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	Обмены	Сравнения	Тип	Размер
quicksort	5	53	1	10
	44	930	1	100
	633	13291	1	1000
	6181	175515	1	10000
	62631	2124560	1	100000
	623174	26056792	1	1000000
	6224627	296354139	1	10000000
	62294608	3289584954	1	100000000
	11	52	2	10
	102	989	2	100
	1178	13738	2	1000
	11221	177169	2	10000
	112598	2141393	2	100000
	1121487	25442601	2	1000000
	11229452	301987393	2	10000000
	112284778	3309646130	2	100000000
	19	47	3	10
	228	992	3	100
	3055	13592	3	1000
	38855	175781	3	10000
	464747	2161663	3	100000
	5448197	25490206	3	1000000
	62214719	297312517	3	10000000
	703645950	3324110532	3	100000000
	15	55	4	10
	224	996	4	100
	3044	14496	4	1000
	38529	179216	4	10000
	466519	2144926	4	100000
	5450157	25377638	4	1000000
	62351922	295301216	4	10000000
	702836445	3336228955	4	100000000

Метод	Обмены	Сравнения	Тип	Размер
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
	18	34	3	10
	749	1048	3	100
	20445	24642	3	1000
	357807	414563	3	10000
	4509333	5228887	3	100000
	53328825	62202547	3	1000000
	624185108	729555898	3	10000000
	7232161631	8446422892	3	100000000
	16	36	4	10
	773	1069	4	100
	23147	27302	4	1000
	355233	412089	4	10000
	4471405	5190851	4	100000
	53114228	61987266	4	1000000
	620718103	726090029	4	10000000
	7602809429	8817071102	4	100000000

Метод	Средн. число сравн.	Средн. число обм.	Размер
quicksort	51,8	12,5	10
	976,8	149,5	100
	13779,3	1977,5	1000
	176920,3	23696,5	10000
	2143135,5	276623,8	100000
	25591809,3	3160753,8	1000000
	297738816,3	35505180,0	10000000
	3314892642,8	395265445,3	100000000
shellsort	32,5	19,7	10
	805,5	680,7	100
	16971,0	17094,0	1000
	256129,0	264666,0	10000
	3210253,3	3321806,0	100000
	38955138,8	40125368,3	1000000
	450477655,0	459265953,7	10000000
	5315773558,3	5474045547,3	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

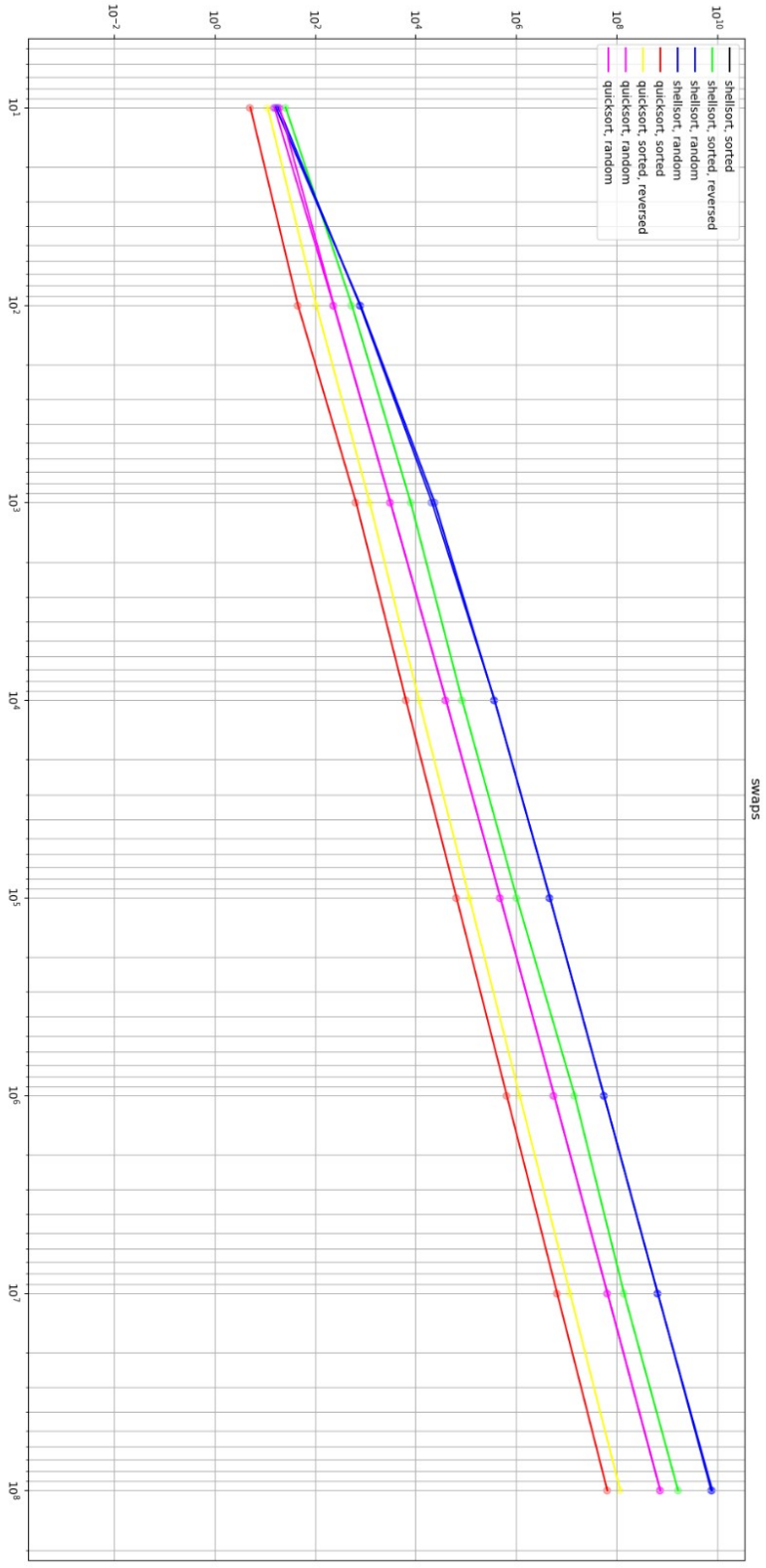
Количество обменов (swaps) shellsort на отсортированном массиве (shellsort, sorted) равно 0.

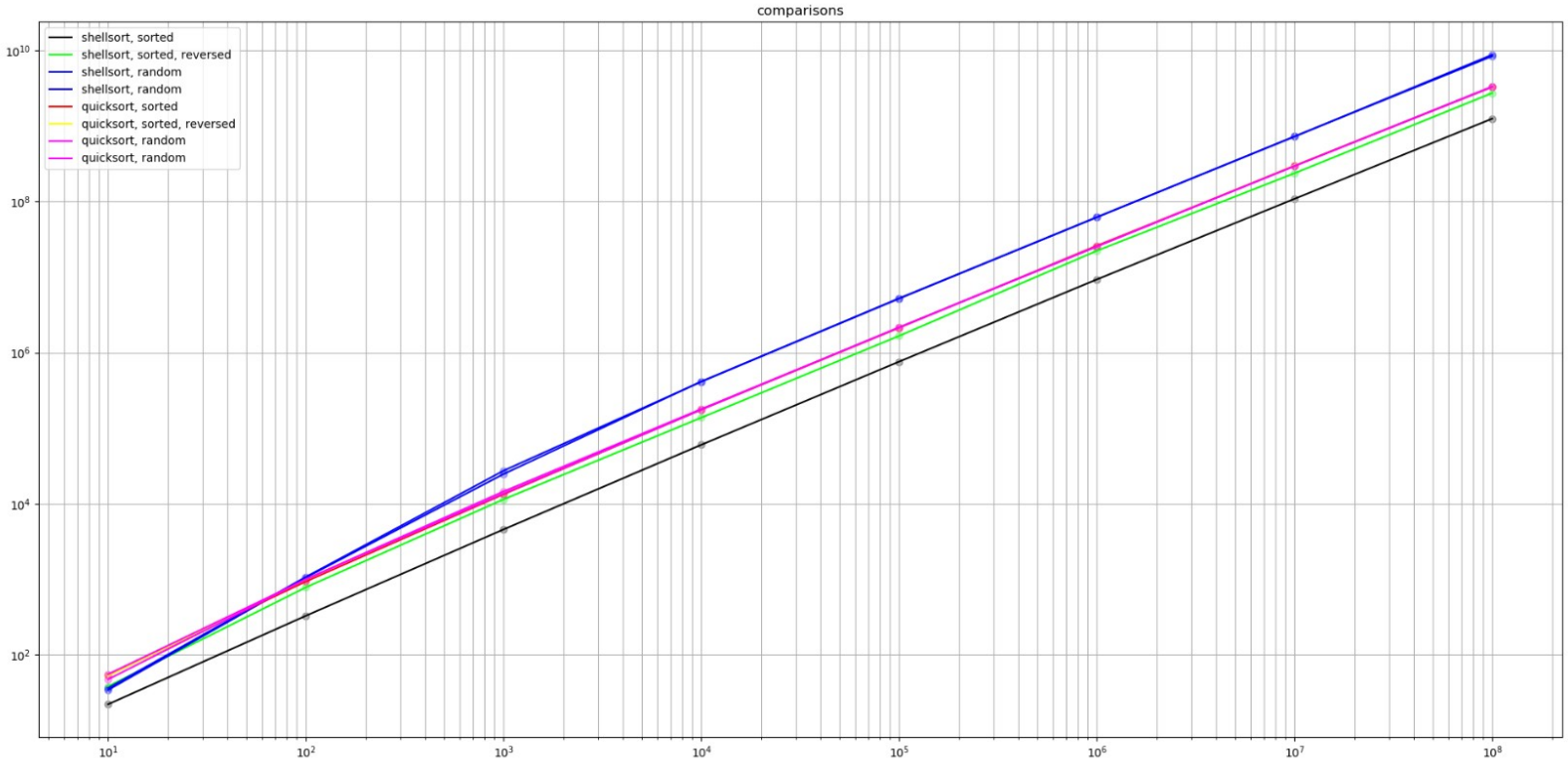
Все четыре графика сравнений (comparisons) quicksort очень близки друг к другу и накладываются друг на друга.

Легенда:

- чёрный цвет – shellsort, sorted
- зелёный – shellsort, sorted, reversed
- синий – shellsort, random

- красный – quicksort, sorted
- жёлтый – quicksort, sorted, reversed
- magenta – quicksort, random




comparisons
shellsort, sorted
shellsort, sorted, reversed
shellsort, random
shellsort, random
quicksort, sorted
quicksort, sorted, reversed
quicksort, random
quicksort, 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. Гарантируется 0 != p != n
- 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]. pivot должен быть не больше максимума в массиве и не меньше минимума

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, либо она возвращала значение, равное нулю или размеру массива, что приводило к возможности бесконечной рекурсии. Для решения этой проблемы в комментариях к этой функции написаны упрощённые короткие доказательства корректности работы функции (написание доказательств также помогало отлаживать код). Для гарантии 0 != p имеется заплатка, которая перемещает минимальный элемент в начало в таком случае

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