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

Отчет по заданию №1

«Методы сортировки»

Вариант 3 / 3 / 2 / 3

Выполнил:
студент 104 группы
Гаухов В. К.

Преподаватель:
Сенюкова О. В.

Москва
2020

Содержание

Постановка задачи

В задании требуется сравнить два метода сортировки - метод сортировки простым выбором и метод Шелла. Требуется экспериментально сравнить эти методы сортировки и произвести их теоретическую оценку. Сравнение производится по количеству обменов и сравнений элементов массива, над которым ведется сортировка. На вход даётся массив дробных чисел типа double. Сортировка ведётся по неубыванию модулей.

Результаты экспериментов

Сортировка простым выбором

n	Параметр	Номер сгенерированного массива				Среднее значение
		1	2	3	4	
10	Сравнения	45	45	45	45	45
	Перемещения	0	5	7	8	5
100	Сравнения	4950	4950	4950	4950	4950
	Перемещения	0	50	97	92	59.75
1000	Сравнения	499500	499500	499500	499500	499500
	Перемещения	0	500	993	992	621.25
10000	Сравнения	49995000	49995000	49995000	49995000	49995000
	Перемещения	0	5000	9983	9982	6241.25

Таблица 1: Результаты работы сортировки простым выбором

Оценка работы сортировки простым выбором

Время работы алгоритма T находится из формулы $T = T_1 + T_2$, где T_1, T_2 - время, затраченное на сравнение элементов и обмен значений между собой.

В сортировке простым выбором сначала находится элемент с наименьшим по модулю значением, этот элемент меняется местами с первым элементом массива. Из оставшихся $n-1$ элементов также находится элемент с наименьшим по модулю значением, который меняется со вторым элементом. Всего потребуется $n-1$ таких шагов. На k шаге происходит $n-k$ сравнений. Всего сравнений - $\frac{n(n-1)}{2}$, т.е. $T_1 = O(n^2)$. Количество перемещений в худшем случае $n-1$, а в лучшем - 0 т.е $T_2 = O(n)$. Значит, среднее время работы сортировки $T = T_1 + T_2 = O(n^2) + O(n) = O(n^2)$.

Метод Шелла

Сортировка Шелла является усовершенствованным вариантом сортировки вставками. Количество перемещений пропорционально размеру массива, а количество сравнений - размеру массива умноженного на квадрат логарифма размера массива, т.е. в среднем алгоритм работает за $O(n * (logn)^2)$. В худшем случае сортировка работает за $O(n^2)$[1].

При сравнении таблиц можно заметить, что при увеличении количества элементов число сравнений в простой сортировке в разы превосходит число сравнений в сортировке Шелла. Рост числа перемещений в сортировке Шелла менее значителен.

n	**Параметр**	**Номер сгенерированного массива**				**Среднее значение**
		1	2	3	4	
10	Сравнения	22	27	29	31	27.25
	Перемещения	0	13	11	16	10
100	Сравнения	503	668	819	883	718.25
	Перемещения	0	260	377	428	266.25
1000	Сравнения	8006	11716	14186	14312	12055
	Перемещения	0	4700	6711	6829	4560
10000	Сравнения	120005	172578	238495	238907	192496.25
	Перемещения	0	62560	123611	124020	77547.75

Таблица 2: Результаты работы сортировки Шелла

Структура программы и спецификация функций

- void swap(double arr[ ], int i, int j)

  Функция принимает на вход массив arr и номера элементов i, j, между которыми происходит обмен значений. Ничего не возвращает.

- double random_num(void)

  Функция генерирует случайное число типа double и возвращает его значение.

- double *gen_arr(int type, int count)

  Функция генерирует массив double размера count в зависимости от значения type(1 = ORDER - упорядоченный массив, 2 = REV_ORDER - обратно упорядоченный массив, остальное - массив случайных значений), возвращает указатель на массив.

- double *arr_cpy(double *arr, int count)

  Функция генерирует массив double размера count, состоящий из элементов массива arr. Возвращает указатель на полученный массив.

- int cmp(double x, double y)

  Функция сравнивает по модулю значения x и y, если |x|>|y| возвращает 1, иначе - 0.

- void Shell_Sort(double arr[], int size)

  Функция реализует сортировку массива arr размера size методом Шелла. В качестве значений шага step выбрана последовательность $n/2$, $n/4$ ... 1, которую предложил сам Шелл. Функция не возвращает значение.

- void Simple_Choose(double arr[], int size)

  Функция реализует сортировку массива arr размера size методом простого выбора.

- void test(double* arr, int count)

  Функция проверяет массив arr размера count на упорядоченность. Если находятся два неупорядоченных элемента - печатает сообщение об ошибке и значения элементов. Ничего не возвращает.

Отладка программы, тестирование функций

Тестирование и отладка программы производилась с помощью функции void test(double* arr, int count), которая проверяла массив на упорядоченность.

Результаты работы сортировок

Исходный упорядоченный массив:
0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0
После сортировки Шелла:
0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0
После сортировки простым выбором:
0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0
Исходный обратно упорядоченный массив:
10.0 9.0 8.0 7.0 6.0 5.0 4.0 3.0 2.0 1.0
После сортировки Шелла:
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0
После сортировки простым выбором:
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0
Исходный массив случайных значений:
-1.301755e+082 -2.354801e+187 -1.582684e+263 -1.650100e-198 1.005085e+158 -1.408468e-263 -1.650398e+264 -1.071105e+279 6.814479e+155 -6.883421e+261
После сортировки Шелла:
-1.408468e-263 -1.650100e-198 -1.301755e+082 6.814479e+155 1.005085e+158 -2.354801e+187 -6.883421e+261 -1.582684e+263 -1.650398e+264 -1.071105e+279
После сортировки простым выбором:
-1.408468e-263 -1.650100e-198 -1.301755e+082 6.814479e+155 1.005085e+158 -2.354801e+187 -6.883421e+261 -1.582684e+263 -1.650398e+264 -1.071105e+279

Анализ допущенных ошибок

Изначально для генерации случайного числа double использовались тригонометрические функции, но из-за недостаточной точности вычисления функций компьютером диапазон возможных случайно полученных значений был сильно ограничен.

Список литературы

[1] Д. Кнут. Искусство программирования. Том 3. Сортировка и поиск, 2-е изд. Калифорния, 1997.