import tkinter as tk

from tkinter import messagebox

import time

import unittest

import git

"""

Программа по сортировке пользовательских данных

"""

# Функция сортировки выбором

def selection\_sort(arr):

"""

Функция сортировки выбором

"""

for i in range(len(arr)):

min\_idx = i

for j in range(i+1, len(arr)):

if arr[j] < arr[min\_idx]:

min\_idx = j

arr[i], arr[min\_idx] = arr[min\_idx], arr[i]

return arr

# Функция сортировки пузырьком

def bubble\_sort(arr):

"""

Функция сортировки пузырьком

"""

n = len(arr)

for i in range(n-1):

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

return arr

# Функция сортировки вставками

def insertion\_sort(arr):

"""

Функция сортировки вставками

"""

for i in range(1, len(arr)):

key = arr[i]

j = i-1

while j >= 0 and key < arr[j]:

arr[j+1] = arr[j]

j -= 1

arr[j+1] = key

return arr

# Функция сортировки слиянием

def merge\_sort(arr):

"""

Функция сортировки слиянием

"""

if len(arr) > 1:

mid = len(arr)//2

L = arr[:mid]

R = arr[mid:]

merge\_sort(L)

merge\_sort(R)

i = j = k = 0

while i < len(L) and j < len(R):

if L[i] < R[j]:

arr[k] = L[i]

i += 1

else:

arr[k] = R[j]

j += 1

k += 1

while i < len(L):

arr[k] = L[i]

i += 1

k += 1

while j < len(R):

arr[k] = R[j]

j += 1

k += 1

return arr

# Функция быстрой сортировки

def quicksort(arr):

"""

Функция быстрой сортировки

"""

if len(arr) <= 1:

return arr

else:

pivot = arr[0]

less = [x for x in arr[1:] if x <= pivot]

greater = [x for x in arr[1:] if x > pivot]

return quicksort(less) + [pivot] + quicksort(greater)

# Функция очистки полей вывода

def clear\_output():

"""

Функция очистки полей вывода

"""

output\_label.config(text="")

time\_label.config(text="")

git\_label.config(text="")

# Функция начала сортировки

def start\_sorting():

"""

Функция начала сортировки

"""

input\_text = input\_sequence.get()

# Проверка ввода чисел через запятую

if not validate\_input(input\_text):

messagebox.showerror("Ошибка", "Введите числа через запятую")

return

# Выбор алгоритма сортировки

sort\_type = selected\_sort.get()

try:

sequence = [float(x) for x in input\_text.split(',')]

sorted\_sequence, sorted\_time = sort\_sequence(sequence, sort\_type)

output\_text.delete("1.0", tk.END) # Очистка поля вывода

output\_text.insert(tk.END, "Результат сортировки: {}\n".format(str(sorted\_sequence)))

output\_text.insert(tk.END, "Время выполнения сортировки: {:.8f} секунд".format(sorted\_time))

input\_sequence.delete(0, tk.END) # Очистка окна ввода

save\_results(sorted\_sequence, sort\_type, sorted\_time) # Сохранение результатов сортировки в файл

except ValueError:

messagebox.showerror("Ошибка", "Введена некорректная последовательность чисел")

# Функция сортировки выбором

def sort\_sequence(sequence, sort\_type):

"""

Функция сортировки выбором

"""

time.perf\_counter()

start\_time = time.perf\_counter()

if sort\_type == "Сортировка выбором":

sorted\_sequence = selection\_sort(sequence)

elif sort\_type == "Сортировка пузырьком":

sorted\_sequence = bubble\_sort(sequence)

elif sort\_type == "Сортировка вставками":

sorted\_sequence = insertion\_sort(sequence)

elif sort\_type == "Сортировка слиянием":

sorted\_sequence = merge\_sort(sequence)

elif sort\_type == "Быстрая сортировка quicksort":

sorted\_sequence = quicksort(sequence)

else:

messagebox.showerror("Ошибка", "Выбран неправильный тип сортировки")

return

end\_time = time.perf\_counter()

sorted\_time = end\_time - start\_time

return sorted\_sequence, sorted\_time

# Функция для отображения информации о текущей версии Git

def show\_git\_info():

"""

Функция отображения информации о текущей версии

"""

try:

repo = git.Repo(search\_parent\_directories=True)

git\_info = f"Текущая ветка: {repo.active\_branch}\n" \

f"Последний коммит: {repo.head.commit}\n" \

f"Автор последнего коммита: {repo.head.commit.author}\n" \

f"Дата последнего коммита: {repo.head.commit.committed\_datetime}"

git\_label.config(text=git\_info)

except git.InvalidGitRepositoryError:

messagebox.showerror("Ошибка", "Не найден репозиторий Git")

# Функция проверки ввода цифр через запятую

def validate\_input(input\_str):

"""

Функция проверки ввода цифр через запяту

"""

nums = input\_str.split(",")

for num in nums:

try:

float(num)

except ValueError:

return False

return True

# Функция для сохранения результатов сортировки в файл

def save\_results(sorted\_sequence, sort\_type, sorted\_time):

"""

Функция для сохранения результатов сортировки в файл

"""

try:

with open("sorting\_results.txt", "a") as file:

file.write(f"Метод сортировки: {sort\_type}\n")

file.write(f"Результат сортировки: {str(sorted\_sequence)}\n")

file.write(f"Время выполнения сортировки: {sorted\_time} секунд\n")

file.write("\n")

messagebox.showinfo("Сохранение", "Результаты сортировки сохранены в файле sorting\_results.txt")

except IOError:

messagebox.showerror("Ошибка", "Не удалось сохранить результаты сортировки в файл")

# Функция проверки кода с помощью модуля unittest

class TestSortingFunctions(unittest.TestCase):

"""

Функция проверки кода с помощью модуля unittest

"""

def test\_selection\_sort(self):

self.assertEqual(selection\_sort([3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]), [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9])

def test\_bubble\_sort(self):

self.assertEqual(bubble\_sort([3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]), [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9])

def test\_insertion\_sort(self):

self.assertEqual(insertion\_sort([3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]), [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9])

def test\_merge\_sort(self):

self.assertEqual(merge\_sort([3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]), [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9])

def test\_quicksort(self):

self.assertEqual(quicksort([3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]), [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9])

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main(argv=[''], exit=False)

# Создание основного окна

root = tk.Tk()

root.title("Сортировка данных")

# Создание и размещение элементов интерфейса

#Создание метки описания поля ввода

input\_label = tk.Label(root, text="Введите последовательность чисел через запятую:")

input\_label.pack()

#Создание поле ввода

input\_sequence = tk.Entry(root, width=80)

input\_sequence.pack()

#Создание метки описания поля метода сортировки

sort\_label = tk.Label(root, text="Выберите тип сортировки:")

sort\_label.pack()

#Создание раскрывающегося списка с методами сортировки

sort\_options = ["Сортировка выбором", "Сортировка пузырьком", "Сортировка вставками", "Сортировка слиянием", "Быстрая сортировка quicksort"]

selected\_sort = tk.StringVar(root)

selected\_sort.set(sort\_options[0])

sort\_menu = tk.OptionMenu(root, selected\_sort, \*sort\_options)

sort\_menu.pack()

# Создание кнопки для запуска сортировки

sort\_button = tk.Button(root, text="Начать сортировку", command=start\_sorting)

sort\_button.pack()

output\_text = tk.Text(root, height=5, width=60)

output\_text.pack()

time\_label = tk.Label(root, text="")

time\_label.pack()

# Создание кнопки и окна вывода для отслеживания контроля версий Git

git\_button = tk.Button(root, text="Показать информацию Git", command=show\_git\_info)

git\_button.pack()

git\_label = tk.Label(root, text="")

git\_label.pack()

# Запуск основного цикла приложения

root.mainloop()