УДК 681.142.2(075.8)

А. Ф. Деон

D-ПОСЛЕДОВАТЕЛЬНОСТИ В БЫСТРОЙ СОРТИРОВКЕ ХОАРА

Рассмотрены вопросы формирования длительных по количеству компьютерных операций D-последовательностей для определения скоростных свойств быстрой сортировки Хоара на одномерных массивах целочисленной информации.

E-mail: deonalex@mail.ru

Ключевые слова: *быстродействие, алгоритм, числа, размерность.*

Алгоритм быстрой сортировки Хоара относится к группе сложных методов упорядочивания массивов вместе с методом слияния, алгоритмом Шелла, методами древовидных сортировок. Для больших массивов все они превосходят по быстродействию простые методы минимакса, перестановки, парных сравнений, сдвига и вставки. Формульный анализ количества выполняемых операций включает минимальные и максимальные оценки. Максимальное количество операций в быстрой сортировке Хоара дают длительные по числу операций *D*-последовательности исходных данных, подлежащих сортировке.

Разделяющий элемент Хоара. Пусть имеется упорядоченный массив, например по возрастанию. Назовем элемент массива *разделяющим*, если слева от него находятся элементы с меньшими или равными значениями. Тогда справа будут находиться элементы с большими или равными значениями. В упорядоченном массиве любой элемент является разделяющим. Но могут существовать массивы, когда левые и правые части относительно разделяющего элемента неупорядочены. В этом случае разделяющий элемент точно находится на том же месте в целиком упорядоченном массиве. Это следствие несложной теоремы, доказательство которой начинается с утверждения, что любая непрерывная часть упорядоченного массива упорядочена. Хоар (Hoare) предложил алгоритм поиска такого разделяющего элемента в произвольном массиве. Это означает, что появилась возможность сразу определить место разделяющего элемента в будущем упорядоченном массиве.

Ниже представлена программа *DH01*, в которой функция *HoarePosPrint*( ) определяет позицию разделяющего элемента и предоставляет распечатки дополнительной информации по ходу поиска:

```
// Program DH01 (Win32)
// Разделяющий элемент в быстрой сортировке Хоара
#include <conio.h>                                    // _getch
```

```
#include <iostream>
using namespace std;
#include <iomanip>                                          // setw
template<typename Type >
int inline HoarePosPrint( Type* const a,int left,int right )
{
Type v = a[right];                  // значение разделяющего элемента
int i = left;                                  // начало поиска слева
int j = right - 1;                            // начало поиска справа
int m = j;                                        // рабочий индекс
Type r;                                       // рабочая переменная
while( true )
{
while( a[i] < v &&i< m ) ++i;                  // меньший элемент
while( v < a[j] && j > left ) –j;               // больший элемент
cout<< "i = " <<i<< " j = " << j <<endl;
if(i>= j ) break;                                 // обмен не нужен
if( a[i] == a[j] ) break;                         // обмен не нужен
r = a[i];                                         // обмен a[i] и a[j]
a[i] = a[j];
a[j] = r;
for(int k = left; k <= right; k++ )
cout<<setw (4) << a[k];
cout<<endl;
}
if( a[i] > v )                    // обмен разделяющего элемента
{
r = a[i];
a[i] = v;
a[right] = r;
}
return i;                         // позиция разделяющего элемента
}
//_____________________________________________________________
voidmain()
{
constint n = 6;                   // количество элементов в массиве
int a[n] = { 20, 50, 10, 30, 60, 40 };          // массив целых чисел
for(inti = 0; i< n; i++ )
cout<<setw (4) << a[i];
cout<<endl;
```

```
int L = HoarePosPrint<int> ( a, 0, n-1 );                    // разделяющий
cout<< "L = " << L <<endl;
for(i = 0; i< n; i++ ) cout<<setw (4) << a[i];
_getch();                                                    // просмотр результата
}
```

***D*-последовательности в перестановках Хоара.** Для определения количества операций, выполняемых в функции *HoarePosPrint*( ), необходимо подсчитать количество итераций, выполняемых в вечном цикле при поиске индекса разделяющего элемента. Для этого необходимо выполнить программу $DH01$ в различных вариантах представления исходного массива. Интерес представляют последовательности элементов, которые дают минимальное и максимальное число итераций в вечном цикле. В качестве примера ниже показана последовательность с одинаковыми элементами:

5 5 5 5 5

В этой последовательности одна итерация вечного цикла, поскольку отсутствует необходимость в перестановке и упорядочивать нечего.

В следующей последовательности на правой грани находится элемент с разделяющим значением 3,такой вид назовем *D*-последовательностью:

5 4 1 2 3

Прежде чем разделяющее значение 3 окажется на месте разделяющего элемента, произойдет перестановка значений элементов. В результате преобразования исходной *D*-последовательности получается другая последовательность:

2 1 3 5 4

особенностью которой является то, что она создала две новые *D*-последовательности. После установки разделяющего элемента 3в новых подпоследовательностях слева 2, 1 и справа 5, 4 также требуются максимальные перестановки для установки в них разделяющих элементов.

D-последовательность характеризуется следующими свойствами:

1) начальная *D*-последовательность содержит два элемента, упорядоченных в обратном порядке;

2) при поиске места разделяющего элемента необходимо произвести максимальное число перестановок значений в массиве;

3) после установки разделяющего элемента правая и левая подпоследовательности должны быть *D*-последовательностями;

4) равное число элементов в правой и левой подпоследовательностях.

Эти свойства обеспечивают максимальное количество итераций вечного цикла и хорошо подходят для оценки максимального числа

операций. Свойство 4 является следствием требования максимального числа перестановок, которые будет необходимо произвести в новых подпоследовательностях относительно установленного разделяющего элемента. Из этого также следует, что разделяющий элемент будет находиться посередине получающейся последовательности. В таблице представлено несколько D-последовательностей и количество элементов в них. Интересно отметить, что количество элементов в следующей D-последовательности удваивается, и добавляется один новый разделяющий элемент.

Таблица

Начальные D-последовательности

Число элементов	D-последовательность
2	2, 1
5	5, 4, 1, 2, 3
11	8, 7, 10, 11, 9, 3, 2, 1, 4, 5, 6
23	17, 16, 13, 14, 15, 21, 23, 22, 19, 20, 18, 6, 5, 4, 1, 2, 3, 9, 11, 10, 7, 8, 12

Для того чтобы экспериментально проверить количество итераций, в тело вечного цикла включают счетчик cc, а также счетчики индексов ci и cj, как показано ниже:

```
unsignedint cc = 0, ci = 0, cj = 0;                    // счетчики итераций
while( true )
{
c++;                                                   // увеличение счетчика
while( a[i] < v &&i< m ) { ++i; ++ci; }                          // меньший
while( v < a[j] && j > left ) { –j; ++cj; }                      // больший
if(i>= j ) break;                                      // обмен не нужен
if( a[i] == a[j] ) break;                              // обмен не нужен
r = a[i];                                              // обмен a[i] и a[j]
a[i] = a[j];
a[j] = r;
}

cout<< "n = " <<right-left+1;
cout<< " ci = " << ci << " cj = " <<cj<<endl;
```

В распечатке приведены значения счетчиков cc, ci, cj для нескольких D-последовательностей при различном количестве элементов n

в массиве:

$$n = 2 \quad cc = 1 \quad ci = 0 \quad cj = 0$$
$$n = 5 \quad cc = 3 \quad ci = 2 \quad cj = 2$$
$$n = 11 \quad cc = 6 \quad ci = 5 \quad cj = 5$$
$$n = 23 \quad cc = 12 \quad ci = 11 \quad cj = 11$$
$$n = 47 \quad cc = 24 \quad ci = 23 \quad cj = 23$$

Итак, для подсчета минимального и максимального числа операций[1] при поиске места вставки по Хоару необходимо произвести расчеты для исходного массива, который содержит либо одинаковые значения, либо значения D-последовательности. Установку разделяющего элемента выполняет функция *HoarePosPrint*( ).В теле данной функции выполняются начальные действия *Typev = a*[*right*], *inti = left*, *intj = right – 1*, *intm = j*, на которые затрачивается $F1$ операций для запоминания и вычитания:

$$F1 = 5$$

Количество итераций вечного цикла обозначим как z_{av} для массива из n элементов с одинаковыми значениями и как z_D для массива с D-последовательностью:

$$z_{av} = 1;$$
$$z_D = \frac{n+1}{2}.$$

В теле внешнего вечного цикла *while* (*true*) выполняется первый внутренний цикл $while(a[i] < v \,\&\&\, i < m)$ *++i* для группы меньших элементов. На каждой итерации внутреннего цикла выполняется три или пять операций: сравнение *a*[*i*]<*v*, сравнение $i < m$, булева конъюнкция *&&* и увеличение индекса *++i*. С учетом внешнего цикла всего будет выполнено I_{av} и I_D операций для соответствующих значений исходного массива.

$$lI_{av} = \sum_{k=1}^{z_{av}} 3 = \sum_{k=1}^{1} 3 = 3;$$
$$I_D = \sum_{k=1}^{z_D} 3 + \sum_{k=1}^{z_D - 1} (2+3) = 3z_D + 5(z_D - 1) = 8\frac{n+1}{2} - 5 = 4n - 1.$$

Затем выполняется второй внутренний цикл $while(v < a[j] \&\& j > left)$–$j$ для группы больших элементов. Всего за время внешнего

[1] С е д ж в и к Р. Фундаментальные алгоритмы на С++: Анализ: Структуры данных: Сортировка: Поиск: пер. с англ. – СПб.: ООО ДианаСофтЮП, 2002. – 688 с.

цикла будет выполнено J_{av} и J_D операций для равноупорядоченного и D-упорядоченного исходного массива:

$$J_{av} = \sum_{k=1}^{z_{av}} 3 = \sum_{k=1}^{1} 3 = 3;$$

$$J_D = \sum_{k=1}^{z_D} 3 + \sum_{k=1}^{z_{D-1}} (2+3) = 3z_D + 5(z_D - 1) = 8\frac{n+1}{2} - 5 = 4n - 1.$$

После перестановок очередных меньших и больших элементов выполняются две инструкции условия для определения момента окончания формирования меньшей и большей групп. Оба условия $i >= j$ и $a[i] == a[j]$ затрачивают две операции. Всего во внешнем цикле будет затрачено C_{av} или C_D операций для двух вариантов исходного массива:

$$C_{av} = z_{av} \cdot 2 = 1 \cdot 2 = 2;$$

$$C_D = z_D \cdot 2 = \frac{n+1}{2} \cdot 2 = n + 1.$$

Поскольку инструкции условия для прекращения внешнего цикла выполняются раньше перестановки значений элементов, то количество перестановок будет на единицу меньше числа итераций внешнего цикла. Каждая перестановка занимает по три операции запоминания. Всего во внешнем цикле будет выполнено E_{av} и E_D операций для каждого варианта исходного массива соответственно:

$$E_{av} = z_{av} \cdot 3 - 1 = 1 \cdot 3 - 1 = 2;$$

$$E_D = z_D \cdot 3 - 1 = \frac{n+1}{2} \cdot 3 - 1 = \frac{3}{2}n + \frac{1}{2}.$$

Внешний вечный цикл завершен. Далее проверяется необходимость установки разделяющего значения. Эту операцию обозначим как $F2_{av}$ для массива с одинаковыми значениями и как $F2_D$ для массива с D-последовательностью:

$$F2_{av} = 1;$$

$$F2_D = 4.$$

Если предположить, что все операции занимают одинаковое время, что соответствует упорядочиванию массива целых чисел, то в функции *HoarePosPrint*( ) выполняются HP_{av} и HP_D операций при различных вариантах исходного массива:

$$HP_{av} = F1 + I_{av} + J_{av} + C_{av} + E_{av} + F2_{av} = \\ = 5 + 3 + 3 + 2 + 2 + 1 = 14;$$

$$HP_D = F1 + I_D + J_D + C_D + E_D + F2_D =$$
$$= 5 + (4n-1) + (4n-1) + (n+1) + \left(\frac{3}{2}n + \frac{1}{2}\right) + 4 =$$
$$= \frac{21}{2}n + \frac{17}{2}.$$

Таким образом, рассмотрены способы формирования и количественные оценки скоростных свойств D-последовательностей. Тексты функций, участвующих в данном исследовании, представлены на языке программирования исторического C++ с учетом шаблона для типа элементов массива.

Статья поступила в редакцию 10.05.2012