Содержание

Постановка задачи

В процессе работы были выполнены следующие задачи:

- Реализация на языке программирования C двух функций, которые решают задачу упорядочивания массива из элементов типа long long int по невозрастанию модулей. В основе одной функции лежит метод простого выбора, в основе другой - метод Шелла.
- Реализация функций генерации массива для сортировки, создающей массив, в котором:
  - элементы уже упорядочены по невозрастанию модулей (прямой порядок)
  - элементы упорядочены по неубыванию модулей (обратный порядок)
  - расстановка элементов случайна
- Экспериментальное сравнение двух методов сортировки: сравнение количества перемещений и сравнений элементов массива при разных сортировках.
- Проведение теоретической оценки каждого из методов.
- Анализ ошибок, допущенных во время работы.

Результаты экспериментов

Метод простого выбора

Реализация сортировки простым выбором в условиях данной задачи заключается в следующем: находится максимальный по модулю элемент массива, и он переставляется с первым элементом. Далее аналогичная процедура применяется ко всем элементам, кроме первого. И так далее.

- Количество сравнений элементов ([1] 16-17)

  Замечательное свойство этого метода заключается в сокращении списка от просмотра к просмотру. На каждом последующем просмотре выполняется на одно сравнение меньше, так что последовательно выполняется (n - 1), (n - 2), (n - 3), …, 2, 1 сравнений. Общее число сравнений равно произведению среднего числа сравнений за один просмотр на число просмотров, т. е. $\frac{n(n-1)}{2}$ и совершенно не зависит от состояния данных.

- Следовательно сложность алгоритма – $O(n^2)$

- Количество перестановок элементов:

  - Наилучший случай (массив изначально упорядочен) – 0, так как все элементы находятся в нужном порядке

  - Наихудший случай (массив изначально упорядочен в обратном порядке) – $\frac{n}{2}$, так как каждая пара элементов находится в неверном порядке, и при обмене на нужные места будут вставать оба элемента

  - Среднее значение – $n(\ln(n) + g)$, где $g \approx 0.58$ – константа Эйлера ([1] 100-101)

n	**Параметр**	**Номер сгенерированного массива**				**Среднее значение**
		1	2	3	4	
10	Сравнения	45	45	45	45	45,00
	Перемещения	0	5	8	7	4,33
100	Сравнения	4950	4950	4950	4950	4950,00
	Перемещения	0	50	95	93	48,33
1000	Сравнения	499500	499500	499500	499500	499500,00
	Перемещения	0	500	992	993	497,33
10000	Сравнения	49995000	49995000	49995000	49995000	49995000,00
	Перемещения	0	5000	9992	9988	4997,33

Таблица 1: Результаты работы сортировки методом простого выбора

Метод Шелла

Реализация сортировки простым выбором в условиях данной задачи заключается в следующем: пусть k — целое от 1 до n/2. Независимо друг от друга упорядочиваются подпоследовательности из элементов, отстоящих друг от друга на k позиций: Xi, Xi + k, Xi + 2k, Xi + 3k, ... (i = 1, 2, ..., k) Затем k уменьшается и процесс повторяется заново. Последний шаг обязательно должен быть выполнен при k = 1. Значение k можно менять разными способами, один из них таков: вначале k равно (целой части от) n/2, а затем k каждый раз уменьшается вдвое.

- Сложность алгоритма (количество сравнений элементов) ([2] 108-110)
  - Наилучший случай – $O(nlog^2n)$
  - Наихудший случай – $O(n^2)$, так как количество элементов в подпоследовательности Xi зависит линейно от n. А в свою очередь k принимает значения от $\frac{n}{2}$ до 1, значит количество рассмотренных подпоследовательностей тоже зависит от n. Следовательно, наихудшая сложность – $O(n^2)$.
  - Среднее значение – $O\left(n^{1.5+\frac{s}{2}}\right)$, где $s = \frac{1}{2^t-1}$
- Количество перемещений элементов ([2] 105-108)
  - Наилучший случай (массив изначально упорядочен) – 0
  - Наихудший случай – $\frac{1}{8}n^2$
  - Среднее значение – $f(n,h)$, где h – упорядоченная перестановка, $q = \frac{n}{h}$ и $r = n\ mod\ h$

$$f(n,h) = \frac{2^{2q-1}q!\,q!}{(2q+1)!}\left(\binom{h}{2}q(q+1) + \binom{r}{2}(q+1) - 1/2\binom{h-r}{2}\right)$$

n	Параметр	Номер сгенерированного массива				Среднее значение
		1	2	3	4	
10	Сравнения	22	27	29	25	26,00
	Перемещения	0	13	12	7	8,33
100	Сравнения	503	668	912	917	694,33
	Перемещения	0	260	457	467	239,00
1000	Сравнения	8006	11716	15086	15092	11602,67
	Перемещения	0	4700	7582	7609	4094,00
10000	Сравнения	120005	172578	276437	259365	189673,33
	Перемещения	0	62560	161425	144450	74661,67

Таблица 2: Результаты работы сортировки методом Шелла

Вывод

Эксперимент показал, что сложность сортировки методом Шелла меньше сложности сортировки методом простого выбора, значения совпадают только в наихудшем случае для сортировки методом Шелла. Что касается количества перемещений элементов, тут все не так однозначно, однако среднее значение все же меньше у сортировки методом простого выбора. Но несмотря на это, сортировка методом Шелла более эффективна.

Структура программы и спецификация функций

Глобальные переменные

- **`static int cnt_selection_sort_cmp`** – количество сравнений элементов при сортировке методом простого выбора
- **`static int cnt_selection_sort_swap`** – количество перемещений элементов при сортировке методом простого выбора
- **`static int cnt_shell_sort_cmp`** – количество сравнений элементов при сортировке методом Шелла
- **`static int cnt_shell_sort_swap`** – количество перемещений элементов при сортировке методом Шелла

Функции

- **`void swap(long long *a, long long *b)`** – функция, реализующая обмен значений двух переменных. Принимает на вход два указателя на числа a и b, не имеет возвращаемого значения.
- **`void selection_sort(long long *arr, int n)`** – функция, реализующая сортировку методом простого выбора по невозрастанию модулей. Принимает на вход указатель на начало сортируемого массива arr и количество элементов данного массива n, не имеет возвращаемого значения.
- **`void shell_sort(long long* arr, int n)`** – функция, реализующая сортировку методом Шелла по невозрастанию модулей. Принимает на вход указатель на начало сортируемого массива arr и количество элементов данного массива n, не имеет возвращаемого значения.
- **`int straight_ordered_compare(const void* a, const void* b)`** – функция, реализующая сравнение по невозрастанию модулей (в прямом порядке) для функции сортировки qsort из стандартной библиотеки языка C. Необходима для генерации массива, элементы которого упорядочены согласно условию задачи. Функция принимает на вход два указателя на числа a и b. Возвращает:
  - значение 1, если модуль числа a меньше модуля числа b
  - значение 0, если модули чисел a и b равны
  - значение -1, если модуль числа a больше модуля числа b
- **`int reverse_ordered_compare(const void* a, const void* b)`** – функция, реализующая сравнение по неубыванию модулей (в обратном порядке) для функции сортировки qsort из стандартной библиотеки языка C. Необходима для генерации массива, элементы которого упорядочены в обратном порядке согласно условию задачи. Функция принимает на вход два

указателя на числа a и b. Возвращает:

- значение 1, если модуль числа a больше модуля числа b
- значение 0, если модули чисел a и b равны
- значение -1, если модуль числа a меньше модуля числа b

- **`void straight_ordered_array_generating(long long* arr, int n)`** – функция, генерирующая массив, элементы которого упорядочены по невозрастанию модулей (в прямом порядке). Принимает на вход указатель на начало генерируемого массива arr и количество элементов данного массива n, не имеет возвращаемого значения.

- **`void reverse_ordered_array_generating(long long* arr, int n)`** функция, генерирующая массив, элементы которого упорядочены по неубыванию модулей (в обратном порядке). Принимает на вход указатель на начало генерируемого массива arr и количество элементов данного массива n, не имеет возвращаемого значения.

- **`void random_array_generating(long long* arr, int n)`** – функция, генерирующая массив, элементы которого неупорядочены. Принимает на вход указатель на начало генерируемого массива arr и количество элементов данного массива n, не имеет возвращаемого значения.

- **`void print_array(long long* arr, int n)`** – функция вывода значений массива. Принимает на вход указатель на начало массива arr и количество элементов данного массива n, не имеет возвращаемого значения.

- **`double find_average_value(int* arr, int n)`** – функция вычисления среднего значения элементов массива. Принимает на вход указатель на начало массива arr и количество элементов данного массива n, возвращает среднее значение элементов массива.

- **`void free_up_memory_of_matrix(int** arr, int n)`** – функция освобождения памяти, выделенной под двумерный массив. Принимает на вход указатель на начало массива arr и количество строк данного массива n, не имеет возвращаемого значения.

- **`void print_header(char* name)`** – функция вывода заголовков таблицы результатов сравнения. Принимает на вход строку с названием метода сортировки, не имеет возвращаемого значения.

- **`void print_footer()`** – функция вывода последней строки таблицы результатов сравнения. Не входных данных, не имеет возвращаемого значения.

Структура программы

Программа функционирует в двух режимах. В начале работы требуется выбрать один из них:

Compare select sort and shell sort by yourself

В этом режиме пользователю предоставляется возможность лично сравнить два метода сортировки и убедиться в корректности работы программы. Программа потребует ввести следующие параметры для сравнения:

- Количество элементов в генерируемом массиве
- Тип упорядочивания элементов в генерируемом массиве: прямой, обратный, случайный
- Хочет ли пользователь вывести сгенерированный и отсортированные массивы

Программа выделит память для двух генерируемых массивов (один массив будет отсортирован методом простого выбора, а другой – методом Шелла), одинаково заполнит их элементами, упорядоченными согласно выбору пользователя. Отсортирует двумя способами, параллельно ведя подсчет количества сравнений значений элементов и их перемещений. Далее по желанию пользователя выведет сгенерированный массив и два отсортированных разными способами. После программа выведет таблицу с результатами сравнений значений элементов и их перемещений. Далее программа освободит память, выделенную под два сгенерированных массива, после чего успешно закончит свое выполнение.

View the result of a full comparison

В этом режиме программа сразу выведет две таблицы пологого сравнения сортировок методом простого выбора и методом Шелла.

Программа выделит память для двух таблиц, в которых будут храниться значения параметров (количество сравнений значений элементов и их перемещений при сортировке), и для двух массивов, в которых будут храниться средние значения параметров. Далее программа будет генерировать массивы разных длин и с разной упорядоченностью элементов: в прямом порядке, обратном и случайном, параллельно ведя подсчет количества сравнений значений элементов и их перемещений и заполняя таблицы значениями параметров и массивы - средними значениями параметров. Далее программа выведет две таблицы полного сравнения методов сортировок, освободит память, выделенную подо все массивы, и успешно закончит свое выполнение.

Отладка программы, тестирование функций

На основе приведенных тестов была проверена корректность работы программы, найдены ошибки при генерации массива. После их устранения было проведено повторное тестирование, результаты которого подтвердили правильность работы программы.

Сортировка методом простого выбора

	arr[0]	**arr[1]**	**arr[2]**	**arr[3]**	**arr[4]**	**arr[5]**
Ввод	-2097024362	-2085528704	1883940024	1520443392	1181503488	390061584
Вывод	-2097024362	-2085528704	1883940024	1520443392	1181503488	390061584

Таблица 3: Тест №1 (в генерируемом массиве элементы упорядочены)

	arr[0]	**arr[1]**	**arr[2]**	**arr[3]**	**arr[4]**	**arr[5]**
Ввод	-175792224	929069568	-950310848	-1347756824	1395662520	1821413120
Вывод	1821413120	1395662520	-1347756824	-950310848	929069568	-175792224

Таблица 4: Тест №2 (в генерируемом массиве элементы упорядочены в обратном порядке)

	arr[0]	**arr[1]**	**arr[2]**	**arr[3]**	**arr[4]**	**arr[5]**
Ввод	-445552768	1622093152	-435601372	-1149299620	2101800184	1737483712
Вывод	2101800184	1737483712	1622093152	-1149299620	-445552768	-435601372

Таблица 5: Тест №3 (в генерируемом массиве элементы неупорядочены)

	arr[0]	**arr[1]**	**arr[2]**	**arr[3]**	**arr[4]**	**arr[5]**
Ввод	-611220664	-1681091076	36596640	250786816	1024169308	1165672110
Вывод	-1681091076	1165672110	1024169308	-611220664	250786816	36596640

Таблица 6: Тест №4 (в генерируемом массиве элементы неупорядочены)

Сортировка методом Шелла

	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]
Ввод	1984261440	-1895874040	1804025546	-924202539	557662617	486915336
Вывод	1984261440	-1895874040	1804025546	-924202539	557662617	486915336

Таблица 7: Тест №1 (в генерируемом массиве элементы упорядочены)

	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]
Ввод	22420120	161165952	-821832184	-921450688	-1136101488	1804312340
Вывод	1804312340	-1136101488	-921450688	-821832184	161165952	22420120

Таблица 8: Тест №2 (в генерируемом массиве элементы упорядочены в обратном порядке)

	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]
Ввод	745733056	554581652	-1382680832	255998898	762489440	-576903020
Вывод	-1382680832	762489440	745733056	-576903020	554581652	255998898

Таблица 9: Тест №3 (в генерируемом массиве элементы неупорядочены)

	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]
Ввод	-1010899310	-2064547984	314161792	81696724	-895362968	1726387200
Вывод	-2064547984	1726387200	-1010899310	-895362968	314161792	81696724

Таблица 10: Тест №4 (в генерируемом массиве элементы неупорядочены)

Анализ допущенных ошибок

В процессе работы были допущены ошибки, имеющие отношения только к генерации массивов и выводу таблиц с результатами сравнения сортировок. В реализации других функций ошибок допущено не было. В ходе работы были оптимизированы некоторые алгоритмы, связанные с параллельным заполнением таблиц.

Список литературы

[1]. Лорин Г. Сортировка и системы сортировки. — М.: Наука, 1983.
[2]. Кнут Д. Искусство программирования для ЭВМ. Т.3. — М.: Мир, 1978.