***Лабораторная работа 3*. Методы внутренней сортировки массивов**

***Цель работы —* изучение эффективности основных методов внутренней сортировки массивов.**

Упорядочение данных (сортировка) процесс, используемый в прикладном программировании очень часто. При больших объёмах данных именно методы сортировки определяют эффективность всей системы. Цель сортировки — облегчить поиск элементов в отсортированном множестве данных.

***1.1.* *Постановка задачи***

В общем виде задача сортировки выглядит следующим образом. Пусть надо упорядочить n элементов R1, R2, …, Rn, которые являются записями. Каждая запись Rj имеет свой ключ Kj, который и управляет процессом сортировки. Помимо ключа, запись может содержать дополнительную информацию, которая не влияет на сортировку, но присутствует в записи.

Задача сортировки — найти такую перестановку записей

Rp1, Rp2, …, Rpn ,

чтобы ключи расположились бы в следующем порядке Kp1, Kp2,…, Kpn.

При сортировке либо перемещаются сами записи, либо создаётся вспомогательная таблица, которая описывает перестановку и обеспечивает доступ к записям в соответствии с порядком их ключей.

В дальнейшем под множеством данных будем понимать массив A[n] целых или вещественных чисел. Задача сортировки такого массива состоит в преобразовании массива  в массив  . Обычно сортировку подразделяют на два класса: внутреннюю, когда все записи хранятся в оперативной памяти, и внешнюю, когда при сортировке используются внешние запоминающие устройства. Далее рассматривается только внутренняя сортировка.

***Основные требования к методам сортировки:***

1. **Экономия памяти.** Все перестановки элементов массива должны выполняться на том же месте (то есть должен быть отсортирован исходный массив A, а не получен результирующий массив B).
2. **Устойчивость сортировки.** Все элементы с равными значениями после сортировки должны сохранять порядок следования относительно друг друга.

Для оценки быстродействия алгоритмов различных методов сортировки обычно используют два показателя:

– количество присваиваний;

– количество сравнений.

Все методы внутренней сортировки можно разделить на две группы:

– прямые методы сортировки;

– улучшенные методы сортировки.

Прямые методы сортировки в соответствии с принципами, положенными в основу методов, разделяются на три подгруппы:

1. Сортировка **прямым включением**;
2. Сортировка **прямым выбором**;
3. Сортировка **прямым обменом** (метод “пузырька”).

Улучшенные методы сортировки основываются на тех же принципах, что и прямые, но используют некоторые оригинальные идеи для ускорения процесса сортировки. Прямые методы на практике используются довольно редко, так как имеют относительно низкое быстродействие. Однако они хорошо показывают суть основанных на них улучшенных методов. Кроме того, в некоторых случаях (как правило, при небольшой длине массива или особом исходном расположении элементов) некоторые из прямых методов могут даже превзойти улучшенные методы.

В данной лабораторной работе необходимо изучить ряд известных алгоритмов сортировки и создать программу, реализующую

1. Метод сортировки прямым включением;
2. Метод сортировки прямым выбором;
3. Метод сортировки прямым обменом;
4. Метод быстрой сортировки.

Разрабатываемая программа должна обеспечивать

* вывод на экран меню;
* ввод исходной информации;
* формирование массивов с большим числом элементов;
* выбор метода сортировки;
* сортировку массива;
* печать результата;
* замеры времени выполнения сортировки массива;
* подсчёт числа сравнений и перемещений элементов в процессе сортировки.

Демонстрация работоспособности разработанной программы должна обеспечивать два варианта контроля: контроль работоспособности каждого из методов и контроль временных характеристик всех реализованных методов.

***1.2. Учебно-методические цели работы***

Данная задача представляет собой вариант широко распространенной задачи сортировки, которая применяется в широком классе реальных задач, и умение и знание алгоритмов сортировки и их особенностей является необходимой частью образования специалиста по программному обеспечению. Наряду с

ознакомлением с проблемой упорядочивания массивов и методами ее решения, выполнение работы направлено на достижение следующих учебно-методических целей:

• приобретение навыков и приемов использования массивов, доступа к их элементам и преобразований массивов;

• практическое освоение принципов программирования ;

• освоение и изучение встроенных функций (применение генератора псевдослучайных чисел и замеры датчиков времени);

• получения навыков сравнительного анализа эффективности разных алгоритмов.

***1.3. Рекомендации по выполнению работы***

Программа должна обеспечить сортировку массивов размером произвольной длины до 10000 элементов и выводить для контроля:

- при небольшом количестве элементов (например, менее 20) - неупорядоченный массив и массив после сортировки для каждого из предложенных алгоритмов;

- при значительном объеме данных выводить число сравнений и обменов элементов и время сортировки одного и того же массива для всех четырех предложенных алгоритмов.

Содержимое массива рекомендуется формировать с помощью генератора

псевдослучайных чисел.

В ходе выполнения лабораторной работы предполагается реализация методов для четырех широко используемых *алгоритмов сортировки*:

• метод прямого выбора;

• метод прямого включения;

• метод прямого выбора;

• метод быстрой сортировки Хоара.

Для простоты изложения описания алгоритмов проводятся на примере задач сортировки массивов по возрастанию.

***1.4. Сортировка с помощью прямого включения***

**Принцип метода:**

Массив длиной N разбивается на две части: упорядоченную и неупорядоченную. Элементы из неотсортированной части поочерёдно выбираются и вставляются в отсортированную часть так, чтобы не нарушать в ней упорядоченность элементов. В начале работы алгоритма в качестве отсортированной части массива принимают только один первый элемент, а в качестве неотсортированной части — все остальные элементы.

Алгоритм состоит из **n-1** прохода (**n** – размер массива), включающего следующие действия:

* взятие очередного **i**-го неотсортированного элемента и сохранение его во вспомогательной переменной **x**;
* поочерёдный сдвиг элементов отсортированной части массива вправо (начиная с **i-1** элемента) с целью определения позиции **j**, в которой присутствие взятого элемента **x** не нарушит упорядоченности элементов;
* включение взятого элемента x в найденную **j**-ю позицию.

**Алгоритм метода прямого включения можно представить в виде:**

**для i=2 до n выполнить** {взятие очередного элемента}

****

**пока ( И )** {поиск места включения}

****

****

**всё-пока**

**** {вставка элемента}

**всё-для**

Схема работы алгоритма прямого включения на примере массива 3,7,2,14,5,1,8 может быть представлена следующим образом:

Итерация Результат итерации

* 1. 3 7 │ 2 14 5 1 8
  2. 2 3 7 │ 14 5 1 8
  3. 2 3 7 14 │ 5 1 8
  4. 2 3 5 7 14 │ 1 8
  5. 1 2 3 5 7 14 │ 8
  6. 1 2 3 5 7 8 14 │

***1.5. Сортировка с помощью прямого выбора***

Аналогично, как и в методе прямого включения, исходный массив разбивается на две части: отсортированную и неотсортированную.

**Принцип метода:**

* выбирается наименьший элемент массива;
* наименьший элемент обменивается местами с первым элементом;
* процесс повторяется с оставшимися n-1, n-2, … элементами до тех пор, пока не останется один, самый большой элемент.

Схема процесса сортировки с помощью прямого выбора на примере массива 3,7,2,14,5,1,8 выглядит следующим образом:

Итерация Результат итерации

1. 1 │ 7 2 14 5 3 8
2. 1 2 │ 7 14 5 3 8
3. 1 2 3 │ 14 5 7 8
4. 1 2 3 5 │ 14 7 8
5. 1 2 3 5 7 │ 14 8
6. 1 2 3 5 7 8 │ 14

Метод прямого выбора в некотором смысле противоположен методу прямого включения. Если при прямом включении для внесения одного элемента в упорядоченную часть массива требуется просмотреть *все элементы упорядоченной части*, то при прямом выборе для внесения элемента надо просматривать *все элементы неупорядоченной части*.

**Алгоритм метода прямого выбора можно представить в виде:**

**для i=1 до n-1 выполнить** {определение места для размещения очередного элемента}

****

**для  до  выполнить** {поиск наименьшего в неотсортированной части}

**если  то  всё-если**

**всё-для**

**** {обмен местами}

****

**всё-для**

***1.6. Сортировка с помощью прямого обмена***

**Принцип метода:**

Справа налево поочерёдно сравниваются два соседних элемента, и если их взаиморасположение не соответствует заданному условию упорядоченности, то они меняются местами.

* После завершения первой итерации наименьший элемент массива будет помещён в самый левый край массива – в элемент с индексом 1.
* На второй итерации выполняются те же действия, но процесс завершается перед элементом , который уже помещён на своё место. Вторая итерация приводит к размещению в позиции  второго по величине элемента заданного массива.
* Таким образом, каждая итерация приводит наименьший элемент в неотсортированной части массива в самую левую позицию этой части, после чего отсортированная часть массива за счёт этого элемента по количеству возрастает на единицу.

Этот метод широко известен как метод «пузырька» (BubbleSort), так как при каждой итерации наименьшие элементы «всплывают» к левому краю массива.

Схема процесса «пузырьковой» сортировки для массива 3,7,2,14,5,1,8 может быть представлена в виде:

Итерация Результат итерации

0 3 7 2 14 5 1 8

1 1 │ 3 7 2 14 5 8

2 1 2 │ 3 7 5 14 8

3 1 2 3 │ 5 7 8 14

4 1 2 3 5 │ 7 8 14

5 1 2 3 5 7 │ 8 14

6 1 2 3 5 7 8 │ 14

***Замечание***.

1. При прямом обмене *тяжёлые* элементы опускаются на «дно» медленно (на один уровень при каждой итерации), а *лёгкие* всплывают на «поверхность» очень быстро.
2. В рассмотренном примере последние четыре прохода не повлияли на взаимное расположение элементов массива, то есть фактически сортировка завершилась раньше, чем выполнен алгоритм сортировки. С учётом этого можно предложить вариант улучшения алгоритма – запоминать, были или не были перестановки в процессе прохода.

**Алгоритм метода прямого обмена можно представить в виде:**

**для** i=2 **до** n **выполнить**

**для** j =n **до** i **шаг** (-1) **выполнить**

**если  то**

****

**всё-если**

**всё-для**

**всё-для**

***1.7. Метод быстрой сортировки (метод Хоара)***

Метод «пузырька» является наименее эффективным из прямых методов сортировки, так как требует большого количества сравнений и обменов. Однако быстрая сортировка, являющаяся усовершенствованием метода прямого обмена, представляет собой самый хороший из известных методов. Быстрая сортировка, предложенная Хоаром, использует обмены на больших расстояниях.

**Метод Хоара основан на следующем алгоритме:**

1. Выбирается какой-либо элемент  массива A (например, вблизи середины массива).
2. Элементы исходного массива перемещаются так, чтобы меньшие или равные  были расположены левее , а большие или равные  — правее . После такого разделения массива  будет находиться на своём месте в отсортированном массиве, то есть один элемент будет «отсортирован».
3. Процесс разделения применяется к получившимся частям массива (меньше и больше ), затем к частям частей до тех пор, пока каждая из частей не будет состоять из одного и только одного элемента.

Разделение массива можно проводить в соответствии со следующим алгоритмом:

****

****

**повторять**  {цикл с постусловием}

**пока**  **выполнить  всё-пока**

**пока**  **выполнить  всё-пока**

**если  то**

****

**всё-если**

**до ** {окончание цикла с постусловием}

**Вариант быстрой сортировки на Питон:**

def quicksort(nums, fst, lst):

   if fst >= lst: return

   i, j = fst, lst

   pivot = nums[random.randint(fst, lst)]

   while i <= j:

       while nums[i] < pivot: i += 1

       while nums[j] > pivot: j -= 1

       if i <= j:

           nums[i], nums[j] = nums[j], nums[i]

           i, j = i + 1, j - 1

   quicksort(nums, fst, j)

   quicksort(nums, i, lst)

***1.8. ЗАДАНИЕ***

1. Сравнить эффективность прямых методов сортировки (число итераций, сравнений и обменов) для числовых массивов, содержащих различное число элементов (20, 500, 1000, 3000, 5000, 10000), выбираемых случайным образом. Для 20 элементов предусмотреть ввод с клавиатуры. Оценить время сортировки, построить соответствующие таблицы.
2. Исследовать влияние начальной упорядоченности массива (уже отсортированный, отсортированный в обратном порядке, частично отсортированный – при разных размерах отсортированной части 25%, 50% и 75%).
3. Сравнить эффективность быстрой сортировки и прямых методов. Определить размеры массивов, когда прямые методы эффективнее. Составить таблицы, иллюстрирующие сделанные выводы.