Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Оценка времени выполнения программ»

Выполнил:

студент группы 20ВВ2

Беляков А.М.

Проверили:

Митрохин М.А.

Юрова О.В.

Пенза 2021

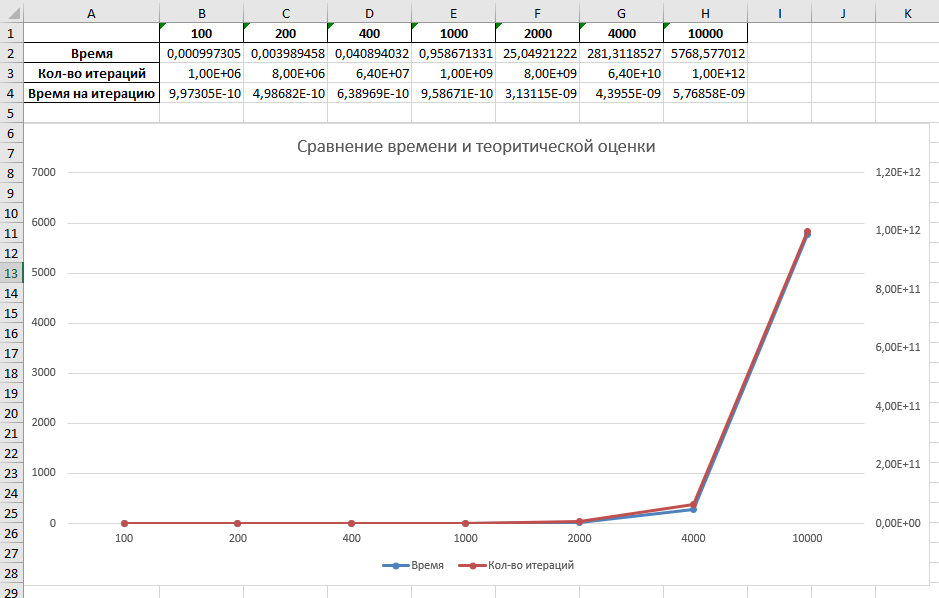
Задание 1:

1. Вычислить порядок сложности программы (О-символику).

2. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200,400, 1000, 2000, 4000, 10000.

3. Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

1.1) O()

1.2,3)

Листинг:

import numpy as np  
import time  
import pandas as pd  
  
  
def multiply(size):  
 matrix\_1 = np.ones([size, size], dtype=int)  
 matrix\_2 = np.ones([size, size], dtype=int)  
 print(size)  
 return timed(lambda: np.dot(matrix\_1, matrix\_2))  
  
  
def timed(function):  
 def wrapper():  
 start\_time = time.time()  
 value = function()  
 duration = time.time() - start\_time  
 return duration  
  
 return wrapper()  
  
  
def tests():  
 sizes = [100, 200, 400, 1000, 2000, 4000, 10000]  
  
 results = [[timed(lambda: multiply(i)) for i in sizes]]  
 results.append([i \*\* 3 for i in sizes])  
 results.append([results[0][i] / results[1][i] for i in range(len(sizes))])  
 result = pd.DataFrame(results, index=['Время', 'Кол-во итераций', 'Время на итерацию'],  
 columns=[str(i) for i in sizes])  
 result.to\_excel("results\_1.xlsx", sheet\_name="Sheet\_first")  
  
  
tests()

Задание 2:

1. Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.

2. Оценить время работы каждого из реализованных алгоритмов на массиве,

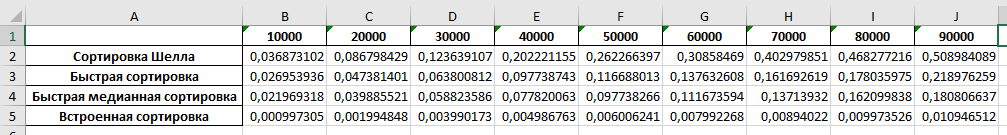
представляющем собой возрастающую последовательность чисел.

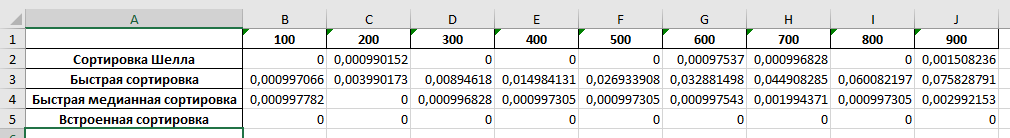
3. Оценить время работы каждого из реализованных алгоритмов на массиве,

представляющем собой убывающую последовательность чисел.

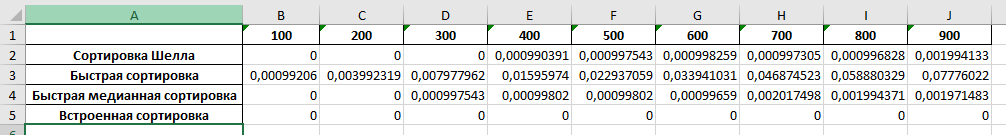
4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.

5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

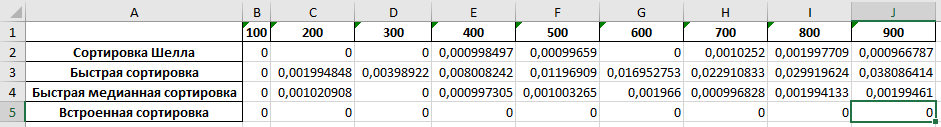
2.1,5) 

2.2,5) 

Использовать списки большей длинны не возможно из за переполнения стека рекурсии у быстрой сортировки.

2.3,5) 

Использовать списки большей длинны не возможно из за переполнения стека рекурсии у быстрой сортировки.

2.4,5) 

Листинг:

from random import randint  
import time  
import pandas as pd  
  
  
def shell\_sort(data):  
 data = data.copy()  
 last\_index = len(data) - 1  
 step = len(data) // 2  
 while step > 0:  
 for i in range(step, last\_index + 1, 1):  
 j = i  
 delta = j - step  
 while delta >= 0 and data[delta] > data[j]:  
 data[delta], data[j] = data[j], data[delta]  
 j = delta  
 delta = j - step  
 step //= 2  
 return data  
  
def quick\_sort\_normal(array):  
 if len(array) <= 1:  
 return array  
  
 support\_element = array[0]  
  
 left = list(filter(lambda x: x < support\_element, array))  
 center = [i for i in array if i == support\_element]  
 right = list(filter(lambda x: x > support\_element, array))  
  
 return quick\_sort\_normal(left) + center + quick\_sort\_normal(right)  
  
  
def quick\_sort\_median(array):  
 if len(array) <= 1:  
 return array  
  
 support\_element = (array[0] + array[-1] + array[len(array) // 2]) // 3  
  
 left = list(filter(lambda x: x < support\_element, array))  
 center = [i for i in array if i == support\_element]  
 right = list(filter(lambda x: x > support\_element, array))  
  
 return quick\_sort\_median(left) + center + quick\_sort\_median(right)  
  
  
def random\_elements(lenght, minimum, maximum):  
 return [randint(minimum, maximum) for \_ in range(lenght)]  
  
  
def growing\_elements(lenght):  
 return [i for i in range(lenght)]  
  
  
def decrease\_elements(lenght):  
 return [i for i in range(lenght, 0, -1)]  
  
  
def growing\_decrease\_elements(lenght):  
 return [i for i in range(int(lenght / 2))] + [i for i in range(int(lenght / 2), 0, -1)]  
  
  
def timed(function):  
 def wrapper():  
 start\_time = time.time()  
 value = function()  
 duration = time.time() - start\_time  
 return duration  
  
 return wrapper()  
  
  
def tests(lenghts, minimum, maximum):  
 indexs = ['Сортировка Шелла', 'Быстрая сортировка', 'Быстрая медианная сортировка', 'Встроенная сортировка']  
 place = "results\_2.xlsx"  
 massives = [[random\_elements(i\*100, minimum, maximum) for i in lenghts], [growing\_elements(i) for i in lenghts],  
 [decrease\_elements(i) for i in lenghts],  
 [growing\_decrease\_elements(i) for i in lenghts]]  
 results = [[[timed(lambda: shell\_sort(j)) for j in i] for i in massives],  
 [[timed(lambda: quick\_sort\_normal(j)) for j in i] for i in massives],  
 [[timed(lambda: quick\_sort\_median(j)) for j in i] for i in massives],  
 [[timed(lambda: sorted(j)) for j in i] for i in massives]]  
 result = pd.DataFrame([results[i][0] for i in range(len(results))], index=[indexs],  
 columns=[str(len(i)) for i in massives[0]])  
 result.to\_excel(place, sheet\_name="random\_elements")  
  
 result = pd.DataFrame([results[i][1] for i in range(len(results))], index=[indexs],  
 columns=[str(len(i)) for i in massives[1]])  
 with pd.ExcelWriter(place, mode='a') as writer:  
 result.to\_excel(writer, sheet\_name='growing\_elements')  
  
 result = pd.DataFrame([results[i][2] for i in range(len(results))], index=[indexs],  
 columns=[str(len(i)) for i in massives[2]])  
 with pd.ExcelWriter(place, mode='a') as writer:  
 result.to\_excel(writer, sheet\_name='decrease\_elements')  
  
 result = pd.DataFrame([results[i][3] for i in range(len(results))], index=[indexs],  
 columns=[str(len(i)) for i in massives[3]])  
 with pd.ExcelWriter(place, mode='a') as writer:  
 result.to\_excel(writer, sheet\_name='growing\_decrease\_elements')  
  
  
lenghts = [i for i in range(100, 1000, 100)]  
tests(lenghts, 0, 1000)

Вывод: Во время выполнения данной лабораторной работы мы научились оценивать сложность алгоритмов.