Национальный исследовательский университет «Высшая школа экономики» Специальность «Компьютерная безопасность»

Отчет
По лабораторной работе №1
«Сортировки»

по дисциплине «Методы программирования»
направления «Компьютерная безопасность»

Вариант 9

Сааков А. СКБ-182

Москва, 2021

Общее задание:

1) Реализовать сортировки для массива объектов в соответствии с вариантом.
2) Перегрузить операторы сравнения (>, <, >=, <=) для сравнения объектов.
3) Входные данные для сортировки массива обязательно считывать из внешних источников: текстовый файл, файл MS Excel, MS Access, данные из СУБД (любое на выбор).
4) Выбрать 7-10 наборов данных для сортировки размерности от 100 и более (но не менее 100000). Засечь (программно) время сортировки каждым алгоритмом. По полученным точкам построить графики зависимости времени сортировки от размерности массива для каждого из алгоритмов сортировки на одной оси координат. Сделать вывод о том, в каком случае, какой из методов лучше применять. Графики можно строить программно или в любой из прикладных программ (MS Excel, Matlab, MathCad и т.д.).

Вариант 9:
Массив данных об экспортируемых товарах:
наименование товара, страна (куда экспортируется товар), объем поставляемой продукции, сумма в рублях (сравнение по полям – наименование, объем, страна)

Сортировки: Сортировка пузырьком, шейкер-сортировка, сортировка слиянием.

Программа состоит из четырех файлов: «main.py», «product_class.py», «gendata.py», «sorts.py»

«main.py»:

```
import copy
from gendata import generator
from sorts import bubble_sort, cocktail_sort, merge_sort
from datetime import datetime
from product_class import Product


data_size = [100, 200, 500, 1000, 5000, 10000, 50000, 100000]

for size in data_size:
    data = []
    generator(size)  # создаем файл с данными
    with open(f'./data_{size}.txt', mode='r', encoding='UTF-8') as database:
        for one_data in database:
            data.append(Product(one_data.strip()))  # переносим данные из
файла в массив

    data_bubble = copy.deepcopy(data)
    data_cocktail = copy.deepcopy(data)
    data_merge = copy.deepcopy(data)

    start1 = datetime.now()
    bubble_sort(data_bubble)  # сортировка пузырьком
    end1 = datetime.now()

    start2 = datetime.now()
    cocktail_sort(data_cocktail)  # шейкер-сортировка
    end2 = datetime.now()

    start3 = datetime.now()
    merge_sort(data_merge)  # сортировка слиянием
    end3 = datetime.now()

    print(str(size) + 'bubble sort:' + str(end1 - start1) + '\n')
    print(str(size) + 'cocktail sort:' + str(end2 - start2) + '\n')
    print(str(size) + 'merge sort:' + str(end3 - start3) + '\n')

    with open('./time.txt', mode='a', encoding='UTF-8') as result:
        result.write(f"size: {size} bubble sort: {str(end1 - start1)}" + '\
n')
        result.write(f"size: {size} cocktail sort: {str(end2 - start2)}" + '\
n')
        result.write(f"size: {size} merge sort: {str(end3 - start3)}" + '\n')
```

«product_class.py»:

```
class Product:  # класс сотрудник
    name: str
    country: str
    volume: str
    price: str

    def __init__(self, line: str):  # инициализация
        name, country, volume, price = line.split(', ')
        self.name = name
        self.country = country
        self.volume = volume
        self.price = price

    def __gt__(self, other):  # перегрузка  '>'
        if self.name > other.name:
            return True
        elif self.name == other.name:
            if self.volume > other.volume:
                return True
            elif self.volume == other.volume:
                if self.country > other.country:
                    return True
                else:
                    return False
            else:
                return False
        else:
            return False

    def __lt__(self, other):  # перегрузка '<'
        if self.name < other.name:
            return True
        elif self.name == other.name:
            if self.volume < other.volume:
                return True
            elif self.volume == other.volume:
                if self.country < other.country:
                    return True
                else:
                    return False
            else:
                return False
        else:
            return False

    def __ge__(self, other):  # перегрузка '>='
        if self.name > other.name:
            return True
        elif self.name == other.name:
            if self.volume > other.volume:
                return True
            elif self.volume == other.volume:
                if self.country > other.country:
                    return True
                elif self.country == other.country:
                    return True
                else:
                    return False
            else:
                return False
        else:
```

```
            return False

    def __le__(self, other):  # перегрузка '<='
        if self.name < other.name:
            return True
        elif self.name == other.name:
            if self.volume < other.volume:
                return True
            elif self.volume == other.volume:
                if self.country < other.country:
                    return True
                elif self.country == other.country:
                    return True
                else:
                    return False
            else:
                return False
        else:
            return False
```

«gendata.py»:

```
import random


data_name = ['рыба', 'хлеб', 'масло', 'яблоки', 'груши', 'зерно', 'шоколад']
data_country = ['США', 'Китай', 'Германия', 'Австрия', 'Венгрия',
'Аргентина', 'Австралия', 'Перу']
data_volume = ['1 тонна', '2 тонны', '5 тонн', '10 тонн', '15 тонн', '20
тонн', '30 тонн', '50 тонн']
data_price = ['500000 рублей', '1000000 рублей', '1250000 рублей', '14850000
рублей', '12340000 рублей', '7050000 рублей']


def generator(size):
    res = []
    for i in range(size):
        product = ''
        product += data_name[random.randint(0, len(data_name) - 1)] + ', '
        product += data_country[random.randint(0, len(data_country) - 1)] +
', '
        product += data_volume[random.randint(0, len(data_volume) - 1)] + ',
'
        product += data_price[random.randint(0, len(data_price) - 1)]
        res.append(product)

    with open(f'./data_{size}.txt', mode='w', encoding='UTF-8') as
data_product:
        for one_product in res:
            data_product.write(f"{one_product}" + '\n')  # записываем один
элемент в файл
```

«sorts.py»:

```
def bubble_sort(array):  # пузырьком

    for i in range(len(array) - 1):
        for j in range(len(array) - i - 1):
            if array[j] > array[j + 1]:
                array[j], array[j + 1] = array[j + 1], array[j]
    return array


def cocktail_sort(array):
    length = len(array)
    swapped = True
    start_index = 0
    end_index = length - 1

    while swapped:
        swapped = False
        for i in range(start_index, end_index):
            if array[i] > array[i + 1]:
                array[i], array[i + 1] = array[i + 1], array[i]
                swapped = True
        end_index = end_index - 1

        if not swapped:
            break
        swapped = False
        for i in range(end_index - 1, start_index - 1, -1):
            if array[i] > array[i + 1]:
                array[i], array[i + 1] = array[i + 1], array[i]
                swapped = True
        start_index = start_index + 1

    return array


def merge_sort(array):  # слиянием

    if len(array) >= 2:
        middle = int(len(array) / 2)
        left = array[:middle]
        right = array[middle:]

        merge_sort(left)
        merge_sort(right)

        i, j, k = 0, 0, 0

        while i < len(left) and j < len(right):
            if left[i] < right[j]:
                array[k] = left[i]
                i = i + 1
            else:
                array[k] = right[j]
                j = j + 1

            k = k + 1

        while i < len(left):
            array[k] = left[i]
            i = i + 1
            k = k + 1
```

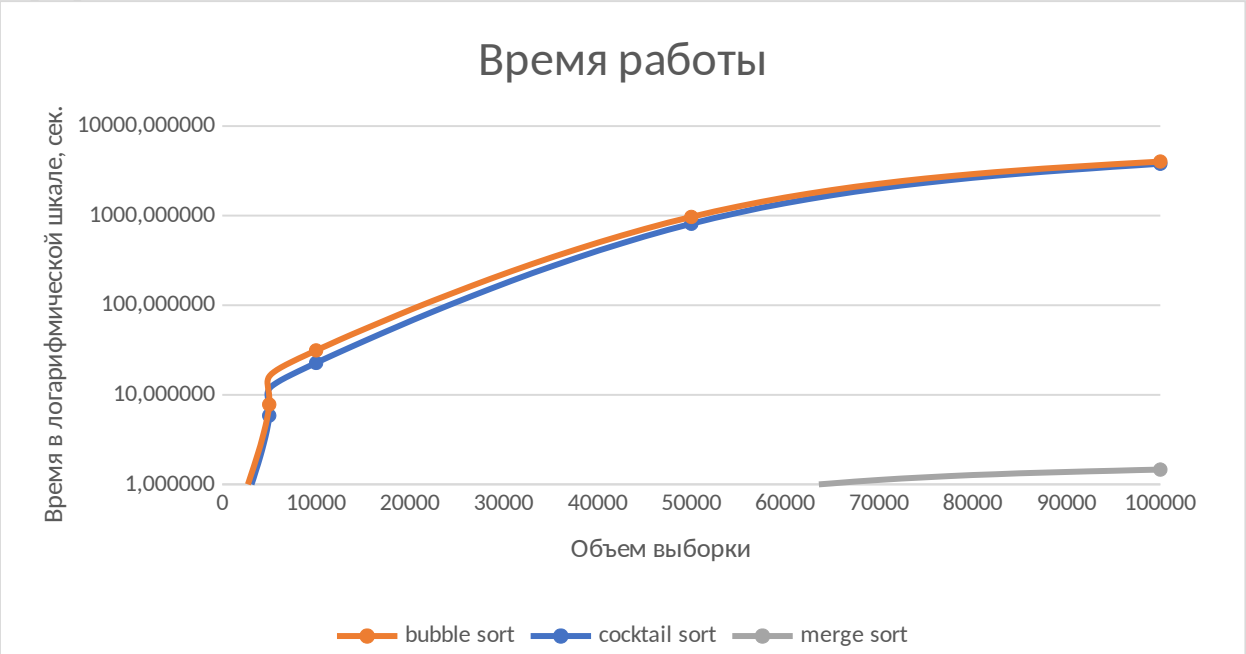
```
    while j < len(right):
        array[k] = right[j]
        j = j + 1
        k = k + 1

    return array
return array
```

Таблица. Время в секундах

	Время		
Размер выборки	bubble sort	cocktail sort	merge sort
100	0,002993	0,002019	0,000000
200	0,010970	0,007980	0,001024
500	0,070847	0,047872	0,003986
1000	0,286243	0,209404	0,006982
5000	7,798263	5,869252	0,063828
10000	31,329193	22,703028	0,105691
50000	968,309824	812,150526	0,700865
100000	4018,270904	3798,230911	1,464073

График



Вывод:

Были проведены испытания на выборках размером 100, 200, 500, 1000, 5000, 10000, 50000, 100000. Сортировка пузырьком и шейкер-сортировка работают значительно медленнее, чем сортировка слиянием. При этом шейкер-сортировка работает немного быстрее, чем сортировка пузырьком, что отчетливо видно на больших объемах выборки.