Сахаров Александр Константинович
г. Барнаул
cak@uni-altai.ru

МОДИФИЦИРОВАННЫЙ МЕТОД СОРТИРОВКИ СЛИЯНИЕМ

Метод сортировки слияниями был предложен в 1945 году одним из величайших математиков XX века Джоном фон Нейманом**.** Это был первый алгоритм с трудоемкостью $O(n \log_2 n)$. Алгоритм основан на многократном слиянии уже упорядоченных и рядом расположенных групп элементов массива. Метод является устойчивым.

Главный недостаток метода (кроме того, что используется дополнительная память) в том, что в конце каждого слияния элементы перезаписываются из дополнительного массива на соответствующие места в исходный массив.

Пусть p1 и p2 – индексы, с которых начинаются упорядоченные и рядом стоящие участки массива A, а len1 и len2 – длины участков, причем 1<=p1<p2<=n, p2=p1+len1. Для слияния этих участков в другой массив B используется следующая процедура:

```
Procedure MERGE(var a:mass;p1,len1,p2,len2:integer;var b:mass);
  Var i,j,j1,k :integer;
  Begin
     i:=p1; j:=p2; j1:=j;
     for k:=p1 to p1+len1+len2-1 do
       if(j1<=p2+len2-1)and((i>p1+len1-1)or(a[i]>a[j]))
          then begin b[k]:=a[j]; inc(j1); if j1<=n then inc(j); end
          else  begin b[k]:=a[i]; inc(i) end;
  End;
```

Идея модифицированного метода. Рассмотрим частный случай, когда количество элементов массива $n = 2^k$. Так как массив A можно рассматривать как совокупность n упорядоченных групп по одному элементу в каждой, то к каждой соседней паре элементов можно применить процедуру MERGE. В результате этого, в массиве B получим n div 2 упорядоченных групп по два элемента. На втором шаге, с помощью той же процедуры, объединяем рядом стоящие упорядоченные пары массива B. Результат слияния будет в массиве A. После этого слияния массив A будет состоять уже из n div 4 упорядоченных групп по 4 элемента в каждой. После K-го шага элементы либо массива A (при четном K), либо массива B будут упорядочены.

Проиллюстрируем данный алгоритм на массиве из 8 элементов.

	массив A	7	4	2	5	8	3	6	1
шаг 1	массив B	4	7	2	5	3	8	1	6
шаг 2	массив A	2	4	5	7	1	3	6	8
шаг 3	массив B	1	2	3	4	5	6	7	8

Анализ метода. На каждом шаге количество сравнений и пересылок не зависит от расположения элементов массива. Для любого, даже уже упорядоченного массива, трудоемкость на каждом шаге составляет величину порядка O(n). Тогда общая трудоемкость метода за K шагов

$T_n = O(n \cdot \log_2 n)$.

Построим алгоритм сортировки элементов массива A методом слияния таким образом, чтобы конечная упорядоченная последовательность элементов всегда находилась в массиве A. Для этого необходимо один раз переписать элементы массива A в массив B и выполнить один из следующих пунктов:

1. Если $n=2^k$, то при четном K процесс слияния следует начинать с массива A, при нечетном K – с массива B, иначе выполняем пункт 2.

2. Выделяем участки максимальной длины $m_1=2^{k_1}$, $m_2=2^{k_2}$, …, $m_i=2^{k_i}$ с помощью следующей процедуры:

```
Procedure MK(poz :integer; var m, k :integer);
 Var kol:integer;
 Begin
    kol:=n+1 – poz; m:=1; k:=0;
    while 2*m<=kol do begin m:=m*2;k:=k+1 end; { m=2^k}
    k:=k mod 2;
 End;
```

На каждом участке длиной не менее двух выполняем сортировку элементов методом слияния таким образом, чтобы упорядоченные последовательности двух соседних участков находились в разных массивах, причем элементы первого участка в массиве B. Например, при четном количестве участков получаем

m_1 элементов	m_2	m_3	……			m_i
массив B	A	B	……	A	B	A

Затем, если n нечетное, то элементы последнего участка объединяем с последним элементом массива (процедура MERGE), иначе переписываем их на соответствующие места в другой массив (процедура Move).

```
Procedure Move(var a,b:mass;poz:integer);
 Var i:integer;
 Begin
    for i:=poz  to n do  b[i]:=a[i];
 End;
```

После этого две последние упорядоченные последовательности будут находиться рядом и в одном массиве. После их объединения получаем то же самое в другом массиве. То есть, в результате последовательного объединения соседних участков элементы массива A будут упорядочены.

Для реализации данного метода предлагается следующий рекурсивный алгоритм, где u – номер участка:

```
Procedure MMS_SortM (var a: mass; n: integer);
 Var b: mass;
 Procedure Sort_Rek (u: integer; poz: integer);
  Var p, len, s, m, k, s1, poz1: integer;
  Begin
    len:=1; s:=1; MK(poz, m, k);
   if m=n then s1:=-1 else s1:=1;
   Repeat
     p:=poz;
     While p<poz+m-1 do
       begin
         if (u+k) mod 2=1
           then if s*s1=1 then Merge(b,p,len,p+len,len,a)
                          else  Merge(a,p,len,p+len,len,b)
           else if s*s1=1 then Merge(a,p,len,p+len,len,b)
                          else Merge(b,p,len,p+len,len,a);
         p:=p+2*len;
       end;
     len:=len*2; s:=-s;
   Until len=m;
   poz1:=poz+m;
   if (poz1>n)and(u>1) then if u mod 2 = 0 then Move(a,b,poz)
                                           else  Move(b,a,poz);
   if poz1=n then if u mod 2 = 0 then Merge(a,poz,m,n,1,b)
                                 else Merge(b,poz,m,n,1,a);
   if poz1<n then
     begin
       Sort_Rek(u+1,poz1);
       if u mod 2 = 0 then Merge(a,poz,m,poz1,n+1-poz1,b)
                      else Merge(b,poz,m,poz1,n+1-poz1,a);
     end;
  End;
 Begin {MMS_SortM}
    b:=a; if n>1then  Sort_Rek(1,1);
 End;
```

Если $n=2^k$ или $n=2^k+1$, то модифицированный метод сортировки слиянием является прямым методом.

Сравним время работы рассмотренного метода и всех улучшенных методов. Так как время работы всех методов на массивах равно нулю, то сравнение будем проводить на файлах. Для этого достаточно заменить тип простых переменных integer на longint, тип mass на тип files=file of longint и, учитывая специфику работы с файлами, внести изменения в работе рассмотренных процедур.

```
  Procedure Merge (var a: files; p1, len1, p2, len2: longint; var b: files);
    Var i, j, k, ai, aj: longint;
    Begin
      i:=p1; j:=p2; seek(b,i);
      seek(a,i); read(a,ai);  seek(a,j); read(a,aj);
      for k:=p1 to p1+len1+len2-1 do
         if (j<=p2+len2-1)and((i > p1+len1-1)or(ai > aj))
          then begin
                  write(b,aj); inc(j);
                  if j<n then begin  seek(a,j); read(a,aj); end;
               end
          else  begin
                   write(b,ai); inc(i); seek(a,i); read(a,ai);
                end;
   End;

Procedure MK (poz: longint; var m: longint; var k: integer);
  Var kol: longint;
  Begin
     kol:=n - poz; m:=1; k:=0;
     while 2*m<=kol do begin m:=m*2; k:=k+1 end; {m=2^k}
     k:=k mod 2;
  End;

Procedure Move (var a, b: files; poz: longint);
  Var x: longint;
  Begin
     seek(a, poz); seek(b, poz);
     while not eof(a) do
        begin
          read(a,x); write(b,x);
       end;
  End;

Procedure MMS_SortF (var a: files; n: longint);
  Var b: files;
  Procedure Sort_Rek (u: integer; poz: longint);
    Var p, len, m, poz1: longint; s, k, s1: integer;
    Begin
      len:=1;s:=1;MK(poz,m,k);
      if m=n then s1:=-1 else s1:=1;
      repeat p:=poz;
        while p<poz+m-1 do
          begin
            if (u+k) mod 2=1
```

```
                then if s*s1=1 then Merge(b,p,len,p+len,len,a)
                               else Merge(a,p,len,p+len,len,b)
                else if s*s1=1 then Merge(a,p,len,p+len,len,b)
                               else Merge(b,p,len,p+len,len,a);
              p:=p+2*len;
            end;
          len:=len*2;s:=-s;
        until len=m;
        poz1:=poz+m;
        if (poz1>n-1)and(u>1) then if u mod 2 = 0 then Move(a,b,poz)
                                                  else Move(b,a,poz);
        if poz1=n-1 then if u mod 2 = 0 then Merge(a,poz,m,n-1,1,b)
                                        else Merge(b,poz,m,n-1,1,a);
        if poz1<n-1 then
          begin
            Sort_Rek(u+1,poz1);
            if u mod 2 = 0 then Merge(a,poz,m,poz1,n-poz1,b)
                           else Merge(b,poz,m,poz1,n-poz1,a);
          end;
      End;
    Begin {MMS_SortF}
      assign(b,'b.dat');rewrite(b);
      Move(a,b,0);close(b);reset(b);
      if n>1 then Sort_Rek(1,0);close(b);
    End;
```

Ниже в таблице указано время работы в секундах всех улучшенных методов на файлах при $n=2^k-1$ при различных значениях K. Для данного значения n количество обращений к рекурсивной процедуре Sort_Rek максимально и составляет k-1.

K	10	12	14	16	18	20
Neman_SortF	0.22	1.05	4.73	19.31	81.79	346.21
Shell_SortF	0.11	0.50	2.70	12.93	65.89	348.19
Heap_SortF	0.11	0.33	1.71	7.59	33.77	150.71
Quick_SortF	0.06	0.28	1.38	6.16	28.16	126.43
MMS_SortF	0.06	0.28	1.10	5.06	22.28	96.37

СПИСОК ЛИТЕРАТУРЫ

1. Алкок Д. Язык паскаль в иллюстрациях. М.: Мир, 1991, 192 с.

2. Кнут Д. Искусство программирования, т. 3. Сортировка и поиск. Издательский дом «Вильямс», 2000, 832 с.

3. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М.: МЦНМО, 2000, 960 с.