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

Отчет по заданию №1

«Методы сортировки»

Вариант 3 / 3 / 1 + 5

Выполнил:
студент 101 группы
Бутылкин А. С.

Преподаватель:
Кузьменкова Е. А.

Москва
2022

Содержание

1. Постановка задачи

Необходимо реализовать два метода сортировки массива чисел и провести их экспериментальное сравнение. Требования к программе:

- Сравниваемые сортировки: пирамидальная сортировка (сортировка "кучей"), "пузырьковая" сортировка,
- Память под массив чисел выделяется динамически,
- Тип элементов массива: double (вещественные числа двойной точности),
- Числа упорядочиваются по неубыванию модулей.

2. Результаты экспериментов

В полученных таблицах номер сгенерированного массива означает:

- 1 - массив уже упорядочен,
- 2 - массив упорядочен в обратном порядке,
- 3, 4 - расстановка элементов в массиве случайна.

n	Номер сгенерированного массива				Среднее значение
	1	2	3	4	
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"

n	Номер сгенерированного массива				Среднее значение
	1	2	3	4	
10	0	45	20	4	17,25
100	0	4950	2357	2737	2511
1000	0	499500	246253	245698	247862,75
10000	0	49995000	25198045	25077585	25067657,5

Таблица 2: Количество перемещений при сортировке методом "пузырька"

Теоретическая оценка сортировки методом "пузырька"

Пузырьковая сортировка (см. [1]). Заметим, что после каждого шага внешнего цикла мы рассматриваем на одни элемент меньше, т.е. на i-ом шаге внешнего цикла мы сравниваем (n - i) элементов. Всего сравнений получаем: n + (n - 1) + … + 1 = $\frac{n(n+1)}{2}$, то есть $O(n^2)$.

Количество перемещений также равно $O(n^2)$.

n	Номер сгенерированного массива				Среднее значение
	1	2	3	4	
10	43	35	41	42	40,25
100	1291	944	1082	1052	1092,25
1000	22462	15965	17211	17280	18229,5
10000	325894	226682	239187	239414	257794,25

Таблица 3: Количество сравнений при пирамидальной сортировки

n	Номер сгенерированного массива				Среднее значение
	1	2	3	4	
10	29	12	23	25	22,25
100	861	417	550	518	586,5
1000	14975	7317	8594	8649	9883,75
10000	217263	106697	119440	119626	140756,5

Таблица 4: Количество перемещений при пирамидальной сортировки

Теоретическая оценка пирамидальной сортировки

Пирамидальная сортировка (см. [1]). При вставке элемента в массив (i-ый элемент): в худшем случае мы проходим всю высоту дерева, т.е. $\lceil log_2(i) \rceil$, тогда получаем в худшем случае $\sum_{i=1}^{n} \lceil log_2(i) \rceil$ сравнений и перемещений. Аналогично получаем при взятии максимума. В итоге количество сравнений и перемещений $O(n \cdot log(n))$

Итог

Согласно полученным данным можно сделать вывод, что при малых n (n < 100) лучше использовать сортировку "пузырьком", однако при большем n лучше использовать пирамидальную сортировку. С учетом малого количества тестов, а также с учетом того, что в

половине тестов элементы массива были упорядочены (т.е. эти тесты нельзя считать случайными), можно сделать вывод, что полученные результаты эксперимента соответствуют теоретической оценке.

3. Структура программы и спецификация функций

Список реализованных функций: `abs_d, gen_mas_1, gen_mas_2, gen_mas_34, count_out, comp, perm, bub_sort, heap_sort, test`.
Подробное описание каждой функции:

- `double abs_d(double a)`

  Параметры:

  - `double a` - число типа double.

  Функционал:

  Возвращает модуль числа типа double.

- `void gen_mas_1(double *a, int n)`

  Параметры:

  - `double *a` - указатель на первый элемент массива,
  - `int n` - количество элементов массива.

  Функционал:

  Заполняет выделенную память (элементы a1[0 … n - 1]) случайными числами типа double по неубыванию модулей.

- `void gen_mas_2(double *a, int n)`

  Параметры:

  - `double *a` - указатель на первый элемент массива,
  - `int n` - количество элементов массива.

  Функционал:

  Заполняет выделенную память (элементы a2[0 … n - 1]) случайными числами типа double по невозрастанию модулей.

- `void gen_mas_34(double *a, int n)`

  Параметры:

  - `double *a` - указатель на первый элемент массива,
  - `int n` - количество элементов массива.

  Функционал:

  Заполняет выделенную память (элементы a3[0 … n - 1], a4[0 … n - 1]) случайными числами типа double.

- `void count_out(int k)`

  Параметры:

  - `int k` - параметр.

  Функционал:

  Функция имеет статистические переменные: `count_comp`, `count_perm`, в которых хранится количество сравнений и перемещений соответсвенно. При k == 1, увеличиваем `count_comp` на 1, при k == 2, увеличиваем `count_perm` на 1, при k == 0, выводим значения статистических переменных и обнуляем их.

- `bool comp(double x, double y)`

  Параметры:

  - `double x` - число типа double,
  - `double y` - число типа double.

  Функционал:

  Возвращает true, если модуль числа x больше модуля числа y, и false в обратном случае. Также вызывает функцию `count_out` с параметром 1.

- `void perm(double *x, double *y)`

  Параметры:

  - `double *x` - указатель на первый элемент для перемещения,
  - `double *y` - указатель на второй элемент для перемещения.

  Функционал:

  Меняет местами элементы массива, чьи адреса в памяти были переданы. Также вызывает функцию `count_out` с параметром 2.

- `void bub_sort(double *a, int n)`

  Параметры:

  - `double *a` - указатель на первый элемент массива,
  - `int n` - количество элементов массива.

  Функционал:

  Функция реализует алгоритм сортировки методом "пузырька". Для сравнения или обмена элементов массива вызывает функцию `comp` или `perm` соответсвенно.

- `void heap_sort(double *a, int n)`

  Параметры:

  - `double *a` - указатель на первый элемент массива,
  - `int n` - количество элементов массива.

  Функционал:

  Функция реализует алгоритм пирамидальной сортировки. Для сравнения или обмена элементов массива вызывает функцию `comp` или `perm` соответсвенно.

- `bool test(double *a, double *an, int n)`

Параметры:

- `double *a` - указатель на первый элемент массива, отсортированного с помощью одной из функций (`bub_sort, heap_sort`),
- `double *an` - указатель на первый элемент массива (массив до сортировки массива `a`),
- `int n` - количество элементов массива.

Функционал:

Функция проверяет, что элементы массива `a`, отсортированы по неубыванию модулей элементов. Также проверяет, что все числа из массива `an` присутствуют в массиве `a` в том же количестве.

4. Отладка программы, тестирование функций

Этапы тестирования:

1. Ручное тестирование функции `abs_d`
2. Ручное тестирование функций `gen_mas_1, gen_mas_2, gen_mas_34` для разных n (проверка на упорядоченность для `gen_mas_1, gen_mas_2`)
3. Ручное тестирование функций `count_out, comp, perm`
4. Ручное тестирование функций `bub_sort, heap_sort, test` с разными входными массивами

5. Анализ допущенных ошибок

В ходе тестирования и отладки была выявлена ошибка в функции `test,` не было проверки на то, что одинаковых элементов в массиве одно и то же количество (была проверка на то ,что элемент входит в оба массива). Также была реализована более безопасная версия сравнения элементов типа double(вместо "==" сравниваем разность чисел через ">")

Список литературы

[1] Кнут Д., Искусство программирования для ЭВМ. Том 3 Первое издание. — М.:«Мир», 1978.