Разбор задач Шестой Интернет-олимпиады

Введение

В базовой номинации Шестой Интернет-олимпиады сезона 2008-2009 участникам было предложено для решения восемь задач. В олимпиаде приняло участие 106 команд, из них 90 решили хотя бы одну задачу.

Наиболее простой оказалась задача «C. Привет» — ее решили 84 команды. Наиболее сложной — задача «F. Период» — ее решили 17 команд.

Условия задач, результаты олимпиады, тесты и решения жюри можно найти на сайте интернет-олимпиад `http://neerc.ifmo.ru/school/io`.

Задача A. Окружности-2

Автор задачи: Антон Ахи
Автор разбора: Антон Ахи

Данная задача имеет в качестве ответа формулу $n^2 - n + 2$. Эта формула выдает правельный ответ для всех $n > 0$, поэтому случай $n = 0$ необходимо рассмотреть отдельно (в таком случае ответ 1).

Докажем эту формулу по индукции. Для случая $n = 1$ формула верна. Пусть формула верна для $k - 1$, докажем,что формула верна для k. Для этого рассмотрим, наиболее выгодный способ размещения k-ой окружности. Очевидно, что необходимо, чтобы новая окружность пересекала все старые (по два пересечения с каждой), и при этом чтобы не возникало ситуаций, когда более двух окружностей пересекаются в одной точке. Так всегда можно сделать. Таким образом на новой окружности возникнет $2(k-1)$ точки пересечения, а значит она будет разделена на $2(k-1)$ дуг, каждая из которых делит одну из предыдущих частей на две. Таким образом, если для $k - 1$ была верна формула $(k-1)^2-(k-1)+2 = k^2-2k+1-k+1+2 = k^2-3k+4$, то для k получаем $k^2-3k+4+2(k-1) = k^2-k+2$. Таким образом формула доказана.

Код программы, решающей данную задачу:

```
var
  n : longint;

begin
  reset(Input, 'circles2.in');
  rewrite(Output, 'circles2.out');
  read(n);
  if (n = 0) then
    writeln('1')
  else
    writeln(n * n - n + 2);
end.
```

Задача B. Цепная дробь

Автор задачи: Александр Торопов
Автор разбора: Антон Ахи

Будем решать задачу с помощью сведения ее к аналогичной с меньшими числами. Пусть на i-ом шаге требуется вычислить цепную дробь для $\frac{p}{q}$. Тогда $a_i = \left\lfloor \frac{p}{q} \right\rfloor$, $\frac{p}{q} = a_i + \frac{1}{x}$, $x = \frac{q}{p \mod q}$. Таким образом мы свели задачу к решению задачи для дроби $\frac{q}{p \mod q}$. Заметим, что сумма числителя и знаменателя уменьшилась. Так как она будет все время уменьшаться, то рано или поздно получим дробь $\frac{1}{1}$.

Код программы, решающей данную задачу:

```
var
  p, q, tmp, n, i : longint;
  a : array [1..1000] of longint;

begin
  reset(Input, 'frac.in');
  rewrite(Output, 'frac.out');
  read(p, q);
  n := 0;
  while (q > 0) do begin
    Inc(n);
    a[n] := p div q;
    tmp := p mod q;
    p := q;
    q := tmp;
  end;
  writeln(n);
  for i := 1 to n do
   write(a[i], ' ');
end.
```

Задача C. Привет

Автор задачи: Антон Ахи
Автор разбора: Федор Царев

Отдельно найдем число рукопожатий A, число B раз, которое слово «Привет» было произнесено мальчиком, и число C раз, которое слово «Привет» было произнесено девочкой.

Для того, чтобы найти число рукопожатий, рассмотрим мальчика, который вошел в комнату i-ым по порядку (из мальчиков). Он пожмет руку $(i-1)$ мальчику. Значит, общее число рукопожатий будет равно $A = 1 + 2 + \ldots + (n-1) = \frac{n \cdot (n-1)}{2}$.

Так как каждый мальчик скажет «Привет» каждой девочке, то $B = n \cdot m$.

Далее, каждая девочка скажет «Привет» каждому мальчику и каждой другой девочке, поэтому $C = n \cdot m + m \cdot (m-1)$.

Из этого следует, что ответы на задачу равны $A = \frac{n \cdot (n-1)}{2}$ и $B + C = 2 \cdot n \cdot m + m \cdot (m-1)$. Отметим, что для вычисления этих чисел следует использовать 64-битный тип данных, так как оно может быть не представимо в рамках 32-битного типа.

Приведем программную реализацию.

```
var
  n, m : int64;
begin
  reset(input, 'hello.in');
  rewrite(output, 'hello.out');
  read(n, m);
  writeln(n * (n - 1) div 2, ' ', 2 * n * m + m * (m - 1));
end.
```

Задача D. Метро

Автор задачи: Владимир Ульянцев
Автор разбора: Антон Феськов

Прежде чем решать задачу, рассмотрим упрощенный ее вариант: какое минимальное число станций может содержать метро, подчиняющееся правилам из условия, если известно только количество n линий? Ответ прост: каждая из n линий пересекается с $n-1$ линией метро, значит минимальное число станций есть $f(n) = \frac{n \cdot (n-1)}{2}$. Делить пополам надо, так как иначе каждая станция будет посчитана дважды — как пересечение линии A с линией B, и как пересечение B с A.

Вернемся к решению поставленной задачи. Поскольку ни про какие линии кроме первой ничего не известно, то, помимо станций на первой линии, в метро должно быть еще хотя бы $f(n-1)$ станций. Эта оценка является оценкой снизу и, очевидно, достигается. Таким образом ответ на задачу есть $k + f(n-1) = k + \frac{(n-1) \cdot (n-2)}{2}$

Фрагмент реализации на языке Паскаль:

```
read(n, k);
writeln((n - 1) * (n - 2) div 2 + k);
```

Время работы и необходимая дополнительная память — $O(1)$.

Задача E. Двоичное число

Автор задачи: Антон Ахи
Автор разбора: Федор Царев

Для решения этой задачи необходимо заметить, что после выполнения n указанных операций к числу n получится число, двоичная запись которого содержит разрядов, столько же, сколько двоичная запись n содержит единиц, причем во всех этих разрядах находятся единицы. Например, из числа $14_{10} = 1110_2$ получится число $7_{10} = 111_2$. Говоря иными словами, если в двоичной записи числа n содержится k единиц, то ответом на задачу является число $2^k - 1$.

Таким образом, для решения исходной задачи необходимо найти число единиц в двоичной записи числа n. Для решения этой задачи существует несколько алгоритмов. Опишем один из них. Этот алгоритм основан на базовой операции «удаления из числа младшей единицы», которая выражается формулой $x = x \wedge (x-1)$, где как $\wedge$ обозначена операция побитового «И».

Докажем, что после применения этой операции в числе x самая младшая единица в двоичной записи заменится на ноль. Обозначим номер разряда, в котором эта единица находится как k. Заметим, что в числе $x-1$ в этом разряде находится ноль, в всех более младших — единицы. В числе x все наоборот: в k-ом разряде единица, а более младших — нули. Поэтому в числе $x \wedge (x-1)$ все разряды от k-ого и младше содержат нули. Таким образом, в в двоичной записи этого числа на одну единицу меньше, чем в двоичной записи числа x.

Приведем программную реализацию этого алгоритма.

```
function oneCount(x : int64) : integer;
begin
  result := 0;
  while (x > 0) do begin
    result := result + 1;
    x := x and (x - 1);
  end;
end;
```

С помощью этой функции исходную задачу можно решить следующим образом.

```
k := oneCount(n);
m := 1;
for i := 1 to k do begin
  m := m * 2;
end;
writeln(m - 1);
```

Отметим также, что в этой задаче необходимо использовать 64-битный тип данных, так как даже исходное число n не может быть представлено в рамках 32-битного типа.

Задача F. Период

Автор задачи: Владимир Ульянцев
Автор разбора: Антон Феськов

Первое, что стоит сделать при решении задачи — переформулировать условие. Последняя цифра числа n в k-ичной системе счисления — это всего лишь остаток от деления n на k. Значит приведенная в условии последовательность имеет следующий вид: $A_0 = 1$, а $A_i = (A_{i-1} \cdot n) \bmod k$ при $i > 0$.

Заметим, что если какие-то два элемента этой последовательности совпадают, то и следующие за ними также совпадают. Таким образом, задача сведена к нахождению двух ближайших совпадающих чисел в последовательности A. Эта задача в свою очередь эквивалентна нахождению минимального натурального j, такого что A_j встречается среди $A_0, A_1, \ldots A_{j-1}$.

Несложно видеть, что среди первых $k+1$ элементов A встретятся два равных. Число k мало, потому можно действовать таким образом: последовательно находим члены последовательности, запоминая на каком шаге был получен элемент x в массиве *step*. Если вновь полученный элемент уже встречался ранее, то выводим разницу между текущим шагом и тем, на котором был ранее получен этот элемент.

Фрагмент решения, на языке Паскаль:

```
fillchar(step, sizeof(step), -1);
read(n, k);
n := n mod k;
i := 1;
j := 0;
while (step[i] = -1) do begin
          step[i] := j;
          inc(j);
          i := (i * n) mod k;
end;
writeln(j - step[i]);
```

Здесь `j` — номер текущего шага, причем $i = A_j$. Время работы алгоритма есть $O(k)$

Задача G. Точная степень

Автор задачи: Федор Царев
Автор разбора: Федор Царев

Рассмотрим некоторое представление числа x в виде $x = a^b$. Так как по условию $b > 1$, то $a \leq \sqrt{x}$. Так как $x \leq 10^9$, то $a \leq \sqrt{10^9}$, поэтому существует менее сорока тысяч возможных значений числа a. Эти значения основания степени можно перебрать, а для каждого из них — проверить существует ли соответствующее значение показателя степени b.

Эту проверку можно организовать с помощью последовательного вычисления степеней a^1, a^2, a^3, … до тех пор, пока очередное значение не превзойдет число x. Особо отметим, что на этом этапе следует использовать 64-битные типы данных (`int64` в языке *Delphi*, `long long` в языках *C* и *C++*, `long` — в языке *Java*), так как, например, при $x = 10^9$, $a = 10$ число $a^9 = 10^9 \leq x$ и может быть представлено с помощью 32-битного типа данных, а $a^{10} = 10^10$ уже превышает x и с помощью 32-битного типа данных представлено быть не может.

Приведем программную реализацию этого алгоритма на языке программирования *Delphi*.

```
var
  ansa, ansb : array[1..1000] of longint;
  cnt : lognint;
  x : longint;
  a : longint;
  b : longint;
  t : int64;
begin
  read(x);
  a := 1;
  while (a * a <= x) do begin
    b := 1;
    t := a;
    while (t <= x) do begin
      if (t = x) then begin
        cnt := cnt + 1;
        ansa[cnt] := a;
        ansb[cnt] := b;
        break;
      end;
      b := b + 1;
      t := t * a;
    end;
    a := a + 1;
  end;
  writeln(cnt);
  for i := 1 to cnt do begin
    writeln(ansa[i], ' ', ansb[i]);
  end;
end.
```

Задача H. Времечко

Автор задачи: Владимир Ульянцев
Автор разбора: Федор Царев

Решение этой задачи удобно разделить на две части — вычисление числа вхождений десятичной записи заданного числа в строку и перебор всех времен в течение суток с шагом в минуту.

Так как рассматриваемые в задачи строки имеют длину не более четырех символов, то для решения первой части задачи можно воспользоваться наиболее простым алгоритмом — перебрать все позиции в первой строке и для каждой из них проверить соответствующую подстроку. Реализуем это алгоритм в виде функции.

```
function substringCount(t : string; s : string) : integer;
```

```
var
  n, m : integer;
  i, j : integer;
  good : booolean;
begin
  n := length(t);
  m := length(s);
  result := 0;
  for i := 1 to n - m + 1 do begin
    good := true;
    for j := 1 to m do begin
      if (t[i + j - 1] <> t[j]) then begin
        good := false;
        break;
      end;
    end;
    if (good) then begin
      result := result + 1;
    end;
  end;
end;
```

В этой функции параметр `t` (от слова text) обозначает строку, в которой осуществляется поиск, а параметр `s` (от слова string) — строку, вычисление числа вхождений которой производится.

Время работы этого алгоритм есть $O(|t| \cdot |s|)$, где как $|s|$ и $|t|$ обозначены длины строк s и t соответственно. Отметим, что поиска подстроки в строке существуют и более быстрые алгоритма, например, алгоритм Кнута-Морриса-Пратта со временем работы $O(|s| + |t|)$. Об этом алгоритме можно прочитать в [1].

Перейдем ко второй части решения. Перебор всех моментов времени с шагом в минуту можно осуществить с помощью двух вложенных циклов. После этого необходимо найти представление времени в виде строки, и с помощью функции `substringCount` найти число вхождений представления заданного числа n в представление текущего времени.

```
s := inttostr(n);
ans := 0;
for h := 0 to 23 do begin
  for m := 0 to 59 do begin
    t := inttostr(h div 10) + inttostr(h mod 10) +
         inttostr(m div 10) + inttostr(m mod 10);
    ans := ans + substringCount(t, s);
  end;
end;
writeln(ans);
```

Список литературы

[1] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: Построение и Анализ. — М.: МЦНМО, 1999.