

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
имени М. В. ЛОМОНОСОВА  
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

ОТЧЕТ ПО ЗАДАНИЮ №1

**«Методы сортировки»**

**Вариант 2 4 3 4**

Исполнитель: студент 106 группы

Магомедов Абдуррахман

Преподаватели:

Соловьев Михаил Александрович

Корухова Людмила Сергеевна

Манушин Дмитрий Валерьевич

МОСКВА

2020

## Оглавление

Постановка задачи.....	3
Результаты экспериментов.....	4
Графики.....	5
Структура программы и спецификации функций.....	7
Отладка программы, тестирование функций.....	9
Анализ допущенных ошибок.....	10
Литература.....	11

## Постановка задачи

Необходимо реализовать два метода сортировки чисел типа `long long int` по невозроанию модулей и провести их сравнение.

Первый метод сортировки – быстрая сортировка или quicksort, второй – сортировка Шелла (shellsort). Память будет выделяться динамически. При выполнении сортировки каждого вида будет подсчитываться число сравнений и перемещений элементов.

Рассматриваются длины массивов  $10$ ,  $10^2$ ,  $10^3$ ,  $10^4$ ,  $10^5$ ,  $10^6$ ,  $10^7$ ,  $10^8$ . Для каждой длины массива будет генерироваться четыре массива:

1. массив, в котором элементы упорядочены
2. массив, в котором элементы упорядочены в обратном порядке
- 3, 4. массивы со случайной расстановкой элементов

Результаты запуска программы будут записаны в таблицу.

Код доступен на <https://github.com/abrikos110/polygon/tree/master/sorts-homework>

## Результаты экспериментов

quicksort

n	Параметр	Номер сгенерированного массива			4 Среднее значение
		1	2	3	
	Сравнения	35	30	42	35 35.5
10	Перемещения	0	5	10	8 5.75
	Сравнения	671	670	813	896 762.5
100	Перемещения	0	50	162	162 93.5
	Сравнения	9976	9972	14130	14331 12102.25
1000	Перемещения	0	500	2316	2287 1275.75
	Сравнения	134219	134214	177464	172089 154496.5
10000	Перемещения	0	5000	31074	31497 16892.75
	Сравнения	1668927	1668926	2179320	2192024 1927299.25
100000	Перемещения	0	50000	390539	389427 207491.5
	Сравнения	19951424	19951420	27961392	27397214 23815362.5
1000000	Перемещения	0	500000	4632993	4638899 2442973.0
	Сравнения	233759915	233759910	317513594	311935159 274242144.5
10000000	Перемещения	0	5000000	53927058	54285919 28303244.25
	Сравнения	2666003370	2666003368	3551494913	3598468328 3120492494.75
100000000	Перемещения	0	50000000	619542986	619082528 322156378.5

shellsort

n	Параметр	Номер сгенерированного массива			4 Среднее значение
		1	2	3	
	Сравнения	22	38	35	27 30.5
10	Перемещения	0	25	16	8 12.25
	Сравнения	327	778	1065	1026 799.0
100	Перемещения	0	520	772	732 506.0
	Сравнения	4550	11390	23303	21591 15208.5
1000	Перемещения	0	7690	19128	17397 11053.75
	Сравнения	60156	137708	425883	416874 260155.25
10000	Перемещения	0	80958	369106	360043 202526.75
	Сравнения	758531	1662744	5137526	5407488 3241572.25
100000	Перемещения	0	984680	4418005	4687899 2522646.0
	Сравнения	9310065	22320677	62196841	61855813 38920849.0
1000000	Перемещения	0	13933052	53323091	52981905 30059512.0
	Сравнения	108851935	237412758	725492690	725604358 449340435.25
10000000	Перемещения	0	132894650	620122606	620232921 343312544.25
	Сравнения	1255334358	2744265881	8491922258	8765015802 5314134574.75
100000000	Перемещения	0	1587165582	7277665982	7550747330 4103894723.5

Серьёзных противоречий экспериментальных данных с теоретическими оценками сложности алгоритмов обнаружено не было

## Графики

Для просмотра графиков необходим цвет. Графики строились в логарифмическом масштабе с помощью matplotlib

Количество обменов (swaps) shellsort на отсортированном массиве (shellsort, sorted) равно 0

Все четыре графика сравнений (comparisons) quicksort очень близки друг к другу и накладываются друг на друга

Легенда:

- чёрный цвет – shellsort, sorted
- зелёный – shellsort, sorted, reversed
- синий – shellsort, random
- красный – quicksort, sorted
- жёлтый – quicksort, sorted, reversed
- magenta – quicksort, random



## Структура программы и спецификации функций

### **some\_funcs.h:**

- `long long int rand_ll(unsigned only_nonnegative);` // Возвращает

неотрицательное случайное число, если `only_nonnegative != 0`, иначе

возвращает случайное число во всём диапазоне допустимых значений типа `long long int`

- `void swap_vp(void *a, void *b, int sz);` // Меняет местами `sz` байт

по соответствующим указателям. Области памяти не должны пересекаться.

Используется глобальная `extern` переменная для подсчёта числа вызовов

- `long long int median3(long long int, long long int, long long`

`int);` // Возвращает медиану трёх чисел (использовалась в `quicksort`)

### **shellsort.h:**

- `void insertionsort(int n, long long int *a, int step, int (*less)(long`

`long int, long long int));` // Сортировка вставками по неубыванию на массиве `a`

длины `n` с шагом `step` (`a[0]`, `a[step]`, `a[2*step]`, ..., `a[(n-1)/step*step]`) и

функцией сравнения `less`, которая возвращает значение `!=0`, если первый

аргумент должен стоять раньше второго, иначе возвращает `0`

- `int step(int n, int i);` // `i`-ый (нумерация с `0`) шаг для сортировки

вставками массива длины `n` в сортировке Шелла. Здесь используется

последовательность  $f(j) = (2^j - 3)(2^{j+1} - 3)$ , которая обеспечивает асимптотику

$O(n^{4/3})$  в худшем случае [1]. Выбирается максимальное `j`, для которого  $f(j) < n$ ,

возвращается  $f(j-i)$

- `void shellsort(int n, long long int *a, int (*less)(long long int, long`

`long int), int (*step)(int, int));` // Параметры аналогичны `insertionsort`, только

есть дополнительный параметр указателя на функцию шага для сортировки

вставками

### **quicksort.h**

- `int partition(int n, long long int *a, int (*less)(long long int, long long int), long long int pivot);` // Функция разделяет массив на две части: если  $0 \leq i < p$ , то  $a[i] \leq pivot$ , если  $p \leq i < n$ , то  $a[i] \geq pivot$

- `void quicksort(int n, long long int *a, int (*less)(long long int, long long int));` // Параметры аналогичны `insertionsort`, сортирует массив с помощью `quicksort`. Средняя сложность –  $O(n \log n)$ , худшая –  $O(n^2)$  [3]

#### **main.c:**

- `long long int lliabs(long long int x);` // Возвращает абсолютное значение числа

- `int less(long long int a, long long int b);` // Возвращает `lliabs(a) > lliabs(b)` (для соответствия сортировки заданию). Используется глобальная переменная для подсчёта количества вызовов

- `void fill_array(int n, long long int *a, int filltype, int (*less)(long long int, long long int));` // Заполняет массив `a` длины `n` в соответствии с номером типа заполнения `filltype` (см. раздел Постановка задачи)

- `int main(void);` // Выделяет необходимую память, тестирует сортировки на массивах разной длины и выводит результат в `stdout`, освобождает память



## Отладка программы, тестирование функций

Функция `rand_lll` тестировалась построением графиков распределения, ошибки определялись на глаз. Проверялся знак возвращаемого значения `rand_lll(1)`

Функции `swar_vp`, `median3` отдельно не проверялись. В комментариях `median3` написано, что происходит в коде

Функции `insertionsort`, `step` отдельно не проверялись, функция `shellsort` проверялась на правильность сортировки

Функции `partition`, `quicksort` проверялись на соответствие требованиям, были найдены ошибки (см. раздел Анализ допущенных ошибок)

Функции `lliabs`, `less`, `fill_array` отдельно не проверялись

Функция `main` проверялась вручную с просмотром выводимых данных, были найдены ошибки

## Анализ допущенных ошибок

Функция `partition` много раз переписывалась, так как она иногда не обеспечивала правильное разделение массива, что приводило к неправильной сортировке

В функции `main` сначала тестировались `shellsort` и `quicksort` на разных массивах. После исправления этой ошибки появилась другая: `shellsort` запускался после `quicksort` на том же массиве, что приводило к тестированию `shellsort` на отсортированном массиве (ошибка была найдена просмотром данных — количество перемещений было равно 0). Сейчас эти ошибки исправлены

## Литература

1. Sedgewick, R. (1986). *A new upper bound for Shellsort*. *Journal of Algorithms*, 7(2), 159–173. doi:10.1016/0196-6774(86)90001-5
2. Incerpi, J., & Sedgewick, R. (1985). *Improved upper bounds on shellsort*. *Journal of Computer and System Sciences*, 31(2), 210–224. doi:10.1016/0022-0000(85)90042-x
3. <https://neerc.ifmo.ru/wiki/index.php?title=%D0%91%D1%8B%D1%81%D1%82%D1%80%D0%B0%D1%8F%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0>
4. <https://ru.wikipedia.org/wiki/%D0%91%D1%8B%D1%81%D1%82%D1%80%D0%B0%D1%8F%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0>
5. <https://neerc.ifmo.ru/wiki/index.php?title=%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0%D0%A8%D0%B5%D0%BB%D0%BB%D0%B0>
6. <https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0%D0%A8%D0%B5%D0%BB%D0%BB%D0%B0>
7. Образец отчёта по заданию 1 (лежит в папке <https://github.com/abrikos110/polygon/tree/master/sorts-homework/docs> )