М. С. Долинский,
Гомельский государственный университет имени Франциска Скорины, Беларусь

ГЕНЕРАЦИЯ КОМБИНАТОРНЫХ ОБЪЕКТОВ С ПОМОЩЬЮ РЕКУРСИВНЫХ ПРОЦЕДУР И ФУНКЦИЙ

Аннотация
В статье описана методика изучения темы «Рекурсивная генерация комбинаторных объектов» при подготовке школьников к олимпиадам по информатике. Технической основой является разработанная под управлением автора инструментальная система дистанционного обучения (http://dl.gsu.by).
Ключевые слова: рекурсия, комбинаторика, олимпиады по информатике, инструментальная система дистанционного обучения.
DOI: 10.32517/2221-1993-2019-18-4-59-63

Введение

С сентября 1996 года на базе средней школы № 27 г. Гомеля (Республика Беларусь), а с сентября 1999 года дополнительно и на базе сайта дистанционного обучения DL.GSU.BY ведется работа [2, 3] по факультативному изучению информатики и программирования школьниками разных возрастов. Ключевая особенность этого обучения — его раннее начало, фактически с первого класса. Как следствие, уже в пятом-шестом классах есть ученики, которым имеет смысл объяснять такие темы, как рекурсия. Поскольку традиционные подходы рассчитаны на обучение как минимум старшеклассников, то необходимы более простое и наглядное объяснение и более медленное продвижение по учебному материалу с явным выделением и обозначением всех этапов этого продвижения.

Введение в решение задач с помощью рекурсии изложено в работе [1]. В текущей статье читателям предлагается материал по теме «Рекурсивная генерация комбинаторных объектов», включающий условия задач и примеры их решения с помощью генерации таких комбинаторных объектов, как:

- множество всех подмножеств;
- сочетания;
- перестановки;
- перестановки с повторениями;
- размещения.

Проверка решений осуществляется автоматически на сайте DL.GSU.BY.

Предлагаем вдумчивым читателям после чтения условия задачи попытаться решить ее самостоятельно; если же это сделать не удалось, рекомендуется аналогичную попытку предпринять, воспользовавшись приведенными рекомендациями по решению задачи.

1. Множество всех подмножеств

Задача 1. Собиратель-2.

Сережа недавно нашел магические кубики — N штук с записанными на них какими-то разными числами. Сначала он решил сгруппировать некоторые кубики так, чтобы в сумме на них было 2010. С этой задачей Сережа справился легко, но его брат Руслан предложил ему решить кое-что посложнее, а именно: подсчитать количество способов, которыми можно набрать при помощи этих кубиков некоторую сумму M.

Формат ввода	*Пример ввода*
N M a_1 ... a_N Здесь: a_i — число, написанное на кубике; $1 \le N \le 20$; $1 \le M \le 10^{18}$; $1 \le a_i \le 10^{16}$	5 6 1 2 3 4 5
Формат вывода	*Пример вывода*
Ans — количество способов	3

Контактная информация

Долинский Михаил Семенович, канд. тех. наук, доцент, доцент кафедры математических проблем управления и информатики, Гомельский государственный университет имени Франциска Скорины, Беларусь; *адрес:* 246000, Республика Беларусь, г. Гомель, ул. Советская, д. 104; *e-mail:* dolinsky@gsu.by

M. S. Dolinsky,
Francisk Skorina Gomel State University, Belarus

GENERATING COMBINATORIAL OBJECTS USING RECURSIVE PROCEDURES AND FUNCTIONS

Abstract
The article describes the methodology to solve problems of Olympiads in informatics generating combinatorial objects using recursion. Distance learning system DL.GSU.BY is the effective technical base for teaching.
Keywords: recursion, combinatorics, teaching for programming, Olympiads in informatics, distance learning tools.

Для решения задачи достаточно рекурсивно перебрать все возможные комбинации элементов массива a, фактически рассматривая по очереди элементы множества всех подмножеств, состоящих из элементов массива a.

Для каждого такого подмножества проверяем сумму его элементов, и если сумма равна M, то прибавляем 1 к ответу. В рекурсивную процедуру передаем два параметра: N — количество элементов, которые осталось перебрать (взять или не взять каждый из них в подмножество), и M — сумму, которую осталось набрать, за вычетом суммы тех элементов $a[i]$, которые уже взяты в текущее подмножество. В связи с указанными ограничениями используем для элементов и их сумм тип int64.

Полный текст решения:

```
var
 a      : array [1..20] of int64;
 N, i   : longint;
 ans, M : int64;

procedure Rec(N: longint; M: int64);
begin
 if (N>=0) and (M=0) then
  begin inc(ans); exit; end;
 if (N=0) and (M<>0) then exit;
 Rec(N-1, M-a[n]);
 Rec(N-1, M)
end;

begin
  readln(N, M);
  for i:=1 to N do readln(a[i]);
  ans:=0;
  Rec(N, M);
  writeln(ans);
end.
```

2. Количество сочетаний из N по M

Задача 2. Двенадцатизначное число.

Задано число K. Необходимо выяснить, сколько существует двенадцатизначных чисел, в записи которых ровно K четных цифр (это цифры 0, 2, 4, 6, 8).

Формат ввода	*Пример ввода*
K — количество четных цифр; $0 \le K \le 12$	1
Формат вывода	*Пример вывода*
Ans — количество двенадцатизначных чисел, в записи которых ровно K четных цифр	2880859375

Пояснение.

Для $K = 1$ соответствующими двенадцатизначными числами (то есть числами, в записи которых ровно одна четная цифра) являются: 111111111110, 111111111112, 111111111114 и т. д.

Идея решения заключается в следующем.

Если $K = 0$,

то ответ: 5^12 — количество способов разместить нечетные цифры (1, 3, 5, 7, 9) на 12 позициях,

иначе ответ: 5^($N - K$)*(количество способов расставить K четных цифр по 12 позициям),

где:

5^($N - K$) — количество способов разместить пять нечетных цифр (1, 3, 5, 7, 9) на ($N - K$) позициях;

(количество способов расставить K четных цифр по 12 позициям) — это следующая сумма:

«поставить четыре четные цифры (2, 4, 6, 8) на первую позицию и пять четных цифр (0, 2, 4, 6, 8) на все остальные позиции (число сочетаний из 11 по $K - 1$)»

+

«поставить пять четных цифр (0, 2, 4, 6, 8) на $K - 1$ позицию, кроме первой (число сочетаний из 11 по $K - 1$)»:

$$4*C(11, K - 1) + C(11, K).$$

Число сочетаний из M по N вычисляется рекурсивно по стандартной формуле:

$$C(N, M) = C(N - 1, M - 1) + C(N - 1, M).$$

Исключения:

$C(N, M) = 1$, если $M = 0$, $N > 0$ или $0 \le N = M$;

$C(N, M) = 0$, если $0 \le N < M$.

Полный текст решения:

```
var
 p5   : array [0..12] of int64;
 i, k : longint;
 ans  : int64;

function C(N, M: longint): longint;
begin
 if ((M=0) and (N>0)) or
    ((M=N) and (M>0))
  then C:=1
  else if (M>N) and (N>=0)
   then C:=0
   else C:=C(N-1, M-1)+C(N-1, M)
end;

begin
 p5[0]:=1;
 for i:=1 to 12 do p5[i]:=p5[i-1]*5;
  readln(k);
  if k=0
    then ans:=p5[12]
    else ans:=p5[12-k]*(4*p5[k-1]*C(11,
k-1)+p5[k]*C(11, k));
  writeln(ans);
end.
```

3. Вывод сочетаний из N по M

Задача 3. Хорошие пароли.

Хороший пароль состоит из L ($3 \le L \le 15$) символов (из множества a...z), имеет хотя бы одну гласную (a, e, i, o, u) и хотя бы две согласные, все символы пароля должны идти в алфавитном порядке (т. е. abc — хороший пароль, bac — нехороший пароль).

По заданной длине L и C символам напишите программу, выводящую все хорошие пароли длиной L, которые могут быть сформированы из этих символов. Пароли должны быть выведены в алфавитном порядке, по одному в строке.

Формат ввода	*Пример ввода*
Строка 1: Два разделенных пробелами целых числа — *L* и *C*. Строка 2: С разделенных пробелами строчных латинских символов — множество символов, из которых можно формировать пароли	4 6 a t c i s w
Формат вывода	*Пример вывода*
Строки 1..?: Каждая выходная строка содержит одно слово длиной *L* символов (без пробелов). Все строки должны идти в алфавитном порядке	acis acit aciw acst acsw actw aist aisw aitw astw cist cisw citw istw

Пояснение.

Пароли длиной в четыре символа составляем из шести заданных символов.

Основная идея решения — рекурсивное построение всех возможных паролей (сочетаний из заданных букв):

```
procedure Rec(Nt, ni: longint; pwd: string);
var
 i : longint;
begin
 if (Nt=L) and Good(pwd) then writeln(pwd);
 if Nt=L then exit;
 inc(Nt);
 for i:=ni to C do
  Rec(Nt,i+1,Pwd+p[i]);
end;
```

При этом на позицию *Nt* (от 1 до *L*) могут попадать только буквы, которые в строке *P* (заданных символов без пробелов и в отсортированном порядке) находятся правее уже взятой буквы (*ni* — ее номер в строке *P*). *pwd* — текущее состояние уже набранной начальной части пароля.

Функция *Good*(*pwd*) «в лоб» проверяет «хорошесть» пароля (т. е. пароль имеет хотя бы одну гласную (a, e, i, o, u) и хотя бы две согласных):

```
function Good(pwd: string): boolean;
var
 kv, kc, i : longint;
begin
 kv:=0; kc:=0;
 for i:=1 to L do
  if pos(pwd[i], 'aeiou')=0
   then inc(kc)
   else inc(kv);
 Good:=(kv>=1) and (kc>=2);
end;
```

Полный текст решения:

```
var
 s, p    : string;
 L, C, i : longint;

procedure SortP;
var
 i, j : longint;
 t    : char;
begin
 for i:=1 to C do
  for j:=1 to C-1 do
   if p[j]>p[j+1]
    then
     begin
      t:=p[j]; p[j]:=p[j+1]; p[j+1]:=t;
     end;
end;

function Good(pwd: string): boolean;
var
 kv, kc, i : longint;
begin
 kv:=0; kc:=0;
 for i:=1 to L do
  if pos(pwd[i], 'aeiou')=0
   then inc(kc)
   else inc(kv);
 Good:=(kv>=1) and (kc>=2);
end;

procedure Rec(Nt, ni: longint; pwd: string);
var
 i : longint;
begin
 if (Nt=L) and Good(pwd) then writeln(pwd);
 if Nt=L then exit;
 inc(Nt);
 for i:=ni to C do
  Rec(Nt, i+1, Pwd+p[i]);
end;

begin
 readln(L, C);
 readln(s);
 p:='';
 for i:=1 to length(s) do
  if s[i]<>' ' then p:=p+s[i];
 SortP;
 Rec(0, 1, '');
end.
```

4. Перестановки

Задача 4. Перестановки.

Задан массив из *N* чисел. Коэффициент интересности массива *A* вычисляется по следующей формуле:

$$A[1]*1 + A[2]*2 + A[3]*3 + ... + A[N]*N,$$

т. е. это сумма произведений чисел массива на их порядковый номер.

Необходимо подсчитать количество возможных перестановок заданного массива *A*, коэффициент интересности которых нацело делится на *K*.

Формат ввода	*Пример ввода*
N K a_1 ... a_N Здесь: $1 \le N, K \le 10$, $1 \le a_i \le 20$	3 2 1 2 3
Формат вывода	*Пример вывода*
Ans	2

Пояснение.

Для решения задачи требуется сгенерировать все перестановки чисел от 1 до *N*:

```
procedure Rec(NT: longint);
var
 i : longint;
begin
 if NT=0
  then
   begin
    <анализируем очередную перестановку>
    exit;
   end;
 for i:=1 to N do
  if x[i]=0
   then
    begin
     x[i]:=1; inc(kb); b[kb]:=i; Rec(NT-1);
     x[i]:=0; dec(kb);
    end;
end;
```

Вызов процедуры

```
Rec(N);
```

Здесь *NT* указывает, сколько элементов осталось добавить в перестановку.

Для генерации перестановок из *N* элементов используется глобальный массив *x*, хранящий признаки использования элементов для *i* от 1 до *N*:

x[*i*] = 0 означает, что *i*-й элемент еще не включен в перестановку;

x[*i*] = 1 означает, что *i*-й элемент уже включен в перестановку.

Массив *b* хранит номера элементов перестановки в порядке включения.

В данной задаче можно вычислять сумму произведений по ходу вызовов рекурсии, и тогда нет необходимости хранить номера чисел в очередной перестановке.

Полный текст решения:

```
var
 a, x            : array [1..10] of longint;
 N, K, i, Ans, kb : longint;

procedure Rec(NT, Sum: longint);
var
 i : longint;
begin
 if NT=0
  then
   begin
    if (Sum mod K)=0 then inc(Ans);
    exit;
   end;
 for i:=1 to N do
  if x[i]=0
   then
    begin
     x[i]:=1; inc(kb); Rec(NT-1, Sum+a[i]*kb);
     x[i]:=0; dec(kb);
    end;
end;

begin
 readln(N, K);
 for i:=1 to N do readln(a[i]);
 for i:=1 to N do x[i]:=0;
 Ans:=0;
 Rec(N, 0);
 writeln(Ans);
end.
```

5. Перестановки с повторениями

Задача 5. Разбиение строки.

Руслан написал слово *S* прописными латинскими буквами (A..Z). Также у Руслана есть *N* типов бумажек с различными словами. Соединяя в произвольном порядке бумажки, Руслан может получить различные слова (каждый тип можно использовать несколько раз).

Помогите Руслану определить, сколько существует различных способов соединить бумажки так, чтобы получилось слово *S*.

Например, слово RUSLAN из бумажек со словами R, U, S, L, A, N, AN можно получить двумя способами:

R + U + S + L + A + N,

R + U + S + L + AN.

Формат ввода	*Пример ввода*
S *N* a_1 ... a_N Здесь: *S* — слово, которое требуется получить; длина слова от 1 до 20 символов; *N* — количество бумажек со словами; $1 \le N \le 20$; a_i — слово, которое написано на *i*-й бумажке; длина слова на бумажке — от 1 до 20 символов	AAAA 2 A AA
Формат вывода	*Пример вывода*
Ans — количество различных способов соединить бумажки так, чтобы получилось слово *S*	5

Фактически это задача на генерацию всех перестановок (с возможностью повторения). Будем рекурсивно пытаться «сопоставить» очередной кусочек строки с началом эталонной строки. Если сопоставление удалось, удаляем этот фрагмент из эталонной строки и продолжаем процесс, пока не получим пустую строку.

Полный текст решения:

```
var
 a         : array [1..20] of string;
 S         : string;
 N, i, ans : longint;

procedure Rec(S: string);
var
 i, p : longint;
 x    : string;
begin
 if S=''
  then inc(ans)
  else for i:=1 to N do
   begin
    if pos(a[i], s)=1
     then
      begin
       x:=s;
       delete(x, 1, length(a[i]));
```

```
      Rec(x);
     end;
   end;
end;

begin
 readln(S);
 readln(N);
 for i:=1 to N do readln(a[i]);
 ans:=0;
 Rec(S);
 writeln(ans);
end.
```

6. Размещения

Задача 6. Поиск знаков.

Дано выражение вида: 1?2?3?4?5?...?(N – 1)?N. Необходимо определить, какое наибольшее простое число можно получить в результате вычисления этого выражения, заменив знаки «?» на «+» или «–». Число называется простым, если у него нет никаких делителей кроме единицы и его самого.

Формат ввода	*Пример ввода*
N ($2 \le N \le 21$)	4
Формат вывода	*Пример вывода*
Ans — искомое наибольшее простое число	2

Рекурсивно генерируем все возможные алгебраические суммы. Каждую получившуюся сумму проверяем на простоту. Там же отсекаем как непростые все суммы, меньшие либо равные 1. Если получившееся число простое и больше, чем ранее запомненное, запоминаем его.

Полный текст решения:

```
var
 N, Ans : longint;

function Simple(x: longint): boolean;
var
 i : longint;
begin
 Simple:=false;
 if x<=1 then exit;
 for i:=2 to x div 2 do
  if (x mod i)=0 then exit;
 Simple:=true;
end;

procedure Rec(N, Sum: longint);
begin
 if N=1
  then
   begin
    if Simple(Sum) and (Sum>Ans) then Ans:=Sum;
    exit;
   end;
 Rec(N-1, Sum+N);
 Rec(N-1, Sum-N)
end;

begin
 readln(N);
 Ans:=0;
 Rec(N, 1);
 writeln(Ans);
end.
```

Заключение

В данной статье представлена методика изучения темы «Рекурсивная генерация комбинаторных объектов» с учениками пятых—восьмых классов, предлагающая наиболее простой, по мнению автора, способ постепенного осознания школьниками механизма рекурсии и способа решения задач с ее помощью. Методика включает в себя последовательность задач, снабженных при необходимости предварительными общими пояснениями и последующими полными решениями предлагаемых задач.

Список использованных источников

1. *Долинский М. С.* Введение в решение задач с помощью рекурсивных процедур и функций // Информатика в школе. 2016. № 9.

2. *Долинский М. С.* Гомельская школа олимпиадного программирования // Информатика и образование. 2015. № 7.

3. *Долинский М. С., Кугейко М. А.* Гомельская инструментальная система дистанционного обучения // Информатика и образование. 2010. № 11.