**МІНІСТЕРСТВО ОСВІТИ І НАУКИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ.ІГОРЯ СІКОРСЬКОГО»**

**НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ**

Лабораторна робота №5

«Дерева»

Варіант 2

Виконав:  
                                                                     Студент 2 курсу  
                                                                     Групи ФІ-21  
                                                                     Голуб Михайло

Перевірив:  
Лавренюк А. М.

Київ 2024

# ЗАВДАННЯ

Побудувати двійкове дерево пошуку з цілих чисел, що вводяться. Вивести його на екран у вигляді дерева. Знайти вершину, яка містить задане число. Визначити максимальний елемент в цьому дереві.

# ХІД РОБОТИ

Створимо клас Node, який має методи рекурсивні методи depth, append та find:

class Node:  
 def \_\_init\_\_(self, value, parent=None, right\_arm=None, left\_arm=None):  
 self.value = value  
 self.parent = parent  
 self.right\_arm = right\_arm  
 self.left\_arm = left\_arm  
  
 def \_\_str\_\_(self):  
 return str(self.value)  
  
 def depth(self):  
 if self.parent is None:  
 return 0  
 else:  
 return self.parent.depth() + 1  
  
 def append(self, value):  
 if self.value >= value:  
 if self.left\_arm is None:  
 node = Node(value, self)  
 self.left\_arm = node  
 return  
 else:  
 self.left\_arm.append(value)  
 else:  
 if self.right\_arm is None:  
 node = Node(value, self)  
 self.right\_arm = node  
 return  
 else:  
 self.right\_arm.append(value)  
 return  
  
 def find(self, key):  
 if key < self.value:  
 if self.left\_arm is None:  
 return 0  
 else:  
 return self.left\_arm.find(key)  
 elif key > self.value:  
 if self.right\_arm is None:  
 return 0  
 else:  
 return self.right\_arm.find(key)  
 else:  
 return self

Створимо клас Tree, який має методи find, draw, append, find\_max:

class Tree:  
 def \_\_init\_\_(self, root):  
 if isinstance(root, Node):  
 self.root = root  
 else:  
 self.root = Node(root)  
  
 def find(self, key):  
 return self.root.find(key)  
  
 def draw(self, x\_spacing=25, y\_spacing=30):  
 turtle.speed(0)  
 previous\_level = []  
 level = [self.root]  
 level\_counter = 0  
 while True:  
 *# draw level* turtle.penup()  
 turtle.pencolor("black")  
 turtle.goto(-x\_spacing \* (len(level) - 1) / 2, -level\_counter \* y\_spacing)  
 for node in level:  
 if not node is None:  
 turtle.write(node.value, align="center", font=("Verdana", 15, "normal"))  
 turtle.setheading(0)  
 turtle.forward(x\_spacing)  
  
 *# draw connections* for node in level:  
 if not node is None:  
 if not node.parent is None:  
 turtle.penup()  
 turtle.goto(-x\_spacing \* (len(level) - 1) / 2 + x\_spacing \* level.index(node),  
 -level\_counter \* y\_spacing)  
 turtle.pendown()  
 if node.parent.left\_arm == node:  
 turtle.pencolor("red")  
 else:  
 turtle.pencolor("black")  
 turtle.goto(  
 -x\_spacing \* (len(previous\_level) - 1) / 2 + x\_spacing \* previous\_level.index(node.parent),  
 -level\_counter \* y\_spacing + y\_spacing)  
  
 *# iterate* level\_counter += 1  
 nodes\_exist = False  
 new\_level = []  
 for node in level:  
 if not node is None:  
 new\_level.append(node.left\_arm)  
 new\_level.append(node.right\_arm)  
 nodes\_exist = True  
 else:  
 new\_level.append(None)  
 new\_level.append(None)  
 previous\_level = copy.copy(level)  
 level = copy.copy(new\_level)  
 if not nodes\_exist:  
 return 0  
  
 def append(self, value):  
 self.root.append(value)  
  
 def find\_max(self):  
 nodes = [self.root]  
 values = []  
 nodes\_detected = True  
 while nodes\_detected:  
 new\_nodes = []  
 nodes\_detected = False  
 for node in nodes:  
 if not node is None:  
 nodes\_detected = True  
 values.append(node.value)  
 new\_nodes.append(node.right\_arm)  
 new\_nodes.append(node.left\_arm)  
 nodes = copy.copy(new\_nodes)  
 maximum = max(values)  
 print(maximum)  
 return self.find(maximum)

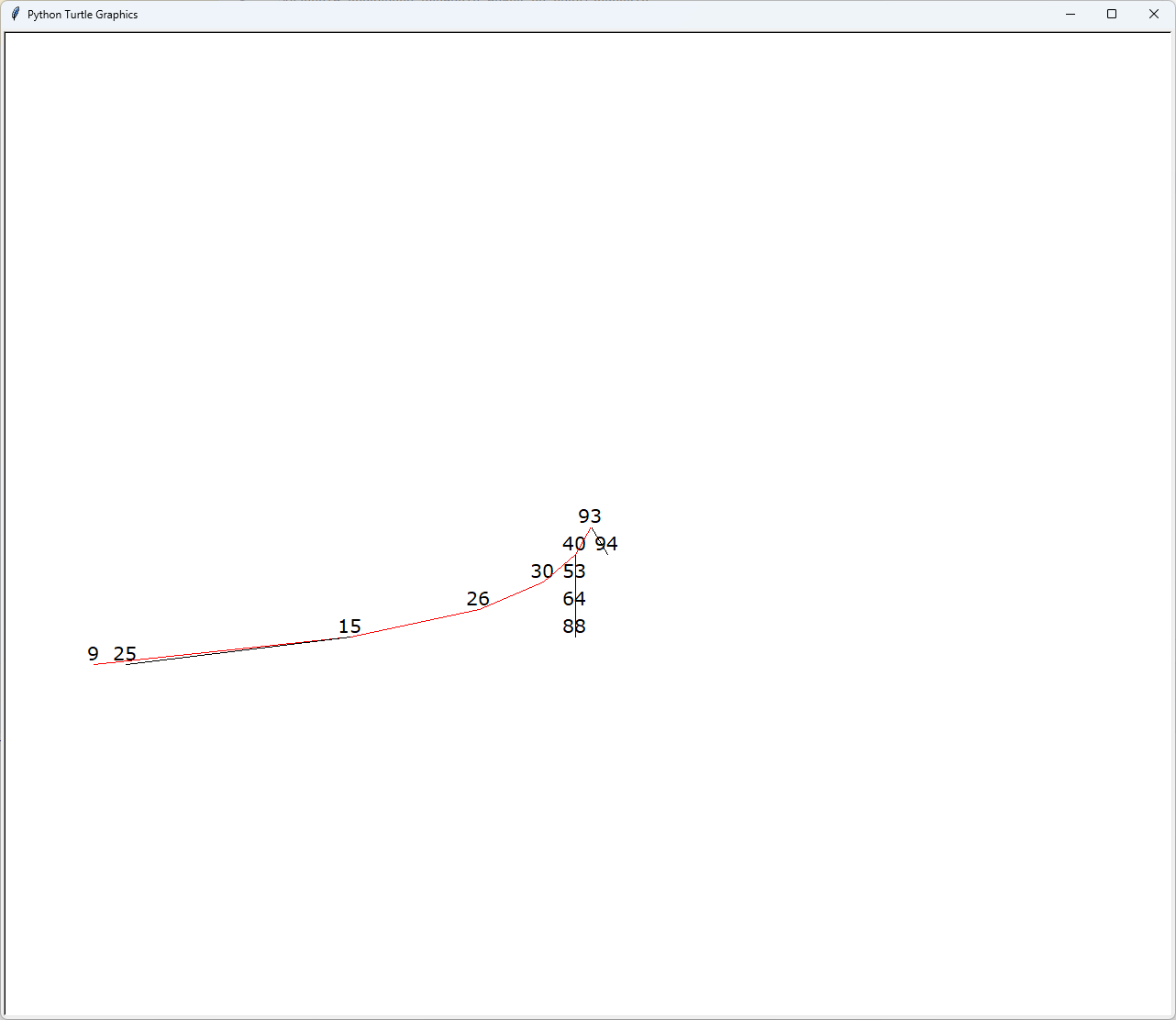
Створимо функцію tree\_random, яка випадковим чином генерує дерево із заданою кількістю вершин:

def random\_tree(nodes\_n, min=0, max=99):  
 width = max - min  
 tree = Tree(int(random() \* width + min))  
 for n in range(nodes\_n):  
 value = int(random() \* width + min)  
 tree.append(value)  
 return tree

Отже, створюємо випадкове дерево з 10 вершин та виводимо значення його найбільшого елемента:

test = random\_tree(10)  
print(test.find\_max())  
test.draw(35)  
input()

Приклад відображення дерева (ліві зв’язки червоним, праві зв’язки чорним):



Результат пошуку максимального елемента в дереві: 94