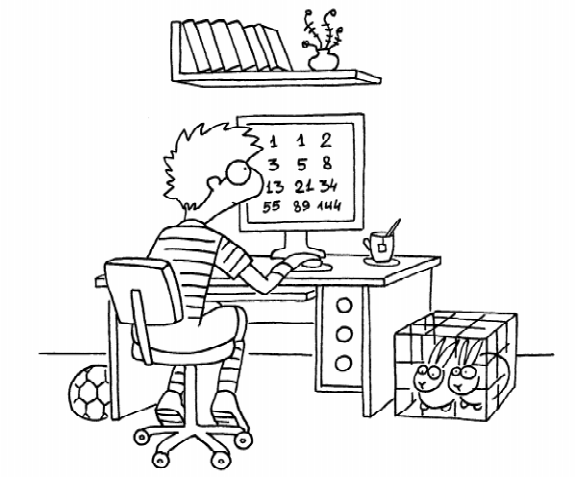
Ульянцев Владимир Игоревич,
Царёв Фёдор Николаевич

ЗАДАЧА «СТРОКИ ФИБОНАЧЧИ»

Этой статьей мы продолжаем цикл публикаций олимпиадных задач для школьников по информатике и программированию с разборами.

Решение таких задач и изучение разборов поможет Вам повысить уровень практических навыков программирования и подготовиться к олимпиадам по информатике и программированию.

В этой статье рассматривается задача «Строки Фибоначчи», которая предлагалась в первой Интернет-олимпиаде базового уровня сезона 2009–2010. Интернет-олимпиады по информатике базового уровня проводятся Санкт-Петербургским государственным университетом информационных технологий, механики и оптики. Сайт этих олимпиад находится по адресу http://neerc.ifmo.ru/school/io/.



УСЛОВИЕ ЗАДАЧИ

В математике достаточно часто применяются так называемые рекуррентные соотношения. Обычно они применяются для задания числовых последовательностей, но могут применяться и для задания последовательностей строк.

Одним из примеров строк, задаваемых рекуррентным соотношением, являются *строки Фибоначчи* F_0, F_1, ... Они задаются следующим образом: $F_0 = a$, $F_1 = b$, $F_i = F_{i-2}F_{i-1}$, $i > 1$. Первые семь строк Фибоначчи выглядят следующим образом: *a*, *b*, *ab*, *bab*, *abbab*, *babbabab*, *abbabbabbabbab*.

Дима занимается в кружке олимпиадного программирования и интересуется алгоритмами на строках. Недавно он узнал о строках Фибоначчи. Он быстро понял, что их длина с увеличением номера *i* растет очень быстро, поэтому задача нахождения всех символов строки F_i требует слишком большого объема памяти. Поэтому он решил ограничиться задачей нахождения некоторых символов.

Напишите программу, которая находит *k*-ый символ строки F_n.

Формат входного файла

Входной файл содержит несколько наборов входных данных. Первая строка входного файла содержит целое число T наборов входных данных ($1 \le T \le 100$).

Каждая из последующих T строк описывает один набор входных данных и содержит по два целых числа: n и k ($0 \le n \le 45$, $1 \le k \le |F_n|$, через $|F_n|$ обозначена длина строки F_n, позиции символов в строке нумеруются с единицы).

Формат выходного файла

Выведите в выходной файл T строк, каждая из которых должна содержать ровно один символ – ответ для соответствующего набора входных данных.

Примеры входных и выходных данных

fib1.in	fib1.out
4 0 1 1 1 3 2 7 7	 a b a a

РАЗБОР ЗАДАЧИ

Заметим, что длина i-ой строки Фибоначчи $|F_i|$ равна i-ому числу Фибоначчи, так как для длин строк Фибоначчи справедливо рекуррентное соотношение $|F_0| = |F_1| = 1$, $|F_i| = |F_{i-2}|+|F_{i-1}|$, $i > 1$. При данных в задаче ограничениях ($|F_n| = 1134903170$ при $n = 45$) невозможно хранить в явном виде строку Фибоначчи, поэтому решение задачи, основанное на непосредственном построении соответствующей строки Фибоначчи, не будет удовлетворять ограничению по времени работы.

Верное решение задачи основано на следующем рассуждении. Пусть требуется найти k-ый символ в n-ой строке Фибоначчи. Из условия задачи следует, что если $n = 0$, а $k = 1$, то этот символ – «a»; если $n = 1$, то этот символ – «b».

Теперь рассмотрим случай $n > 1$. Заметим, что если $k = |F_{n-2}|$, то искомый символ до конкатенации, в результате которой была получена строка F_n, содержался в строке F_{n-2}, и нам достаточно найти k-ый символ в F_{n-2}. В противном случае этот символ до конкатенации содержался в строке F_{n-1}, и нам требуется найти символ с номером $(k–|F_{n-2}|)$ в строке F_{n-1}. Таким образом, мы всякий раз сводим задачу к задаче меньшей размерности, уменьшая значение n не менее чем на единицу. Из этого следует, что время обработки каждого набора входных данных составляет $O(n)$.

Приведем программную реализацию описанного алгоритма на языке программирования *Pascal* (см. листинг 1).

Листинг 1. Реализация алгоритма

```
const
  MAXN = 45;
var
  t, n, j, k : longint;
 len : array [0..MAXN] of longint; //len[i] - длина i-ой строки Фибоначчи
begin
  assign(input, 'fib1.in');
  reset(input);
  assign(output, 'fib1.out');
  rewrite(output);
  // Заполняем массив len
  len[0] := 1;
  len[1] := 1;
  for j := 2 to MAXN do begin
    len[j] := len[j - 1] + len[j - 2];
  end;
  read(t);
  for j := 1 to t do begin
    read(n, k);
    while (n > 1) do begin
      // Сведение к задаче меньшей размерности
      if (k <= len[n - 2]) then
        n := n - 2
      else begin
        k := k - len[n - 2];
        n := n - 1;
      end;
    end;
    // Вывод ответа
    if (n = 0) then
      writeln('a')
    else
      writeln('b');
  end;
end.
```

Ульянцев Владимир Игоревич,
студент третьего курса кафедры «Компьютерные технологии» СПбГУ ИТМО, член жюри Интернет-олимпиад по информатике базового уровня,

Царёв Фёдор Николаевич,
аспирант кафедры «Компьютерные технологии» СПбГУ ИТМО, чемпион мира по программированию среди студентов 2008 года, член жюри Интернет-олимпиад по информатике базового уровня.