ВАРИАНТ 37101 - Решение

1. В теории чисел натуральное число называется B-гладким, если все его простые делители не превосходят B. Разработайте алгоритм проверки чисел в диапазоне от P до Q на B-гладкость.

Решение. Построим массив простых чисел с помощью решета Эратосфена. Далее для каждого числа *n* из диапазона от P до Q проверяем, являются ли простые числа, большие B, но меньшие или равные *n* / 2, делителями числа *n*. Если это так, то число не является B-гладким. В противном случае число является B-гладким.

Для использования решета Эратосфена необходимо построить массив чисел в заданном диапазоне от 1 до Q / 2. Поскольку число 1 не является простым, в элемент массива с индексом 1 занесём значение 0 – будет удобнее, если индекс массива и число в массиве совпадают. Затем для чисел *i* в диапазоне от 2 до $\sqrt{Q/2}$, начиная с числа *i*, вычёркиваем из массива (заменяем нулями) все числа с шагом *i* (само число *i* не вычёркивается). Для нахождения следующего значения *i* нужно найти первый незачёркнутый (ненулевой) элемент массива после текущего значения *i*.

```
алг B-гладкость()
нач
  цел p, q, b, primes[q / 2], n, i, j
  лог is_smooth

  ввод p, q, b

  если p < 1 или q < 1 или b < 1 или p > q то
    вывод "Некорректные исходные данные"
  иначе

    // Построение массива простых чисел
    primes[1] = 0
    для i от 2 до q / 2
    нц
      primes[i] = i
    кц

    i = 2
    пока i <= целая_часть(sqrt(q / 2))
    нц
      для j от 2 * i до q / 2 шаг i
      нц
        primes[j] = 0
      кц
      выполнить
        i = i + 1
      до primes[i] <> 0
    кц

   // Проверка чисел от p до q на b-гладкость
   для n от p до q
   нц
     is_smooth = истина
     i = b + 1
     пока i <= n / 2 и is_smooth
     нц
       если primes[i] <> 0 и n mod i = 0 то
         is_smooth = ложь
       всё
       i = i + 1
     кц
     если is_smooth то
       вывод n, " является ", b, "-гладким числом"
     иначе
       вывод n, " не является ", b, "-гладким числом"
     всё
   кц

  всё
кон
```

2. Марина и Светлана разговаривают по телефону и хотят выбрать секретное число так, чтобы оно осталось неизвестным постороннему, возможно подслушивающему их разговор. Для этого Марина подбирает натуральное число $a \le 256$ такое, что числа $R_{257}(a^i)$ различны при всех $1 \le i \le 256$ и $R_{257}(a^{256}) = 1$, где $R_{257}(t)$ – остаток от деления числа t на 257. Затем Марина загадывает натуральное число $x \le 256$, а Светлана – натуральное число $y \le 256$. После этого Марина сообщает числа a и $R_{257}(a^x)$ Светлане, а Светлана ей – число $R_{257}(a^y)$. Теперь они обе вычисляют их секретное число $R_{257}(a^{xy})$. Составьте алгоритм для нахождения этого секретного числа, если известно, что $R_{257}(a^x) = 9$, $R_{257}(a^y) = 256$.

Решение.

1. Вводим значение a и проверяем условия $R_{257}(a^1) \ne R_{257}(a^2) \ne \ldots \ne R_{257}(a^{256})$ и $R_{257}(a^{256}) = 1$. Если условия не выполняются, то вводим новое значение a.

2. Генерируем случайное значение x $(1 < x \le 256)$, для которого $R_{257}(a^x) = 9$, и случайное значение y $(1 < y \le 256)$, для которого $R_{257}(a^y) = 256$.

3. Вычисляем $k_1 = R_{257}(256^x)$ и $k_2 = R_{257}(9^y)$.

4. Проверяем $k_1 = k_2$ и выводим $k = k_1$.

3. По квадратной матрице A размера n построить матрицу B того же размера, где b_{ij} определяется следующим образом. Через a_{ij} проведём в A линии, параллельные сторонам прямоугольника до пересечения с главной диагональю (главная диагональ квадратной матрицы – диагональ, которая проходит через верхний левый и нижний правый углы); b_{ij} определяется как минимум среди элементов треугольника в A. На рис. 1 треугольник, заштрихованный косыми линиями, соответствует случаю, когда a_{ij} находится выше главной диагонали, а треугольник, заштрихованный вертикальными линиями, соответствует случаю, когда a_{ij} находится ниже главной диагонали.

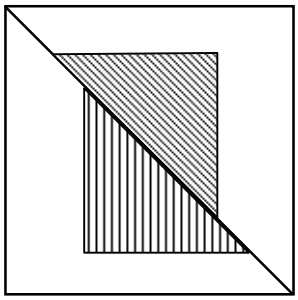

Рис. 1

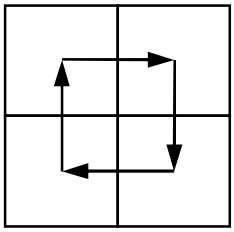

Рис. 2

Решение. Рассмотрим отдельно случаи, когда элемент a_{ij} находится выше главной диагонали, на главной диагонали и ниже главной диагонали. Для случая выше главной диагонали b_{ij} определяется как минимум среди элементов a_{km}, $k = i, \ldots, j$, $m = k, \ldots, j$. Случай, когда a_{ij} находится на главной диагонали, является вырожденным, треугольник состоит из одного элемента a_{ij}, который в данном случае и является минимумом. Для случая ниже главной диагонали b_{ij} определяется как минимум среди элементов a_{km}, $k = j, \ldots, i$, $m = j, \ldots, k$.

```
алг ФормированиеМатрицы()
нач
  цел n, i, j, k, m
  вещ A[n, n], B[n, n]

  ввод n
  если n <= 0 то
    вывод "Некорректные исходные данные"
  иначе
    для i от 1 до n
    нц
      для j от 1 до n
      нц
        ввод A[i, j]
      кц
```

```
    кц

    // Выше главной диагонали
    для i от 1 до n – 1
    нц
      для j от i + 1 до n
      нц
        B[i, j] = A[i, i]
        для k от i до j
        нц
          для m от k до j
          нц
            если A[k, m] < B[i, j] то
              B[i, j] = A[k, m]
            всё
          кц
        кц
      кц
    кц

    // На главной диагонали
    для i от 1 до n
    нц
      B[i, i] = A[i, j]
    кц

    // Ниже главной диагонали
    для i от 2 до n
    нц
      для j от 1 до i – 1
      нц
        B[i, j] = A[j, j]
        для k от j до i
        нц
          для m от j до k
          нц
            если A[k, m] < B[i, j] то
              B[i, j] = A[k, m]
            всё
          кц
        кц
      кц
    кц

    для i от 1 до n
    нц
      для j от 1 до n
      нц
        вывод B[i, j]
      кц
    кц
  всё
кон
```

4. На листе бумаги нарисована квадратная таблица размера $2n$. В клетках написаны различные целые числа. Необходимо получить новую таблицу, переставляя блоки размера $n \times n$ в соответствии с рис. 2.

Решение. Перебираем элементы $a_{i,j}$, $i = 1, \ldots, n$, $j = 1, \ldots, n$ и переставляем сразу 4 элемента по схеме $c \leftarrow a_{i,j} \leftarrow a_{i+n,j} \leftarrow a_{i+n,j+n} \leftarrow a_{i,j+n} \leftarrow c$.

```
алг Обмен()
нач
  цел n, i, j, c
  вещ A[2 * n, 2 * n]

  ввод n
  если n <= 0 то
    вывод "Некорректные исходные данные"
  иначе
    для i от 1 до 2 * n
    нц
      для j от 1 до 2 * n
      нц
        ввод A[i, j]
      кц
    кц

    для i от 1 до n
```

```
    нц
      для j от 1 до n
      нц
        c = A[i, j]
        A[i, j] = A[i + n, j]
        A[i + n, j] =  A[i + n, j + n]
        A[i + n, j + n] = A[i, j + n]
        A[i, j + n] = c
      кц
    кц

    для i от 1 до 2 * n
    нц
      для j от 1 до 2 * n
      нц
        вывод A[i, j]
      кц
    кц
  всё
кон
```

5. В теории чисел задача Знама спрашивает, какие множества k целых чисел имеют свойство, что каждое целое в множестве является собственным делителем произведения других целых чисел в множестве плюс 1. То есть, если дано число k, какие существуют множества целых чисел $\{n_1, ..., n_k\}$ таких, что для любого i число n_i делит, но не равно $\left(\prod_{j\neq i}^{k} n_j + 1\right)$.

Разработайте алгоритм нахождения числа решений задачи Знама для k в диапазоне от P до Q. Принять верхнюю границу $n_i = 10^{11}$.

Решение. Задача решается перебором всех вариантов. Но надо сформировать все наборы значений из k чисел. Поскольку k заранее неизвестно, невозможно написать несколько циклов. Будем использовать следующий алгоритм. Сначала формируем массив из k элементов, каждый из которых равен минимальному возможному значению (в данном случае это 2). Далее увеличиваем последний элемент массива на 1, пока он не станет равным максимальному значению. После этого последнему элементу снова присваиваем минимальное значение, а предыдущий элемент увеличиваем на 1. Если предыдущий элемент (или несколько предыдущих) равен (равны) максимальному значению, присваиваем ему (им) минимальное значение, а на 1 увеличиваем первый с конца элемент, который не равен максимальному значению. Когда все элементы массива будут равны максимальному значению, заканчиваем перебор.

В наборе чисел, удовлетворяющем условию задачи Знама, может быть только одно чётное число, т.к. иначе значение выражения $\left(\prod_{j\neq i}^{k} n_j + 1\right)$ будет нечётным, и если n_i – чётное число, то оно точно не будет делителем этого выражения. Поэтому каждый сформированный набор можно проверить на наличие чётных чисел и все чётные элементы, кроме первого, увеличить на 1 и установить шаг изменения последнего элемента равным 2. Если первые (k – 1) элементы – нечётные, оставляем шаг изменения последнего элемента равным 1.

Также в наборе не может быть одинаковых чисел, т.к. в этом случае выражение $\left(\prod_{j\neq i}^{k} n_j + 1\right)$ также не будет делиться на один из сомножителей произведения.

Следовательно, первая подходящая комбинация состоит из чисел 2, 3, 5, 7 и т.д. Кроме того, при переходе к следующему возможному набору надо учитывать, что каждое следующее число должно быть больше предыдущего, т.к. это позволит нам не рассматривать наборы, состоящие из одинаковых чисел, расположенных в разном порядке.

```
алг ЗадачаЗнама()
нач
  цел p, q, k, r, step, i, j
  цел n[q]
```

```
лог all, found
цел константа limit = 1e11

ввод p, q

для k от p до q
нц
  r = 0

  // Заполняем массив исходными значениями. Первый элемент массива положим равным 2, а остальные – 3, 5 и т.д.
  n[1] = 2
  для i от 2 до k
  нц
    n[i] = 2 * i – 1
  кц

  step = 2
  all = ложь
  пока не all
  нц
    если УдовлетворяетУсловию(n, k) то
      r = r + 1
    всё

    // Переход к следующему набору
    если n[k] + step <= limit то          // если можно изменить последний элемент набора
      n[k] = n[k] + step                  // меняем его
    иначе                                 // ищем первый с конца элемент, который можно изменить
      found = ложь
      i = k – 1
      пока i >= 1 и не found
      нц
        если n[i] + 1 <= limit – ((k – i) * 2 – 1) то
         found = истина
        иначе
          i = i – 1
        всё
      кц
      если не found то                    // если нет элемента, который можно изменить
        all = истина                      // завершаем цикл по перебору комбинаций
      иначе
        n[i] = n[i] + 1
        found = ложь                      // проверяем наличие чётного элемента до i-го элемента
        j = i – 1
        пока j >= 1 и не found
        нц
          если n[j] mod 2 = 0 то
            found = истина
          всё
          j = j – 1
        кц
        если found то                     // если до i-го элемента есть чётный элемент
          если n[i] mod 2 = 0 то          // если i-ый элемент тоже чётный
            n[i] = n[i] + 1               // делаем его нечётным
          всё
          для j от i + 1 до k             // остальным элементам присваиваем следующие нечётные значения
          нц
            n[j] = n[j – 1] + 2
          кц
          step = 2                        // шаг = 2, т.к. есть чётный элемент, остальные должны быть нечётным
        иначе                             // если до i-го элемента нет чётного элемента
          если n[i] mod 2 = 0 то          // если i-ый элемент является чётным
            n[i + 1] = n[i] + 1           // (i + 1)-ому элементу присваиваем следующее (нечётное) значение
            для j от i + 2 до k           // остальным элементам присваиваем следующие нечётные значения
            нц
              n[j] = n[j – 1] + 2
            кц
            step = 2                      // шаг = 2, т.к. есть чётный элемент, остальные должны быть нечётным
          иначе                           // если i-ый элемент является нечётным
            n[i + 1] = n[i] + 1           // присваиваем следующему элементу следующее (чётное) значение
            если i + 2 <= k то            // ещё следующему (если он есть) присваиваем нечётное значение
              n[i + 2] = n[i + 1] + 1
            всё
            для j от i + 3 до k           // остальным элементам  присваиваем следующие нечётные значения
            нц
              n[j] = n[j – 1] + 2
            кц
            если n[k] mod 2 = 0 то        // если чётным оказался последний элемент, значит, все предыдущие
              step = 1                    // являются нечётными, и шаг = 1
            иначе
              step = 2
            всё
          всё
```

```
          всё
        всё
      всё
    кц

    вывод "Количество решений для набора из ", k, " чисел равно ", r
  кц
кон

алг УдовлетворяетУсловию(арг цел n[k], арг цел k)
нач
  цел i, j, p
  лог f

  f = истина
  i = 1
  пока i <= k и f
  нц
    p = 1
    для j от 1 до i - 1
    нц
      p = p * n[j]
    кц
    для j от i + 1 до k
    нц
      p = p * n[j]
    кц
    если (p + 1) = n[i] или (p + 1) mod n[i] <> 0 то
      f = ложь
    всё
    i = i + 1
  кц
  вернуть f
кон
```