А.К. Сахаров

РЕКУРСИВНЫЕ МЕТОДЫ РЕШЕНИЯ КОМБИНАТОРНЫХ ЗАДАЧ

В статье рассматриваются рекурсивные методы нахождения перестановок из n элементов, сочетаний и размещений из n элементов по k. Элементами являются символы строки S. Символы в строке могут повторяться. Для элементов другого типа можно использовать массивы.

Ключевые слова: рекурсия, перестановки, сочетание, размещение, комбинаторные задачи.

A.K. Saharov

RECURSIVE METHODS OF COMBINATORIAL PROBLEM SOLVING

The article deals with the recursive methods of finding the permutation of n elements, combinations and distribution of n elements from k. The elements are the symbols of line S. The symbols in the line can repeat themselves. Arrays can be used for the elements of other types.

Key words: recursion, permutation, combination, distribution, combinatorial problems.

Рассмотрим случай, когда все символы строки S различны. Для нахождения перестановок, сочетаний и размещений из этих символов достаточно найти соответствующие расстановки из индексов элементов строки, т. е. расстановки из первых N натуральных чисел. Очередную комбинацию индексов будем хранить в массиве X.

Для вывода очередной комбинации используем процедуру

```
Procedure Print(n:integer);
var i:integer;
    Begin
       for i:=1 to n do write(x[i]:4);
       inc(kol);writeln(' - ',kol);
    End;
```

1. Перестановки из n различных элементов.

Для нахождения всех перестановок из натуральных чисел от 1 до N рассмотрим два рекурсивных алгоритма.

Вариант 1. Сначала формируем массив X: $x[i] = i$, $1 \le i \le n$. Затем, начиная с первого элемента, для элемента с номером i выполняем $n - i + 1$ раз следующие действия:

1) если $i < n - 1$, то находим перестановки со следующего $i + 1$ элемента (рекурсивный вызов с меньшим количеством элементов), иначе выводим комбинацию на экран;
2) элементы массива, начиная с номера i и до n, сдвигаем влево на один разряд по кругу.

```
Procedure P1(i:integer);
    var j, k, w:integer;
    Begin
       for j:=i to n do
       begin
          if i<n-1 then P1(i+1) else Print(n);
          w:=x[i];
          for k:=i to n-1 do x[k]:=x[k+1];
          x[n]:=w;
       end;
    End;
```

Вариант 2. При формировании массива X используется логический массив d из n элементов. До обращения к процедуре $P2$ значения $d[i] = true$, $1 \le i \le n$.

```
Procedure P2(i:integer);
    var j:integer;
    Begin
       for j:=1 to n do
          if d[j] then begin
                          x[i]:=j; d[j]:=false;
                          if i<n then P2(i+1) else Print(n);
                          d[j]:=true;
                       end;
    End;
```

2. Сочетания из n различных элементов по k.

Для нахождения всех сочетаний из N различных предметов по K рассмотрим частные случаи. При любом, но фиксированном k, алгоритм нахождения сочетаний очевиден. Например:

$k = 1$ – в цикле выводим все элементы массива: `for i1:=1 to n do write(x[i1:4]);`

$k = 2$ – двукратный цикл:

```
for i1:=1 to n-1 do
    for i2:=i1+1 to n do
           writeln(x[i1]:4,x[i2]:4);
```

$k = 3$ – трехкратный цикл:

```
for i1:=1 to n-2 do
    for i2:=i1+1 to n-1 do
       for i3:=i2+1 to n do
       writeln(x[i1]:4,x[i2]:4,x[i3]:4);
```

Построим рекурсивную процедуру, которая вызывает себя k раз. Заголовок процедуры должен содержать три параметра: i – счетчик количества вызовов, L, R – границы изменения некоторого локального параметра цикла. На частных примерах выявляется следующая закономерность: при очередном вызове процедуры значения i и R увеличиваются на единицу, а значение L на единицу больше значения предыдущего параметра.

Вызов процедуры – $CNK(1, 1, n - k + 1)$.

3. Размещения из n различных элементов по k.

Для нахождения всех размещений из n различных элементов по k достаточно найти все сочетания из n элементов по k и к каждому из них применить алгоритм нахождения перестановок из k элементов. Можно предложить другой вариант:

```
Procedure RNK(i:integer);
    var j:integer;
    Begin
       for j:=1 to n do
          if d[j] then begin
                          x[i]:=j; d[j]:=false;
                          if i<k then RNK(i+1) else Print(k);
                          d[j]:=true;
                       end;
    End;
```

4. Расстановки с частичным повторением.

Ранее предложенные рекурсивные алгоритмы можно применять лишь в том случае, когда все элементы различны. Рассмотрим случай, когда символы строки S могут повторяться. Для нахождения перестановок, сочетаний и размещений из символов строки S предполагается, что каждый символ может входить в очередную расстановку не более того количества раз, сколько раз он встречается в строке.

Рассмотрим этапы создания рекурсивных алгоритмов.

После ввода строки S (и для сочетания и размещения) формируем:

- строку $S1$ из различных символов строки S;
- массив повторений $Pov[i]$ i-го символа строки $S1$ в строке S;
- массив последней расстановки $Last[i]$ для сочетаний.

Очередная последовательность символов соответствующей расстановки находится с учетом массива повторений символов.

Приведем полный текст программы.

```
uses Crt;
Type mas=array[1..255] of integer;
Var i,n1,k,p:byte; s1:string; kol:longint;
    d:array[1..255] of boolean;
    x, last, pov : mas;
Procedure Create(var s1:string;var pov:mas;var n1,k,p:byte);
   var i,j,n,m:byte;s:string;ch:char;
   Begin
      repeat write('s = ');readln(s) until s<>''; n:=length(s);
      writeln(' 1 - CNK');writeln(' 2 - RNK');writeln(' 3 - PERES');
      repeat ch:=readkey until (ch>='1')and(ch<='3'); p:=ord(ch)-48;
      if p=3 then k:=n
              else begin writeln('n=',n);
                      repeat write('k=');readln(k) until (k>=1)and(k<=n)
                   end;
      s1:=''; for i:=1 to n do if pos(s[i],s1)=0 then s1:=s1+s[i]; {строка S1}
      n1:=length(s1);
      for i:=1 to n1 do {массив Pov}
         begin
            m:=0; for j:=1 to n do if s1[i]=s[j] then inc(m);
            pov[i]:=m;
         end;
   End;
Procedure Print;
   var i:integer;
   Begin
      for i:=1 to k do write(s1[x[i]]);
      inc(kol); writeln(' - ',kol); if kol mod 24=0 then readkey;
   End;
Procedure Form_Last(var last:mas;Pov:mas);
   var i,j:byte;
   Begin j:=n1;
      for i:=k downto 1 do
         begin
            last[i]:=j; dec(pov[j]); if pov[j]=0 then dec(j);
         end;
   End;
Procedure Cnk_P(i,L,R:integer);
   var j,p:integer;
   Begin
      for j:=L to R do
         begin x[i]:=j;
            if i<k then
                    begin
                       dec(pov[j]);if pov[j]>0 then p:=0 else p:=1;
                       Cnk_P(i+1,j+p,last[i+1]);
                       inc(pov[j]);
                    end else Print;
         end;
   End;
```

```
Procedure Rnk_P(i:integer);
   var j:integer;
   Begin
      for j:=1 to n1 do
         if d[j] then
                    begin
                       x[i]:=j;dec(pov[j]);
                       if pov[j]=0 then d[j]:=false;
                       if i<k then Rnk_P(i+1) else Print;
                       inc(pov[j]);d[j]:=true;
                    end;
   End;
BEGIN
   Clrscr; kol:=0;
   Create(s1,pov,n1,k,p);
   if p=1 then begin
                Form_Last(last,pov);
                Cnk_P(1,1,last[1]);
               end
          else begin
                for i:=1 to n1 do d[i]:=true;
                Rnk_P(1)
               end;
   readln;
END.
```

Отметим, что рекурсивные алгоритмы, которые приведены в пункте 4, будут работают и для того случая, когда все элементы исходной строки различны.