Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

**Реферат на тему:**

**“Сортировка методом Шелла”**

Выполнил: студент 1 курса 141 группы  
направления 010500.62 Математическое обеспечение и администрирование информационных систем (профиль Параллельное программирование)

факультета компьютерных наук и информационных технологий

Акимов Артемий Андреевич

Саратов 2014

1. **Введение**

Данная сортировка является улучшением сортировки вставками.

Для алгоритма сортировки, который каждый раз перемещает запись только на одну позицию, среднее время будет в лучшем случае пропорционально N2, потому что в процессе сортировки каждая запись должна пройти в среднем 1/3N позиций. Поэтому, если желательно получить метод, существенно превосходящий по скорости метод простых вставок, необходим механизм чтобы записи могли перемещаться большими скачками, а не короткими шажками.

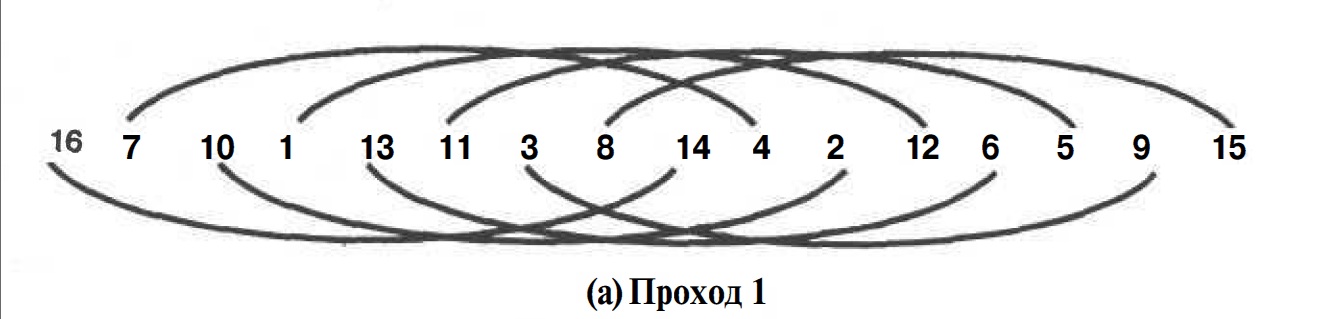
Такой метод предложен в 1959 году Дональдом Л. Шеллом и известен во всем мире под именем своего автора.

Сортировка вставками не относится к категории быстродействующих, поскольку единственный вид операции обмена, который она использует, выполняется над двумя соседними элементами, в связи с чем элемент может передвигаться вдоль массива лишь на одно место за один раз. Например, если элемент с наименьшим значением ключа оказывается в конце массива, потребуется сделать N шагов, чтобы поместить его в надлежащее место. Сортировка методом Шелла представляет собой простейшее расширение метода вставок, быстродействие которого выше за счет обеспечения возможности обмена местами элементов, которые находятся далеко один от друго­го.

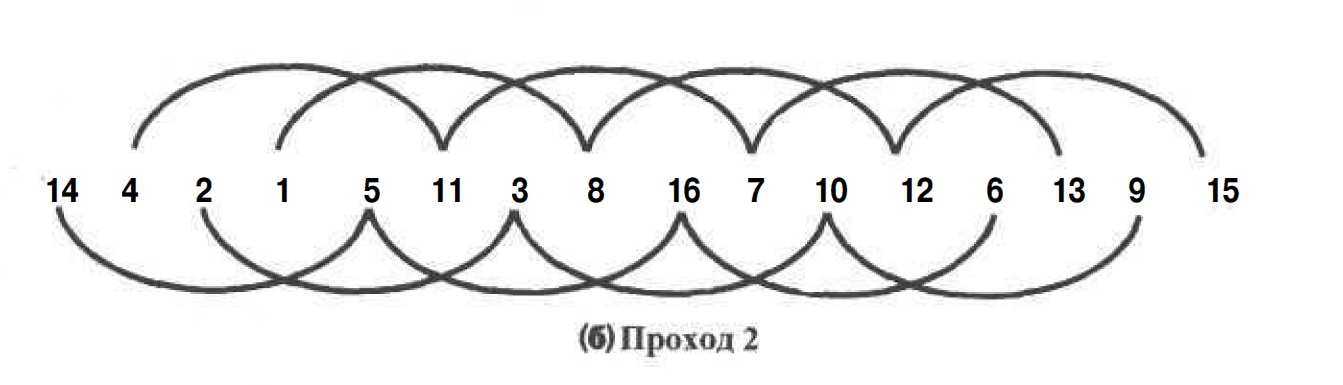
1. **Разбор сортировки**

Идея заключается в переупорядочении файла та­ким образом, чтобы придать ему следующее свой­ство: совокупность h-ых элементов исходного фай­ла (начиная с любого) образует отсортированный файл. В таком случае говорят, что файл h-упорядочен (h-sorted). Другими словами, h-упорядоченный файл представляет собой h независимо отсортированных взаимно проникающих друг в друга файлов. В про­цессе h-сортировки при достаточно больших значе­ниях h можно менять местами элементы массива, расположенные достаточно далеко друг от друга, и тем самым ускорить последующую h-сортировку при меньших значениях h. Использование такого рода процедуры для любой последовательности значений h, которая заканчивается единицей, завершается получением упорядоченного файла: именно в этом и заключается суть сортировки методом Шелла.

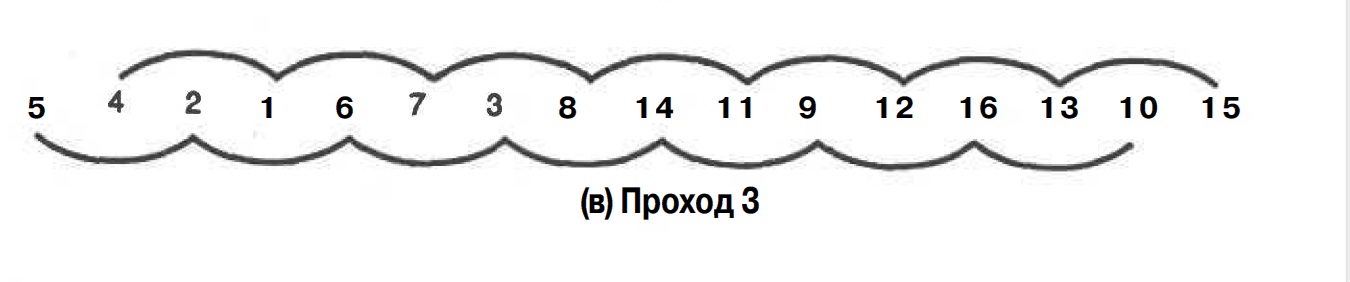
Для примера возьмем файл из 16 элементов. Сначала просматриваются пары с шагом 8. Это пары элементов 1-9, 2-10, 3-11, 4-12, 5-13, 6-14, 7-15, 8-16. Если значения элементов в паре не упорядочены по возрастанию, то элементы меняются местами. Назовем этот этап 8-сортировкой.



Второй этап — сортировка со смещением 4, на котором элементы в файле делятся на четверки: 1-5-9-13, 2-6-10-14, 3-7-11-15, 4-8-12-16. Выполняется сортировка в каждой четверке.



Третий этап — сортировка со смещением 2, когда элементы в файле делятся на 2 группы по 8: 1-3-5-7-9-11-13-15 и 2-4-6-8-10-12-14-16. Выполняется сортировка в каждой восьмерке.



Четвертый этап - наконец весь файл упорядочивается методом вставок (сортировка со смещением 1). Поскольку дальние элементы уже переместились на свое место или находятся вблизи от него, этот этап будет значительно менее трудоемким, чем при сортировке вставками без предварительных «дальних» обменов.

Очевидно, лишь последняя сортировка необходима, чтобы расположить все элементы по своим местам. Так зачем нужны остальные? На самом деле они продвигают элементы максимально близко к соответствующим позициям, так что в последней стадии число перемещений будет весьма невелико. Последовательность и так почти отсортирована. Ускорение подтверждено многочисленными исследованиями и на практике оказывается довольно существенным.

Метод сортировки Шелла ещё называется с «убывающим смещением», поскольку каждый проход характеризуется смещением h, таким, что сортируются записи, каждая из которых отстоит от предыдущей на h позиций. Последовательность значений смещений 8, 4, 2, 1 не следует считать неизменной, можно пользоваться любой последовательностью смещений ht-1, ht-2, …, h0. но последнее смещение h0 должно быть равно 1. Например, со смещениями 7, 5, 3, 1. Выбор значений смещений на последовательных проходах имеет весьма существенное значение для скорости сортировки.

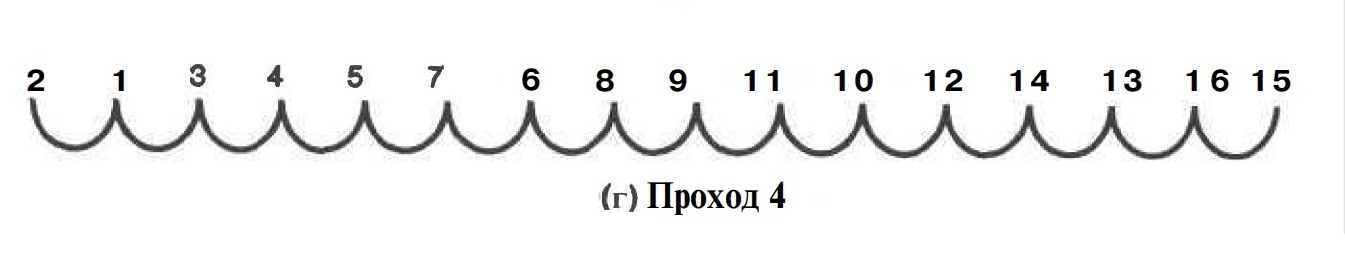
Использованный в примере набор ..., 8, 4, 2, 1 - неплохой выбор, особенно, когда количество элементов - степень двойки. Однако гораздо лучший вариант предложил Р.Седжвик. Его последовательность имеет вид

ins[s] = 9\*2s – 9\*2s/2 + 1, если s четно

ins[s] = 8\*2s – 6\*2(s+1) /2 +1, если s нечетно

При использовании таких приращений среднее количество операций: O(N7/6), в худшем случае - порядка O(N4/3).

Один из способов реализации сортировки мето­дом Шелла заключается в том, что для каждого h независимо используется сортировка вставками на каждом из h подфайлов. Несмотря на очевидную простоту этого процесса, возможен еще более про­стой подход именно благодаря тому, что подфайлы независимы. В процессе h-сортировки файла мы просто вставляем элемент среди предшествующих ему элементов в соответствующем h-подфайле, пе­ремещая большие по значению элементы в право. Эта задача решается путем использова­ния программы сортировки вставками, при которой каждый шаг перемещения по файлу в сторону уве­личения или уменьшения равен h, но не равен 1.



С учетом данного обстоятельства программная реали­зация сортировки методом Шелла сводится всего лишь к тому, что для каждого значения шага осуще­ствляется проход по файлу, характерный для сорти­ровки вставками. Работа программы показана на рисунке ниже:

Возникает вопрос: какую последовательность шагов следует использовать? В общем случае на этот вопрос трудно найти правильный ответ. В литерату­ре опубликованы результаты исследований различ­ных последовательностей шагов, некоторые из них хорошо зарекомендовали себя на практике, однако наилучшую последовательность, по-видимому, отыскать не удалось. В общем случае на практике используются убыва­ющие последовательности шагов, близкие к геометрической прогрессии, в результате чего число шагов находится в ло­гарифмической зависимости от размеров файлов. Напри­мер, если размер следующего шага равен примерно поло­вине предыдущего, то для сортировки файла, состоящего из 1 миллиона элементов, потребуется примерно 20 шагов, если такое соотношение примерно равно одной четвертой, то достаточно будет 10 шагов. Использование как можно меньшего числа шагов — это весьма важное требование, которое нетрудно учесть, но при этом в последовательно­сти шагов необходимо выдерживать различные арифмети­ческие соотношения, такие как величины их общих дели­телей и ряд других свойств.

Практический результат от обнаружения хорошей пос­ледовательности шагов, по-видимому, ограничен повыше­нием быстродействия алгоритма на 25%, в то же время сама проблема представляет собой увлекательную загадку — примером того, какие сложные вопросы характерны для использования на первый взгляд простого алгоритма.

Последовательность шагов 1 4 13 40 121 364 1093 3280 9841 ..., в которой соотношение соседних шагов составляет приблизительно одну треть, была рекомендована Кнутом в 1969 г. Она просто вы­числяется (начав с 1, получить значение следующего шага, умножив предыдущее зна­чение на 3 и добавив 1) и обеспечивает реализацию сравнительно эффективной сор­тировки даже в случае относительно больших файлов.

Многие другие последовательности шагов позволяют получить еще более эффективную сортировку. Одной из та­ких последовательностей является 1 8 23 77 281 1073 4193 16577 ..., т.е. последова­тельность 4i+1 + 3 • 2i + 1 для i> 0. Приведенная последователь­ность обеспечивает повышенное быстродействие для самых трудных случаев сортировки. Эта последовательность, равно как и последовательность Кнута, а также многие другие последовательности шагов, обладают похожими динамическими характеристиками для файлов больших размеров. Вполне возможно, что существуют лучшие последовательности.

С другой стороны, существуют и плохие последовательности шагов: например, 1 2 4 8 16 32 64 128 256 512 1024 2048 ... (первая последовательность шагов, предложенная Шеллом еще в 1959 г. (см. раздел ссылок)), скорее всего, служит причиной низкой эффективности сортировки, поскольку элементы на нечетных позициях не сравниваются с элементами на четных позициях вплоть до последнего прохода. Этот эффект заметен на файлах с произвольной организацией, он становится катастрофи­ческим в наихудших случаях: эффективность метода резко снижается и время выпол­нения сортировки становится пропорциональным квадрату N, если, например, поло­вина элементов файла с меньшими значениями находится в четных позициях, а другая половина элементов (с большими значениями) — в нечетных позициях.

# **Анализ времени работы**

**Анализ Метода Шелла**

Для рационального выбора последовательности значений смещений сортировки ht-1…, h0 для алгоритма нужно проанализировать время выполнения как функцию от этих смещений. Такой анализ приводит к постановки очень красивых, но еще не до конца решенных математических задач; никому до сих пор не удалось найти наилучшею возможную последовательность смещений для больших N. Тем не менее известно довольно много интересных свойств сортировки методом Шелла с убывающим смещением.

Метод, предложенный Дональдом Л. Шеллом, является неустойчивой сортировкой по месту. Эффективность метода Шелла объясняется тем, что сдвигаемые элементы быстро попадают на нужные места. Среднее время для сортировки Шелла равняется O(N1.25), для худшего случая оценкой является O(N1.5). Также следует сказать, что дополнительной памяти данный алгоритм не требует, но и не гарантирует сохранение порядка элементов с одинаковыми значениями.

Время выполнения сортировки, пропорциональная N1.2 значительно лучше квадратичной зависимости N2, которой подчиняется большинство простых алгоритмов сортировки. Однако, прежде чем вы решите использовать сортировку Шелла, следует иметь в виду, что быстрая сортировка дает даже еще лучшие результаты.

Наше описание эффективности сортировки методом Шелла не отличается особой точностью, поскольку никому еще не удалось выполнить точный анализ этого алгоритма. Этот пробел в наших знаниях затрудняет не только оценку различных последовательностей шагов, но и аналитическое сравнение сортировки методом Шелла с другими методами сортировки. Не известна даже функциональная форма для опре­деления времени выполнения сортировки методом Шелла (более того, эта форма зависит от выбора последовательности шагов). Кнут обнаружил, что функциональные формы N(logN)2 и N1.25 довольно точно описывают ситуацию, а дальнейшие иссле­дования показали, что для некоторых видов последовательностей подходят более сложные функции вида .

**Лемма**. Сортировка методом Шелла выполняет менее O(N3/2) операций сравнения для последовательности шагов 1 4 13 40 121 364 1093 3280 9841...

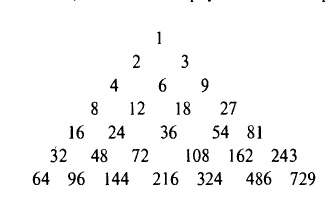
**Лемма**. Сортировка методом Шелла выполняет менее O(N4/3) операций сравне­ния для последовательности шагов 1 8 23 77 281 1073 4193 16577...

Последовательности шагов, которые рассматривались до сих пор, эффективны в силу того, что следующие один за другим элементы последовательности взаимно про­сты. Другое семейство последовательностей шагов эффективно именно благодаря тому, что такие элементы не являются взаимно простыми.

В частности, следует, что в процессе завершающей сортировки вставками 2- и 3-упорядоченного файла каждый элемент перемещается самое большее на одну позицию. Это значит, что такой файл может быть отсортиро­ван за один проход пузырьковой сортировки (отпадает необходимость в дополнитель­ном цикле). Теперь, если файл 4-упорядочен и 6-упорядочен, то из этого следует, что каждый элемент перемещается максимум на одну позицию, если выполнить его 2-упорядочение (поскольку каждый подфайл 2- и 3-упорядочен); и если файл 6-упоря­дочен и 9-упорядочен, то каждый элемент перемещается самое большее на одну по­зицию при его 3-сортировке. Продолжая рассуждения в этом направлении, мы приходим к идее, которую Пратт опубликовал в 1971 г. (см. раздел ссылок).

Метода Пратта основан на использовании треугольника шагов, причем каждое входящее в этот треугольник число в два раза больше числа, стоящего сверху спра­ва, и в три раза больше числа, стоящего в треугольнике сверху слева.

|  |  |
| --- | --- |
|  | |
|  | |  | |



Если мы используем эти числа снизу-вверх и справа налево как последователь­ность шагов в рамках сортировки методом Шелла, то каждому шагу х в нижнем ряду предшествуют значения 2х и Зх, так что каждый подфайл оказывается 2-упорядочен и 3-упорядочен и ни один элемент не передвигается больше, чем на одну позицию в процессе всей сортировки!

**Лемма.** Сортировка методом Шелла выполняет менее О(N (logN)2) операций сравнения для последовательности шагов 1 2 3 4 6 9 8 12 18 27 16 24 36 54 81...

Число шагов из треугольника, которое меньше N по величине, и подавно будет меньше (log2N)2.

Последовательности шагов, предложенные Праттом, на практике проявляют тенденцию работать хуже, чем другие, поскольку, их слишком много.

Проблема построения хорошей последовательности шагов для сортировки мето­дом Шелла представляет собой прекрасный пример сложного поведения простых алгоритмов.

# **Заключение**

Сортировка методом Шелла выбирается для многих приложений, поскольку она обеспечивает приемлемое время сортировки даже для достаточно больших файлов и реализуется в виде компактной программы, которую легко заставить работать. Другие методы сортировки, возможно, более эффективны, но работают всего лишь в два раза быстрее (а то и того меньше) при небольших значениях N, но при этом они существенно сложнее. Поэтому, если необходимо решить проблему сортировки, но нет желания изучать интерфейс системной сортировки, следует воспользоваться сортировкой методом Шелла.

# **5. Приложение**

*Примеры программ (на C++)*

1. *Выбор длины по Седжвику*

#include <iostream>

#include <ctime>

using namespace std;

int increment(int inc[], int size) {

int p1, p2, p3, s;

p1 = p2 = p3 = 1;

s = -1;

do {

if (++s % 2) {

inc[s] = 8\*p1 - 6\*p2 + 1;

} else {

inc[s] = 9\*p1 - 9\*p3 + 1;

p2 \*= 2;

p3 \*= 2;

}

p1 \*= 2;

} while(3\*inc[s] < size);

return s > 0 ? --s : 0;

}

int main() {

int inc,size, i, j, seq[40];

int n;

cout<<"Power= ";

cin >> n;

int \*a = new int [n];

for (int i = 0; i < n; i++)

cin >> a[i];

int s;

s = increment(seq, n);

while (s >= 0) {

inc = seq[s--];

for (i = inc; i < n; i++) {

int temp = a[i];

for (j = i-inc; (j >= 0) && (a[j] > temp); j -= inc)

a[j+inc] = a[j];

a[j+inc] = temp;

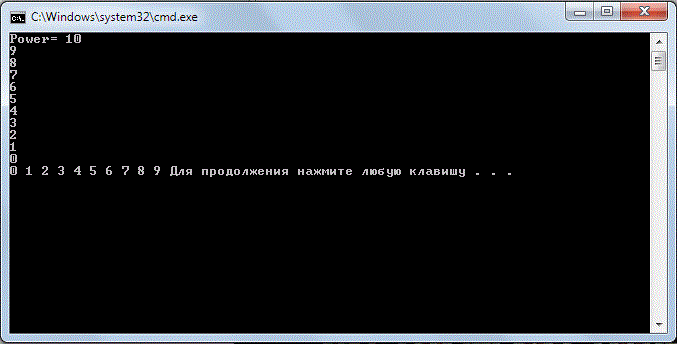
}

}

for (int i = 0; i < n; i++)

cout << a[i] << ' ';

return 0;





1. *Выбор длины по Шеллу (стандартный способ)*

#include <iostream>

using namespace std;

int main()

{

int \*a,n,temp;

cin >> n;

a=new int[n];

for (int i = 0; i < n; i++)

{

cin >> a[i];

}

int inc = n / 2;

while (inc > 0)

{

for (int i = inc; i < n; i++)

{

int j = i-inc;

while (j >= 0 )

{if (a[j]>a[j+inc]

)

{temp=a[j];

a[j]=a[j+inc];

a[j+inc]=temp;

j=j-inc;

}

else j=-1;

}

}

inc = inc / 2;

}

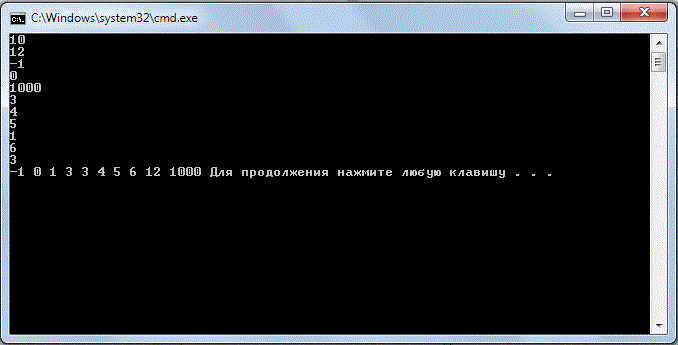
for (int i = 0; i < n; i++)

{

cout << a[i] << ' ';

}

return 0;

**

****

*4.Сортировка в танце.*

Интересный пример работы сортировки Шелла в исполнении венгерских программистов.

****

Видео:****

(Если не установлено приложение для просмотра видеофайлов с расширением \*.3gp, см. Используемый материал, пк. 5, 6.)

**Список литературы:**

1. Огнева М.В., Кудрина Е.В.

Основы программирования на языке С++: Учеб. Пособие в 2ч. Часть 2. – Саратов: ООО Издательский Центр “Наука”, 2009. – 100 c.

1. Р. Седжвик

Фундаментальные алгоритмы на С++. Анализ/Структуры данных/ Сортировка/Поиск: Пер. с англ./Роберт Седжвик. - К.: Издательство «ДиаСофт», 2001.- 688 с.

1. Дж. Макконнелл

Основы современных алгоритмов. 2-е дополненное издание Москва:

Техносфера, 2004. - 368с.

1. <http://algolist.manual.ru/sort/shell_sort.php>
2. <http://www.youtube.com/watch?v=CmPA7zE8mx0>

или

1. <http://windowsplayer.ru/download/> (3gp player)