МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. М.В. ЛОМОНОСОВА

Факультет вычислительной математики и кибернетики

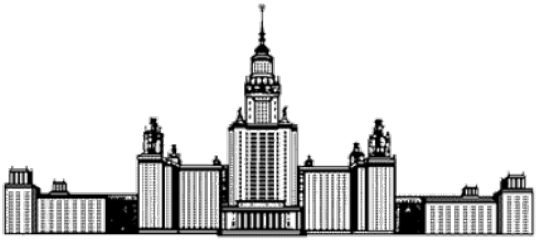
Отчет по заданию № 1

«Методы сортировки»

Вариант 3 3 2 4

Исполнитель:
Студент гр. 106
Кондрашов Даниил Сергеевич

Преподаватели:
Корухова Людмила Сергеевна
Манушин Дмитрий Валерьевич



Москва, 2024

Содержание

1 Постановка задачи

1. Требуется реализовать два метода сортировки одномерного массива чисел типа *double* (элементы упорядочиваются по неубыванию модулей):
   1 – Сортировка методом простого выбора
   2 – Быстрая сортировка, основанная на рекурсивной реализации

2. Провести их экспериментальное сравнение, и при реализации каждого метода вычислить число сравнений элементов и число перемещений, занести результаты в таблицу, посчитать среднее, основываясь на трех случаях заполнения массивов:
   1 – элементы упорядочены,
   2 – элементы упорядочены в обратном порядке,
   3 – расстановка элементов случайна

   (4 массив был убран из-за ненадобности, потому что каждое значение в таблице - это среднее значение, полученное после 10000 прогонов различных массивов одной длины, именно с этим связано появление дробных чисел в таблице)

3. Сравнить асимптотическую сложность алгоритмов.

4. На основе полученных результатов сделать вывод.

Кол-во элементов	**Номер сортировки**	**1**	**2**	**3**	**Среднее значение**
10	Сравнения				
	Перемещения				
...	Сравнения				
	Перемещения				

2 Результаты экспериментов

В результате проведенных экспериментов была подтверждена асимптотическая сложность алгоритмов: $O(n^2)$ для сортировки выбором и $O(n \cdot \log_2 n)$ для быстрой сорттировки.

Selection sort

N	Номер сортировки	1	2	3	Среднее значение
10	Сравнения	45,0	45,0	45,0	45,0
	Перемещения	6,8	6,8	7,0	6,9
50	Сравнения	1 225,0	1 225,0	1 225,0	1 225,0
	Перемещения	45,5	45,2	45,7	45,4
100	Сравнения	4 950,0	4 950,0	4 950,0	4 950,0
	Перемещения	94,9	94,6	94,5	94,7
500	Сравнения	124 750,0	124 750,0	124 750,0	124 750,0
	Перемещения	492,4	493,1	493,7	493,1
1000	Сравнения	499 500,0	499 500,0	499 500,0	499 500,0
	Перемещения	992,1	992,4	992,3	992,3
5000	Сравнения	12 497 500,0	12 497 500,0	12 497 500,0	12 497 500,0
	Перемещения	4 990,4	4 990,4	4 991,2	4 990,7
10000	Сравнения	49 995 000,0	49 995 000,0	49 995 000,0	49 995 000,0
	Перемещения	9 989,1	9 989,6	9 990,6	9 989,8

Quick sort

N	Номер сортировки	1	2	3	Среднее значение
10	Сравнения	50.3	50.7	47.8	49.6
	Перемещения	12.8	12.9	13.4	13.0
50	Сравнения	523.4	514.9	398.2	478.8
	Перемещения	90.0	89.8	93.9	91.2
100	Сравнения	1293.0	1328.5	912.8	1178.1
	Перемещения	202.1	200.7	211.3	204.7
500	Сравнения	12972.5	13215.3	6278.6	10822.1
	Перемещения	1215.5	1212.2	1334.1	1253.9
1000	Сравнения	32118.6	31994.0	13939.0	26017.2
	Перемещения	2605.0	2607.1	2893.0	2701.7
5000	Сравнения	291617.3	306776.7	85866.4	228086.8
	Перемещения	15159.5	15144.3	17223.3	15842.4
10000	Сравнения	819394.8	828068.8	185012.2	610825.2
	Перемещения	32316.7	32028.3	36936.6	33760.5

* количество массивов было сокращено с 4 до 3, потому что каждое значение в таблице это средний результат из 10000 прогонов различных массивов одной длины

Вывод: В результате проведеных опытов были подтверждены формулы, использующиеся для нахождения асимптотической сложности алгоритмов, и было сделано заключение о том, что quick sort быстрее, чем selection sort, но для этого нужны наиболее случайные значения, при росте количества элементов массива эта разница становится заметнее еще больше.

3 Структура программы и спецификация функций

Для более оптимизированной работы программы использовались функции, которые работали как с массивом данных, так и с численными перменными.

Список функций:

1. change() – функция меняет местами два элемента массивов, которые ей подаются.
2. selection_sort() – функция сортирует массив методом выбора и подсчитывает количество изменений и сравнений, которые были при этом сделаны.
3. fast_sort() – функция сортирует массив методом быстрой сортировки рекурсией и подсчитывает количество изменений и сравнений, которые были при этом сделаны.
4. sort_q() – сортирует массив по возрастанию, используя алгоритм selection sort.
5. sort_rev() – сортирует массив в обратнмо порядке, используя алгоритм selection sort.
6. filling() – заполняет массив случайным числами, используя условие, которое было выбрано.
7. sred() – подсчитывает среднее из всех элементов массива

4 Отладка программы, тестирование программы

Для проверки корректности сортировки, перед выводом результатов, массивы, полученные после выполнения функций, проверялись на упорядоченность. В случае неправильного расположения элементов, в консоль выводилось сообщение об ошибке.
Для отладки и соотвествующей тестировки программы в каждую из сортирующий функций был добавлен вывод массива после каждого прохода по нему.

Далее представлены примеры прохода по **случайным** массивам и их сортировка нашими функциями

selection_sort
изначально функции подается неупорядоченный массив из 7 элементов, после 5 проходов возвращается упорядоченный массив

172.00	5992.90	-4900.79	-7914.01	24101.29	5888.81	-12516.97
172.00	-4900.79	5992.90	-7914.01	24101.29	5888.81	-12516.97
172.00	-4900.79	5888.81	-7914.01	24101.29	5992.90	-12516.97
172.00	-4900.79	5888.81	5992.90	24101.29	-7914.01	-12516.97
172.00	-4900.79	5888.81	5992.90	-7914.01	24101.29	-12516.97
172.00	-4900.79	5888.81	5992.90	-7914.01	-12516.97	24101.29

quick_sort
изначально функции подается неупорядоченный массив из 7 элементов, и возвращается массив, упорядоченный согласно нашему условию.
| - показывает разбиение на подмассивы с которыми работает рекурсивная функция

172.00	5992.90	-4900.79	5888.81	24101.29	-7914.01	-12516.97
172.00	-4900.79	5992.90	5888.81	-7914.01	24101.29	-12516.97
172.00	-4900.79	5888.81	5992.90	-7914.01	-12516.97	24101.29

P.S. при проверке отсортированного массива не стоит забывать, что элементы сортируются по неубыванию модулей

5 Анализ допущенных ошибок

При написании функции **fast_sort()** для сортировки массива методом быстрой сортировки (рекурсивная реализация), была допущена ошибка:
значение pivot (опорного элемента) сохранялось, как значение индекса этого элемента в массиве, а не как вещественное число, из-за чего при выполнение функции значение pivot могло меняться в течение одного прохода, что привело к неправильной сортировке массивов.

6 Литература

Список литературы

[1] Трифонов Н. П., Пильщиков В. Н. Задания практикума на ЭВМ (1 курс). Методическая разработка(составители). — М.: ВМК МГУ, 2001.

[2] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. Второе издание. — М.: «Вильямс», 2005.

[3] Головинов Г. Основы программирования на TeX. Том 1. Начало. — М.: МФТИ, 2024.

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

[5] Лорин Г. Сортировка и системы сортировки. — М.: Наука, 1983.