**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**отчет**

**по лабораторной работе №1**

**по дисциплине «Построение и анализ алгоритмов»**

**Тема: «Программирование и исследование алгоритмов сортировки»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6382 |  | Черкасова Е.И. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2017

**Цель работы.**

Реализовать выбранный алгоритм сортировки и провести экспериментальное испытание (исследование) алгоритма (программной реализации).

**Постановка задачи.**

Вариант 21.

В наборе S имеется n вещественных чисел. Задано также вещественное число x. Содержатся ли в S два таких элемента, что их сумма равна x.

Указание. Если набор S отсортирован, то решить задачу можно за время O(n).

**Требования.**

Построить, обосновать и испытать алгоритм, решающий поставленную задачу. Провести экспериментальное испытание алгоритма сортировки Шелла. При этом оценить эффективность алгоритма: время выполнения (с использованием компьютерного таймера). По результатам испытаний необходимо проанализировать:

1) зависимость времени выполнения от количества элементов в заданной последовательности.

2) соотношение измеренных и теоретически предсказанных значений.

Рекомендуется провести сравнения (по тем показателям, по которым это возможно) с каким-либо «эталонным» алгоритмом (реализацией), например, с системной функцией qsort или с какой-либо ещё библиотечной функцией.

**Описание алгоритмов.**

1. Алгоритм, решаюющий поставленную задачу.

**Шаг 1.** После упорядочивания по возрастанию вещественных чисел из S, находится элемент, меньший или равный x.

**Шаг 2.** Теперь рассматривается только часть последовательности от начала до рассматрриваеого элемента (“усеченная” последовательность). Вычисляется разница между этим элементом и х. Если в последовательности присутствует 0 и вычисленная разница равна 0, то ответ на задачу положительный. Если она не 0, алгоритм переходит к шагу 3.

**Шаг 3.**Двигаясь от найденного элемента к началу последовательности, каждый следующий элемент сравнивается с данной разницей и если оказывается ей равным, то это означает положительный ответ на задачу. Если же разница больше проверяемого элемента, то учитывая, что последовательность упорядочена, дальнейшие проверки не имеют смысла. Если в “усеченной” последовательности еще есть элементы, то рассматриваемым становится следующий за элементом, найденном на предыдущем шаге, и алгоритм возвращается к шагу 2. Иначе ответ на задачу отрицательный, т.е. в последовательности S нет таких 2 элементов, что их сумма равна х.

2. Сортировка Шелла

Это [алгоритм сортировки](https://ru.wikipedia.org/wiki/Алгоритм_сортировки), являющийся усовершенствованным вариантом [сортировки вставками](https://ru.wikipedia.org/wiki/Сортировка_вставками). Идея метода Шелла состоит в сравнении элементов, стоящих не только рядом, но и на определённом расстоянии друг от друга.

При сортировке Шелла сначала сравниваются и сортируются между собой значения, стоящие один от другого на некотором расстоянии  (о выборе значения  [см. ниже](https://ru.wikipedia.org/wiki/Сортировка_Шелла" \l "Выбор_длины_промежутков)). После этого процедура повторяется для некоторых меньших значений , а завершается сортировка Шелла упорядочиванием элементов при  (то есть обычной [сортировкой вставками](https://ru.wikipedia.org/wiki/Сортировка_вставкой)). Эффективность сортировки Шелла в определённых случаях обеспечивается тем, что элементы «быстрее» встают на свои места (в простых методах сортировки, например, [пузырьковой](https://ru.wikipedia.org/wiki/Сортировка_методом_пузырька), каждая перестановка двух элементов уменьшает количество [инверсий](https://ru.wikipedia.org/wiki/Инверсия_(перестановка)) в списке максимум на 1, а при сортировке Шелла это число может быть больше).

Невзирая на то, что сортировка Шелла во многих случаях медленнее, чем [быстрая сортировка](https://ru.wikipedia.org/wiki/Быстрая_сортировка), она имеет ряд преимуществ:

- отсутствие потребности в памяти под стек;

- отсутствие деградации при неудачных наборах данных — быстрая сортировка легко деградирует до O(n²), что хуже, чем худшее гарантированное время для сортировки Шелла.

* Среднее время работы алгоритма зависит от длин промежутков - d, на которых будут находиться сортируемые элементы исходного массива ёмкостью  на каждом шаге алгоритма. Существует несколько подходов к выбору этих значений:

**ДОП вики**

**Спецификация программы.**

Ограничения на входные данные: входные данные - массив вещественных чисел (с количеством элементов от 2 до 1 000 000), больших 0, но меньших 1000000.

Место и форма представления входных данных: входные данные расположены либо в файле, либо вводятся с клавиатуры, либо генерируются случайным образом самой программой (в указанном диапазоне и указанной длины), представляют собой последовательность вещественных чисел, где все элементы, кроме последнего, образуют последовательность - S, а последний есть х.

Выходные данные. Результат работы программы отображается в консоли. При необходимости могут выводиться в файл.

Также в программе присутствуют проверки на все входные данные, включая вспомогательные для диалога с пользователем. На любом этапе выполнения программы при некорректном вводе данных есть возможность их повторного ввода или же можно завершить работу.

Реализовано сравнение времени работы используемого метода сортировки и библиотечной функции qsort, проверка отсортированного массива.

**Алгоритм работы программы.**

1. Выбор способа ввода данных

2. Считывание исходных данных и их проверка

3. Копирование данных в 2 вектора (для сортировок)

4. Сортировка S последовательности 2 методами

5. Работа функции, решающей задачу

6. Проверка отсортированных массивов на идентичность

7. Рассчет затраченного времени на каждую сортировку и на решение поставленной задачи

**Описание основных функций.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Тип возвращаемого значения* | *Имя Функции* | *Входные данные* | *Выходные данные* | *Назначение* |
| void | cinclear | - | - | очистка буфера ввода |
| void | safe\_exit | - | - | закрытие файлов перед выходом из программы до ее завершения |
| bool | need\_exit | - | true - выход  false - повторный ввод | осуществление выбора дальнейшего действия при ошибке во входных данных |
| bool | incheck | string& buf - cсылка на введенные данные  int& n - ссылка для передачи числа в программу | true - число, удолетворяющее требованиям задачи  false - не число | проверка введенной строки: является ли числом в определенных границах |
| void | generate | ostream &file - файл для записи результата; size\_t n - количество элементов | - | генерация случайной последовательности вещественных чисел |
| void | see\_file | fstream &file - ссфлка на файловый поток | - | вывод содержимого файла на консоль, не извлекая данные |
| void | read\_vector | istream& my\_stream - ссылка на поток для чтения vector<float> &s - вектор, куда будут считаны данные | - | для считывания данных в вектор вещественных чисел |
| bool | read\_vector | vector<float> &s - вектор вещ. чисел | true - успешно  false - пустой вектор | вывести на экран вектор |
| void | template <typename T> shell\_sort | T arr[] -  size\_t distance - | - | сортировка Шелла |
| bool | if\_there\_is\_sum | vector<float>& s\_ptr - вектор данных  const float& x - сумма чисел | true - да  false - нет | ответ на поставленную задачу |

**Пример работы программы.**

Press Enter to start

Select type of data input. 1 - file, 2 - keyboard, 3 - generate

1

File is:

23.23 32.32 10.1 19.1

29.2

The sequence S is:

23.23 32.32 10.1 19.1

The x is: 29.2

>After shell\_sort

The sequence S is:

10.1 19.1 23.23 32.32

>After qsort

The sequence S is:

10.1 19.1 23.23 32.32

>Checking sorts

Perfect! The sorts gave equal sequences!

>Solving task : if in S there is 2 elements with sum of 29.2 ?

Yes! 19.1 and 10.1

The amount of elements in S is 4

Time of shell\_sort : 11

Time of qsort : 4

Time to solve the task with previous input sorting : 3

**Тестирование программы.**

Рассмотрим тесты на корректных входных данных.

|  |  |  |
| --- | --- | --- |
| **Неотсортированный массив** | **Отсортированный массив** | **Дополнительные данные** |
| -72 -81 -95 -62 -78 -56 -88 | -95 -88 -81 -78 -72 -62 -56 | Перемещения: 42  Сравнения: 28  Время(нс): 2758602 |
| -9 15 -19 -9 -31 -35 22 -39 28 19 -13 -27 -21 13 25 -46 -45 -1 25 49 -23 11 12 -33 29 11 -28 -37 -1 21 11 -42 31 17 30 -3 33 38 -20 -38 24 28 -17 -27 8 33 26 9 27 35 -25 -32 48 -34 50 29 38 -14 3 22 45 43 -11 28 -45 -35 6 -24 12 15 7 19 13 -19 13 18 -4 -10 -4 -35 37 -1 -3 -33 11 -42 -8 -11 14 36 -1 -7 -6 -3 19 -25 17 8 7 -15 | -46 -45 -45 -42 -42 -39 -38 -37 -35 -35 -35 -34 -33 -33 -32 -31 -28 -27 -27 -25 -25 -24 -23 -21 -20 -19 -19 -17 -15 -14 -13 -11 -11 -10 -9 -9 -8 -7 -6 -4 -4 -3 -3 -3 -1 -1 -1 -1 3 6 7 7 8 8 9 11 11 11 11 12 12 13 13 13 14 15 15 17 17 18 19 19 19 21 22 22 24 25 25 26 27 28 28 28 29 29 30 31 33 33 35 36 37 38 38 43 45 48 49 50 | Перемещения: 1200  Сравнения: 1099  Время(нс): 27672784 |
| 16806 17005 17825 15174 14531 14256 17443 15108 12060 17006 12092 16633 16062 15619 16508 18763 18214 10076 16511 18260 | 10076 12060 12092 14256 14531 15108 15174 15619 16062 16508 16511 16633 16806 17005 17006 17443 17825 18214 18260 18763 | Перемещения: 160  Сравнения:131  Время(нс): 13385597 |
| 8 5 | 5 8 | Перемещения: 4  Сравнения:2  Время(нс): 908871 |

Рассмотрим тесты на некорректных входных данных, при которых программа корректно завершает работу.

|  |  |
| --- | --- |
| **Входные данные** | **Результат** |
| 2 \* (ввод с клавиатуры) | You can enter only numbers! Continue, please. Invalid character will not be written |
| 3 | There is only 1 number in your file. Correct it and restart the programm! |
| 1 2 6 | This numbers are ordered! |
| 2 2 2 2 | This numbers are ordered! |
| ; , ! (ввод с файла) | There is invalid character in your file: ;  Correct it and try again |
| 4ы (ввод с файла) | There is invalid character in your file: \321  Correct it and try again |
| 0,2 0,7 | There is invalid character in your file: ,  Correct it and try again |

Примечание: при каждой проверке на корректных данных сортировка программы сходилась с сортировкой функции qsort.

**Исследование.**

В данной лаборатнорной работе реализовывался алгоритм сортировки методом естественного слияния. Для начала рассмотрим зависимость времени выполнения (см. рис. 1), количества сравнений (см. рис. 2) и количества присваиваний(см. рис. 3) от размера массива.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер массива | Время, мс | Сравнения | Присваивания | Время qsort, мс |
| 100 | 12.267761 | 1105 | 1200 | 0.005763 |
| 1 000 | 18.920836 | 17185 | 18000 | 0.183000 |
| 10 000 | 121.212545 | 248727 | 260000 | 1.329810 |
| 100 000 | 1432.628522 | 3116666 | 3200000 | 14.651561 |
| 500 000 | 8091.203434 | 17603879 | 18000000 | 85.488438 |
| 1 000 000 | 17856.240760 | 37222992 | 38000000 | 164.034629 |
| 2 000 000 | 44743.813228 | 78439538 | 80000000 | 320.788530 |
| 3 000 000 | 63092.164381 | 122696783 | 126000000 | 506.485552 |
| 4 000 000 | 81344.483574 | 164880194 | 168000000 | 650.889992 |
| 5 000 000 | 104492.426248 | 214518608 | 220000000 | 820.157487 |
| 6 000 000 | 128705.611313 | 257387291 | 264000000 | 981.501852 |

Рисунок 1 - График зависимости времени сортировки от длины массива

Рисунок 2 - График зависимости количества присваиваний от длины массива

Рисунок 3 - График зависимости количества сравнений от длины массива

Следует отметить, что распротранена также такая сортировка слиянием, которая использует простое слияние. В чем разница модификаций? В случае простого слияния частичная упорядоченность сортируемых данных не дает никакого преимущества. Это объясняется тем, что на каждом проходе сливаются серии фиксированной длины. При естественном слиянии длина серий не ограничивается, а определяется количеством элементов в уже упорядоченных подпоследовательностях, выделяемых на каждом проходе.

Сравним наш алгоритм сортировки с сортировкой qsort. Исходя из экспериментальных данных, сортировка qsort более выгодна в отношении времени и на малых массивах, и на больших. Частично это может быть обосновано неидеальностью функции естественного слияния по сравнению с библиотечной функцией qsort. И исходя из анализа алгоритма естественного слияния, сортировка слиянием затрачивает больший объем памяти, чем qsort.

**Вывод.**

В результате выполнения этой лабораторной работы была исследована сортировка массива естественным слиянием, найдена зависимость количества базовых операций и времени от длины сортируемого массива и проведено сравнение с сортировкой простым слиянием и библиотечной функцией сортировки qsort.

**ПРИЛОЖЕНИЕ 1**

**Исходный код программы**