**Тема 3. СОРТИРОВКА МАССИВОВ**

В данной теме представлены принципиальные алгоритмы внутренней сортировки. Задача сортировки состоит в упорядочивании элементов в не­убывающем (невозрастающем) порядке. Основным условием, предъявляе­мым к алгоритмам сортировки, является экономное использование доступной памяти. Это предполагает, что перестановки элементов с целью их упорядо­чивания должны выполняться на том же месте, т. е. методы, в которых элементы из массива а передаются в результирующий массив b, представля­ют существенно меньший интерес.

Эффективность алгоритмов сортировки массива можно оценить по та­ким критериям, как число необходимых сравнений элементов (*С*) и число пе­рестановок элементов (*М*).

Эти числа фактически являются функциями от п - числа сортируемых элементов. Хотя хорошие алгоритмы сортировки требуют порядка n \* log(n) сравнений, рассмотрим несколько простых и очевидных методов, назы­ваемых прямыми, где требуется порядка n2 сравнений элементов. Разбор прямых методов целесообразен ввиду следующих причин:

1. Прямые методы особенно удобны для объяснения характерных черт основных принципов большинства сортировок.
2. Программы этих методов легко понимать, и они коротки. Следует помнить, что сами программы также занимают память.
3. Усложненные методы требуют небольшого числа операций, но эти операции обычно сами более сложны, и поэтому для достаточно малых n прямые методы оказываются быстрее, хотя при больших n их использовать, конечно, не следует.

Методы сортировки «на том же месте» можно разбить в соответствии с определяющими их принципами на три основные категории:

* Сортировки с помощью включения (by insertion).
* Сортировки с помощью выбора (by selection).
* Сортировки с помощью обмена (by exchange).

Все рассматриваемые алгоритмы будут оперировать массивом а, в ко­тором будут храниться переставляемые на месте элементы. Данный массив объявляется следующим образом:

int a[n];

1. **Сортировка с помощью прямого включения**

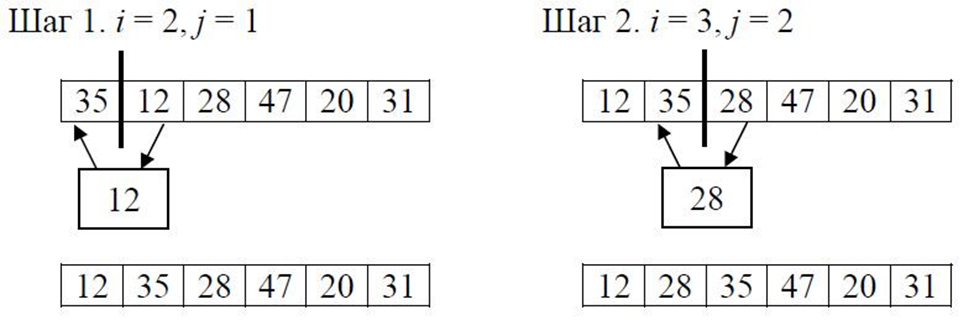
При сортировке элементов массива с помощью прямого включения массив делят на две части: отсортированную или «готовую» а1, ..., ai-1 и неотсортированную или «исходную» аi …, an.

В начале работы алгоритмы в качестве отсортированной части массива принимается только один первый элемент, а в качестве неотсортированной части - все остальные элементы.

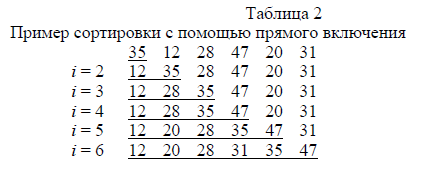
На каждом шаге, начиная с i = 2 и увеличивая i каждый раз на единицу, из неотсортированной последовательности извлекается i-й элемент и встав­ляется в отсортированную так, чтобы не нарушить в ней упорядоченности элементов. Каждый шаг алгоритма включает четыре действия:

1. Взять i-й элемент массива, он же является первым элементом в неотсортированной части, и сохранить его в дополнительной переменной.
2. Найти позицию j в отсортированной части массива, куда будет вставлен i-й элемент, значение которого теперь хранится в дополнительной переменной. Вставка этого элемента не должна нарушить упорядоченности элементов отсортированной части массива.
3. Сдвиг элементов массива с i - 1 позиции по *j-*ю на один элемент вправо, чтобы освободить найденную позицию для вставки.
4. Вставка взятого элемента в найденную j-ю позицию.

На рис.1 приведены два первых шага работы алгоритма сортировки с помощью прямого включения. Отсортированная и неотсортированная части отделены вертикальной чертой.

Рис. 1. Первые два шага алгоритма сортировки с помощью прямого включения

В табл. 2 показан пример работы алгоритма сортировки массива из шести элементов. Отсортированная часть массива подчеркнута.



Анализ алгоритма сортировки с прямым включением. Число сравне­ний ключей (Ci) при i-м проходе самое большее равно i - 1, самое меньшее - 1. Если предположить, что все перестановки из n элементов равновероятны, то среднее число сравнений - i / 2. Число же перестановок *Мi* равно Ci + 2.

Данный алгоритм показывает наилучшие результаты работы в случае уже упорядоченной исходной части массива, наихудшие - когда элементы первоначально расположены в обратном порядке. Алгоритм с прямым вклю­чением можно легко улучшить, если обратить внимание на то, что готовая последовательность, в которую надо вставить новый элемент, сама уже упо­рядочена. Естественным является остановиться на двоичном поиске, при ко­тором делается попытка сравнения с серединой готовой последовательности, а затем процесс деления пополам идет до тех пор, пока не будет найдена точ­ка включения. Такой модифицированный алгоритм сортировки называется методом с двоичным включением (binary insertion).

К несчастью, улучшения, порожденные введением двоичного поиска, касаются лишь числа сравнения, а не числа необходимых перестановок. А поскольку движение элемента, т. е. самого элемента, и связанной с ним информации занимает значительно больше времени, чем сравнение двух ключей, то фактически улучшения не столь уж существенны, ведь важный член М так и продолжает оставаться порядка n . И на самом деле, сортировка уже отсортированного массива потребует больше времени, чем в случае по­следовательной сортировки с прямыми включениями.

Этот пример показывает, что «очевидные улучшения» часто дают не столь уж большой выигрыш, как это кажется на первый взгляд, а в некоторых случаях (случающихся на самом деле) эти «улучшения» могут фактически привести к ухудшениям. После всего сказанного сортировка с помощью включения уже не кажется столь удобным методом для цифровых машин: включение одного элемента с последующим сдвигом на одну позицию целой группы элементов не экономно. Остается впечатление, что лучший результат дадут методы, где передвижка, пусть и на большие расстояния, будет связана лишь с одним-единственным элементом.

1. **Сортировка с помощью прямого выбора**

При сортировке с помощью прямого выбора массив также делится на две части: отсортированную или «готовую» последовательность а1, ..., ai-1 и неотсортированную или «исходную» - аь ., an.

Метод прямого выбора в некотором смысле противоположен методу прямого включения. При прямом включении на каждом шаге рассматрива­ются только один (первый) элемент исходной последовательности и все эле­менты готовой последовательности. При этом отыскивается точка включения этого элемента в «готовую» последовательность так, чтобы при вставке не нарушить ее упорядоченности.

При прямом же выборе происходит поиск одного элемента из исходной последовательности, который обладает наименьшим (наибольшим) значением, и уже найденный элемент помещается в конец готовой последовательности.

На момент начала сортировки методом прямого выбора готовая после­довательность считается пустой, соответственно, исходная последователь­ность включает в себя все элементы массива.

Алгоритм сортировки с помощью прямого выбора можно описать сле­дующим образом:

1. Из всего массива выбирается элемент с наименьшим значением.
2. Он меняется местами с первым элементом а1.
3. Затем этот процесс повторяется с оставшимися n - 1 элементами, n - 2 элементами и т. д. до тех пор, пока не останется один элемент с наи­большим значением.

Рассмотрим несколько первых шагов алгоритма сортировки с помо­щью прямого выбора на примере упорядочивания по возрастанию следую­щего массива:

35 12 28 47 20 31

На первом шаге готовая последовательность не содержит ни одного элемента. Выбираем наименьший элемент из исходной последовательности, куда пока что входят все элементы массива. Таким элементом является вто­рой элемент массива, значение которого - 12. Он меняется местами с первым элементом.

Теперь готовая последовательность включает в себя один элемент - 12. На рис. 2, иллюстрирующем текущее состояние массива, готовая последо­вательность подчеркнута сплошной линией. Наименьший элемент исходной последовательности - 20, он должен занять место второго элемента. Оба эти элемента выделены на рис. 11 полужирным шрифтом.

12 35 28 47 20 31

Рис. 2. Первый шаг алгоритма сортировки с помощью прямого выбора

На втором шаге готовая последовательность состоит уже из двух эле­ментов: 12 и 20. Наименьший элемент исходной последовательности - 28. Он является третьим элементом массива и именно эту позицию он должен занять. Поэтому на рис. 3 полужирным шрифтом выделен только один дан­ный элемент.

12 20 28 47 35 31

Рис. 3. Второй шаг алгоритма сортировки с помощью прямого выбора

В табл. 3 наглядно показаны все шаги алгоритма сортировки с помо­щью прямого выбора: отмечены готовые последовательности и переставляе­мые элементы.

Таблица 3

Пример сортировки с помощью прямого выбора

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 35 | 12 | 28 | 47 | 20 | 31 |
| i = 2 | 12 | 35 | 28 | 47 | 20 | 31 |
| i = 3 | 12 | 20 | 28 | 47 | 35 | 31 |
| i = 4 | 12 | 20 | 28 | 47 | 35 | 31 |
| i = 5 | 12 | 20 | 28 | 31 | 35 | 47 |
| i = 6 | 12 | 20 | 28 | 31 | 35 | 47 |

Анализ алгоритма сортировки с прямым выбором. При работе дан­ного алгоритма число сравнений элементов (С) не зависит от начального по­рядка элементов. Можно сказать, что в этом смысле поведение данного ме­тода менее естественно, чем поведение прямого включения. Для С имеем

С = (n2 - n) / 2.

В случае изначально упорядоченных элементов число перестановок М минимально:

Mmin = 3 \* (n - l).

Если же первоначально элементы располагались в обратном порядке, число перестановок будет максимальным:

Mmax = n2 / 4 + 3 \* (n - 1).

В общем случае алгоритм с прямым выбором, как правило, предпочти­тельнее алгоритма прямого включения. Однако если элементы в начале упо­рядочены или почти упорядочены, алгоритм с прямым включением выпол­нит сортировку быстрее.

1. **Сортировка с помощью прямого обмена**

Алгоритм прямого обмена основывается на сравнении и перестановке пары соседних элементов и продолжении этого процесса до тех пор, пока не будут упорядочены все элементы. Обмен местами двух элементов представ­ляет собой характерную особенность данного метода, хотя, конечно, в обоих рассматривавшихся до этого методах переставляемые элементы также «обменивались» местами.

1. **Пузырьковая сортировка**

Как и в методе прямого выбора при сортировке данным методом вы­полняется ряд проходов по массиву, сдвигая каждый раз наименьший эле­мент оставшейся последовательности к началу массива. Однако в отличие от рассмотренных ранее методов при пузырьковой сортировке происходит сравнивание и перестановка только соседних двух элементов.

Если расположить элементы массива вертикально, то за первый проход элемент с наименьшим значением, т. е. самый «легкий» элемент, поднимется на самый верх массива и станет его первым элементом. В результате сле­дующего прохода второй по «легкости» элемент поднимется к началу масси­ва и станет его вторым элементом. Такое продвижение элементов по массиву вызывает ассоциации с пузырьком воздуха, всплывающим в воде. Отсюда и название данного метода - метод «пузырька» или «пузырьковая сортировка».

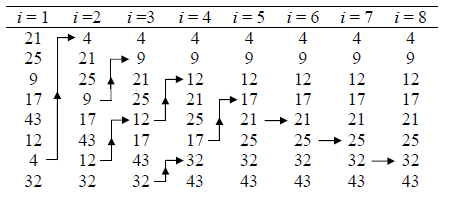
Описать алгоритм сортировки методом пузырька можно следующим образом. На i-ом проходе алгоритма (1 ≤ i ≤ n) рассматриваются в обратном порядке элементы массива с n-го по i-й включительно. Сравниваются только соседние элементы. При этом, если производится сортировка по возрастанию и элемент a[j] оказывается меньше предшествующего ему a[j - 1], то они ме­няются местами.

Если говорить терминами «готовой» и «исходной» последовательно­стей, то элементы до i-го представляют собой готовую последовательность, а с i-го по n-й - исходную. В начале готовая последовательность пуста. По мере работы алгоритма элементы «всплывают» из исходной последовательности и занимают место в конце готовой.

В табл. 4 показана работа алгоритма сортировки методом пузырька. Перемещение элементов к началу массива показано стрелками.

Таблица 4

Пример пузырьковой сортировки



Улучшения этого алгоритма напрашиваются сами собой. На примере в табл. 4 видно, что три последних прохода не влияют на порядок элементов поскольку, они уже отсортированы. Очевидный прием улучшения этого алгоритма - запоминать, были или не были перестановки в процессе некоторо­го прохода. Если в последнем проходе перестановок не было, то работа алго­ритма может быть завершена.

Это улучшение, однако, можно опять же улучшить, если запоминать не только сам факт, что обмен имел место, но и положение (индекс) последнего обмена. Ясно, что все пары соседних элементов выше этого индекса k уже упорядочены. Поэтому просмотр можно заканчивать на этом индексе, а не идти до заранее определенного нижнего предела *i*.

1. **Шейкерная сортировка**

Пример пузырьковой сортировки, представленный в табл. 4, отражает некоторую своеобразную асимметрию работы алгоритма: легкий пузырек всплывает сразу - за один проход, а тяжелый тонет очень медленно - за один проход на одну позицию. Например, массив

15 29 31 55 70 93 8

с помощью пузырьковой сортировки будет упорядочен за один проход, а для сортировки массива

93 8 15 29 31 55 70

потребуется шесть проходов. Это наводит на мысль о следующем улучше­нии: чередовать направление просмотра на каждом последующем проходе.

Алгоритм, реализующий такой подход, называется «шейкерной сортировкой». Табл. 5 иллюстрирует сортировку данным методом тех же (табл. 4) восьми элементов.



Переменные L и R содержат индексы элементов, до которых дол­жен происходить просмотр при движении влево (или вверх) и вправо (или вниз) соответственно.

Если ввести переменную, которая будет отвечать за то, были ли перестановки элементов, то алгоритм шейкерной сортировки выполнит сортировку за пять проходов, в то время как для пузырьковой сортировки без улучшений потребовалось бы семь проходов.

Если же к алгоритму шейкерной сортировки добавить переменную k, содержащую индекс последнего перестановленного элемента, то число проходов сократится до четырех, как показано на примере в табл. 5.

Анализ пузырьковой и шейкерной сортировок. Число сравнений в базовом обменном алгоритме – алгоритме пузырьковой сортировке – равно

С = (n2 – n) / 2,

а минимальное и максимальное число перестановок элементов равно

Mmin = 0, Мmax = 3 \* (n2 – n) / 4.

Анализ же улучшенных методов, особенно шейкерной сортировки, довольно сложен. Стоит отметить, что все перечисленные выше усовершенствования не влияют на число перемещений, они лишь сокращают число излишних проверок. К несчастью, обмен местами двух элементов – чаще всего более дорогостоящая операция, чем их сравнение. Поэтому очевидные на первый взгляд улучшения дают не такой уж большой выигрыш, как ожидалось.

Шейкерная сортировка с успехом используется лишь в тех случаях, когда известно, что элементы почти упорядочены, что на практике бывает весьма редко. Анализ показывает, что «обменная» сортировка и ее усовершенствования фактически оказываются хуже сортировок с помощью включений и с помощью выбора.

**3.4. Сортировка Шелла**

Все методы прямой сортировки фактически передвигают каждый эле­мент на всяком элементарном шаге на одну позицию. Поэтому они требуют порядка n2 таких шагов. Следовательно, в основу любых улучшений алго­ритмов сортировки должен быть положен принцип перемещения на каждом такте элементов на большие расстояния. Можно показать, что среднее рас­стояние, на которое должен продвигаться каждый из n элементов во время сортировки, равно n / 3 позиций. Эта цифра является целью в поиске улуч­шений, т. е. в поиске более эффективных методов сортировки.

Рассмотрим улучшенный метод, основанный на методе прямого включения - сортировке с помощью включений с уменьшающимися расстояниями. В 1959 г. Д. Шеллом было предложено усовершенствование алгоритма сортировки с помощью прямого включения. Рассмотрим его рабо­ту на примере следующего массива:

35 28 49 79 45 11 89 70 91 67 54 19 13 24

Сначала отдельно группируются элементы, отстоящие друг от друга на расстоянии 4. Таких групп будет четыре, они показаны на рис. 13 (а - г). Элементы, принадлежащие одной группе, объединены дугами.

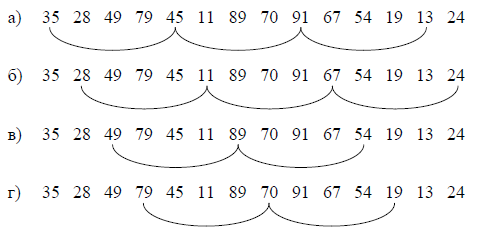


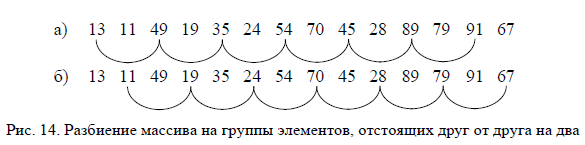
Рис. 13. Разбиение массива на группы элементов, отстоящих друг от друга на четыре

Следующим шагом выполняется сортировка внутри каждой из групп методом прямого включения. Сначала сортируются элементы 35, 45, 91, 13, затем 28, 11, 67, 24, следом 49, 89, 54, и, наконец, 79, 70, 19. В результате по­лучаем массив:

13 11 49 19 35 24 54 70 45 28 89 79 91 67

Такой процесс называется четверной сортировкой. Следует отметить, что алгоритм сортировки Шелла также является алгоритмом сортировки «на месте». Поэтому все перестановки происходят в одном и том же массиве.

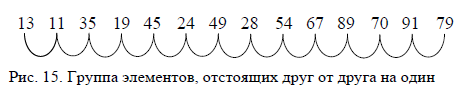
Следующим проходом элементы группируются так, что теперь в одну группу входят элементы, отстоящие друг от друг на две позиции. Таких групп две, они показаны на рис. 14 (а - б).



Вновь в каждой группе выполняется сортировка с помощью прямого включения. Это называется двойной сортировкой, ее результатом будет массив:

1. 11 35 19 45 24 49 28 54 67 89 70 91 79

И наконец, на третьем проходе идет обычная или одинарная сортировка с помощью прямого включения (рис. 15).



Результатом работы алгоритма сортировки Шелла является отсортиро­ванный массив

11 13 19 24 28 35 45 49 54 67 70 79 89 91

На первый взгляд можно засомневаться: если необходимо несколько процессов сортировки, причем и каждый включаются все элементы, то не добавят ли они больше работы, чем сэкономят? Однако на каждом этапе либо сортируется относительно мало элементов, либо элементы уже довольно хо­рошо упорядочены и требуется сравнительно немного перестановок.

Очевидно, что такой метод в результате дает упорядоченный массив. Видно также, что каждый проход от предыдущих только выигрывает (т. к. каждая i-сортировка объединяет две группы, уже отсортированные 2i-сортировкой).

Расстояния в группах можно уменьшать по-разному, лишь бы послед­нее было единичным, ведь в самом плохом случае последний проход и сде­лает всю работу. Однако совсем не очевидно, что такой прием «уменьшаю­щихся расстояний» может дать лучшие результаты, если расстояния не будут степенями двойки. При выполнении лабораторных работ рекомендуется на­чальный шаг взять равным hнач = n / 3, где n - количество элементов массива, и уменьшать его на каждом проходе вдвое:

hi -1 = hi / 2.

Как отмечалось выше *hконеч* = 1.

Анализ сортировки Шелла. Анализ этого алгоритма поставил не­сколько весьма трудных математических проблем, многие из которых так еще и не решены. В частности, не известно, какие расстояния дают наилуч­шие результаты. Известно, однако, что выбор расстояний должен быть та­ким, чтобы взаимодействие различных цепочек проходило как можно чаще.

В работе Кнут [3] показал, что имеет смысл использовать следующую последовательность (она записана в обратном порядке): 1, 4, 13, 40, 121, ..., где hk-1 = 3hk + 1, ht (или hконеч) = 1 и t = log3(n) - 1. Он рекомендует и другую последовательность: 1, 3, 7, 15, 31, ... где hk-1 = 2hk + 1, ht = 1 и t = log2(n) - 1.

Математический анализ показывает, что в последнем случае для сортировки

n элементов методом Шелла затраты пропорциональны n1.2 , что значительно лучше n2, необходимых для методов прямой сортировки.

1. **Сравнение различных алгоритмов сортировки**

Проанализируем эффективность различных алгоритмов сортировки массивов, состоящих из n сортируемых элементов. Анализ будет проводиться по двум критериям: числу необходимых сравнений элементов С и числу пе­рестановок элементов М. Для всех прямых методов сортировки можно дать точные аналитические формулы. Они приведены в табл. 6.

Таблица 6

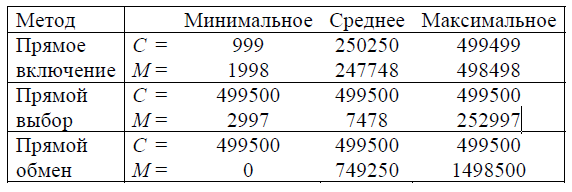
Сравнение прямых методов сортировки



В табл. 7 приведены фактические значения показателей C и M для массива с числом элементов равным тысячи.

Таблица 7

Значения показателей C и M при n = 1000



Для сортировки Шелла, как и для других усовершенствованных мето­дов, простых и точных формул не существует. Однако, в случае сортировки Шелла вычислительные затраты составляют в среднем n1.2 , в то время как для прямых методов затраты составили бы n2.

Для практических целей полезно иметь некоторые экспериментальные данные об эффективности того или иного алгоритма. В табл. 8 показано вре­мя работы (в секундах), обсуждавшихся выше методов сортировки, реализованных на персональной ЭВМ Lilith. Три столбца содержат времена сортировки уже упорядоченного массива, массива со случайными числами и массива, расположенного в обратном порядке. В начале приводятся цифры для массива, состоящего из 256 элементов, а затем - из 2048 элементов.

|  |  |
| --- | --- |
| Метод | Массив |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Упорядоченный | Случайный | В обратном порядке |

|  |  |  |  |
| --- | --- | --- | --- |
| n = 256 | | | |
| Прямое включение | 0.02 | 0.82 | 1.64 |
| Двоичное включение | 0.12 | 0.70 | 1.30 |
| Прямой выбор | 0.94 | 0.96 | 1.18 |
| Метод пузырька | 1.26 | 2.04 | 2.80 |
| Метод шейкера | 0.02 | 1.66 | 2.92 |
| Сортировка Шелла | 0.10 | 0.24 | 0.28 |
| n = 2048 | | | |
| Прямое включение | 0.22 | 50.74 | 103.80 |
| Двоичное включение | 1.16 | 37.66 | 76.06 |
| Прямой выбор | 58.18 | 58.34 | 73.46 |
| Метод пузырька | 80.18 | 128.84 | 178.66 |
| Метод шейкера | 0.16 | 104.44 | 187.36 |
| Сортировка Шелла | 0.80 | 7.08 | 12.34 |

Таблица 8

Очевидно преимущество сортировки Шелла, как улучшенного метода, по сравнению с прямыми методами сортировки. Кроме того, можно отметить следующее:

Время работы различных методов

1. Улучшение двоичного включения по сравнению с прямым включе­нием в действительности почти ничего не даст, а в случае упорядоченного массива даже получается отрицательный эффект.
2. Пузырьковая сортировка определенно наихудшая из всех сравни­ваемых. Ее усовершенствованная версия - шейкерная сортировка - продол­жает оставаться плохой по сравнению с прямым включением и прямым выбо­ром (за исключением патологического случая уже упорядоченного массива).