Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

по дисциплине «Логика и основы алгоритмизации в инженерных задачах»

по лабораторной работе №4

«Бинарное дерево поиска»

Выполнили:

студенты группы 22ВВС1

Эрмедов А.Э.

Казаров И.И.

Приняли:

к.т.н, доцент Юрова О.В.

к.э.н, доцент Акифьев И.В.

Пенза 2023

**Название**

Оценка времени выполнения программ

**Практическая часть**

Задание:

1. Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).
2. \* На основе приведенного кода реализуйте структуру данных *Очередь*.
3. \* На основе приведенного кода реализуйте структуру данных *Стек*.

**Листинг SLZ40.py**

import matplotlib.pyplot as plt

class Node:

def \_\_init\_\_(self, value):

self.value = value

self.left = None

self.right = None

def insert\_node(root, value):

if root is None:

return Node(value)

if value < root.value:

root.left = insert\_node(root.left, value)

elif value > root.value:

root.right = insert\_node(root.right, value)

else:

root.right = insert\_node(root.right, value) # Вставка справа для одинаковых значений

return root

def visualize\_tree(root, x, y, ax):

if root is None:

return

ax.text(x, y, str(root.value), fontsize=12, fontweight='bold', ha='center', va='center')

if root.left:

ax.plot([x, x-1], [y-1, y+1], color='black') # Левая связь

visualize\_tree(root.left, x-1, y+1, ax)

if root.right:

ax.plot([x, x+1], [y-1, y+1], color='black') # Правая связь

visualize\_tree(root.right, x+1, y+1, ax)

def search\_value(root, value):

if root is None or root.value == value:

return root

if value < root.value:

return search\_value(root.left, value)

else:

return search\_value(root.right, value)

# Создание дерева

root\_value = int(input("Введите значение корневого узла: "))

root = Node(root\_value)

more\_values = input("Введите значения узлов через пробел: ").split()

for value in more\_values:

root = insert\_node(root, int(value))

# Визуализация дерева

fig, ax = plt.subplots(figsize=(8, 6))

visualize\_tree(root, 0, 0, ax)

ax.set\_xlim(-5, 5)

ax.set\_ylim(0, 10)

ax.axis('off')

plt.title('Binary Search Tree')

plt.show()

# Поиск значения в дереве

search\_value\_input = int(input("Введите значение для поиска: "))

result = search\_value(root, search\_value\_input)

if result:

print("Значение найдено в дереве!")

else:

print("Значение не найдено в дереве.")

**Результат**

****

****

**Листинг SLZ41.py**

import matplotlib.pyplot as plt

class Node:

def \_\_init\_\_(self, value):

self.value = value

self.left = None

self.right = None

def insert\_node(root, value):

if root is None:

return Node(value)

if value < root.value:

root.left = insert\_node(root.left, value)

elif value > root.value:

root.right = insert\_node(root.right, value)

# Ignore duplicate values

return root

def visualize\_tree(root, x, y, ax):

if root is None:

return

ax.text(x, y, str(root.value), fontsize=12, fontweight='bold', ha='center', va='center')

if root.left:

ax.plot([x, x-1], [y-1, y+1], color='black') # Левая связь

visualize\_tree(root.left, x-1, y+1, ax)

if root.right:

ax.plot([x, x+1], [y-1, y+1], color='black') # Правая связь

visualize\_tree(root.right, x+1, y+1, ax)

def search\_value(root, value):

if root is None or root.value == value:

return root

if value < root.value:

return search\_value(root.left, value)

else:

return search\_value(root.right, value)

# Создание дерева

root\_value = int(input("Введите значение корневого узла: "))

root = Node(root\_value)

more\_values = input("Введите значения узлов через пробел: ").split()

for value in more\_values:

root = insert\_node(root, int(value))

# Визуализация дерева

fig, ax = plt.subplots(figsize=(8, 6))

visualize\_tree(root, 0, 0, ax)

ax.set\_xlim(-5, 5)

ax.set\_ylim(0, 10)

ax.axis('off')

plt.title('Binary Search Tree')

plt.show()

# Поиск значения в дереве

search\_value\_input = int(input("Введите значение для поиска: "))

result = search\_value(root, search\_value\_input)

if result:

print("Значение найдено в дереве!")

else:

print("Значение не найдено в дереве.")

**Результат**





**Вывод**

Задание:

1. Реализовали алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.
2. Реализовали функцию подсчёта числа вхождений заданного элемента в дерево.

3. \* Изменили функцию добавления элементов для исключения добавления одинаковых символов.

1. \* Оценили сложность процедуры поиска по значению в бинарном дереве.