюється новий вузол. Якщо ключ вже існує, збільшується лічильник цього ключа. В іншому випадку ключ додається у відповідне піддерево залежно від його значення.

def delete(node, key):

    if node is None:

        return node

    if key < node.val:

        node.left = delete(node.left, key)

    elif key > node.val:

        node.right = delete(node.right, key)

    else:

        if node.left is None:

            return node.right

        if node.right is None:

            return node.left

        temp = min\_node(node.right)

        node.val = temp.val

        node.right = delete(node.right, temp.val)

    return node

Функція видаляє вузол з двійкового дерева пошуку за вказаним ключем. Вона рекурсивно шукає вузол, який потрібно видалити, і забезпечує правильне перепідключення дітей вузла для збереження структури дерева.

def min\_node(node):

    while node.left:

        node = node.left

    return node

Ця функція знаходить вузол з мінімальним ключем у дереві. Використовується при видаленні вузла, що має двох дітей, для знаходження наступника, який займе місце видаленого вузла.

def print\_tree(node, indent="", pos="root", last=True):

    if node:

        line\_prefix = indent + ("└── " if last else "├── ")

        print(f"{line\_prefix}{node.val}({node.count})")

        indent\_next = indent + ("    " if last else "|   ")

        print\_tree(node.left, indent\_next, 'left', node.right is None)

        print\_tree(node.right, indent\_next, 'right', True)

Функція виводить на екран структуру двійкового дерева пошуку. Вона рекурсивно проходить дерево і виводить кожен вузол у вигляді візуально впорядкованого дерева, що полегшує його сприйняття.

def count\_and\_build(prefix, node):

    subtree = None

    count = 0

    def traverse(n):

        nonlocal subtree, count

        if n:

            if n.val.lower().startswith(prefix.lower()):

                count += n.count

                subtree = insert(subtree, n.val)

            traverse(n.left)

            traverse(n.right)

    traverse(node)

    return count, subtree

Ця функція рахує вузли та будує піддерево зі словами, які починаються на задану букву або префікс. Вона використовується для створення фільтрованої копії дерева та підрахунку кількості таких вузлів.

def remove\_prefix(node, prefix):

    def traverse(n):

        if n:

            n.left = traverse(n.left)

            n.right = traverse(n.right)

            if n.val.lower().startswith(prefix.lower()):

                return delete(n, n.val)

        return n

    return traverse(node)

Функція видаляє з дерева всі вузли, значення яких починаються на певну букву або префікс. Вона рекурсивно проходить дерево, видаляючи відповідні вузли.

def build\_bst(file\_path):

    root = None

    with open(file\_path, 'r') as file:

        for line in file:

            for word in line.split():

                root = insert(root, word)

    return root

Функція читає слова з текстового файлу та побудовує двійкове дерево пошуку. Це основна функція для створення дерева з вхідних даних.

### Результат





