Министерство и науки высшего образования

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»

Работу выполнила студентка группы 21ВВ2:

Алёшина А.В.

Мубаряков А.А.

Приняли:

Юрова О.В.

МитрохинМ.А.

Пенза 2022

Листинг

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <iostream>

#include <windows.h>

#include <fstream>

void QuickSort(int\* items, int left, int right) //вызов функции: qs(items, 0, count-1);

{

int i, j;

int x, y;

i = left; j = right;

/\* выбор компаранда \*/

x = items[(left + right) / 2];

do {

while ((items[i] < x) && (i < right)) i++;

while ((x < items[j]) && (j > left)) j--;

if (i <= j) {

y = items[i];

items[i] = items[j];

items[j] = y;

i++; j--;

}

} while (i <= j);

if (left < j) QuickSort(items, left, j);

if (i < right) QuickSort(items, i, right);

}

void Shell\_sort(int\* array, int size) {

for (int s = size / 2; s > 0; s /= 2) {

for (int i = 0; i < size; i++) {

for (int j = i + s; j < size; j += s) {

if (array[i] > array[j]) {

int temp = array[j];

array[j] = array[i];

array[i] = temp;

}

}

}

}

}

void DecreasingSort(int n, int mass[])

{

int newN, location;

for (int i = 1; i < n; i++)

{

newN = mass[i];

location = i - 1;

while (location >= 0 && mass[location] < newN)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = newN;

}

}

void AscendingSort(int n, int mass[])

{

int newN, location;

for (int i = 1; i < n; i++)

{

newN = mass[i];

location = i - 1;

while (location >= 0 && mass[location] > newN)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = newN;

}

}

clock\_t Calculate(short int\*\* A, short int\*\* B, short int\*\* C,int N)

{

clock\_t start = clock();

using namespace std;

int i, j, k;

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

C[i][j] = 0;

for (k = 0; k < N; k++)

C[i][j] += A[i][k] \* B[k][j];

}

}

return clock() - start;

}

clock\_t Gemm\_v1(int N)

{

std::cout << "Запущена Gemm\_v1 от " << N << " элементов" << std::endl;

using namespace std;

short int\*\* A = new short int\* [N];

for (int i = 0; i < N; ++i)

{

A[i] = new short int[N];

for (int counter = 0; counter < N; ++counter)

{

A[i][counter] = rand() % 10;

}

}

short int\*\* B = new short int\* [N];

for (int i = 0; i < N; ++i)

{

B[i] = new short int[N];

for (int counter = 0; counter < N; ++counter)

{

B[i][counter] = rand() % 10;

}

}

short int\*\* C = new short int\* [N];

for (int i = 0; i < N; ++i)

{

C[i] = new short int[N];

for (int counter = 0; counter < N; ++counter)

{

C[i][counter] = rand() % 10;

}

}

clock\_t result = Calculate(A, B, C, N);

for (int i = 0; i < N; ++i)

{

delete[] A[i];

}

delete[] A;

for (int i = 0; i < N; ++i)

{

delete[] B[i];

}

delete[] B;

for (int i = 0; i < N; ++i)

{

delete[] C[i];

}

delete[] C;

return result;

}

int initArray(int\* array, const int SIZE, byte type)

{

if (SIZE % 2)return 1;

switch (type)

{

case 1: for (int counter = 0; counter < SIZE; ++counter)array[counter] = rand() % 200; break; //Случайный

case 2: for (int counter = 0; counter < SIZE; ++counter) array[counter] = counter \* 2; break; //Возрастающий

case 3: for (int counter = 0; counter < SIZE; ++counter) array[counter] = (SIZE - counter) \* 2; break; //Убывающий

case 4: for (int counter = 0; counter < SIZE / 2; ++counter) array[counter] = counter \* 2; for (int counter = SIZE / 2; counter < SIZE; ++counter) array[counter] = (SIZE - counter) \* 2; break; //Возрастающая горка

default: std::cerr << "\nError call initArray. Uncorrected type calling. Array\'s initializtion failed\n"; return 1;

}

return 0;

}

int funcMatrix()

{

std::cout << "Программа запущена, дождитесь окончания работы программы . . ." << std::endl;

std::cout << "Расчитываем время перемножения матриц" << std::endl;

clock\_t timeGemm[7] = { Gemm\_v1(100), Gemm\_v1(200), Gemm\_v1(400), Gemm\_v1(1000), Gemm\_v1(2000), Gemm\_v1(4000), Gemm\_v1(10000) };

std::cout << "Успешно" << std::endl;

std::cout << "Результат:" << std::endl;

std::cout << "Матрица 100х100 = " << timeGemm[0] << std::endl;

std::cout << "Матрица 200х200 = " << timeGemm[1] << std::endl;

std::cout << "Матрица 400х400 = " << timeGemm[2] << std::endl;

std::cout << "Матрица 1000х1000 = " << timeGemm[3] << std::endl;

std::cout << "Матрица 2000х2000 = " << timeGemm[4] << std::endl;

std::cout << "Матрица 4000х4000 = " << timeGemm[5] << std::endl;

std::cout << "Матрица 1000х1000 = " << timeGemm[6] << std::endl;

std::ofstream file("ResultMatrix.txt");

if (file)

{

file << "Результат:" << std::endl;

file << "Матрица 100х100 = " << timeGemm[0] << std::endl;

file << "Матрица 200х200 = " << timeGemm[1] << std::endl;

file << "Матрица 400х400 = " << timeGemm[2] << std::endl;

file << "Матрица 1000х1000 = " << timeGemm[3] << std::endl;

file << "Матрица 2000х2000 = " << timeGemm[4] << std::endl;

file << "Матрица 4000х4000 = " << timeGemm[5] << std::endl;

file << "Матрица 1000х1000 = " << timeGemm[6] << std::endl;

}

else return 1;

file.close();

return 0;

}

int funcSort()

{

std::ofstream file("ResultSort.txt");

if (!file)std::cout << "Ошибка открытия файла ResultSort.txt." << std::endl;

std::cout << "Расчитываем время сортировки массивов. . .";

int SIZES[7] = { 100, 200, 400, 1000, 2000, 4000, 10000 };

clock\_t timesSort[4][7];

for (int counter = 0; counter < 7; ++counter)

{

int\* array = new int[SIZES[counter]];

int buff = SIZES[counter];

initArray(array, SIZES[counter], 1);

timesSort[0][counter] = clock();

QuickSort(array, 0, SIZES[counter] - 1);

timesSort[0][counter] = clock() - timesSort[0][counter];

initArray(array, SIZES[counter], 2);

timesSort[1][counter] = clock();

QuickSort(array, 0, SIZES[counter] - 1);

timesSort[1][counter] = clock() - timesSort[1][counter];

initArray(array, SIZES[counter], 3);

timesSort[2][counter] = clock();

QuickSort(array, 0, SIZES[counter] - 1);

timesSort[2][counter] = clock() - timesSort[2][counter];

initArray(array, SIZES[counter], 4);

timesSort[3][counter] = clock();

//QuickSort(array, 0, SIZES[counter] - 1);

timesSort[3][counter] = clock() - timesSort[3][counter];

delete[] array;

}

std::cout << "Результат быстрой сортировки:" << std::endl; if (file) file << "Результат быстрой сортировки:" << std::endl;

std::cout << "\nНеупорядочный набор:" << std::endl; if (file) file << "\nНеупорядочный набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[0][counter] << std::endl;

if(file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[0][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Возрастающий набор:" << std::endl; if (file) file << "Возрастающий набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[1][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[1][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Убывающий набор:" << std::endl; if (file) file << "Убывающий набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[2][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[2][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Набор \"горка\":" << std::endl; if (file) file << "Набор \"горка\":" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[3][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[3][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << std::endl; if (file) file << std::endl;

std::cout << std::endl; if (file) file << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

int\* array = new int[SIZES[counter]];

initArray(array, SIZES[counter], 1);

timesSort[0][counter] = clock();

Shell\_sort(array, SIZES[counter]);

timesSort[0][counter] = clock() - timesSort[0][counter];

initArray(array, SIZES[counter], 2);

timesSort[1][counter] = clock();

Shell\_sort(array, SIZES[counter]);

timesSort[1][counter] = clock() - timesSort[1][counter];

initArray(array, SIZES[counter], 3);

timesSort[2][counter] = clock();

Shell\_sort(array, SIZES[counter]);

timesSort[2][counter] = clock() - timesSort[2][counter];

initArray(array, SIZES[counter], 4);

timesSort[3][counter] = clock();

Shell\_sort(array, SIZES[counter]);

timesSort[3][counter] = clock() - timesSort[3][counter];

}

std::cout << "Результат сортировки Шелла:" << std::endl; if (file) file << "Результат сортировки Шелла:" << std::endl;

std::cout << "\nНеупорядочный набор:" << std::endl; if (file) file << "\nНеупорядочный набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[0][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[0][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Возрастающий набор:" << std::endl; if (file) file << "Возрастающий набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[1][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[1][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Убывающий набор:" << std::endl; if (file) file << "Убывающий набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[2][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[2][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Набор \"горка\":" << std::endl; if (file) file << "Набор \"горка\":" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[3][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[3][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << std::endl; if (file) file << std::endl;

std::cout << std::endl; if (file) file << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

int\* array = new int[SIZES[counter]];

initArray(array, SIZES[counter], 1);

timesSort[0][counter] = clock();

DecreasingSort(SIZES[counter], array);

timesSort[0][counter] = clock() - timesSort[0][counter];

initArray(array, SIZES[counter], 2);

timesSort[1][counter] = clock();

DecreasingSort(SIZES[counter], array);

timesSort[1][counter] = clock() - timesSort[1][counter];

initArray(array, SIZES[counter], 3);

timesSort[2][counter] = clock();

DecreasingSort(SIZES[counter], array);

timesSort[2][counter] = clock() - timesSort[2][counter];

initArray(array, SIZES[counter], 4);

timesSort[3][counter] = clock();

DecreasingSort(SIZES[counter], array);

timesSort[3][counter] = clock() - timesSort[3][counter];

}

std::cout << "Результат сортировки Decreasing:" << std::endl; if (file) file << "Результат сортировки Decreasing:" << std::endl;

std::cout << "\nНеупорядочный набор:" << std::endl; if (file) file << "\nНеупорядочный набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[0][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[0][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Возрастающий набор:" << std::endl; if (file) file << "Возрастающий набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[1][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[1][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Убывающий набор:" << std::endl; if (file) file << "Убывающий набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[2][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[2][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Набор \"горка\":" << std::endl; if (file) file << "Набор \"горка\":" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[3][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[3][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << std::endl; if (file) file << std::endl;

std::cout << std::endl; if (file) file << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

int\* array = new int[SIZES[counter]];

initArray(array, SIZES[counter], 1);

timesSort[0][counter] = clock();

AscendingSort(SIZES[counter], array);

timesSort[0][counter] = clock() - timesSort[0][counter];

initArray(array, SIZES[counter], 2);

timesSort[1][counter] = clock();

AscendingSort(SIZES[counter], array);

timesSort[1][counter] = clock() - timesSort[1][counter];

initArray(array, SIZES[counter], 3);

timesSort[2][counter] = clock();

AscendingSort(SIZES[counter], array);

timesSort[2][counter] = clock() - timesSort[2][counter];

initArray(array, SIZES[counter], 4);

timesSort[3][counter] = clock();

AscendingSort(SIZES[counter], array);

timesSort[3][counter] = clock() - timesSort[3][counter];

}

std::cout << "Результат сортировки Ascending:" << std::endl; if (file) file << "Результат сортировки Ascending:" << std::endl;

std::cout << "\nНеупорядочный набор:" << std::endl; if (file) file << "\nНеупорядочный набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[0][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[0][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Возрастающий набор:" << std::endl; if (file) file << "Возрастающий набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[1][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[1][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Убывающий набор:" << std::endl; if (file) file << "Убывающий набор:" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[2][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[2][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << "Набор \"горка\":" << std::endl; if (file) file << "Набор \"горка\":" << std::endl;

for (int counter = 0; counter < 7; ++counter)

{

std::cout << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[3][counter] << std::endl;

if (file) file << "\n\nПри наборе " << SIZES[counter] << "x" << SIZES[counter] << " - " << timesSort[3][counter] << std::endl;

}

std::cout << std::endl; if (file) file << std::endl;

std::cout << std::endl; if (file) file << std::endl;

std::cout << std::endl; if (file) file << std::endl;

file.close();

return 0;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

funcMatrix();

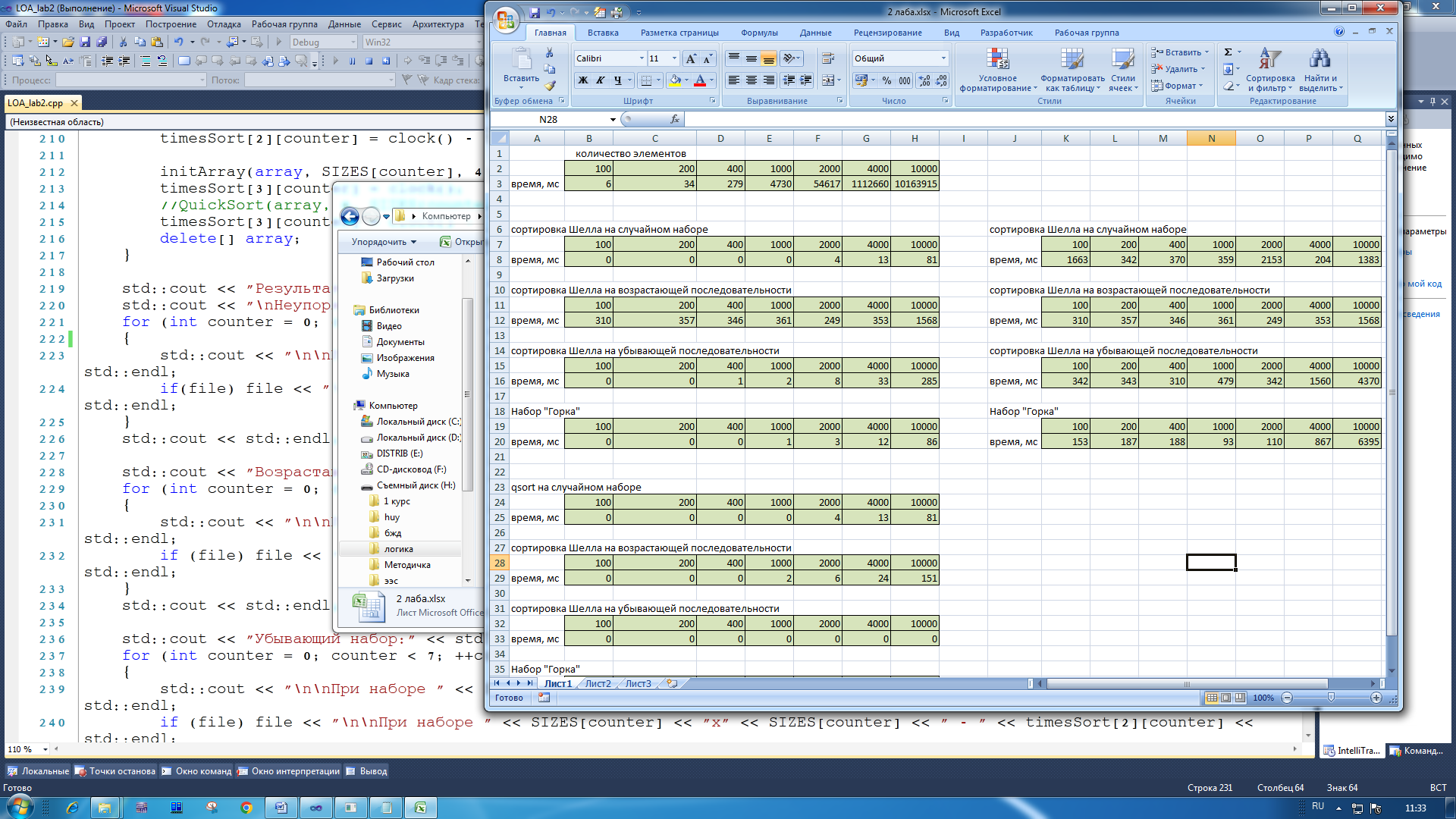
funcSort();

return 0;

}

Задание 1:

1. O(n3)



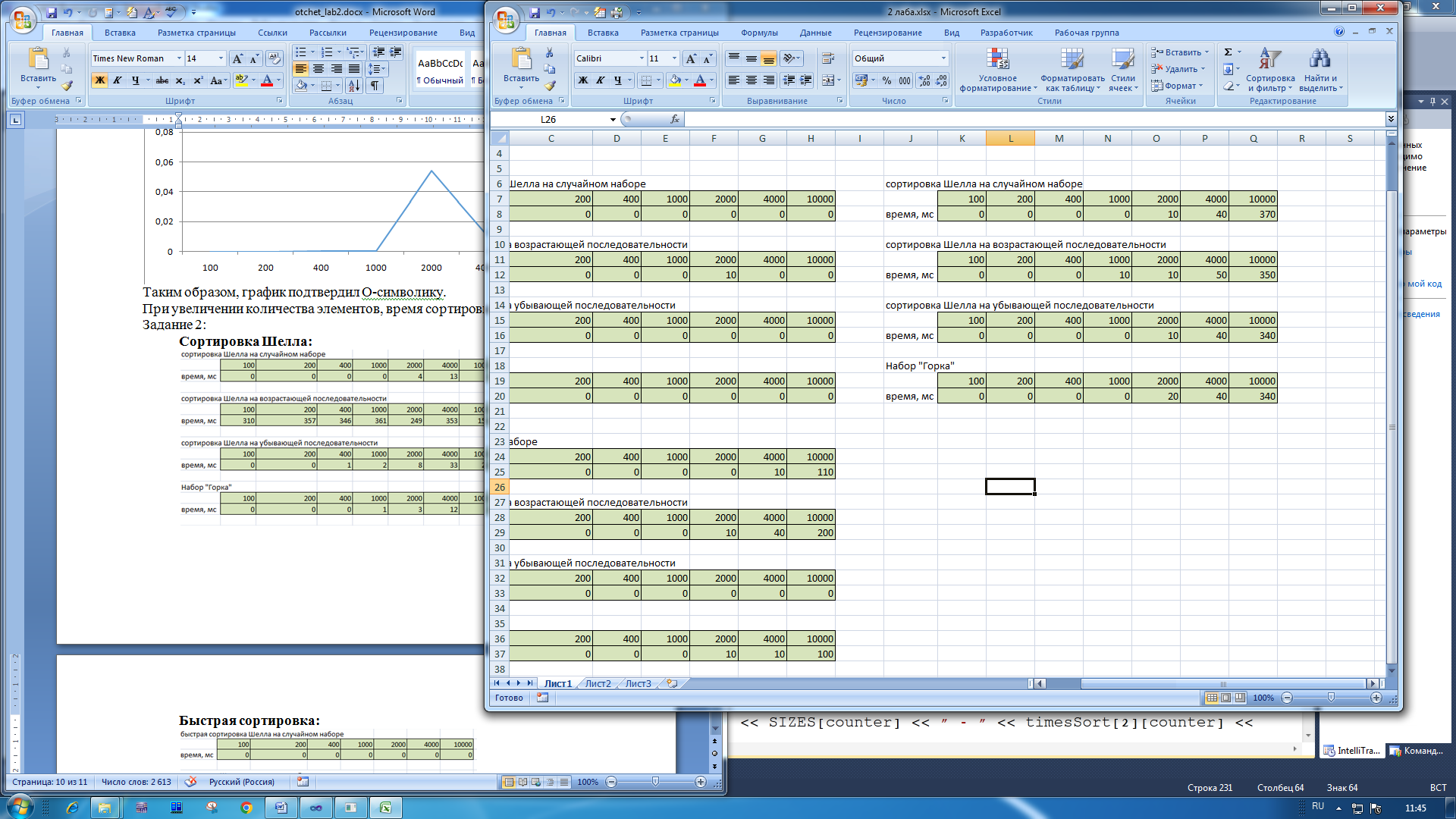
3.

Таким образом, график подтвердил О-символику.

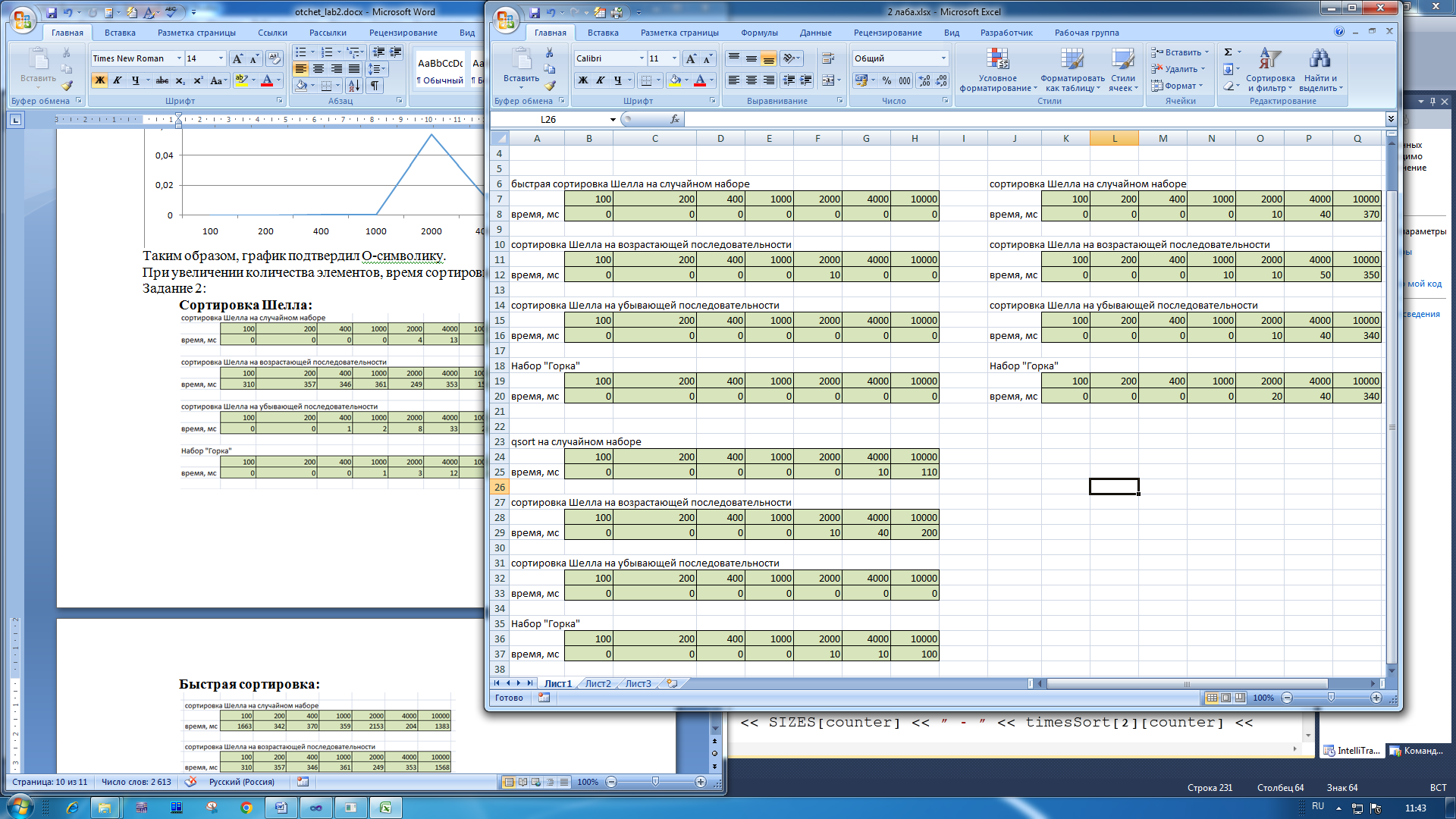
При увеличении количества элементов, время сортировки также увеличивается

Задание 2:

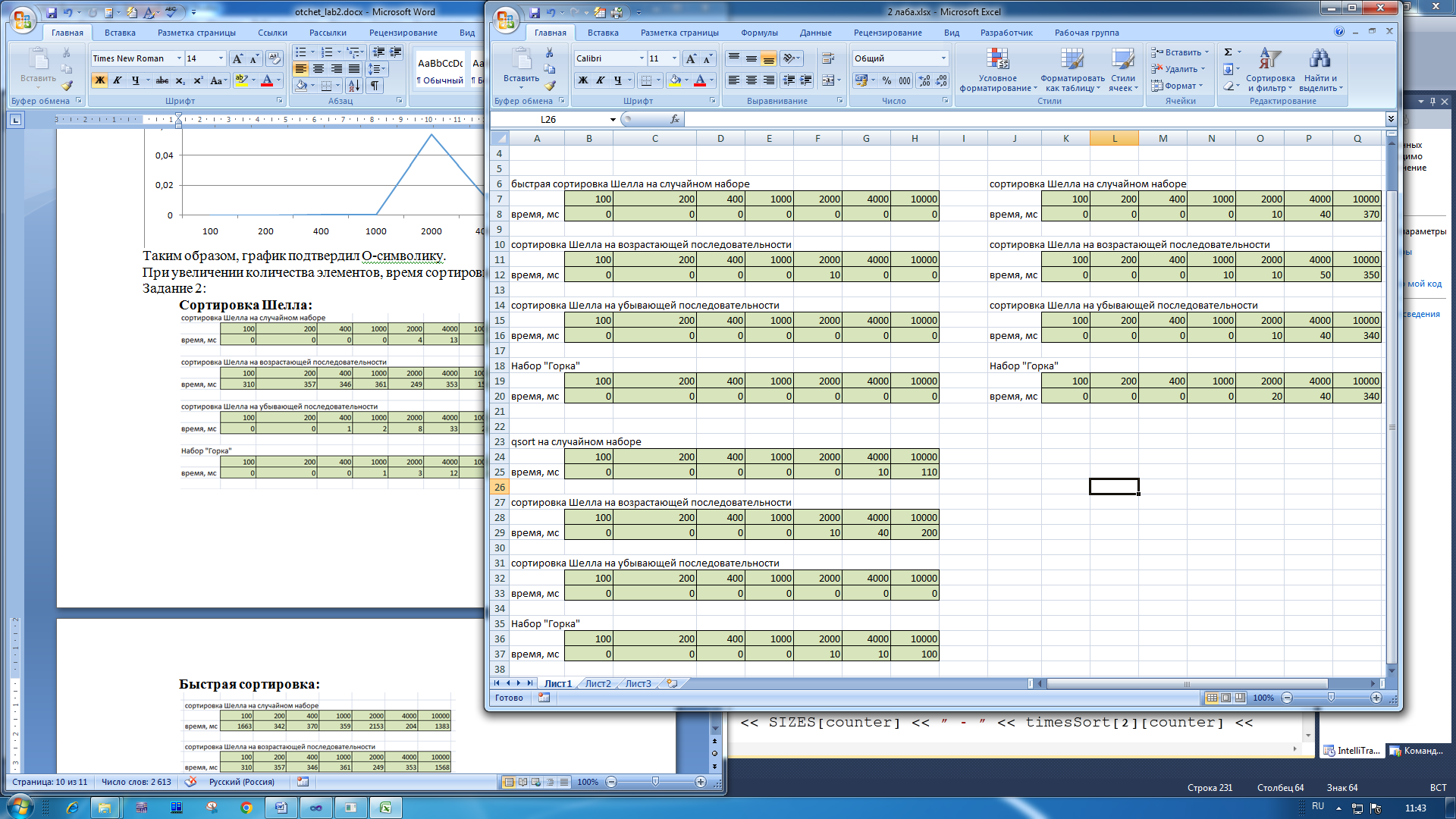
**Сортировка Шелла:**

****

**Быстрая сортировка:**

****

**qsort:**

****

**Вывод:**

Среди сортировок: Шелла, быстрой и qsort, самой долгой оказалась

сортировка Шелла. Быстрая сортировка показала наилучший результат по времени. Во всех случаях сортировки быстрее справлялись на уже упорядоченном по возрастанию или убыванию массиве.