Pascal 10_1. Некоторые комбинаторные алгоритмы

РАЗМЕЩЕНИЯ. ПЕРЕСТАНОВКИ. СОЧЕТАНИЯ

Имеется n различных предметов. Сколько из них можно составить k-расстановок? При этом две расстановки считаются различными, если они либо отличаются друг от друга хотя бы одним элементом, либо состоят из одних и тех же элементов, но расположенных в разном порядке. Такие расстановки называют размещениями без повторений.

> ***Размещением*** *элементов из множества $E=\{a_1,...,a_n\}$ по k называется упорядоченное подмножество из k элементов, принадлежащих E.*

Например: Дано множество $E=\{a_1, a_2, a_3\}$. Найти размещения из E по 2 элемента.
Получаем: (a_1, a_2); (a_2, a_1); (a_1, a_3); (a_3, a_1); (a_2, a_3); (a_3, a_2).
Число размещений обозначают A^k_n.

При составлении k размещений без повторений из n предметов нам надо сделать k выборов. На первом шаге можно выбрать любой из n предметов. На втором шаге выбрать из оставшихся n-1 предметов - ведь повторять сделанный выбор нельзя. На третьем шаге для выбора остается лишь n-2 свободных предметов, на четвертом n-3 предметов... на k-м шаге выбираем из n-k+1 предметов. Получаем, что число k - размещений без повторений из n предметов выражается следующим образом:
$A^k_n = n(n-1)...(n-k+1)$.

Еще эту формулу можно записать следующим образом:
$A^k_n = n!/(n-k)!$

> ***Размещение с повторениями*** *из n элементов множества $E=\{a_1, a_2, ..., a_n\}$ по k - всякая конечная последовательность, состоящая из k членов данного множества E.*
> Два размещения с повторениями считаются различными, если хотя бы на одном месте они имеют различные элементы множества E. Число различных размещений с повторениями из n по k равно n^k.

Задача 1. Напечатать все последовательности длины k из чисел 1..n.

Решение: Будем печатать их в лексикографическом порядке (последовательность *a* предшествует последовательности *b*, если для некоторого i их начальные отрезки длины i равны, а (i+1)-ый член последовательности *a* меньше). Первой будет последовательность <1,1,...,1>, последней - <n,n,...,n>. Будем хранить последнюю напечатанную последовательность в массиве x[1],..,x[k].
last[1],..,last[k] положим равным n. Для того, чтобы перейти к следующей последовательности надо: двигаясь с конца последовательности, найти самый правый член, меньший n, увеличить его на 1, а идущие за ним члены положить равными 1.

Процедура print(x:massiv) - вывод последовательности.
Функция egual(x,last:massiv):boolean - сравнивает элементы массивов,если x<>last, то egual:=false иначе egual:=true.

```
for i:=1 to k do x[i]:=1;
print(x);
for i:=1 to k do last[i]:=n;
   while not(equal(x,last)) do
```

```
begin
    p:=k;
    while not (x[p]<n) do p:=p-1;
    x[p]:=x[p]+1;
    for i:=p+1 to k do x[i]:=1;
    print(x);
end;
```

Перестановки из n элементов - частный случай размещения элементов из E по k, при k=n. Иными словами, *перестановками называют размещения без повторений из n элементов, в которые входят все элементы.* Можно также сказать, что перестановками из n элементов называют всевозможные n-расстановки, каждая из которых содержит все эти элементы по одному разу, и которые отличаются друг от друга лишь порядком элементов.

Число перестановок вычисляется по формуле:
$P_n=A^n_n=n\cdot(n-1)\cdot...\cdot2\cdot1=n!$

Алгоритм генерации перестановок в лексикографическом порядке:

1. Просматриваем a_1, ..., a_n с конца до тех пор, пока не попадется $a_i<a_i+1$. Если таковых нет, то генерация закончена.
2. Рассматриваем a_{i+1}, a_{i+2}, ..., a_n. Найдем первый с конца a_m больший a_i и поменяем их местами.
3. a_{i+1}, a_{i+2}, ..., a_n переставим в порядке возрастания (для этого достаточно еТ переписать с конца).
4. Печатаем найденную перестановку.
5. Возвращаемся к пункту 1.

Первой при этом будет перестановка <1, 2, ..., n>, последней - <n, ..., 2, 1>.

Сочетанием *элементов из $E=\{a_1, ..., a_n\}$ по k называется упорядоченное подмножество из k элементов, принадлежащих E и отличающиеся друг то друга составом, но не порядком элементов.*

Число сочетаний вычисляется по формулам:

$$C_{mn}=n!/(n-m)!\cdot m!$$

Алгоритм генерации сочетаний C^k_n.

1. Для i:=1 до k c_i:=i; печатаем c_i, для i=1..k.
2. С конца находим такое i, что c_i<>n-k+i. Если такого i нет, то генерация сочетаний закончена.
3. c_i:=c_i+1; для m=i+1, i+2, ..., k выполним c_m := c_{m-1} +1; выводим c_i для i=1, ..., k.
4. Возрващаемся к пункту 2.

Задача 2. Перечислить все возрастающие последовательности длины k из чисел 1..n в лексикографическом порядке (все сочетания из n по k).

Например: при n=5, k=2 получаем:
12 13 14 15 23 24 25 34 35 45

Решение: Минимальной будет последовательность <1, 2, ..., k>, а максимальной - <n-k+1, ..., n-1, n>. Каждый i-ый член последовательности можно увеличить, если он меньше n-k+i. После увеличения i-го элемента все следующие должны возрастать с шагом 1.

```
function poisk1(x:massiv):boolean;
var q:boolean;
   i:integer;
begin
     q:=false;
     for i:=m downto 1 do
     if x[i]<>n-m+i then q:=true;
     poisk1:=q;
end.
function poisk2(x:massiv):integer;
var i:integer;
begin
     i:=m;
     while not(x[i]<n-m+i) do
          i:=i-1;
     poisk2:=i;
end;
```

Функции poisk1 и poisk2 организуют поиск i-го элемента, который нужно увеличить на 1. Процедура print печатает полученную последовательность.

```
for j:=1 to m do x[j]:=j;
   print(x);
while poisk1(x) do
     begin
          k:=poisk2(x);
          x[k]:=x[k]+1;
          for j:=k+1 to m do
             x[j]:=x[j-1]+1;
          print(x);
     end;
```

Задача 3. Перечислить все разбиения целого положительного числа n на целые положительные слагаемые (разбиения, отличающиеся лишь порядком слагаемых, считаются за одно).
Например: n=4 разбиения 1+1+1+1, 2+1+1, 2+2, 3+1, 4.

Решение: Договоримся, что

1) в разбиениях слагаемые идут в не возрастающем порядке,

2) сами разбиения мы перечисляем в лексикографическом порядке.

Разбиение храним в начале массива x[1]..x[n], при этом количество входящих в него чисел обозначим k. В начале x[1]=...=x[n]=1, k=n, в конце x[1]=n, k=1. В каком случае x[i] можно увеличить, не меняя предыдущих? Во-первых, должно быть x[i-1]>x[i] или i=1. Во-вторых, i должно быть не последним элементом (увеличение i надо компенсировать уменьшением следующих). Увеличив i, все следующие элементы надо взять минимально возможными.

```
procedure print(x:massiv;m:integer);
var i:integer;
begin
   for i:=1 to m do
      write(x[i],' ');
end;
Begin
    for j:=1 to n do x[j]:=1;
       print(x,n);
    k:=n;
    while k<>1 do
        begin
            i:=k-1;
            while not((i=1) or (x[i-1]>x[i])) do
                i:=i-1;
            x[i]:=x[i]+1;
            sum:=0;
            for j:=i+1 to k do
               sum:=sum+x[j];
               for j:=1 to sum-1 do
                  x[i+j]:=1; k:=i+sum-1;
               print(x,k);
        end;
End.
```

(http://www.distedu.ru/mirror/_inform/bspu.secna.ru/Guide/book1/uch1_3.html)

ПРИМЕР: Вывести все размещения цифр от 1 до 4.

*53-str-rec.pas

```
var posl: string;

procedure perebor(A, w: string; N: integer);
var i: integer;
begin
  if N = Length(w) then begin
    writeln(w);
    Exit;
  end;
  for i:=1 to Length(A) do begin
    w[N+1]:= A[i];
    perebor(A, w, N+1);
  end;
end;

begin
  posl:= '....';
  perebor('1234', posl, 0);
end.
```

ЗАДАНИЯ:

1) **http://informatics.mccme.ru/mod/statements/view3.php?id=263&chapterid=187**
2) **http://informatics.mccme.ru/mod/statements/view.php?id=265**