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

Отчет по заданию №1

«Методы сортировки»

Вариант 2 / 1 / 2 / 3

Выполнил:
студент 102 группы
Плеханов А. Д.

Преподаватель:
Кулагин А. В.

Москва
2020

Содержание

Постановка задачи

В задании требуется реализовать две сортировки массива чисел - сортировку простым выбором и сортировку Шелла. Необходимо провести эксперементальное сравнение, а также привести теоретические оценки сложности данных методов. Сравнение требуется произвести по количеству сравнений и перестановок на массивах чисел длиною 10, 100, 1000, 10000. Тип элементов массива - long long int, сортировка элементов массива происходит в порядке неубывания.

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	9	8	6
100	Сравнения	4950	4950	4950	4950	4950
	Перемещения	0	50	92	97	60
1000	Сравнения	499500	499500	499500	499500	499500
	Перемещения	0	500	988	992	620
10000	Сравнения	49995000	49995000	49995000	49995000	49995000
	Перемещения	0	5000	9989	9987	6244

Таблица 1: Результаты работы сортировки простым выбором

Сортировка простым выбором, применимая к массиву длиной n, имеет, как видно из таблицы, постоянное количество сравнений, равное $\frac{n(n-1)}{2}$, в любом случае. В лучшем случае (когда массив осортирован по неубыванию) не производится никаких обменов. Обмен между элементами массива может происходить каждую итерацию цикла, в котором происходит выбор следующего наименьшего элемента, значит, максимальное количество обменов равно n - 1. Таким образом, алгоритм имеет суммарную сложность, равную $O(n^2) + O(n) = O(n^2)$, в худшем, среднем и лучшем случае.

Сортировка Шелла

n	**Параметр**	**Номер сгенерированного массива**				**Среднее значение**
		1	2	3	4	
10	Сравнения	9	45	36	25	29
	Перемещения	0	45	29	18	23
100	Сравнения	275	620	875	866	659
	Перемещения	0	430	622	623	419
1000	Сравнения	5616	9217	13091	13506	10358
	Перемещения	0	4478	7822	8254	5139
10000	Сравнения	86021	132571	196085	196834	152878
	Перемещения	0	55174	113907	114687	70942

Таблица 2: Результаты работы сортировки Шелла

Сортировка Шелла - усовершенствованная сортировки включенями. Суть усовершенствования заключается в последовательной сортировке подмассивов, полученных из элементов, отстоящих друг от друга на одинаковых расстояних d_i. Сортировка подмассивов осуществляется включениями. В методе, предложенном Д. Шеллом, изначально выбиралась послдеовательность приращений $d_1 = n/2$, ... , $d_k = d_{k+1}/2$. Последним приращением всегда должна быть 1. Выбор такой последовательности приращений даёт сложность $O(n^2)$. В моей сортировке была использована последовательность приращений, предложенная Робертом Сэджвиком:

$d[i] = 9 * 2^i - 9 * 2^{i/2} + 1$, если i - четно
$d[i] = 8 * 2^i - 6 * 2^{(i+1)/2} + 1$, если i - нечетно

При таком выборе последовательности приращений алгоритм имеет сложность $O(n^{7/6})$ в среднем случае и $O(n^{4/3})$ в худшем случае.

Структура программы и спецификация функций

- long long int random_long(void)
  Функция возвращает случайное 64-битное число

- void swap(long long *a, long long *b)
  Функция получает на вход два указателя на 64-битные числа и меняет содержимое указателей местами

- int cmp_order(const void *a, const void *b)
  Функция принимает на вход два указателя, приводит их к типу 64-битного числа и возвращает:
  1, если a > b
  0, если содержимое указателей равно
  -1, если содержимое указателя b больше, чем a

- int cmp_reverse(const void *a, const void *b)
  Функция принимает на вход два указателя на элементы const void, приводит их к типу long long int и возвращает:
  -1, если содержимое указателя a больше, чем b
  0, если содержимое указателей равно
  1, если содержимое указателя b больше, чем a

- long long int *generate_arr(int n, int p)
  Функция получает на вход число n - длину генерируемого массива и параметр p, и возвращает указатель на неубывающий массив (p = 1), произвольный массив (p = 0) или невозрастающий массив(p = -1)

- long long int *duplicate_arr(long long int *a, int n)
  Функция получает на вход указатель на массив и его длину, возвращает указатель на дублированный исходный массив

- void print_arr(long long *a, int n)
  Функция печати массива a длины n

- void select_sort(long long *a, int n)
  Функция получает на вход указатель на массив и его длину, производит его сортировку методом простого выбора, выводя на экран кол-во сравнений и обменов

- int *increment(int *len, int n)
  Функция получает на вход указатель len и число n, строит массив приращений с помощью формул Сэджвика для массива длины n и сохраняет его длину по указателю len. Возвращает указатель на массив приращений.

- void shell_sort(long long *a, int n)
  Функция получает на вход указатель на массив и его длину и производит его сортировку методом Шелла

Отладка программы, тестирование функций

Тестировка функций сортировки производилась с помошью функции void print_arr(long long int *a, int n) на массивах длины до 1000. Также для удобства чтения выведенных на экран массивов применялась модифицированная функция long long int random_long(void), возвращавшая число в пределах $[-999; 999]$ (генерируемое 64-битное число бралось по модулю 1000).

Анализ допущенных ошибок

При реализации сортировки простым выбором в некоторых случаях вызывалась функция void swap(long long int *a, long long int *b) при одинаковых указателях a и b.

Изначально массив приращений для сортировки Шелла строился методом Шелла, что отрицательно сказывалось на эффективности алгоритма в отличие от сортировки с приращениями Сэджвика.

Список литературы

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

[2] Вирт Н. Алгоритмы и структуры данных. —М.: Мир, 1989.