Занятие 16. Сортировка слиянием

Рассмотрим ещё один метод сортировки массива. Его алгоритм строится с помощью алгоритмической стратегии «разделяй и властвуй». Суть этой стратегии в том, что решение задачи состоит из следующих трёх этапов.

1. *Разделение*. Задача разбивается на две или более аналогичные неперекрывающиеся подзадачи меньшей размерности. Неперекрывающиеся — это значит, что решение каждой подзадачи никак не зависит от решения других подзадач.
2. *Покорение*. Рекурсивно решаются подзадачи. Когда размерность подзадачи достаточно мала, эта подзадача решается непосредственно.
3. *Комбинирование* решения исходной задачи из решений подзадач

Мы уже, по сути, применяли стратегию «разделяй и властвуй» при составлении алгоритма двоичного поиска (см. занятие 8), хотя там и не было рекурсии. Чтобы найти элемент в отсортированном массиве, мы делили массив пополам, искали элемент в правой и левой части этого массива (с разным успехом). На этапе соединения мы объявляли индекс найденного элемента, если одна из подзадач к нему привела, или о том, что заданного элемента в массиве нет.

Подобным образом поступим и для сортировки массива методом слияния (merge sort). Для определённости будем говорить о сортировке по возрастанию. Упомянутые выше этапы сводятся к следующему.

1. Разделяем массив из n элементов на две примерно равные части: в первой $\left\lfloor \frac{n}{2} \right\rfloor$ элементов, во второй — все остальные. (Скобки означают, что $\frac{n}{2}$ округляется в меньшую сторону.)
2. Сортируем *методом слияния* каждую часть отдельно.
3. Отсортированные части массива сливаем в один отсортированный массив.

Разбиение на подзадачи можно прекращать, когда в выделенной для сортировки части массива окажется всего один элемент. Массив из одного элемента является упорядоченным. Это и есть условие выхода из рекурсии.

Самой сложной частью задачи сортировки слиянием является само слияние двух отсортированных частей массива с образованием упорядоченного массива. Пусть процедура Merge (A,p,q,r) решает эту задачу: сливает отсортированные части массива A[p..q] (часть массива A от A[p] до A[q] включительно) и A[q+1..r] в отсортированную последовательность A[p..r]. Здесь $p \le q < r$.

Тогда алгоритм сортировки слиянием MergeSort(A,p,r) можно представить следующим образом (см. рис. 1). Смысл параметров процедуры здесь тот же, что и для процедуры Merge: A — имя массива, p — левая, а r — правая граница (индекс) сортируемой части массива.

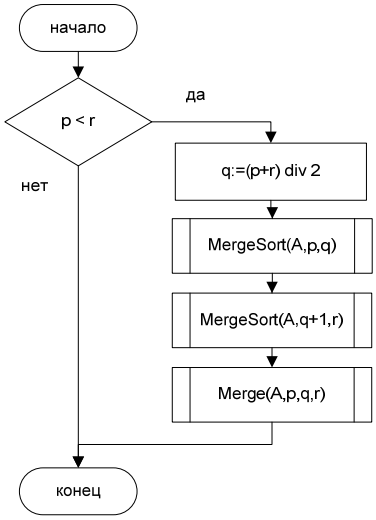

Рисунок 1. Сортировка слиянием MergeSort(A,p,r)

Чтобы отсортировать весь массив A[1..n], надо вызвать описанную процедуру следующим образом:

MergeSort(A,1,n)

Теперь приступим к разработке процедуры слияния Merge.

Представим себе, что на столе лежат две стопки карт, обращённых лицевой стороной вниз. Карты в каждой стопке отсортированы, причём наверху находится карта наименьшего достоинства. Эти две стопки надо объединить в одну выходную, в которой карты также будут отсортированы. Для этого, пока в обеих стопках есть карты, будем брать по одной верхней карте из обеих стопок,

выбирать из них меньшую и помещать её в выходную стопку. В уменьшившейся стопке верхней будет теперь другая карта. После того, как одна из стопок закончится, карты из второй стопки надо будет по одной переложить в выходную стопку. На этом сортировка будет закончена. Чтобы не заботиться о случае, когда одна из стопок закончится, можно немного усовершенствовать алгоритм. Подложим под каждую стопку по одной сигнальной карте особо большого достоинства. Настолько большого, чтобы ни одна из карт в обеих стопках не имела достоинства больше. Тогда можно быть уверенными, что эти сигнальные карты останутся в своих стопках, когда все другие карты уже будут переложены в выходную. Эти сигнальные карты обозначим символом ∞.

На рис. 2 представлен алгоритм процедуры MergeSort(A,p,q,r) слияния подмассивов A[p..q] и A[q+1..r] в подмассив A[p..r].

Здесь через N1 обозначена длина массива A[p..q], через N2 — длина массива A[q+1,r]. Для работы этой процедуры создаются 2 вспомогательных массива L и R. В первый из них переписывается подмассив A[p..q], левая часть массива A[p..r]. В массив R переписывается подмассив A[q+1..r]. Последними элементами массивов L и R будут сигнальные очень большие значения. В цикле с параметром k в конце алгоритма в массив A[p..r] сливаются отсортированные массивы L и R.

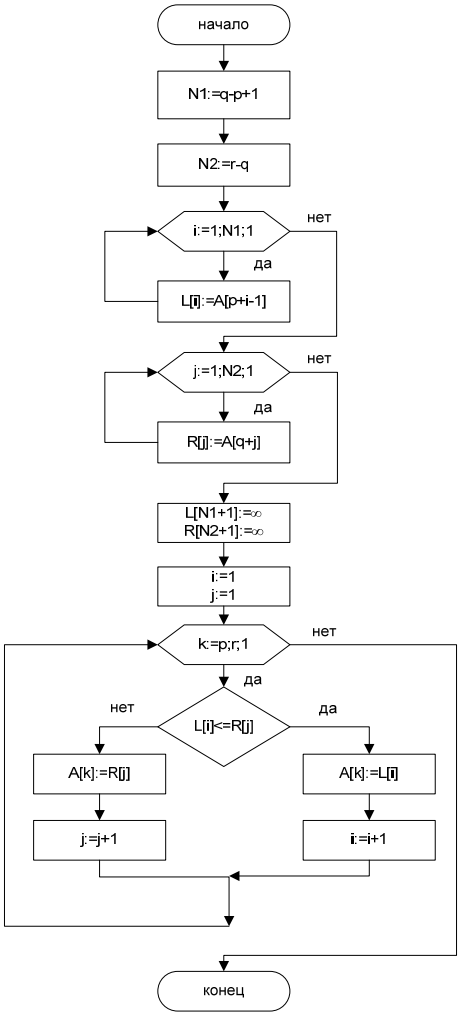

Рисунок 2. Слияние подмассивов Merge(A,p,q,r)

Теперь пишем программу.

```
{$mode delphi}
Program Merge_Sort;
Const   MAXN   =   10000000;        //Максимальное
//количество элементов массива
      BIG = MaxInt;      //Сигнальный элемент

Type TElem = Integer;
     massiv  = array[1..MAXN] of TElem;
     massiv2 = array[1..(MAXN+1) div 2 + 1] of
TElem;


//Слияние отсортированных подмассивов
//A[p..q] и A[q+1..r]
//в отсортированный подмассив A[p..r]
Var Left,Right : massiv2; //Глобальные массивы
Procedure   Merge(Var   A   :   massiv;   p,q,r   :
Integer);
Var i,j,k,N1,N2 : Integer;
Begin
  //Вычисляем длины подмассивов
  N1:=q-p+1;
  N2:=r-q;

  //Копируем элементы подмассивов во вспомогательные массивы
  For i:=1 To N1 Do
  Left[i]:=A[p+i-1];

  For j:=1 To N2 Do
  Right[j]:=A[q+j];

  //Устанавливаем сигнальные элементы
  Left[N1+1]:=BIG;
  Right[N2+1]:=BIG;

  //Выполняем слияние массивов Left и Right
```

```
  i:=1;
  j:=1;

  For k:=p To r Do
  If Left[i]<=Right[j] Then
  Begin
    A[k]:=Left[i];
    i:=i+1
  End
  Else Begin
         A[k]:=Right[j];
         j:=j+1
       End
End;


//Сортировка слиянием
//Выполняется для массива A[p..r]
Procedure MergeSort(Var A : massiv; p,r : Integer);
Var q : Integer;
Begin
  If p<r Then
  Begin
    q:=(p+r) div 2;
    MergeSort(A,p,q);
    MergeSort(A,q+1,r);
    Merge(A,p,q,r)
  End;
End;

//Основной алгоритм
Var a   : massiv;
    f   : Text;
    i,N : Integer;
Begin
  Assign(f,'merge.in');
  Reset(f);
  Read(f,N); //Количество элементов массива)
  For i:=1 To N Do
  Read(f,a[i]);
  Close(f);

  MergeSort(a,1,N);

  Assign(f,'merge.out');
  Rewrite(f);
  For i:=1 To N Do
  Begin
    Write(f,a[i]:10, ' ');
    If i mod 7 = 0 Then
    WriteLn(f) //Данные выводим по 7 чисел в строку
  End;
  Close(f)
End.
```

Вспомогательные массивы L и R в программе обозначены как Left и Right. Так как в подпрограмме Merge уже есть переменная r, а в языке Паскаль регистр символов в написании идентификаторов не различается, то обозначать массив идентификатором R мы не имели права. Размерность этих массивов выбрана такой, чтобы в них могла поместиться половина исходного массива плюс один элемент (сигнальный). При вычислении половины размера исходного массива результат округляем в большую сторону: (MAXN+1) div 2. Массивы Left и Right сделаны глобальными для того, чтобы память под них выделялась лишь 1 раз.

Время работы сортировки слиянием для массива из n элементов пропорционально $n \cdot log(n)$, что значительно лучше прямых методов сортировки (вставкой, прямым выбором и пузырьковой). Однако у этого алгоритма есть один существенный недостаток. Для его работы нужны дополнительные массивы, которые в сумме занимают такой же объём памяти, как и исходный.

Алгоритм сортировки слиянием является устойчивым. Это значит, что относительный порядок равных элементов не изменяется. Часто это является несущественным. Но представьте такую задачу. Массив *name* содержит упорядоченные в алфавитном порядке имена участников соревнований, а массив *score* — набранные ими количества очков. Итоговая таблица должна содержать участников в порядке убывания набранных баллов. Если двое участников набрали одинаковое количество баллов, то в турнирной таблице их имена располагаются по алфавиту. Для решения этой задачи можно отсортировать массив *score* по убыванию, синхронно переставляя элементы массива *name*. В этом случае применение устойчивого алгоритма сортировки гарантирует, что имена участников с одинаковым количеством очков будут расположены в прежнем (алфавитном) порядке.

Задания для самостоятельной работы

1. Что надо изменить в алгоритме сортировки слиянием, описанном в тексте занятия, чтобы массив сортировался по убыванию?
2. Перепишите процедуру Merge так, чтобы в ней не использовались сигнальные значения. Сигналом к остановке должен служить тот факт, что все элементы массива L или массива R скопированы обратно в массив A, после чего в этот массив копируются элементы, оставшиеся в непустом массиве.
3. Решите задачу, сформулированную в конце текста занятия.
4. Проведите вычисления и сравните время сортировки слиянием со временем сортировки массивов при помощи других алгоритмов сортировки, описанных в занятии 7 и частично в занятии 13. Сравнение проведите для малых и больших массивов.

Литература

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ. Второе издание. — Москва, Санкт-Петербург, Киев. Издательство «Вильямс», 2010.