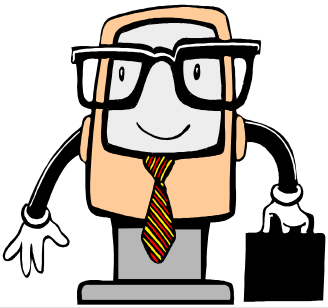

***М.С. Долинский**, к.т.н., доцент кафедры математических проблем управления Гомельского государственного университета им. Ф. Скорины*

Элементы теории чисел: работа с битами

Введение

Автор много лет занимается обучением программированию школьников разных возрастов и первокурсников математического факультета Гомельского государственного университета им. Ф. Скорины (специальности: «Программное обеспечение информационных технологий» и «Прикладная математика») [1-20]. Все это время автор занимался созданием литературы для самостоятельного изучения школьниками и студентами, стараясь представить материал в как можно более простой, наглядной и понятной форме. В данной статье приводится пример такого материала для обучения решению задач по информатике на тему «Работа с битами». Такой материал может быть интересен для преподавателей как в качестве иллюстрации методики обучения, так и по содержанию. В то же время, автору представляется, что этот материал может оказаться весьма полезным и интересным и для школьников, и для студентов, занимающихся самообучением. Всем заинтересованным предлагается следующий порядок работы: откладывать статью в сторону и пытаться самостоятельно выполнить предлагаемое задание: первый раз –

после прочтения условия задачи, второй раз – после прочтения указаний к решению.

Перебор множества всех подмножеств

Часто приходится решать задачи, в которых для получения ответов требуется перебрать и проанализировать все возможные комбинации из некоторого множества элементов. Например, рассмотрим задачу:

08_AprB – «Going to the Movies»

Фермер Джон решил свозить некоторых из своих коров в кино! Поскольку его грузовик имеет ограниченную грузоподъемность C (100 <= C <= 5000) килограммов, он хочет взять максимальный вес коров, не превысив грузоподъемности C машины.

По заданному N (1 <= N <= 16) и их соответствующим весам W_i, определите вес самой тяжелой группы коров, которую ФД может взять в кино.

Формат ввода

* Строка 1: Два разделенных пробелом целых числа: C N
* Строки 2..N+1: Строка i+1 содержит одно целое число: W_i

Пример ввода (файл cowflix.in):

259 5
81
58
42
33
61

Формат вывода

* Строка 1: Одно целое число – вес самой тяжелой группы коров, которую можно увезти.

Пример вывода (файл cowflix.out):

242

Пояснения

81+58+42+61 = 242; эта наибольшая возможная сумма.

Указания к решению

Для получения ответа нужно перебрать все возможные комбинации из одной, двух, трех, ... четырнадцати, пятнадцати и шестнадцати коров. Каждый раз складывать их веса, сравнивать с грузоподъемностью и запоминать наибольший из суммарных весов, не превысивший грузоподъемности.

В таких задачах можно использовать двоичное представление чисел.

Напомним, что двоичное число – это последовательность нулей и единиц, например, последовательность двоичного счета от 0 до 15, представлена в таблице ниже:

10	2
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Оказывается, эта последовательность обеспечивает нам перебор всех возможных комбинаций из четырех элементов. Пусть первый столбик отвечает за наличие/отсутствие первого элемента, второй столбик отвечает за наличие/отсутствие второго элемента, третий столбик отвечает за наличие/отсутствие третьего элемента, а четвертый столбик отвечает за наличие/отсутствие четвертого элемента (например, 0 означает, что элемент отсутствует, а 1, что элемент присутствует). Тогда, строчка 0000 означает, что все элементы отсутствуют (пустое множество), строчка 1111 означает, что все элементы присутствуют, а строчка 1100 означает, что первый и второй элементы присутствуют, а третий и четвертый – отсутствуют.

Таким образом, если мы заведем целочисленную переменную i и посчитаем в ней (с помощью оператора i:=i+1) от 0 до 15, мы получим множество всех комбинаций из 4 элементов. А если у нас 6 элементов, тогда до какого максимального числа нужно посчитать? До 63 (2 в 6-ой степени минус 1). В общем случае, если у нас K элементов, то надо посчитать от 0 до 2 в степени K минус 1.

В процессе счета необходимо конвертировать значение переменной I в двоичное представление и проверить, подходит ли такая комбинация элементов под условие задачи.

```
var
    a : array [0..15] of longint;
    c,n,x,i,k,sum,max : longint;
begin
    assign(input,'cowflix.in'); reset(input);
    assign(output,'cowflix.out'); rewrite(output);
    readln(c,n);
    for i:=1 to n do readln(a[i]);
    max:=0;
    for i:=1 to 32*1024 do
        begin
            x:=i;
            sum:=0; k:=0;
            while x<>0 do
                begin
                    if odd(x) then inc(sum,a[k]);
                    inc(k);
                    x:=x shr 1;
                end;
            if (sum<=C) and (sum>max) then max:=sum;
        end;
    writeln(max);
    close(input); close(output);
end.
```

Пояснения

Веса коров считываются в массив a[i] от 0 до 15. Соответственно, коровы нумеруются от 0 до 15. Переменная k указывает номер коровы и, одновременно, номер бита (цифры в двоичном представлении) числа i. Биты также нумеруются от 0 до 15 справа налево.

Задача 06_AprB – «Cow Pizza»

Коровы любят пиццу. Пицца может иметь T (1 <= T <= 20) различных приправ. Приправы пронумерованы от 1 до T, так что коровы могут заказывать приправы числами. Ваша задача – посчитать, сколько различных пицц можно составить, если известно, что некоторые комбинации приправ не приемлемы для коров. Например, некоторые коровы не едят анчоусы, а другие – комбинации грибов и спаржи.

Задано множество из N (1 <= N <= 52) ограничений, подсчитайте, сколько можно составить пицц, используя все возможные разрешенные комбинации приправ (включая пиццу вовсе без приправ). Каждое ограничение – это множество чисел от 1 до T, указывающих запрещенные приправы. Например, ограничение «5 3» означает, что пицца не может содержать приправу номер 5, а также приправу номер 3.

Формат ввода

* Строка 1: Два разделенных пробелом целых числа T и N

* Строки 2..N+1: Каждая строка описывает ограничения, используя разделенные пробелами целые числа. Первое целое число – количество приправ в ограничениях Z. Следующие Z целых различных чисел описывают запрещенные приправы (их комбинации исключают пиццу).

Пример ввода (файл pizza.in):

```
6 5
1 1
2 4 2
3 3 2 6
1 5
3 3 4 6
```

Пояснения к вводу

Всего 6 приправ с номерами от 1 до 6. Пять ограничений: пицца не должна содержать приправу 1, приправы 4 и 2, ...

Формат вывода

* Строка 1: одно целое число – общее количество пицц, которые можно приготовить, используя заданные приправы с учетом ограничений.

Пример вывода (файл pizza.out):

```
10
```

Пояснения к выводу

Эти 10 пицц таковы – без приправ, 1, 2 3, 2 6, 2, 3 4, 3 6, 3, 4 6, 4.

Указания к решению задачи

Прежде всего, обратим внимание на ошибку в пояснениях к ответу (OUTPUT DETAILS в оригинале). Сам ответ 10 правильный, а вот приведенный список разрешенных комбинаций приправ – нет. Например, пицца с приправой 1 запрещена первым ограничением. И это не ошибка перевода. Желающие могут убедиться, что английский (оригинальный) вариант условий содержит те же числа в OUTPUT DETAILS. Ошибка наме-

ренно не устранена, чтобы лишний раз подчеркнуть – такое возможно в реальных олимпиадах. Если у участника есть возможность – надо уточнить у жюри, а если нет – ПРИНЯТЬ РЕШЕНИЕ САМОМУ.

По сравнению с предыдущей задачей (с перебором всех возможных вариантов) здесь введено усложнение – необходимо исключить некоторые варианты из рассмотрения, и представлены обобщенные списки этих запрещенных вариантов: например, исключить все варианты, содержащие элемент 1 или все варианты, содержащие элементы 4 и 2.

И здесь двоичная система приходит нам на помощь. Можно завести массив чисел p2, являющихся степенями числа 2 от 0 до 20 (максимально есть 20 приправ). И прямо на вводе закодировать запрещенную комбинацию вариантов в число e, а их последовательность – в массив z[i]:

```
const
   p2 : array [0..20] of longint =
      (1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,
      16384,32768,64*1024,128*1024,256*1024, 512*1024,
      1024*1024);
...
   for i:=1 to n do
      begin
         read(k); e:=0;
         for j:=1 to k do
            begin read(x); inc(e,p2[x-1]); end;
         z[i]:=e;
      end;
```

А проверку, запрещает ли комбинация z[i] текущий вариант x, выполнять следующим образом

1) j:=x and z[i];

В результате переменная j получит значение 0 на всех позициях, где в z[j] находится 0, а во всех остальных позициях (где z[j]=1) туда скопируется значение соответствующего бита из переменной x.

Напомним таблицу истинности для операции and

X	z[i]	and
0	0	0
0	1	0
1	0	0
1	1	1

2) j:=j xor z[i];

Теперь, если в j и z[i] на одной позиции стоят одинаковые биты (0 в обоих переменных, или 1 в обоих переменных), то соответствующий бит переменной j получит значение 0. А если в j и z[i] стоят разные значения (0 в одной и 1 в другой, или 1 в одной и 0 в другой), то соответствующий бит переменной j получит значение 1.

Good2:= j<>0;

Этот оператор присваивает функции значение истина, если переменная j не равна 0, а ограничение z[i] не запрещает комбинацию x.

Далее приведен полный текст решения.

```
const
    p2 : array [0..20] of longint =
        (1,2,4,8,16,32,64,128,256,512,1024, 2048,4096,8192,
        16384, 32768,64*1024,128*1024,256*1024,512*1024,
        1024*1024);
var
    z : array [1..52] of longint;
    t,n,i,j,k,x,e : longint;

procedure InputData;
    begin
        readln(t,n);
        for i:=1 to n do
            begin
                read(k); e:=0;
                for j:=1 to k do
                    begin read(x); inc(e,p2[x-1]); end;
                z[i]:=e;
            end;
    end;

function Good2(i,x:longint) : boolean;
    var
        j : longint;
    begin
        j:=x and z[i];
        j:=j xor z[i];
        Good2:= j<>0;
    end;

function Good(x:longint) : boolean;
    var
```

```
        i : longint;
    begin
        i:=1;
        while (i<=n) and Good2(i,x) do inc(i);
        Good:= i>n;
    end;

begin
    assign(input,'pizza.in'); reset(input);
    assign(output,'pizza.out'); rewrite(output);
    InputData;
    j:=1;
    for i:=1 to p2[t]-1 do
        if Good(i) then inc(j);
    writeln(j);
    close(input); close(output);
end.
```

Задача 06_JanB – «The Water Bowls»

Перед коровами 20 баллонов. Вначале баллоны могут быть ориентированы «вверх» либо «вниз». Требуется сориентировать все баллоны «вниз». Однако корова умеет «переворачивать» только по три баллона вместе: средний, а также баллоны слева и справа от него. Если баллон крайний (самый левый или самый правый) – то переворачиваются только два баллона.

По заданному начальному состоянию баллонов (1 – вверх, 0 – вниз) определите минимальное количество «переворачиваний», которое требуется, чтобы перевести все баллоны в состояние «вниз».

Формат ввода

* Строка 1: 20 целых чисел, разделенных одиночными пробелами

Пример ввода (файл bowls.in):

0 0 1 1 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0

Формат вывода

* Строка 1: Минимальное количество переворачиваний баллонов, которое требуется, чтобы перевести их в состояние «вниз», то есть в 0. Для заданных входных состояний всегда это можно сделать.

Пример вывода (файл bowls.out):

3

Пояснения

Переворачивая баллоны 4 9 и 11, мы получаем требуемый результат:

0 0 1 1 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 [начальное состояние]
0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 [после переворачивания 4]
0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 [после переворачивания 9]
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 [после переворачивания 11]

Указания к решению

Во-первых, при вводе необходимо преобразовать введенную последовательность из 0 и 1 в число, представленное такой двоичной последовательностью:

```
b:=0;
for i:=1 to 20 do
    begin
        read(k);
        if k=1 then inc(b,p2[20-i]);
    end;
```

Здесь P2 – снова массив степеней двойки. B – число, соответствующее введенной двоичной последовательности.

Основной цикл перебора может выглядеть следующим образом:

```
min:=maxlongint;
for i:=0 to p2[20]-1 do
    begin
        k:=NumBit(i);
        if (k<min) and Zero(i,b) then min:=k;
    end;
```

Здесь единицы в числе I будут указывать, какие баллоны нужно переворачивать в качестве центральных.

Функция NumBit(i) вычисляет количество единиц в числе i.

Функция Zero(i,b) проверяет, приведет ли переворачивание по закону i исходного состояния баллонов b в требуемое состояние 0.

Если единиц меньше, чем текущее минимальное, а переворачивание обращает в 0, то запоминаем новое количество единиц (баллонов, которые нужно перевернуть в качестве центральных).

Подсчет количества единиц в числе x ведется с помощью операции AND. Напомним, что числа в массиве p2 (степеней двойки) характеризуются тем, что имеют 1 в одном бите и нолики во всех остальных.

```
k:=0;
for i:=0 to 19 do
    if (x and p2[i])<>0 then inc(k);
```

Наконец, проверка, переводит ли комбинация i исходную последовательность b к нулевому состоянию, осуществляется так:

```
{b – исходные баллоны}
{i – определяет баллоны, которые нужно перевернуть}
if (i and p2[0])<>0 then b:=b xor (p2[0] +p2[1]);
if (i and p2[19])<>0 then b:=b xor (p2[19]+p2[18]);
for j:=1 to 18 do
    if (i and p2[j])<>0
        then b:=b xor (p2[j-1]+p2[j]+p2[j+1]);
Zero:= b=0;
```

Напомним также, что, по условию задачи, в случае крайних баллонов (0 и 19) переворачивается только один соседний. А в случае «средних баллонов» (с 1 по 18) переворачивается он и оба его соседа – слева и справа. Переворачивание осуществляется с помощью операции xor.

Если в результате переменная b получает значение 0, значит, комбинация I переворачивает все баллоны в исходное состояние.

Далее приводится полный текст решения.

```
var
    p2 : array [0..20] of longint;
    i,k,j,b,min : longint;
function Numbit(x:longint) : longint;
    var
        k,i : longint;
    begin
        k:=0;
        for i:=0 to 19 do
            if (x and p2[i])<>0 then inc(k);
        Numbit:=k;
    end;
function Zero(i,b:longint):boolean;
    var
        j,p3 : longint;
    begin
{b – исходные баллоны}
{i – определяет баллоны, которые нужно перевернуть}
        if (i and p2[0])<>0 then b:=b xor (p2[0] +p2[1]);
```

```
    if (i and p2[19])<>0 then b:=b xor (p2[19]+p2[18]);
    for j:=1 to 18 do
        if (i and p2[j])<>0
            then b:=b xor (p2[j-1]+p2[j]+p2[j+1]);
    Zero:= b=0;
end;
begin
    assign(input,'bowls.in'); reset(input);
    assign(output,'bowls.out'); rewrite(output);
    p2[0]:=1;
    for i:=1 to 20 do p2[i]:=2*p2[i-1];
    b:=0;
    for i:=1 to 20 do
        begin
            read(k);
            if k=1 then inc(b,p2[20-i]);
        end;
    min:=maxlongint;
    for i:=0 to p2[20]-1 do
        begin
            k:=NumBit(i);
            if (k<min) and Zero(i,b) then min:=k;
        end;
    writeln(min);
    close(input); close(output);
end.
```

Заключение

В данной статье приведен материал для обучения решению задач по информатике на тему «Работа с битами». Технической основой методики является разработанная инструментальная система дистанционного обучения (Distance Learning Belarus – http://dl.gsu.by). Все задачи, приведенные в статье, могут быть сданы в курсе «Методы алгоритмизации».

Литература

1. Долинский, М.С. Об опыте подготовки школьников Гомельской области к республиканским и международным олимпиадам по информатике / М.С. Долинский // Информатизация образования. – 2009. – № 1(54). – С. 29-40.

2. Долинский, М.С. Система интернет-курсов дифференцированного обучения программированию школьников и сту-

дентов / М.С. Долинский, М.А. Кугейко // Информатизация образования. – 2010. – № 1(58). – С. 58-68.

3. Долинский, М.С. Как учить думать школьников и студентов? / М.С. Долинский, М.А. Кугейко // Информатизация образования. – 2010. – № 2(59). – С. 62-72.

4. Долинский, М.С. Технология развивающего дифференцированного обучения программированию младших школьников «с чистого листа» / М.С. Долинский, М.А. Кугейко // Информатизация образования. – 2010. – № 3(60). – С. 12-20.

5. Долинский, М.С. Интернет-курс «Базовое программирование» как средство подготовки к областным олимпиадам по информатике / М.С. Долинский, М.А. Кугейко // Информатизация образования. – 2010. – № 4(61). – С. 3-15.

6. Долинский, М.С. Развитие мышления младших школьников на основе флеш-заданий на рисование, раскраску и конструирование в системе DL.GSU.BY / М.С. Долинский, Ю.В. Решетько, М.А. Кугейко // Информатизация образования. – 2011. – № 1(62). – С. 24-35.

7. Долинский, М.С. Какими должны быть задачи на олимпиадах по информатике / М.С. Долинский, М.А. Кугейко // Информатизация образования. – 2011. – № 1(62). – С. 68-76.

8. Долинский, М.С. Флеш-шаблоны для создания заданий развивающего обучения / М.С. Долинский, Ю.В. Решетько, М.А. Долинская // Информатизация образования. – 2011. – № 2(63). – С. 14-28.

9. Долинский, М.С. Конструирование интерактивных флеш-заданий на развитие мышления / М.С. Долинский, Ю.В. Решетько, М.А. Долинская // Информатизация образования. – 2011. – № 3(64). – С. 21-33.

10. Долинский, М.С. Конструирование интерактивных флеш-заданий на развитие мышления на базе произвольных картинок / М.С. Долинский, Ю.В. Решетько, М.А. Долинская // Информатизация образования. – 2011. – № 4(65). – С. 3-14.

11. Долинский, М.С. Конструирование интерактивных флеш-заданий на базе собственных танов / М.С. Долинский, Ю.В. Решетько, Н.С. Лебедько // Информатизация образования. – 2012. – № 1(66). – С. 24-34.

12. Долинский, М.С. Конструктор интерактивных флеш-заданий как открытая система для создания электронных учебных пособий / М.С. Долинский, Ю.В. Решетько, М.А. Долинская, Н.С. Лебедько // Информатизация образования. – 2012. – № 2(67). – С. 35-45.

13. Долинский, М.С. Электронное учебное пособие «Математика. Начальная школа» / М.С. Долинский, Ю.В. Решетько, Н.С.Лебедько // Информатизация образования. – 2012. – № 3(68). – С. 30-42.

14. Долинский, М.С. Создание электронных учебных пособий для вузовских дисциплин с помощью конструктора флеш-заданий / М.С. Долинский, Ю.В. Решетько // Информатизация образования. – 2012. – № 4(69). – С. 34-45.

15. Долинский, М.С. Интерактивная анимация в электронных учебных пособиях, создаваемых с помощью конструктора флеш-заданий / М.С. Долинский, Ю. В. Решетько, М.А. Долинская // Информатизация образования. – 2013. – № 1(70). – С. 30-38.

16. Долинский, М.С. Учебный интернет-курс и перманентный интернет-конкурс «Математика 1-8 кл.» / М.С. Долинский, Ю.В. Решетько, М.А. Долинская // Информатизация образования. – 2013. – № 2(71). – С. 38-47.

17. Долинский, М.С. Концептуальные основы и практика сквозного развивающего обучения информатике и программированию от детского сада до вуза / М.С. Долинский, Ю.В. Решетько, М.А. Долинская // Информатизация образования. – 2013. – № 3(72). – С. 16-25.

18. Долинский, М.С. Об одном подходе к обучению программированию на первом курсе / М.С. Долинский, М.А. Долинская // Информатизация образования. – 2014. – № 1(73). – С. 32-41

19. Долинский, М.С. Использование форума при обучении программированию первокурсников / М.С. Долинский // Информатизация образования. – 2014. – № 2(74). – С. 22-34.

20. Долинский, М.С. Элементы теории чисел: системы счисления / М.С. Долинский // Информатизация образования. – 2015. – № 1(75), С. 14-28.

Статья поступила 03.07.2015