Алгоритмы генерирования перестановок. Алгоритмы генерирования множества всех подмножеств, k-элементных множеств, разбиение множества

Лекция 20-21

Алгоритм генерирования перестановок

Общее число перестановок из k элементов равно k! (действительно, на первое место можем поставить любое из k чисел, на второе - любое из оставшихся k-1 и т.д.).

Идея алгоритма:

Рекурсия. На j-м шаге в строку, содержащую j-1 символ вставляем символ, равный j. Причем ставим его на все места от 1 до j-го. Останавливаемся, когда в строке станет k элементов.

Пусть k=3

j	1	2	3
S	1	21	321
			231
			213
		12	312
			132
			123

Пример. Составить программу генерирующую все перестановки из первых k натуральных чисел. К вводит пользователь.

```
type stroka=string[10];
Var k:integer;
procedure perest(n:integer; s:stroka);
{n-элемент, который будет вставляться в строку. На данный момент, в строке n-1 символ}
var i:integer;
    st1,st:stroka;
begin
  if n=k+1 then    {В строке n символов, она сформирована, выводим на экран}
             begin
               writeln(s);   exit;
             end
```

```
 else {В строке  меньше n символов}
    for i:=1 to n do    {Вставляем символ n в строку на i-е место (от 1 до n)}
      begin
         st:=s; {Копируем строку s во вспомогательную переменную st}
        str(n,st1); {Переводим цифру n в строку st1}
        insert(st1,st,i); {Вставляем строку st1 в строку st}
          perest(n+1,st); {Вызываем процедуру perest для следующей цифры n+1}
      end;
end;
Begin {Начало основной программы}
  writeln('Введите k'); readln(k);
   perest(1,''); {Изначально вызываем процедуру для элемента 1, вставляем его в пустую строку}
  readln;
end.
```

Алгоритм генерирования множества всех подмножеств

Допустим, что у нас есть множество S, состоящее из элементов a_1, a_2, ..., a_N, т.е. S = {a_1, a_2, ..., a_N} Для простоты можно считать, что a_1, .. a_N - это различные целые числа от 1 до N. Подмножеством данного множества S называется множество S', которое содержит некоторые элементы из S (не обязательно все). У множества из N элементов будет ровно 2^N различных подмножеств.

Для примера возьмем N = 3.

Запишем все числа от 0 до $2^N - 1 = 7$ в двоичной системе счисления: (123)

0 – 000

1 – 001

2 – 010

3 – 011

4 – 100

5 – 101

6 – 110

7 – 111

Если на *i*-той позиции в двоичной записи стоит 1, то *i*-тый элемент входит в подмножество, иначе — не входит. Поэтому данный алгоритм можно реализовать так:

```
var
  kol:longint;
  n,i:integer;
   a:array [1..100] of integer; {Массив из элементов множества (тип может быть любым)}
procedure Generate(x:longint; p:integer); {В процедуре генерируется одно подмножество исходного множества, для этого число x переводится в двоичную систему счисления и на экран выводятся только те элементы множества (массива a), на месте которых в двоичной записи числа x стоит 1}
begin
      write('{'); {Открываем скобку означающую начало подмножества}
  while x<>0 do {Пока число не равно 0}
    begin
```

```
        if x mod 2=1 then write(a[p],' '); {Находим значение цифры в двоичной записи числа путем нахождения остатка от деления числа на 2. Если полученная цифра равна 1, то выводим элемент массива а стоящий на соответствующем месте, начиная от последнего}
        Dec(p); {Уменьшаем на 1 номер текущего элемента массива}
      x:=x div 2;
  end; {Конец цикла пока число x не равно 0}
    Writeln('}'); {Подмножество сгенерировано, закрываем скобку}
end; {Конец процедуры Generate}
Begin  {Начало основной программы}
  writeln('Vvedite kol. el. v mnoj');
  Readln(n);
```

```
writeln('Vvedite el. mnojestva');
  for i:= 1 to n do  read(a[i]);
      kol:=1 shl n; {Операция shl (побитовый сдвиг влево) умножает число, стоящее слева на 2 в степени числа стоящего справа. Т.е. переменной kol присваивается значение 2 в степени n (число различных подмножеств множества из n элементов) }
  for i:= 0 to kol-1 do {Все числа от 0 до 2^n - 1}
    Generate(i,n); {Переводим в двоичную систему счисления и выводим в качестве очередного подмножества все элементы исходного множества, на позициях которых в двоичной записи числа стоят 1}
  Readln;
end.
```

Алгоритмы генерирования m-элементных множеств

Сочетание — это выбор из n предметов нескольких (m), причем порядок не важен.

Из курса комбинаторики известно, что число сочетаний из n по m равно n!/(m! * (n - m)!).

Будем выводить все сочетания в лексикографическом порядке.

В данном примере исходное множество состоит из первых n натуральных чисел. Если рассматривать их как индексы массива, то можно обобщить этот алгоритм на любое множество.

Первое сочетание — это первые m натуральных чисел.

Затем, начиная с последнего элемента, увеличиваем его на 1 и выводим измененный массив, продолжаем действовать таким образом до тех пор, пока данный элемент не примет свое максимальное значение (для последнего элемента n, для предпоследнего n-1, …, для первого n-m+1). Затем переходим к предыдущему элементу и увеличиваем его на 1, а все элементы, расположенные после него, становятся на единицу больше своих предшественников (располагаются по порядку). Например, для n=5, m=3:

123 124 125 134 135 145 234 235 245 345

```
Var
  s: Array[1 .. 10] of integer; {массив для хранения текущего сочетания}
  n,m,i,j: integer;
begin
  writeln('Vvedite n,m');
  readln(n, m);
  for i:=1 to m do s[i]:=i; { Заполняем сочетание числами от 1 до m }
  while true do {Бесконечный цикл}
    begin
          for i:=1 to m do write(s[i],' '); {Выводим текущее сочетание}
      writeln;
      i := m; {Начинаем с последнего элемента сочетания}
```

```
while (i>0) and (s[i]=n-m+i) do i:=i-1; {Пока не исчерпаны все элементы сочетания и i-й элемент принимает свое максимально возможное значение, то переходим к предыдущему элементу}
if i=0 then break; {Если все элементы сочетания принимают свои максимально возможные значения, то все сочетания сгенерированы. Выходим из бесконечного цикла}
inc(s[i]); {Увеличиваем на 1 найденный элемент, который еще не достиг своего максимального значения}
for j:=i+1 to m do s[j]:=s[j-1]+1; {Все элементы, расположенные после него, становятся на единицу больше своих предшественников}
end; {Конец бесконечного цикла}
readln;
end.
```

Алгоритмы генерирования разбиения множества

Разбиение множества — это представление его в виде объединения произвольного количества попарно непересекающихся подмножеств.

Так как в каждом из разбиений участвуют все элементы исходного множества, будем в массиве индексов p записывать в какой блок попадает каждый из элементов в текущем разбиении.

```
type
  mas=array [1..100] of integer;
var
    n,i:integer;
    a,p:mas;
procedure vivod(n:integer; var p:mas);
var i,j,imax:integer;
begin
  imax:=1; {Определяем количество блоков в разбиении}
  for i:=1 to n do
    if p[i]>imax then imax:=p[i];
   for i:=1 to imax do {Проходим по всем блокам данного разбиения}
    begin
```

```
write('{');  {Выводим на экран i-й блок}
      for j:=1 to n do  {Просматриваем все элементы}
        if p[j]=i then write(a[j],' '); {Если элемент принадлежит i-му блоку то выводим его на экран}
          write('} ') {Блок напечатан}
    end;
  writeln; {Разбиение напечатано}
end;
procedure razb(i, j: integer);   {i- рассматриваемый элемент}
var l: integer;  {j- количество блоков в разбиении}
Begin {p - массив пометок, принадлежности к блоку разбиения}
if i>n then vivod(n, p) {Если рассматриваемый элемент больше, чем общее число элементов в множестве, то разбиение сформировано, выводим его}
  else
```

```
for l := 1 to j do   {Просматриваем все блоки}
    begin
      p[i] := l;  {Ставим i-й элемент в l-й блок, l=1,..,j}
        if l=j then razb(i+1, j+1) {Если i-й элемент вставили в последний блок, то переходим к следующему элементу i+1 и добавляем новый блок  j+1}
          else razb(i+1, j) {в противном случае переходим к следующему элементу i+1 не добавляя новый блок}
    end;
end;
 begin
  writeln('Vvedite kol. el. v mnoj');
  readln(n);
```

```
writeln(‘vvedite elementi mnoj’);
  for i:= 1 to n do
    read(a[i]);
  razb(1,1);
  readln;
end.
```

Домашнее задание

1. Составить опорный конспект лекции по теме «Алгоритмы генерирования перестановок. Алгоритмы генерирования множества всех подмножеств, к-элементных множеств, разбиение множества» на основе презентации.

2. Комбинаторика для программистов. Липский В. М.: «Мир», 1988, стр. 10-54.