Лабораторная работа №8

«Алгоритмы сортировки. Сложность алгоритмов»

Работу выполнил Ивашкин Иван, гр. Б02-012

Дата выполнения – 29.03.2021

Цель работы – выяснить зависимость времени выполнения различных алгоритмов сортировки целочисленных массивов от количества элементов в них, а также сравнить эти алгоритмы между собой.

В работе используются: кроссплатформенная среда разработки с открытым кодом «Code::Blocks 20.03»; компилятор MinGW; табличный процессор «Microsoft Excel 2019» среда для построения научной графики «SciDAVis 2.3.0».

Краткие теоретические сведения

Основные понятия

Алгоритмом сортировки называется алгоритм, предназначенный для упорядочивания элементов в списке или массиве. В данной работе рассматриваются алгоритмы сортировки одномерных целочисленных массивов.

При сравнении эффективности различных алгоритмов сортировки учитывают три основных характеристики:

1) *быстродействие*, то есть скорость выполнения алгоритма,
2) дополнительные *затраты памяти* на хранение вспомогательных элементов или массивов,
3) *устойчивость* – свойство, при котором алгоритм не меняет местами элементы, изначально расположенные в верном порядке.

Производя сравнительный анализ алгоритмов, обычно говорят об их *сложности*, имея в виду, с одной стороны, время, затраченное на выполнение всех операций (*временная сложность*), а с другой стороны, объем памяти, необходимый алгоритму для работы (*пространственная сложность*).

В общем случае обе сложности могут являться функциями $f(n)$ длины строки n, представляющей входные данные. Для облегчения их сравнения сложность выражают с использованием нотации «О-большое», которая учитывает только слагаемое самого высокого порядка и игнорирует константные множители.

Сложность алгоритма сортировки может меняться в зависимости от содержания подаваемых на вход данных. Будем говорить о сложности того или иного алгоритма в трех случаях: *наихудшем* (на вход подан массив

элементов, размещенных в порядке, полностью противоположном верному), *наилучшем* (на вход подан уже отсортированный массив) и *среднем*.

Существует множество различных алгоритмов сортировки массивов и списков чисел. Их можно подразделить на две группы. К первой группе относятся *элементарные алгоритмы*, имеющие относительно простую структуру и появившиеся на заре программирования.

Ко второй группе относятся *эффективные алгоритмы*, представляющие собой в большинстве случаев какой-либо серьезно усовершенствованный элементарный алгоритм (или их комбинацию). Эффективные способы сортировки включают более сложные операции с данными, дают экономию памяти и времени и широко используются на практике в отличие от элементарных алгоритмов, сохранившихся лишь в учебных курсах программирования.

В рамках данной работы реализованы и изучены три элементарных алгоритма сортировки: сортировка пузырьком, сортировка выбором, сортировка вставками, – а также один эффективный алгоритм сортировки – сортировка Хоара (или быстрая сортировка).

Приведем краткие теоретические сведения по каждой из указанных сортировок.

Сортировка пузырьком

Сортировка пузырьком (Bubble sort), или сортировка простыми обменами, – это элементарный алгоритм сортировки, основанный на многократном сравнении соседних элементов массива.

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется перестановка элементов. В простейшей реализации проходы по массиву повторяются $N - 1$ раз. Наименьший элемент шаг за шагом «всплывает» на «поверхность» массива, как пузырек в воде, – отсюда и название.

Сортировка выбором

Сортировка выбором (Selection sort) – это элементарный алгоритм сортировки, основанный на последовательном уменьшении области массива, подлежащей обработке (на каждом шаге сортируется «хвост», оставшийся после предыдущего шага). При той же сложности, что и у сортировки пузырьком, выполняет меньшее количество обменов элементов.

Шаг алгоритма состоит из следующих операций:

1) выбирается наименьший элемент в обрабатываемой части массива,

2) выбранный элемент обменивается местами со значением первой неотсортированной позиции («хвост» массива укорачивается на один элемент),
3) аналогично сортируется «хвост» массива, причем ранее отсортированные элементы исключаются из рассмотрения.

Сортировка вставками

Сортировка вставками (Insertion sort) – это элементарный алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается (вставляется) на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

Сортировка Хоара

Сортировка Хоара, или быстрая сортировка (Quick sort) – это эффективный алгоритм сортировки, являющийся одним из самых быстрых и универсальных алгоритмов упорядочивания массивов. Разработан английским математиком Тони Хоаром.

Алгоритм состоит из трёх шагов:
1) выбирается некоторый опорный элемент массива,
2) остальные элементы перераспределяются таким образом, что элементы, меньшие опорного, помещаются перед ним, а большие или равные – после,
3) первые два шага рекурсивно применяются к двум полученным на втором шаге подмассивам, находящимся слева и справа от опорного элемента.

Выбор опорного элемента на первом шаге алгоритма может производиться разными способами. В ранних реализациях в качестве опорного выступал первый элемент массива, что снижало эффективность сортировки Хоара при работе на уже отсортированных массивах.

Для улучшения эффективности может выбираться средний, случайный элемент или (для больших массивов) медиана первого, среднего и последнего элементов. Наиболее оптимальным опорным элементом является медиана всей последовательности, но её вычисление слишком трудоёмко для использования в сортировке.

Основные характеристики сортировок

В заключение приведем таблицу основных характеристик изучаемых в работе алгоритмов сортировки числовых массивов.

В таблице даны временные и пространственные сложности алгоритмов в наихудшем, наилучшем и среднем случаях, а также отмечена устойчивость алгоритмов. Кроме того, приведены найденные теоретически количества операций сравнений и обменов элементов для наихудшего и наилучшего случаев (при характеристике сортировки Хоара мы ограничиваемся оценками для среднего случая)

Таблица 1. **Сложность алгоритмов сортировки**

Характеристика		Сортировка			
		пузырьком	выбором	вставками	Хоара
Число сравнений	наихудший случай	$(n-1)n$	$(n-1)\frac{n}{2}$	$(n-1)\frac{n}{2}$	В среднем: $n\log n$
	наилучший случай	$(n-1)n$	$(n-1)\frac{n}{2}$	$n-1$	
Число обменов	наихудший случай	$(n-1)\frac{n}{2}$	n	$(n-1)\frac{n}{2}$	В среднем: $\frac{n\log n}{6}$
	наилучший случай	0	n	0	
Временная сложность	наихудший случай	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
	наилучший случай	$O(n)$	$O(n)$	$O(n)$	$O(n\log n)$
	средний случай	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n\log n)$
Пространственная сложность	наихудший случай	$O(1)$	$O(1)$	$O(1)$	$O(1)$
	наилучший случай	$O(1)$	$O(1)$	$O(1)$	$O(1)$
	средний случай	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Устойчивость		Есть	Нет	Есть	Зависит от реализации

Ход работы и её результаты

Ссылка на репозиторий данной работы.

Мы начали работу с написания и компиляции функций, реализующих каждый из алгоритмов сортировки. Функции были помещены в отдельные файлы: **`bubble_sort.cpp`**, **`selection_sort.cpp`**, **`insertion_sort.cpp`** и **`qsort.cpp`**.

Далее написанные функции были объединены в файле **`test.cpp`**. Время исполнения каждой функции измерялось в наносекундах с помощью средств библиотеки **`chrono`**. Результаты измерений записывались в файлы расширения **`.txt`** отдельно для каждой сортировки. Впоследствии данные из этих файлов были импортированы в табличный процессор «Microsoft Excel».

Применяемый в работе способ измерения времени надо признать довольно грубым, если дополнительно учесть, что в работе использовалась ОС «Windows». Грубость измерений стала заметна после построения графика временной сложности для сортировки Хоара: разброс точек в этом случае оказался весьма ощутимым.

Программа **`test.cpp`** подавала на вход каждой сортировки массив случайных чисел. За счет постоянства зерна генератора случайных чисел их последовательность всякий раз была одинакова.

Количество элементов в массиве варьировалось следующим образом:
1) для сортировки Хоара – от 5 000 до 5 005 000 с шагом в 10 000,
2) для сортировки вставками – от 1 000 до 10 000 с шагом в 1 000, далее от 10 000 до 100 000 с шагом в 10 000, далее отдельно для 500 000 и 1 000 000.
3) для сортировок выбором и пузырьком – как для сортировки вставками, но без последнего значения в 1 000 000.

Сужение диапазона измерений для элементарных сортировок было обусловлено их низкой эффективностью.

Полученные графики зависимости времени работы каждого алгоритма сортировки (в наносекундах) от количества элементов в массиве приведены в файлах **`elem_graph.pdf`** и **`quick_graph.pdf`** (построены отдельно в связи с разными масштабами по осям координат).

Аналогичным образом выполнено тестирование алгоритмов в наилучших случаях, то при подаче на вход сортировки правильно упорядоченного массива. Данный тест выполнен в программе **`already_sorted_test.cpp`**. Результаты измерений сохранены также в виде отдельных файлов со словом **`sorted`** в названии.

В ходе последнего теста было установлено, что алгоритм сортировки вставками способен распознать упорядоченный массив: его время работы было исчезающе мало (в файле **`Insertion_sort_sorted.txt`** большинство результатов нулевые). Случайные отклонения результатов измерений от нулевых объясняются несовершенством метода измерения времени.

Для других массивов по результатам их работы в наилучших случаях были построены графики временной сложности, представленные в файлах **`quick_sort_sorted.pdf`** и **`elem_sort_sorted.pdf`** (пунктиром отмечены графики для случайных массивов).

Выводы

В работе реализованы и изучены четыре алгоритма сортировки: пузырьком, выбором, вставками и Хоара. Проведенные в работе измерения позволяют расположить указанные алгоритмы в следующем порядке по убыванию эффективности: сортировка Хоара, сортировка вставками, сортировка выбором, сортировка пузырьком.