ФГБОУ ВО

"Уфимский государственный нефтяной технический университет"

Кафедра вычислительной техники и инженерной кибернетики.

Дисциплина: Структуры и алгоритмы обработки данных.

ОТЧЕТ

по лабораторной работе № 1

"Алгоритмы внутренней сортировки"

Вариант 10

Выполнил ст. гр. БПО-15-01 Калимуллина Э. Р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил: доцент Жолобова Г.Н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

оценка, дата, подпись

Уфа 2018

Цель работы: изучение основных методов упорядочивания данных, расположенных в оперативной памяти; определение сложности алгоритмов сортировки и производительности.

Задание на лабораторную работу:

Написать программу, состоящую из следующих пунктов:

1. Сортировка пузырьком;

2. Гномья сортировка;

3. Сортировка методом Шелла;

4. Восходящая сортировка;

5. Выход.

В пп. 1- 4 продемонстрировать работу соответствующей сортировки. В п.5 подсчитать время сортировки для каждой сортировки.

**1. Описание сортировок**

**1.1. Сортировка пузырьком**

Сортировка пузырьком (обменная сортировка) – простой в реализации и малоэффективный алгоритм сортировки. В лучшем случае сортировка пузырьком потребует O(n) времени, а в среднем и худшем – O(n2).

Идея алгоритма заключается в следующем. Соседние элементы последовательности сравниваются между собой и, в случае необходимости, меняются местами.

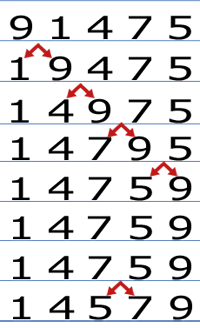


Рисунок 1 – Пример пузырьковой сортировки

Код программы на Си++:

void BubbleSort(int \*A, int n)

{

int tmp;

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (A[j] > A[j + 1])

{

tmp = A[j];

A[j] = A[j + 1];

A[j + 1] = tmp;

}

}

}

}

**1.2. Гномья сортировка**

Гномья сортировка основана на технике, используемой обычным голландским садовым гномом. Это метод, которым садовый гном сортирует линию цветочных горшков. По существу он смотрит на следующий и предыдущий садовые горшки: если они в правильном порядке, он шагает на один горшок вперёд, иначе он меняет их местами и шагает на один горшок назад. Граничные условия: если нет предыдущего горшка, он шагает вперёд; если нет следующего горшка, он закончил.

Гномья сортировка – это оптимизированная глупая сортировка. В глупой сортировке при нахождении неотсортированной пары соседей происходит обмен и возврат в начало массива. В гномьей сортировке просто делается один шаг назад.

Время работы сортировки O(n2).

Код программы на Си++:

void GnomeSort(int \*A, int N)

{

int i = 0;

while(i < N)

{

if(i == 0 || A[i - 1] <= A[i])

{

++i;

}

else

{

int Temp = A[i];

A[i] = A[i - 1];

A[i - 1] = Temp;

--i;

}

}

}

**1.3. Сортировка методом Шелла**

В методе Шелла применяются сравнения и перестановки элементов аналогичные методу вставок, но при этом порядок сравниваемых элементов совершенно другой.

Идея сортировки методом Шелла состоит в том, чтобы сортировать элементы отстоящие друг от друга на некотором расстоянии step. Затем сортировка повторяется при меньших значениях step, и в конце процесс сортировки Шелла завершается при step = 1 (а именно обычной сортировкой вставками).

В лучшем случае сортировка Шелла потребует O(n log n) времени, а в худшем – O(n2).

****

Рисунок 2 – Пример сортировки методом Шелла

Код программы на Си++:

void ShellSort(int \*A, int n)

{

int i, j, step;

int tmp;

for (step = n / 2; step > 0; step /= 2)

for (i = step; i < n; i++)

{

tmp = A[i];

for (j = i; j >= step; j -= step)

{

if (tmp < A[j - step])

A[j] = A[j - step];

else

break;

}

A[j] = tmp;

}

}

**1.4. Восходящая сортировка**

Алгоритм восходящей сортировки слиянием выглядит следующим образом:

1) выделяется временный буфер памяти размером с исходный массив;

2) выполняется попарное слияние элементов, результат записывается во временный массив;

3) меняются местами указатели на временный массив и на исходный массив;

4) выполняется попарное слияние фрагментов длиной 2;

5) все вышеперечисленное продолжается до тех пор, пока не останется один кусок;

6) в конце работы алгоритма удаляется временный массив.

Время работы сортировки O(n log n).

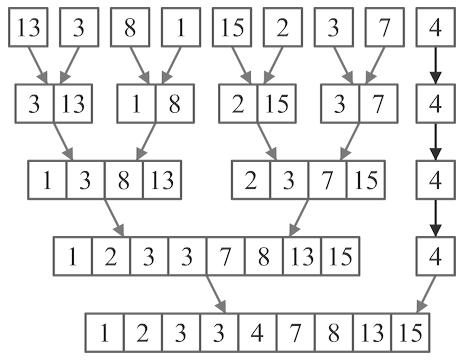


Рисунок 3 – Пример восходящей сортировки

Код программы на Си++:

void MergeSort(int \*A, int n)

{

int step = 1;

int \*temp = (int\*)malloc(n \* sizeof(temp));

while (step < n)

{

int index = 0;

int l = 0;

int m = l + step;

int r = l + step \* 2;

do

{

m = m < n ? m : n;

r = r < n ? r : n;

int i1 = l, i2 = m;

for (; i1 < m && i2 < r; )

{

if (A[i1] < A[i2])

{ temp[index++] = A[i1++]; }

else

{ temp[index++] = A[i2++]; }

}

while (i1 < m)

temp[index++] = A[i1++];

while (i2 < r)

temp[index++] = A[i2++];

l += step \* 2;

m += step \* 2;

r += step \* 2;

} while (l < n);

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

A[i] = temp[i];

step \*= 2;

}

}

**2. Зависимости между временем работы сортировок и количеством данных**

Ниже приведен результат работы программы, благодаря которому можно узнать, сколько по времени выполняется, например, сортировка пузырьком, если количество данных равняется 10 000.

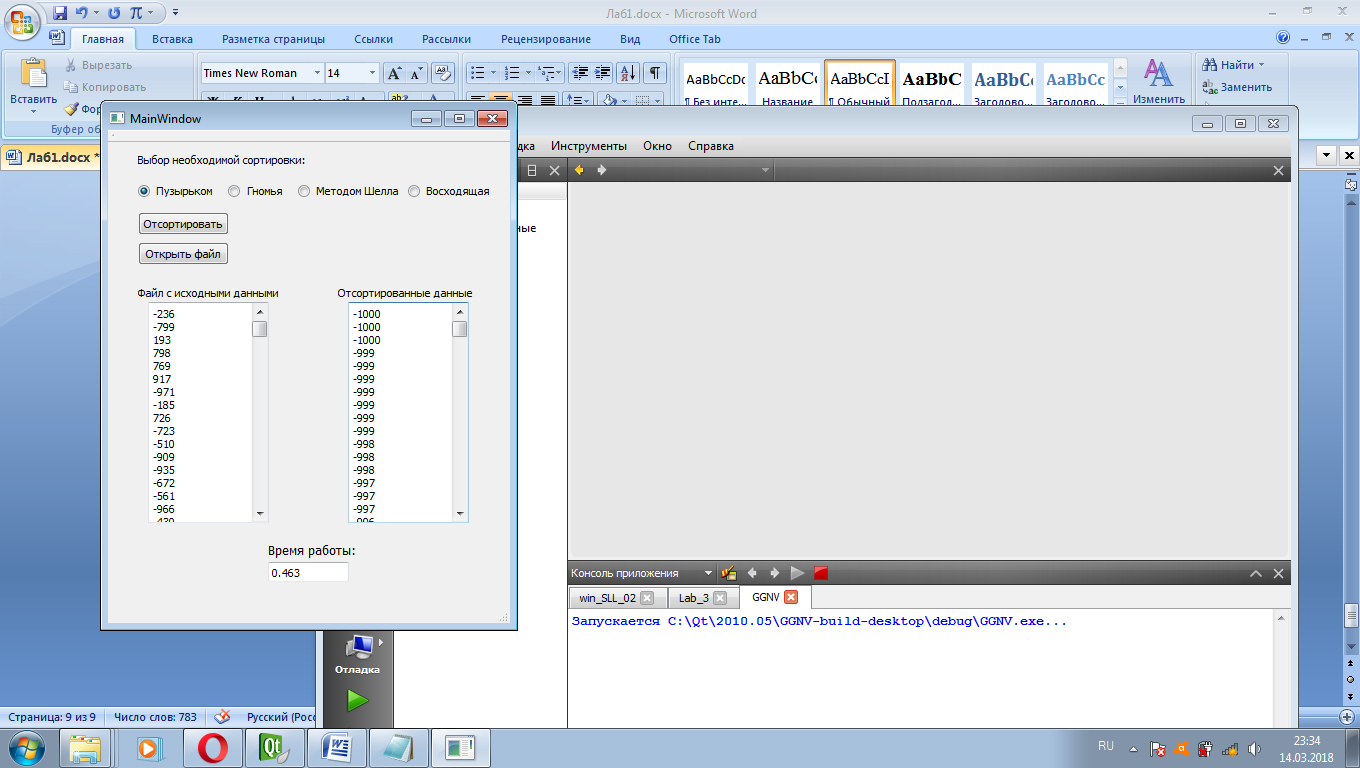
****

Рисунок 4 – Результат работы программы

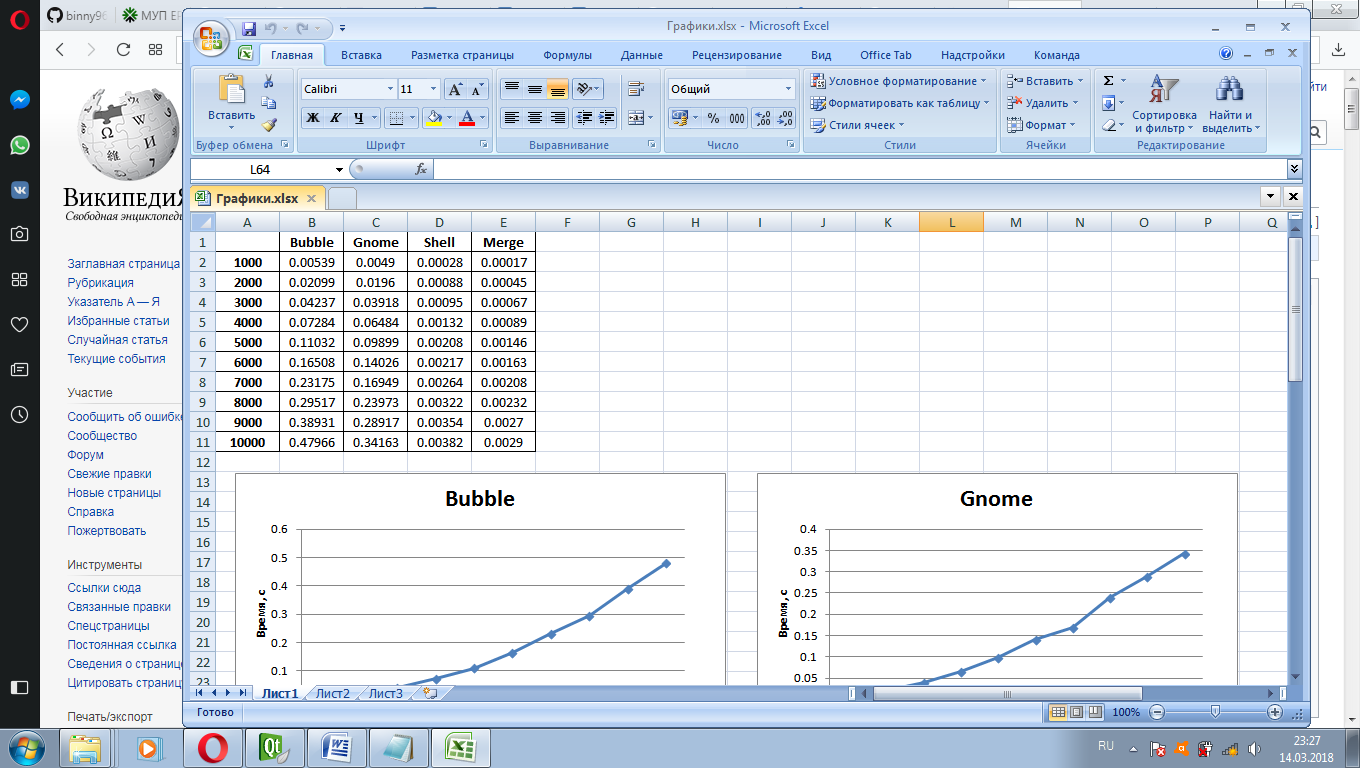


Рисунок 5 – Таблица зависимостей между временем работы

сортировок и количеством данных (1-й столбец)

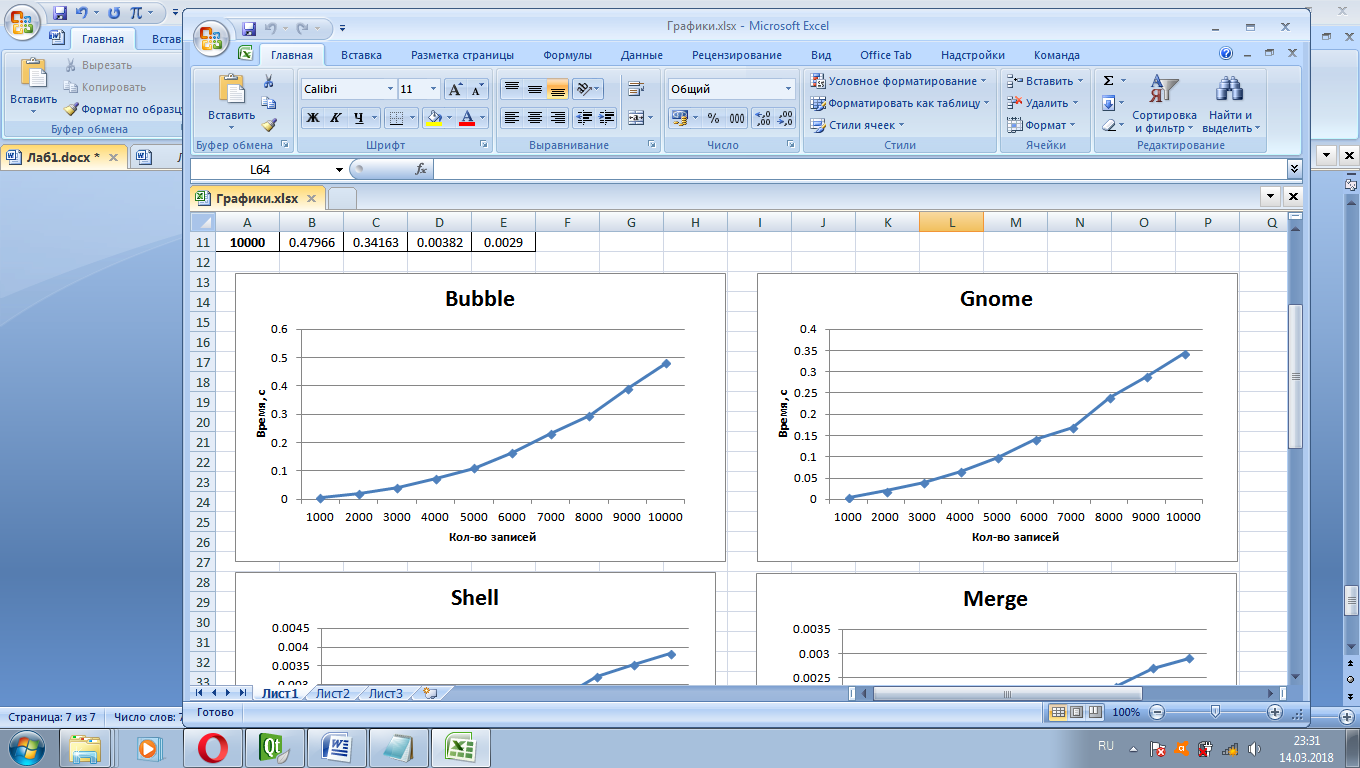


Рисунок 6 – Зависимость сортировки пузырьком

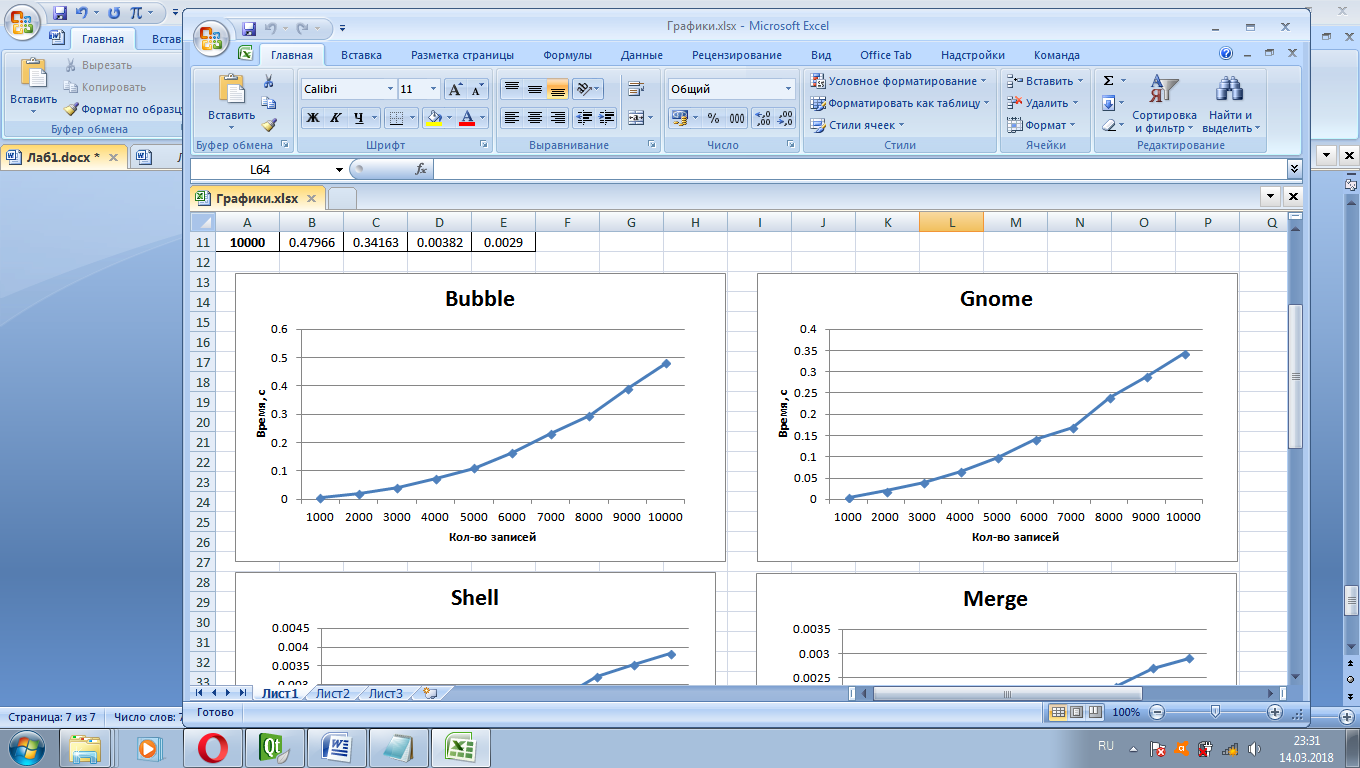


Рисунок 7 – Зависимость гномьей сортировки

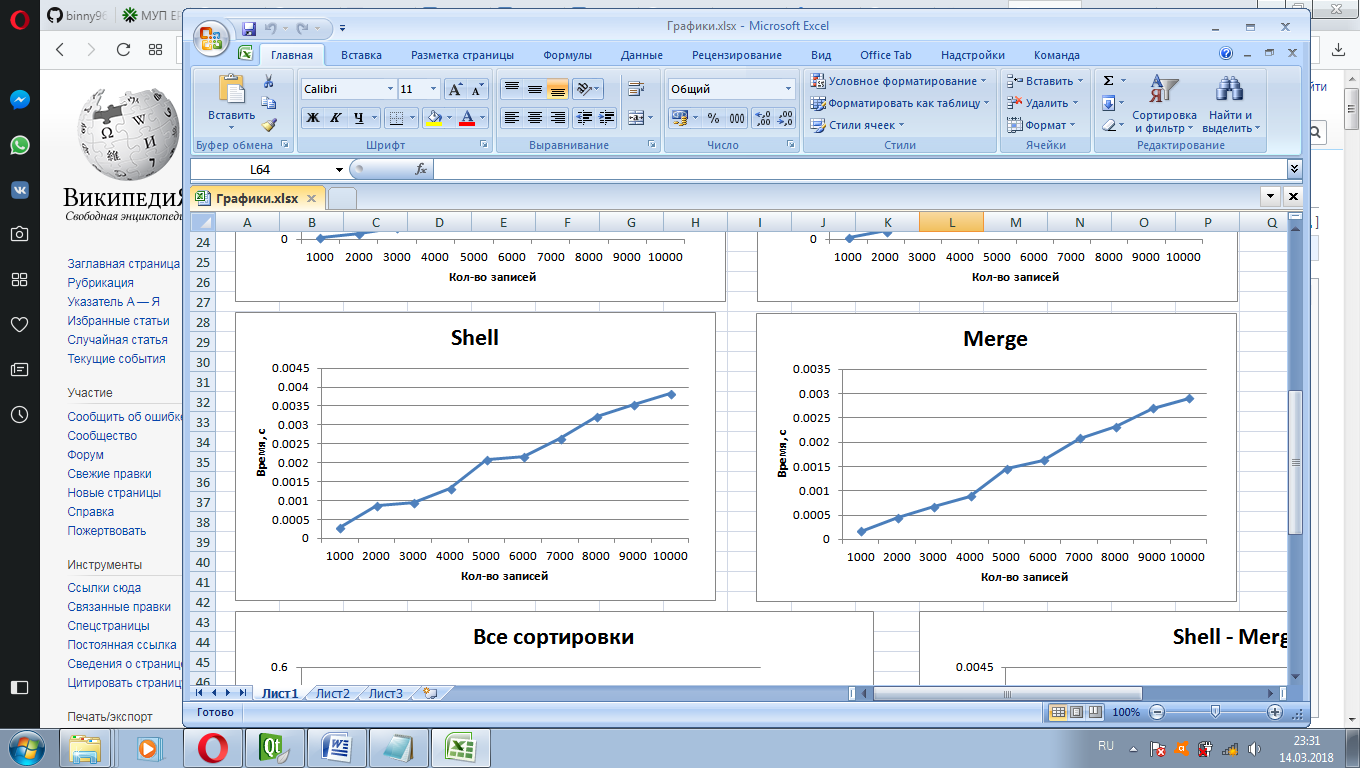


Рисунок 8 – Зависимость сортировки методом Шелла

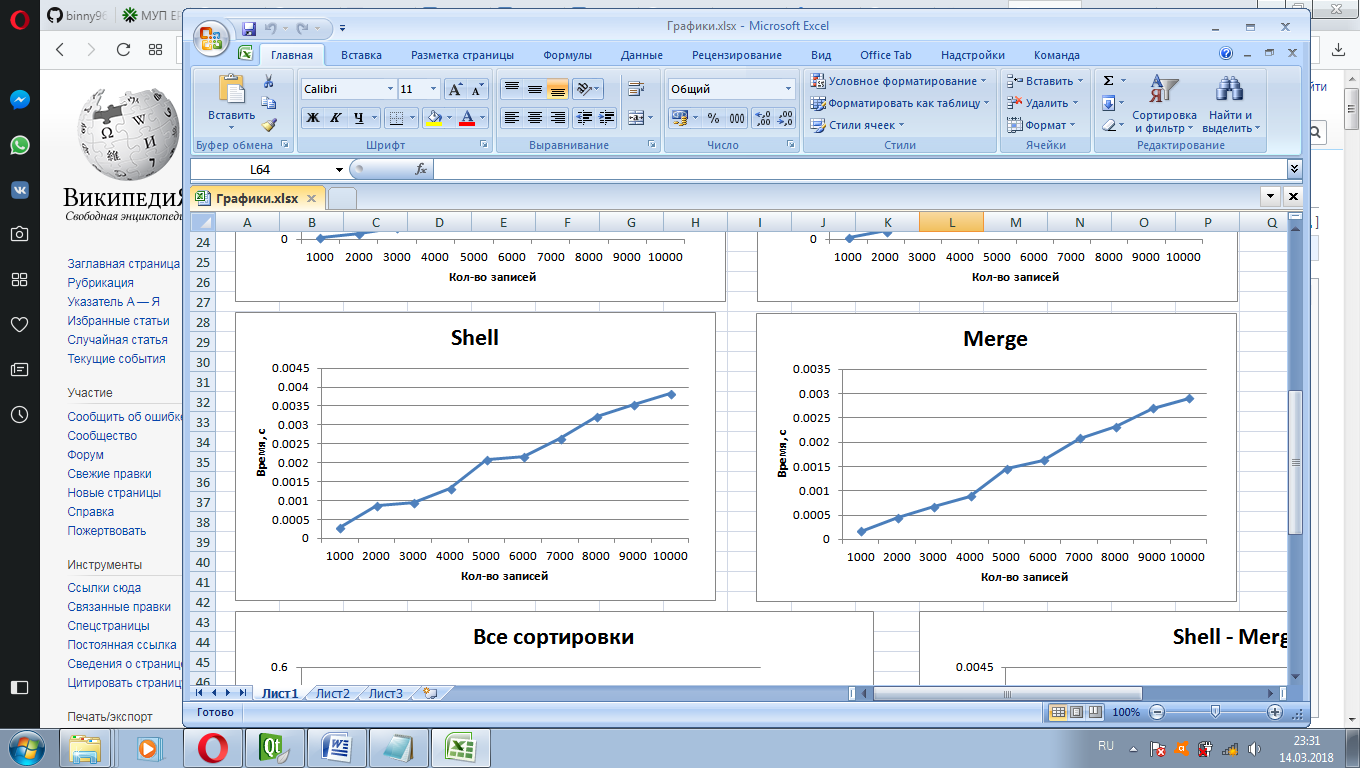


Рисунок 9 – Зависимость восходящей сортировки

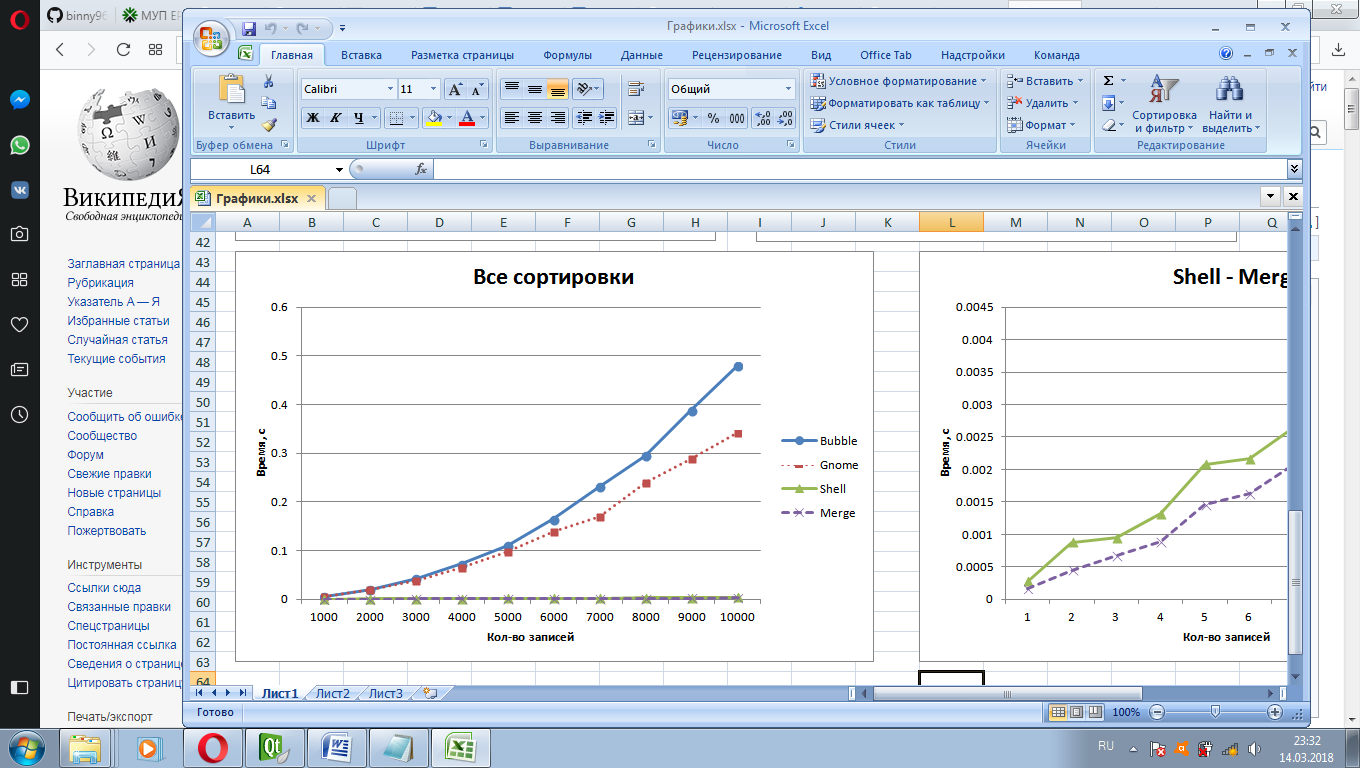


Рисунок 10 – Зависимости всех сортировок

Поскольку на графике недостаточно оценить зависимости сортировки методом Шелла и восходящей сортировки, построим для них отдельный график.

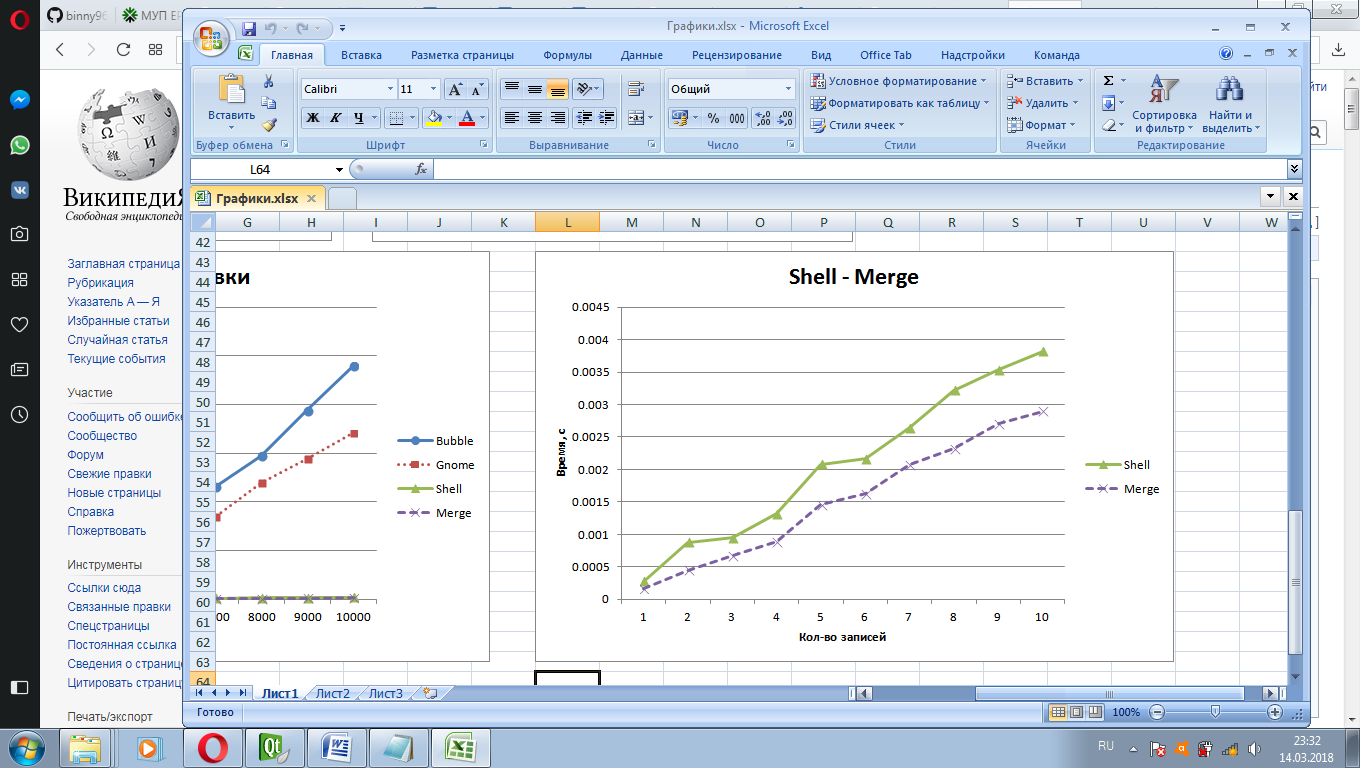


Рисунок 11 – Зависимости сортировки методом Шелла

И восходящей сортировки

**3. Вывод**

Исходя из графиков видно, что быстрее всех работает восходящая сортировка. Примерно также работает сортировка методом Шелла. Самой медленной является сортировка пузырьком.