Школа современного программирования

ГОТОВИМСЯ К ОЛИМПИАДАМ ПО ИНФОРМАТИКЕ

*Ахи Антон Андреевич,*
*Царёв Фёдор Николаевич*

# ЗАДАЧА «НАГРАДЫ»

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.

Рассматриваемая в этой статье задача предлагалась в шестой Интернет-олимпиаде сезона 2009–2010 (олимпиада состоялась 8 мая 2010 года), проводимой Санкт-Петербургским государственным университетом информационных технологий, механики и оптики. Задача предлагалась на олимпиаде в двух вариантах – более простой вариант (с меньшими ограничениями на входные данные) предлагался в базовой номинации, рассчитаной на начинающих участников олимпиад, а вариант с бóльшими ограничениями – в усложненной номинации, где предлагаются задачи уровня городских и всероссийских командных олимпиад по программированию. Сайт этих олимпиад находится по адресу http://neerc.ifmo.ru/school/io.

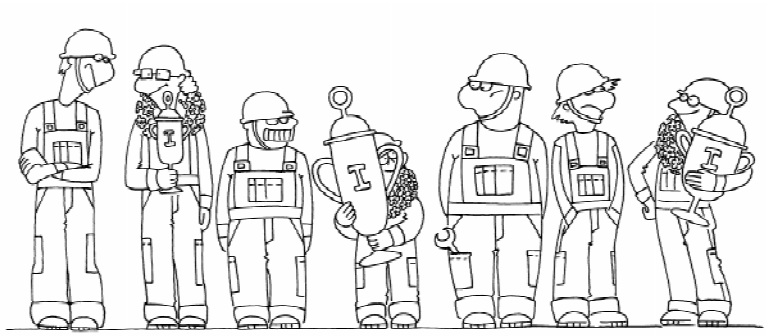
## УСЛОВИЕ ЗАДАЧИ

На одном Очень Важном Предприятии решили наградить некоторых $k$ его работников. Конечно же, решили сделать это в соответствии со следующей Очень Важной Процедурой.

Всех $n$ работников выстроили в один ряд. Причем, получилось так, что каждый работник видит только своих непосредственных соседей в этом ряду. Для повышения уровня производства на Очень Важном Предприятии начальство решило сделать так, чтобы каждый награжденный считал, что наградили именно его и только его. Для этого необходимо, чтобы в ряду не было двух рядом стоящих награжденных работников.

Вам необходимо написать программу для вычисления числа способов раздать таким образом $k$ наград среди $n$ стоящих в ряд работников.

*Для усложненной номинации:* так как это число может быть весьма большим, необходимо найти его остаток от деления на простое число $m$.



Формат входного файла

В первой и единственной строке входного файла заданы два целых числа $n$ и $k$ – количество работников на Очень Важном Предприятии и количество наград ($1 \le k \le n \le 20$).

*Для усложненной номинации:* В первой и единственной строке входного файла заданы три целых неотрицательных числа $n$, $k$ и $m$ – количество работников на Очень Важном Предприятии, количество наград и простой модуль ($1 \le k \le n \le 100000$, $1 \le m \le 10^9$).

Формат выходного файла

В выходной файл выведите единственное целое число – ответ на задачу.

*Для усложненного уровня:* В выходной файл выведите единственное целое число – ответ на задачу, взятый по модулю простого числа $m$.

*Примеры входных и выходных данных*

| grants.in | grants.out |
|---|---|
| 3 2 | 1 |
| 5 2 | 6 |

*Для усложненного уровня:*

| grants.in | grants.out |
|---|---|
| 3 2 569 | 1 |
| 5 2 673 | 6 |

## РАЗБОР ЗАДАЧИ

У данной задачи существует несколько различных решений с разной асимптотикой. Рассмотрим первое и самое простое из них с асимптотикой $O(2^n n)$. Заметим, что каждому набору награждаемых работников, пусть даже некорректному, можно поставить в соответствие битовую строку длины n, где единица на позиции $i$ означает, что $i$-ый работник награжден, а ноль на этой позиции – то, что работник не награжден.

Алгоритм решения заключается в том, чтобы перебрать все битовые строки длины $n$ (их число равно $2^n$), и каждую из них проверить, подходит ли она под ограничения, описанные в условии, – проверить, что в маске ровно $k$ единиц и что в ней нет двух единиц подряд.

Для осуществления такой проверки необходим способ определения того, стоит на $i$-ой позиции в числе `mask` единица или ноль. Составим число, в котором на $i$-ой позиции стоит единица – это число можно получить с помощью операции сдвига влево – оно равно `1 shl i`. После этого вычислим побитовое «И» с числом mask. Если результат не равен нулю, то в числе mask на $i$-ой позиции стоит единица, а иначе – ноль.

Программная реализация этого алгоритма приведена в листинге 1.

Более быстрое решение основано на динамическом программировании [1] и имеет асимптотику времени работы $O(nk)$. Обозначим как $a_{ij}$ число способов из первых $i$ работников наградить $j$, чтобы не было двух награжденных работников рядом. При этом будем считать, что последнего $i$-ого работника не наградили. Ответом для задачи будет значение $a_{n+1,k}$ (добавляем еще одного работника, который никогда не будет награжден).

Выпишем рекуррентные соотношения для динамики. Для подсчета значения $a_{ij}$ рассмотрим два случая (в обоих будем считать, что $j > 0$):

- $i = 2$ и $(i-1)$-ый работник был награжден: в этом случае в $a_{ij}$ должны входить все способы наградить $(j-1)$-го работника из первых $(i-2)$, при этом $(i-2)$-ой не должен быть награжден. Этот случай соответствует $a_{i-2,j-1}$.
- $i < 2$ или $(i-1)$-ый работник не был награжден: в этом случае все $j$ награжденных рабочих должный быть из первых $(i-1)$-ого, при этом $(i-1)$-ый не должен быть награжден. Этот случай соответствует $a_{i-1,j}$.

Таким образом получаем рекуррентное соотношение:

$$a_{ij} = \begin{cases} 1, & j = 0 \\ a_{i-1,j}, & i < 2 \\ a_{i-1,j} + a_{i-2,j-1}, & i \ge 2 \end{cases}$$

Программная реализация описанного алгоритма приведена в листинге 2.

Оба описанных выше алгоритма не могут решить задачу для ограничений,

**Листинг 1.** Реализация алгоритма с временем работы $O(2^n)$

```
var
  good : boolean;
  mask, i, k, n, c, ans : longint;

begin
  reset(input, 'grants.in');
  rewrite(output, 'grants.out');
  read(n, k);
  ans := 0;

  //перебираем все битовые строки длины n («маски»)
  for mask := 0 to (1 shl n) - 1 do begin
    c := 0;    //число единиц в «маске»
    good := true;
    for i := 0 to n - 1 do begin
      //проверяем что в «маске» нет двух единиц подряд
      if (((1 shl i) and mask) <> 0) and
         (((1 shl (i + 1)) and mask) <> 0) then begin
        good := false;
        break;
      end;
      //вычисляем число единиц
      if (((1 shl i) and mask) <> 0) then begin
        inc(c);
      end;
    end;
    //если «маска» «хорошая» и число единиц - k, увеличиваем
    //ответ
    if (good) and (c = k) then begin
      inc(ans);
    end;
  end;
  writeln(ans);
  close(input);
  close(output);
end.
```

предлагаемых в олимпиаде усложненной номинации, так как работают слишком медленно. Опишем алгоритм со временем работы $O(n \log m)$, который работает достаточно быстро для решения задачи усложненного уровня.

Если проанализировать решение, основанное на динамическом программировании, то можно понять, что оно вычисляет число способов выбрать из $(n+1)$-ого элемента $k$ непересекающихся пар соседних элементов. Если бы надо было выбирать не пары, а отдельные элементы, то ответ выражался бы формулой $C_{n+1}^{k}$.

Подойдем к задаче с другой стороны. Если сначала из $A$ элементов выбрать $B$, а затем к каждому из выбранных элементов добавить по одному элементу в пару, то получится, что мы как будто бы из $(A+B)$ элементов выбрали $B$ пар соседних элементов. Проводя это рассуждение в обратную сторону и применяя его к нашей задаче ($A+B=n+1$; $B=k$), получаем, что ответ равен $C_{n+1-k}^{k}$. Значение можно посчитать по известной формуле

$$C_{n+1-k}^{k} = \frac{(n+1-k)!}{k!(n+1-k-k)!} = \frac{(n+1-k)!}{k!(n+1-2k)!} =$$

$$= \frac{(n+1-k)\cdot(n-k)\cdot\ldots\cdot(n+2-2k)}{k!}.$$

В процессе вычисления этой величины могут возникать очень большие числа, но

**Листинг 2.** Реализация алгоритма с временем работы $O(nk)$

```
var
  i, j, n, k: longint;
  a : array [0..5000, 0..5000] of longint;

begin
  reset(input, 'grants.in');
  rewrite(output, 'grants.out');
  read(n, k);
  fillchar(a, sizeof(a), 0);
  for i := 0 to n do
    a[i, 0] := 1;
  for i := 1 to n + 1 do begin
    for j := 1 to i do begin
      a[i, j] := a[i - 1, j];
      if (i >= 2) then
        a[i, j] := a[i, j] + a[i - 2, j - 1];
    end;
  end;
  writeln(a[n + 1, k]);
  close(input);
  close(output);
end.
```

в задаче необходимо узнать ответ по модулю простого числа $m$, а значит нужно производить вычисления сразу по модулю $m$.

Для того чтобы делить, необходимо находить обратное по модулю $m$, что делается с помощью расширенного алгоритма Евклида [1] за $O(\log m)$. Однако и в этом случае остаются проблемы, потому что в числителе и знаменателе этой дроби могут быть числа, делящиеся на $m$. Чтобы избавиться от этой проблемы, необходимо сокращать все числа на $m$ и подсчитывать, сколько раз числитель и знаменатель делятся на $m$. Если числитель делится на $m$ в большей степени, нежели знаменатель, то ответ равен 0.

Также в начале решения необходимо проверять, имеет ли задача смысл, то есть существует ли хотя бы один способ наградить рабочих. Такой способ существует, если выполняется неравенство $2k = n + 1$.

Программная реализация описанного алгоритма приведена в листинге 3.

**Листинг 3.** Реализация алгоритма с временем работы $O(n \log m)$

```
var
  n, k, m, j, ans : int64;
  i : longint;
  d, mul : longint;

//расширенный алгоритм Евклида
procedure gcd(a, b : int64; var x, y : int64);
var
  tmp : int64;
begin
  if (b = 0) then begin
    x := 1;
    y := 0;
    exit;
  end;
```

```
  gcd(b, a mod b, y, x);
  y := y - (a div b) * x;
end;

function inv(a, m : int64) : int64; //получение обратного
var
  x, y : int64;
begin
  gcd(a, m, x, y);
  inv := ((x mod m) + m) mod m;
end;

begin
  reset(input, 'grants.in');
  rewrite(output, 'grants.out');
  read(n, k, m);
  //проверяем, что задача имеет смысл
  if (2 * k > n + 1) then begin
    writeln(0);
    halt(0);
  end;
  d := 0;
  mul := 0;
  ans := 1;
  for i := 1 to k do begin
    j := n + 2 - k - i; //очередное число из числителя
    while (j mod m = 0) do begin //сокращаем на m
      j := j div m;
      inc(mul);
    end;
    ans := (ans * j) mod m; //умножаем по модулю
    j := i;
    while (j mod m = 0) do begin //сокращаем на m
      j := j div m;
      inc(d);
    end;
    //умножаем на обратное по модулю
    ans := (ans * inv(j, m)) mod m;
  end;
  if (mul > d) then begin
    writeln(0)
  end else begin
    writeln(ans);
  end;
  close(input);
  close(output);
end.
```

**Литература**

1. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы. Построение и анализ. М.: Вильямс, 2007.



***Ахи Антон Андреевич,***
***студент третьего курса кафедры «Компьютерные технологии» СПбГУ ИТМО, финалист чемпионата мира по программированию среди студентов 2010 года,***
***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.***