Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчет по заданию №1

«Методы сортировки»

Вариант 2 / 4 / 2 / 3

Выполнил:
студент 101 группы
Александров М. С.

Преподаватель:
Дудина И. А.

Москва
2021

Содержание

Постановка задачи

Требуется реализовать два метода сортировки массива чисел типа **long long int** и привести их экспериментальное сравнение с соответствующей теоретической оценкой. Числа упорядочиваются по невозрастанию модулей, т.е. при сравнении элементов не учитывается знак.

Методы сортировки:

- **метод простого выбора**
- **метод Шелла**

Результаты экспериментов

Теоретическая оценка

N - количество элементов в массиве

1. *Метод простого выбора.*

   (a) Сравнения. На i-ом шаге происходит $(i-1)$ сравнение, поэтому всего их $\sum_{i=1}^{N-1} i = \frac{N^2-N}{2}$.

   (b) Обмены. На каждом шаге происходит ровно один обмен, всего проходов $(N-1)$, поэтому всего будет совершено $(N-1)$ обменов элементов массива.

2. *Метод Шелла* - для этой сортировки существуют разные способы реализации, отличающиеся в последовательности длин промежутков d - на которых будут находиться сортируемые элементы исходного массива на каждом шаге алгоритма. Была выбрана последовательность Томаса Хиббарда, k-ый член которой равен $2^k - 1$. Тогда получим для нее оценку математического ожидания числа сравнений и обменов[1].

   (a) Сравнения.
   $kN^{5/4} + 0.15kN^{5/4}\log_2 N$, где $k \approx 1.07$. (сумма выражений, одно из которых не пронумеровано и находится между (15) и (16)Б а второе - (21)).

   (b) Обмены.
   $\frac{kN^{5/4}+0.15kN^{5/4}\log_2 N}{2}$, где $k \approx 1.07$ (в статье - число перенумераций, а оно в два раза больше числа обменов).

Сравнение результатов

N	**Параметр**	**Номер массива**				**Среднее значение**	**Теор. оценка**
		1 (Отсорт. массив)	2 (Отсорт. в обратном порядке)	3 (Случайной порядок)	4 (Случайный порядок)		
10	Сравнения	45	45	45	45	45	45
	Обмены	0	5	8	8	5.25	9
100	Сравнения	4950	4950	4950	4950	4950	4950
	Обмены	0	50	95	95	60	99
1000	Сравнения	499500	499500	499500	499500	499500	499500
	Обмены	0	500	995	996	622.75	999
10000	Сравнения	49995000	49995000	49995000	49995000	49995000	49995000
	Обмены	0	5000	9989	9986	6243.75	9999

Таблица 1: Результаты работы сортировки методом простого выбора

N	**Параметр**	**Номер массива**				**Среднее значение**	**Теор. оценка**
		1 (Отсорт. массив)	2 (Отсорт. в обратном порядке)	3 (Случайной порядок)	4 (Случайный порядок)		
10	Сравнения	19	25	29	26	24.75	28
	Обмены	0	11	13	9	8.25	14
100	Сравнения	480	614	752	775	655.25	675
	Обмены	0	192	325	351	217	337
1000	Сравнения	7987	10511	14594	14051	11785.75	15011
	Обмены	0	3424	7169	6613	4301.5	7505
10000	Сравнения	113631	144824	254659	242655	188942.25	320267
	Обмены	0	36778	146164	134225	79291.25	160133

Таблица 2: Результаты работы сортировки методом Шелла

Вывод, который можно сделать исходя из получившейся таблицы: сортировка методом Шелла проигрывает сортировке методом простого выбора по числа обменов, но выигрывает в несколько раз по числу сравнений. Из-за этого сортировка методом простого выбора не так эффективна на больших массивах, хоть и число обменов в ней не так велико.

Структура программы и спецификация функций

Описание основной структуры

Для удобного хранения данных была использована структура *test_sort*, которая содержит в себе поля:

1. `int len` - длина массива.
2. `int mode` - как отсортирован массив(1 - невозрастание модулей, 2 - неубывание модулей, 3, 4 - заполнен случайно).
3. `long long *arr_qsort` - массив, к которому будет применен qsort (встроенная функция).
4. `long long *arr_shell_sort` - массив, к которому будет применен shell_sort.
5. `long long *arr_selection_sort` - массив, к которому будет применен selection_sort.
6. `int swaps_shell_sort` - количество обменов при применении сортировки методом Шелла.
7. `int swaps_selection_sort` - количество обменов при применении сортировки методом простого выбора.
8. `int comparisons_shell_sort` - количество сравнений при применении сортировки методом Шелла.
9. `int comparisons_selection_sort` - количество сравнений при применении сортировки методом простого выбора.
10. `int ok_shell_sort` - корректность работы сортировки методом Шелла. 1 - корректно, 0 - некорректно.
11. `int ok_selection_sort` - корректность работы сортировки методом простого выбора. 1 - корректно, 0 - некорректно.

Описание функций программы

1. `long long rand_ll(void)` - возвращает случайное число типа *long long*.
2. `void swap(long long *a, long long *b, int *swaps)` - принимает указатели на 2 переменные и указатель на счетчик обменов, выполняет обмен переменных и увеличивает счетчик.
3. `int cmp(const long long *a, const long long *b, int *comparisons)` - принимает указатели на 2 переменные и указатель на счетчик сравнений, увеличивает счетчик и сравнивает 2 переменные: возвращает 1, если $|a| > |b|$ или $|a| = |b|, b > a$, иначе - 0.
4. `int cmp_for_qsort1(const long long *a, const long long *b)` - то же, что и прошлая функция, но не увеличивает счетчик. используется для генерации отсортированного массива.

5. `int cmp_for_qsort2(const long long *a, const long long *b)` - то же, что и прошлая функция, но возвращает 0 и 1 в других случаях, используется для генерации массива, отсортированного в обратном порядке.

6. `void gen_arr(int n, int mode, long long *arr1, long long *arr2, long long *arr3)` - принимает размер массива, mode для типа генерации и указатели на 3 массива. В зависимости от полученного значения mode генерирует либо отсортированный массив, либо отсортированный в обратном порядке массив, либо массив со случайными значениями. Потом копирует из одного массива в два других и в результате получаем 3 одинаковых массива нужного типа.

7. `void print_arr(long long *a, int n)` - принимает указатель на массив и его длину. После печатает его.

8. `void selection_sort(long long *a, int n, int *comp, int *swaps)` - принимает указатель на массив, его размер и две переменные, куда записывает количество сравнений и обменов ,произошедших во время сортировки. Выполняет сортировку методом простого выбора.

9. `void shell_sort(long long *a, int n, int *comp, int *swaps)` - принимает указатель на массив, его размер и две переменные, куда записывает количество сравнений и обменов ,произошедших во время сортировки. Выполняет сортировку методом Шелла.

10. `int check(long long *a, long long *b, int n)` - принимает указатель на 2 массива, один из которых отсортирован qsort'ом и их размер. Проверяет, правильно ли прошла сортировка путем сравнения с массивом, отстортированным qsort'ом.

11. `void test(test_sort *t)` - принимает указатель на структуру. Инициализирует ее переменные, генерирует массивы нужного типа, сортирует их разными способами, а потом заполняет переменные $ok\_shell\_sort, ok\_selection\_sort$ после проверки.

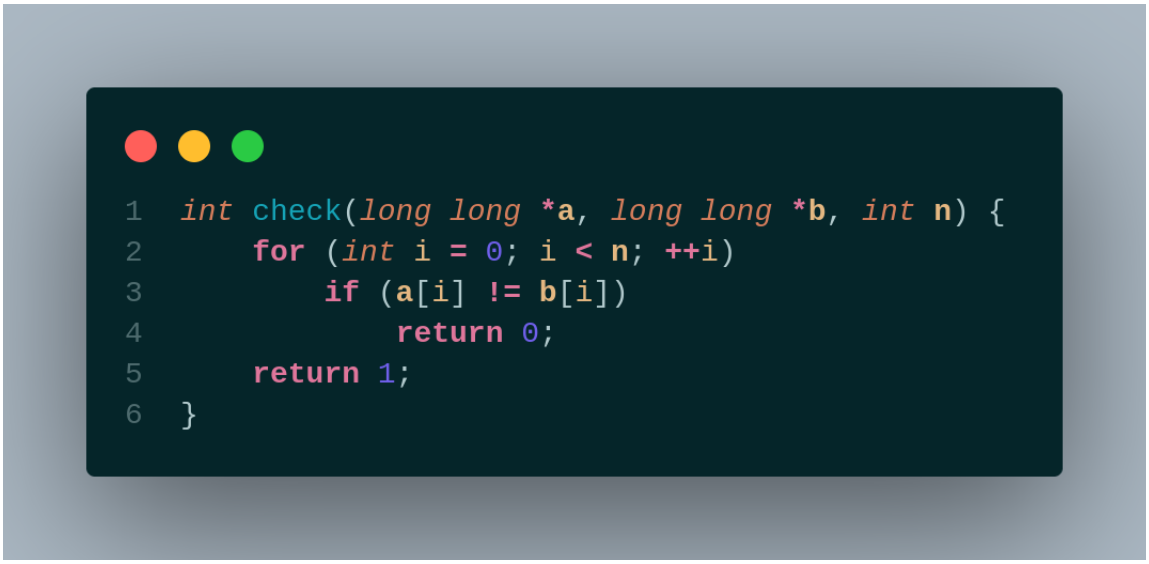
12. `void check_test(test_sort *t, int n)` - принимает указатель на структуру и ее номер. Функция выводит информацию, содержащуюся в структуре: тип массива в начале, количество сравнений и обменов для каждого типа сортировки и правильно ли они прошли.

13. `void free_test(test_sort *t)` - принимает указатель на структуру и освобождает связанную с ней динамическую память.

Отладка программы, тестирование функций

Когда создается массив, также создаются еще 2 его копии, чтобы после быть использованными в сортировке методом простого выбора, в сортировке методом Шелла и во встроенной сортировке qsort - *эталон*, с которым потом будут сравниваться 2 других массива.

Отладка функций происходила сравниванием отсортированного (методом простого выбора или методом Шелла) массива с основным массивом (отсортированным qsort). В результате было найдено несколько ошибок, описанных в следующем разделе. После, сортировки были протестированы на 1000 массивах длиной 1000 элементов (эти массивы были заполнены случайно), и ошибок выявлено не было.



```
1  int check(long long *a, long long *b, int n) {
2      for (int i = 0; i < n; ++i)
3          if (a[i] != b[i])
4              return 0;
5      return 1;
6  }
```

Рис. 1: Функция *check*, выполняющая проверку эквивалентности массивов.

Анализ допущенных ошибок

Функция, возвращающая модуль числа

В программе была использова функция *long long llabs(long long)*, которая при подаче ей в качестве аргумента LLONG_MIN (минимальное число типа long long) возвращает LLONG_MIN, т.е. отрицательное значения (модуль этого числа не помещается в long long) И тогда число LLONG_MIN, которое должно лежать в начале отсортированного массива, попадает в его конец, так как его "модуль"меньше любого другого модуля. Этот случай отдельно отслеживался в компараторах.

Генератор случайных чисел типа long long

Ошибкой было то, что в программе перемножались 4 числа полученных с помощью функции rand() и не было учтено, что результат ограничен типом int, так как rand возвращает число типа int и без явного приведения типа к long long может возникнуть переполнение и мы не сможем получит все числа из множества чисел long long.

Список литературы

[1] А. А. Паперпов, Г. В. Стасевич. Об одном методе упорядочивания информации в запоминающих устройствах цифровых машин, *Пробл. передачи информ.*, 1965, том 1, выпуск 3, 81-98.