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

Отчет по заданию №1

«Методы сортировки»

Вариант 1 / 4 / 1 + 3

Выполнил:
студент 103 группы
Цыбанов И. А.

Преподаватель:
Кузьменкова Е. А.

Москва
2023

Содержание

1. Постановка задачи

Требуется реализовать методы сортировки "пузырьком" и Шелла и провести их экспериментальное сравнение. Для сравнения методов сортировки используются массивы целых чисел (тип `int`). Один массив уже отсортирован, второй отсортирован в обратном порядке, другие 2 массива сгенерированы случайным образом. Сортировка производится по невозрастанию модулей элементов массива.

2. Результаты экспериментов

Теоретические оценки

Далее будем считать, что N – количество элементов сортируемого массива.

Метод «пузырька» ([2], с. 79-80)

Число сравнений: $\frac{1}{2}(N^2 - N)$

Оценка количества перемещений:

- Минимальное значение: 0
- Среднее значение: $\frac{1}{4}(N^2 - N)$
- Максимальное значение: $\frac{1}{2}(N^2 - N)$

Метод Шелла

Анализ метода Шелла представляет собой не до конца решённую математическую задачу. До сих пор неизвестно, какая последовательность смещений является наиболее эффективной. Обычно применяется следующая последовательность: $2^k - 1, \ldots, 15, 7, 3, 1$, где $k = [\log_2 N] - 1$ ([2], с. 82). Она же использована и в моей реализации. Для такой последовательности среднее число перемещений: $1.55N^{\frac{5}{4}} - 4.48N + O(N^{\frac{3}{4}})$ при $N > 100$ ([1], с. 97). В общем случае среднее число перемещений пропорционально $N^{\frac{6}{5}}$ ([2], с. 82). Оценим число сравнений. Первый вложенный цикл содержит $[\log_2 N] - 1$ шагов в силу выбранной последовательности смещений, т.е. можно сказать, что число шагов пропорционально $\log_2 N$. Число шагов во втором вложенном цикле пропорционально N, а в третьем - так же, как и в первом, $\log_2 N$, т.к. при итерации значение счётчика увеличивается на значение смещения. Значит, общее число сравнений можно оценить как $O(N \log_2^2 N)$, т.к. на каждом шаге производится сравнение.

Экспериментальные значения

Далее в таблицах 1-4 номера массивов соответствуют массивам со следующими видами расстановки элементов:

- Массив 1: уже отсортированный массив
- Массив 2: массив, отсортированный в обратном порядке
- Массивы 3, 4: массивы, сгенерированные случайным образом

	Номер сгенерированного массива				**Среднее**
n	**1**	**2**	**3**	**4**	**значение**
10	45	45	45	45	45
100	4950	4950	4950	4950	4950
1000	499500	499500	499500	499500	499500
10000	49995000	49995000	49995000	49995000	49995000

Таблица 1: Количество сравнений при сортировке методом «пузырька»

	Номер сгенерированного массива				Среднее
n	**1**	**2**	**3**	**4**	**значение**
10	0	45	24	19	22
100	0	4950	2469	2579	2499,5
1000	0	499500	243883	248280	247915,75
10000	0	49995000	25078236	24597058	24917573,5

Таблица 2: Количество перемещений при сортировке методом «пузырька»

	Номер сгенерированного массива				Среднее
n	**1**	**2**	**3**	**4**	**значение**
10	70	70	70	70	70
100	7919	7919	7919	7919	7919
1000	798131	798131	798131	798131	798131
10000	98176009	98176009	98176009	98176009	98176009

Таблица 3: Количество сравнений при сортировке методом Шелла

	Номер сгенерированного массива				Среднее
n	**1**	**2**	**3**	**4**	**значение**
10	0	13	12	11	9
100	0	260	451	407	279,5
1000	0	4700	7823	7828	5087,75
10000	0	62560	145062	151744	89841,5

Таблица 4: Количество перемещений при сортировке методом Шелла

Выводы

В методе сортировки "пузырьком"количество сравнений в точности соответствует теоретическим значеням, а количество перемещений достаточно близко к теоретической оценке среднего значения (отклонение составляет 1-3%).

В методе сортировки Шелла количество перемещений отклоняется от теоретической оценки среднего значения не слишком сильно (10-15%) при достаточно больших N ($N > 100$). При меньших значениях N количество перемещений пропорционально $N^{\frac{6}{5}}$, что совпадает с теоретической оценкой среднего значения. Количество сравнений имеет тот же порядок, что и теоретическая оценка, т.е. пропорционально $O(N \log_2^2 N)$.

Сортировка методом Шелла производит большее количество сравнений, чем сортировка "пузырьком но при сортировке методом Шелла производится значительно меньше перемещений. Несмотря на то, что общее число операций в

обоих методах сортировки сравнимо, сортировка методом Шелла является более эффективной, так как перемещение элементов – намного более затратная операция, чем сравнение.

3. Структура программы и спецификация функций

Ниже приведён полный список функций и их характеристика.

- `void generate_random_array(int *a, int *b, int *c, int n)`

  Генерируется массив псевдослучайных чисел (при помощи функций `rand()` и `srand()` стандартной библиотеки `C`), содержащий `n` элементов. Этот массив копируется в массивы `a`, `b`, `c`. Все элементы сгенерированного массива лежат в диапазоне `[-MAX_EL/2, MAX_EL/2]` (`MAX_EL` объявлен при помощи директивы `define` и равен 10^8).

- `void swap(int *a, int i, int j, int *swap_counter, int *cmp_counter)`

  Элементы массива `a` на позициях под номерами `i` и `j` сравниваются. Если позиции не равны и требуется поменять элементы местами, производится перемещение. Увеличивается на 1 счётчик сравнений `cmp_counter` и, если было произведено перемещение элементов, увеличивается на 1 счётчик перемещений `swap_counter`.

- `void bubble_sort(int *a, int n)`

  Производится сортировка массива `a`, содержащего `n` элементов, методом сортировки "пузырьком". Выводится на экран количество совершенных в процессе сортировки сравнений и перемещений элементов.

- `void shell_sort(int *a, int n)`

  Функция аналогична предыдущей, за исключением метода сортировки, – используется метод Шелла.

- `int check_array(int *src, int *res, int n)`

  Функции передаются массивы `src` и `dst`, содержащие `n` элементов. Если массив `res` является отсортированным массивом `src`, возвращается значение 1. Иначе - 0.

- `void print_array(int *arr, int n)`

  На экран выводятся первые `n` элементов массива `arr`.

- `void test_sort(int *a, int *b, int *c, int n)`

  На вход подаются одинаковые массивы `a`, `b`, `c`, содержащие `n` элементов. Первый массив сортируется методом «пузырька», второй – методом Шелла. При помощи третьего массива проверяется корректность сотировок. В случае корректного результата на экран выводится соответствующее сообщение. В случае ошибки на экран выводятся все элементы неверно отсортированного массива (при помощи функции `print_array()`) и сообщение об ошибке.

4. Отладка программы, тестирование функций

На начальном этапе тестирование функций сортировки производилось при помощи сортировки случайно сгенерированных массивов небольшой длины (`n < 20`) и вывода результата на экран.

Затем использовалась функция `test_sort`. Её использование позволило тестировать функции сортировки при больших значениях `n`, для которых слишком трудозатратно проверять результат "вручную".

При использовании этой функции было проведено автоматическое тестирование на 1000 случайно сгенерированных массивах, состоящих из 1000 элементов. Корректный результат на столь большом объёме данных позволяет с уверенностью говорить о корректности реализации сортировок.

Для выявления ошибок работы с динамической памятью использовался инструмент `valgrind`.

5. Анализ допущенных ошибок

В ходе выполнения задания были допущены следующие ошибки:

- Неверно реализован 3-ий вложенный цикл в методе сортировки Шелла. Из-за того, что счётчик `j` двигался не к началу массива, а к концу, массив мог быть отсортирован неверно, т.к. сравнение с первым элементом не производилось уже после 1-ой итерации и далее "окно" элементов, с которыми не будет произведено сравнения, только росло. Следовало оформить 3-ий вложенный цикл следующим образом:
  `for (int j = i - step; j >= 0; j -= step)` вместо
  `for (int j = i + step; j < n; j += step)`.

- Некорректная работа с динамическим массивом в функции `test_sort()`. В цикле, обнуляющем массив `check`, и в цикле, проверяющем массив `check` на наличие ненулевых элементов, счётчик `i` мог выходить за пределы размера массива `check`. Следовало заменить условие в циклах с `i < n` на `i < MAX_EL`.

- Не производилось тестирование функций сортировки на уже отсортированном массиве и массиве, отсортированном в обратном порядке. Соответствующие проверки были добавлены.

Список литературы

[1] Кнут Д. Искусство программирования для ЭВМ. Том 3. – М.: Мир, 1978.

[2] Вирт Н. Алгоритмы и структуры данных. – М.: Мир, 2010.