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

Необходимо реализовать два метода сортировки чисел типа long long int по невозростанию 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

Метод	swaps	comparisons	Данные	n
quicksort	9	41	1	10
	57	827	1	100
	654	12494	1	1000
	6356	170405	1	10000
	64239	2056552	1	100000
	641317	24416160	1	1000000
	6407475	284119238	1	10000000
	64102146	3224160722	1	100000000
	8	39	2	10
	109	852	2	100
	1141	13338	2	1000
	11199	169579	2	10000
	114312	2078720	2	100000
	1141792	24713681	2	1000000
	11407318	285371857	2	10000000
	114095860	3286031528	2	100000000
	12	43	3	10
	235	854	3	100
	3208	12637	3	1000
	39274	168141	3	10000
	472073	2059671	3	100000
	5529138	24182116	3	1000000
	62794698	290271002	3	10000000
	709419630	3238912381	3	100000000
	12	43	4	10
	244	834	4	100
	3143	13635	4	1000
	39548	165531	4	10000
	468976	2117128	4	100000
	5533074	24115015	4	1000000
	63052060	283929809	4	10000000
	709353760	3251918415	4	100000000

Метод	swaps	comparisons	Данные	n
shellsort	0	22	1	10
	0	327	1	100
	0	4550	1	1000
	0	60156	1	10000
	0	758531	1	100000
	0	9310065	1	1000000
	0	108851935	1	10000000
	0	1255334358	1	100000000
	25	38	2	10
	520	778	2	100
	7690	11390	2	1000
	80958	137708	2	10000
	984680	1662744	2	100000
	13933052	22320677	2	1000000
	132894650	237412758	2	10000000
	1587165582	2744265881	2	100000000
	12	30	3	10
	556	857	3	100
	20501	24668	3	1000
	361426	418226	3	10000
	4543629	5263125	3	100000
	53027578	61901585	3	1000000
	621445698	726817606	3	10000000
	7410299820	8624567414	3	100000000
	14	31	4	10
	754	1055	4	100
	19846	23977	4	1000
	363987	420819	4	10000
	4342734	5061977	4	100000
	52993720	61867157	4	1000000
	626201035	731571164	4	10000000
	7190319291	8404588984	4	100000000

Метод	avg. swaps	avg. comps.	n
quicksort	10,3	41,5	10
	161,3	841,8	100
	2036,5	13026,0	1000
	24094,3	168414,0	10000
	279900,0	2078017,8	100000
	3211330,3	24356743,0	1000000
	35915387,8	285922976,5	10000000
	399242849,0	3250255761,5	100000000
shellsort	17,0	30,3	10
	610,0	754,3	100
	16012,3	16146,3	1000
	268790,3	259227,3	10000
	3290347,7	3186594,3	100000
	39984783,3	38849871,0	1000000
	460180461,0	451163365,8	10000000
	5395928231,0	5257189159,3	100000000

В подсчёте средних значений количества обменов **не учитывались** результаты shellsort на отсортированном массиве, так как количество обменов равно 0. Серьёзных противоречий экспериментальных данных с теоретическими оценками сложности алгоритмов обнаружено не было

Графики

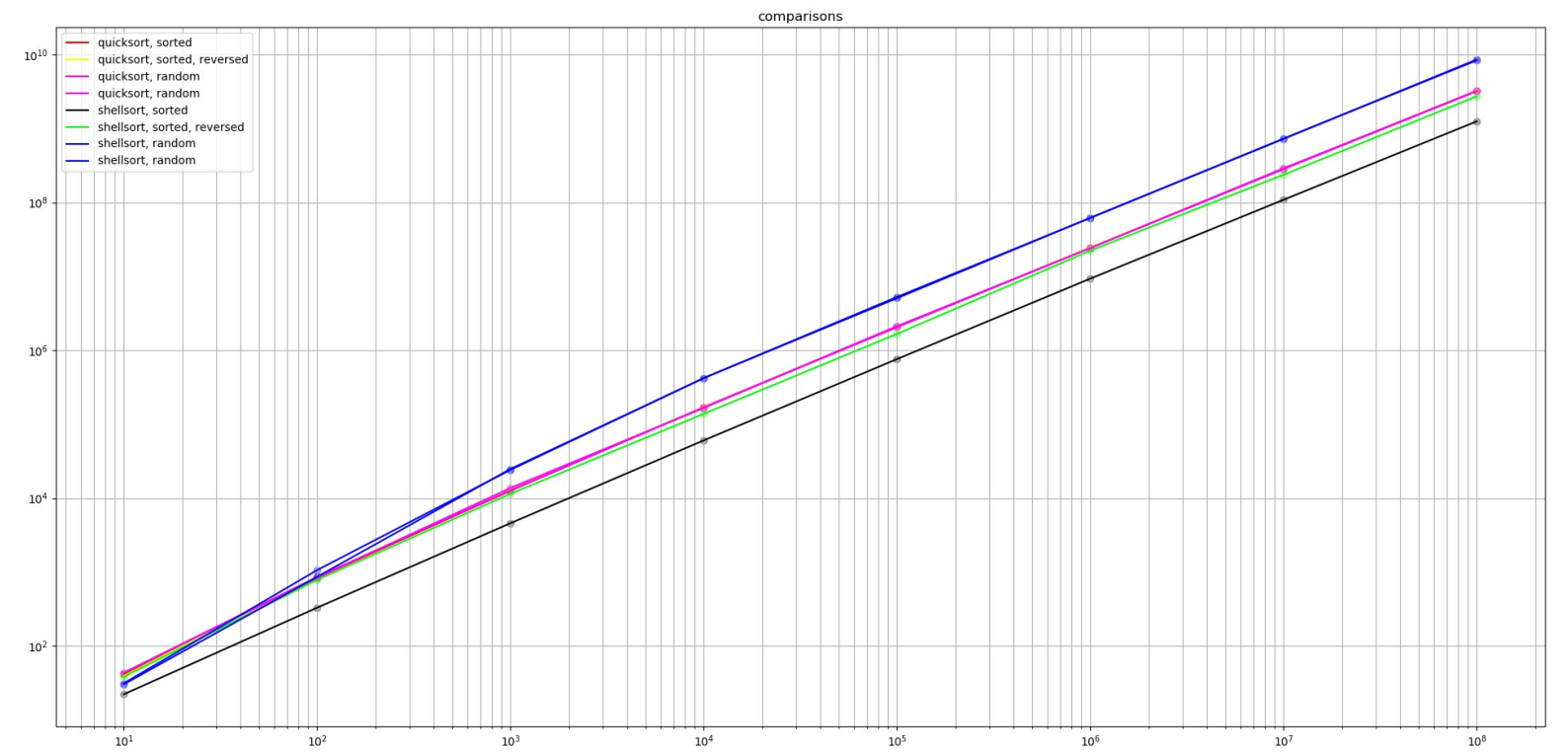
Для просмотра графиков необходим цвет. Графики строились в логарифмическом масштабе с помощью matplotlib

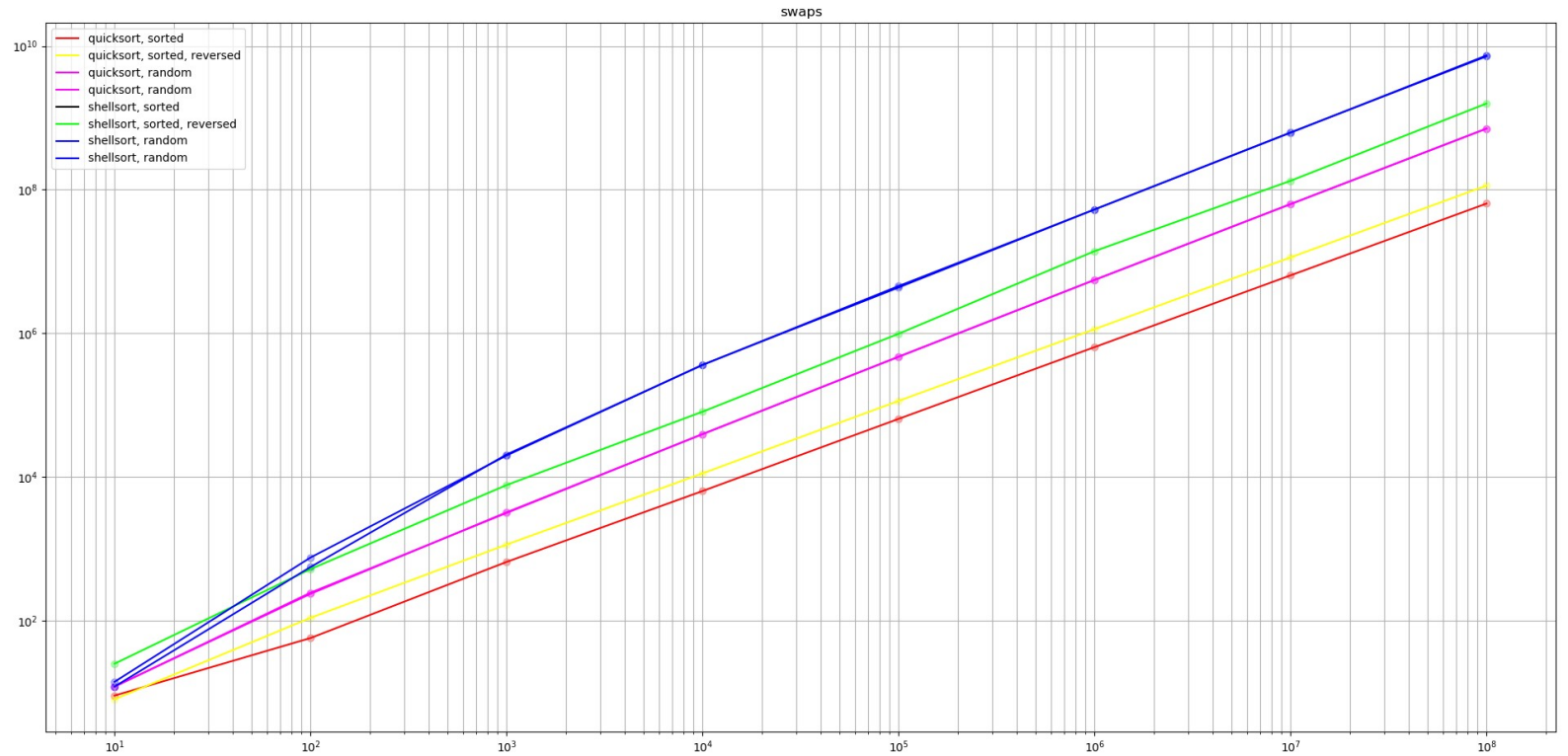
Количество обменов (swaps) shellsort на отсортированном массиве (shellsort, sorted) равно 0

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


comparisons
quicksort, sorted
quicksort, sorted, reversed
quicksort, random
quicksort, random
shellsort, sorted
shellsort, sorted, reversed
shellsort, random
shellsort, random
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}
10^{8}


swaps
quicksort, sorted
quicksort, sorted, reversed
quicksort, random
quicksort, random
shellsort, sorted
shellsort, sorted, reversed
shellsort, random
shellsort, random
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}
10^{8}

Структура программы и спецификации функций

some_funcs.h:

- long long int rand_lli(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 пямяти 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 (используется 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 <= i < p, то a[i] <= pivot, если p <= i < n, то a[i] >= 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_lli тестировалась построением графиков распределения, ошибки определялись на глаз. Проверялся знак возвращаемого значения rand_lli(1)

Функции swap_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, было найдено много ошибок (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