ЕГЭ по информатике 2022: задание 25

Настоящая статья содержит разбор очередного задания, представленного в демонстрационном варианте Единого государственного экзамена по информатике и ИКТ 2022 года [1], и является третьей в цикле статей, описывающих методы решения заданий ЕГЭ 2022, предусматривающих самостоятельное написание программы на одной из доступных на экзамене систем программирования

В первых двух статьях [2][3] мы подробно рассмотрели решения заданий 16, 17 и 24, представленных в проекте демонстрационного варианта Единого государственного экзамена по информатике и ИКТ 2022 года [1]. Однако самостоятельное написание программы для решения задачи не ограничивается только этими тремя заданиями. В настоящей, третьей по счету статье мы продолжим рассмотрение такого типа заданий и подробно разберём следующее такое задание под номером 25.

Ранее, до 2020 года, то есть до перевода ЕГЭ по информатике в компьютерную форму, в демонстрационном варианте Единого государственного экзамена по информатике и ИКТ [4] содержался подробный разбор задания 25, причём на нескольких языках программирования. Перевод ЕГЭ по информатике в компьютерную форму повлёк не только изменение самих заданий (в сторону усложнения), но и внёс изменения в сам демонстрационный вариант: сейчас в таком варианте содержатся только краткие ответы для самопроверки. Настоящей статьёй мы восполним этот недостаток.

Задание 25

Условие

Пусть M – сумма минимального и максимального натуральных делителей целого числа, не считая единицы и самого числа. Если таких делителей у числа нет, то значение M считается равным нулю.

Напишите программу, которая перебирает целые числа, бóльшие 700 000, в порядке возрастания и ищет среди них такие, для которых значение M оканчивается на 8. Выведите первые пять найденных чисел и соответствующие им значения M.

Формат вывода: для каждого из пяти таких найденных чисел в отдельной строке сначала выводится само число, затем – значение M.

Строки выводятся в порядке возрастания найденных чисел.

Решение

Новшеством 2022 года является изменение «стоимости» этого задания. Теперь вместо двух «первичных»

баллов за верное решение этого задания будут давать всего один [1] [5]. В остальном это то же самое задание на делимость, которое было представлено впервые на ЕГЭ в 2021 году [6].

Предупреждение: довольно частой ошибкой в этой задаче является невнимательное прочтение словосочетания «бóльшие 700000», т.е. мы должны начинать проверку с числа 700001, а не 700000. Обязательно обращаем на это внимание.

Эту задачу можно решать простым переборным «неэффективным» способом, а можно решать быстрым «эффективным». В настоящей статье мы рассмотрим их оба.

Первый способ (переборный)

Данный способ предполагает последовательный перебор для каждого проверяемого числа (n) всех натуральных чисел (i), кроме единицы, меньших этого числа (n) и поиск среди них всех возможных делителей. Этот способ работает очень медленно, но он доступен любому, даже начинающему программисту. При незнании других, более эффективных способов, можно попробовать решить задачу и этим алгоритмом в надежде, что его скорости работы окажется достаточно для решения предложенной на экзамене задачи. Справедливости ради отметим, что при исходных данных, указанных в задании 25 «демоверсии» [1], скорости работы данного алгоритма оказалось достаточно.

Далее, из всех найденных делителей нам нужно выбрать минимальный (d1) и максимальный (d2), вычислить их сумму (m), проверить её последнюю цифру и перейти к проверке следующего натурального числа по возрастанию (n).

Мы повторяем эти действия до тех пор, пока не отыщем пять подходящих под условие чисел, параллельно выводя их на экран вместе с вычисленными для них суммами (m).

Количество найденных подходящих чисел (k) мы также храним и используем его в качестве условия для продолжения/окончания цикла.

Рассмотрим вопрос поиска самих делителей подробнее. При переборе всех возможных делителей (i) по возрастанию наименьший делитель является первым найденным, поэтому в соответствующую переменную (d1) мы сохраняем делитель только самый первый, а наибольший – последним, поэтому соответствующую ему переменную (d2) перезаписываем каждым новым найденным делителем.

Отдельно не забываем рассмотреть особые случаи. Так как при отсутствии делителей значение M считается равным нулю, то начальное значение каждой из двух переменных (d1 и d2) для дальнейшего нашего удобства сразу установим равным нулю для каждого нового проверяемого числа (n)

Полученный результат отражён в таблице 1.

700005	233338
700007	100008
700012	350008
700015	140008
700031	24168

Таб. 1

Ниже приведена реализующая описанный алгоритм программа на четырех языках программирования (рис. 1-4).

C++ (№ 25)

```
#include <iostream>
using namespace std;

int main()
{
  long n, m, k, d1, d2, i;
  n = 700001;
  k = 0;
  while (k < 5)
  {
    d1 = 0;
    d2 = 0;
    for (i = 2; i<= n - 1; i++)
      if (n % i == 0)
      {
        if (d1 == 0)
          d1 = i;
        d2 = i;
      }
    m = d1 + d2;
    if (m % 10 == 8)
    {
      cout << n << ' ' << m << endl;
      k++;
    }
    n++;
  }
  return 0;
}
```

Рис. 1

Алгоритмический язык (№ 25)

```
алг
нач
цел n, m, k, d1, d2, i
  n := 700001
  k := 0
  нц пока k < 5
    d1 := 0
    d2 := 0
    нц для i от 2 до n - 1
      если mod(n, i) = 0 то
        если d1 = 0 то
          d1 := i
        все
        d2 := i
      все
    кц
    m := d1 + d2
    если mod(m, 10) = 8 то
      вывод n, " ", m, нс
      k := k + 1
    все
    n := n + 1
  кц
кон
```

Рис. 2

Паскаль (№ 25)

```
var
  n, m, k, d1, d2, i: longint;
begin
  n := 700001;
  k := 0;
  while k < 5 do
  begin
    d1 := 0;
    d2 := 0;
    for i := 2 to n - 1 do
      if n mod i = 0 then
      begin
        if d1 = 0 then
          d1 := i;
        d2 := i
      end;
    m := d1 + d2;
    if m mod 10 = 8 then
    begin
      writeln(n, ' ', m);
      k := k + 1
    end;
    n := n + 1
  end
end.
```

Рис. 3

Python (№ 25)

```
n = 700001
k = 0
while k < 5:
    d1 = 0
    d2 = 0
    for i in range(2, n):
        if n % i == 0:
            if d1 == 0:
                d1 = i
            d2 = i
        m = d1 + d2
    if m % 10 == 8:
        print(n, m)
        k += 1
    n += 1
```

Рис. 4

Второй способ (быстрый)

Решение задания 25 предыдущим способом может оказаться слишком медленным при более сложных исходных данных, поэтому грамотный выпускник школы должен знать и быстрый «эффективный» способ решения этой задачи.

Для этого вспомним, что натуральное число a является делителем натурального числа n, если существует такое натуральное число b, что $n=a*b$. Следовательно, если у натурального числа n существует хотя бы один делитель a, то у n также существует и делитель $b=n/a$. Таким образом, за исключением случаев, когда n является полным квадратом (т.е. a и b совпадут или $a=b$), делители появляются парами и, зная один делитель a, мы можем узнать и делитель b, разделив нацело n на a.

Возьмем для ясности $a<=b$. Тогда a не превосходит $\sqrt{n}$.

Действительно, если предположить противное, т.е. предположить, что $a > \sqrt{n}$, тогда $b \geq a > \sqrt{n}$. Отсюда $a * b > \sqrt{n} * \sqrt{n} = n$. Получили противоречие.

Следовательно, нам незачем проверять весь диапазон в поисках делителей. Если наименьший делитель, отличный от единицы у числа и существует, то он обязательно встретится в диапазоне $[2; \sqrt{n}]$, что существенно сокращает область поиска.

Наибольший делитель, так как его можно представить в виде n/a, где a – тоже является делителем, будет тем больше, чем меньшее значение a при неизменном n мы подставим. Следовательно, парой к наибольшему делителю как раз и будет являться наименьший. Получаем формулу: (наибольший делитель) = $= n /$ (наименьший делитель).

Даже если число является полным квадратом ($n=c^2$) и имеет единственный делитель (c), отличный от единицы и самого числа, то этот делитель (c) будет являться одновременно и наименьшим, и наибольшим. А так как число является полным квадратом, то $n/c=c$, то есть формула, полученная в предыдущем абзаце, верна и в этом случае.

Возвращаясь к нашей задаче, получается, что всё, что нам нужно знать и найти, – это первый (минимальный) делитель, который находится в диапазоне $[2; \sqrt{n}]$. А если в этом диапазоне делителей не нашлось, то M = 0 по условию задачи. Если же один делитель при проверке чисел по возрастанию нашёлся, то дальше (до $\sqrt{n}$) проверять нет смысла, т.к. минимальный делитель мы уже нашли, а максимальный мы просто вычислим.

Таким образом, мы осуществляем перебор чисел-потенциальных делителей (i) искомого числа (n) по возрастанию до тех пор, пока не встретим первый делитель или не выйдем за вышеуказанный диапазон (от 2 до $q = \sqrt{n}$).

Ответ у нас получился тот же, как и полученный ранее другим, более медленным способом (см. таб. 1).

Изложенный подход приводит к существенному росту скорости работы программы, что особенно заметно на больших числах.

Ниже приведена реализующая описанный алгоритм программа на четырех языках программирования (рис. 5-8).

C++ (№ 25 «эффективная»)

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
  long n, m, k, i, q;
  n = 700001;
  k = 0;
  while (k < 5)
  {
    q = sqrt(n);
    i = 2;
    while ((i <= q) && (n % i != 0))
      i++;
    if (i <= q)
      m = i + n / i;
    else
      m = 0;
    if (m % 10 == 8)
    {
      cout << n << ' ' << m << endl;
      k++;
    }
    n++;
  }
  return 0;
}
```

Рис. 5

Алгоритмический язык (№ 25 «эффективная»)

```
алг
нач
  цел n, m, k, i, q
  n := 700001
  k := 0
  нц пока k < 5
    q := int(sqrt(n))
    i := 2
    нц пока i <= q и mod(n, i) <> 0
      i := i + 1
    кц
    если i <= q то
      m := i + div(n, i)
    иначе
      m := 0
    все
    если mod(m, 10) = 8 то
      вывод n, " ", m, нс
      k := k + 1
    все
    n := n + 1
  кц
кон
```

Рис. 6

Паскаль (№ 25 «эффективная»)

```
var
  n, m, k, i, q: longint;
begin
  n := 700001;
  k := 0;
  while k < 5 do
  begin
    q := trunc(sqrt(n));
    i := 2;
    while (i <= q) and (n mod i <> 0) do
      i := i + 1;
    if i <= q then
      m := i + n div i
    else
      m := 0;
    if m mod 10 = 8 then
    begin
      writeln(n, ' ', m);
      k := k + 1
    end;
    n := n + 1
  end
end.
```

Рис. 7

Python (№ 25 «эффективная»)

```
import math
n = 700001;
k = 0;
while k < 5:
    q = math.sqrt(n)
    i = 2
    while i <= q and n % i != 0:
        i += 1
    if i <= q:
        m = i + n // i;
    else:
        m = 0
    if m % 10 == 8:
        print(n, m)
        k += 1
    n += 1
```

Рис. 8

Литература

1. https://doc.fipi.ru/ege/demoversii-specifikacii-kodifikatory/2022/inf_11_2022.zip
2. Муржин Д.В. "ЕГЭ по информатике 2022: задачи 16 и 17" // Потенциал (Математика. Физика. Информатика). – 2021. – № 8.
3. Муржин Д.В. "ЕГЭ по информатике 2022: задание 24" // Потенциал (Математика. Физика. Информатика). – 2021. – № 10.
4. https://doc.fipi.ru/ege/demoversii-specifikacii-kodifikatory/2020/inf_ege_2020.zip
5. https://doc.fipi.ru/ege/demoversii-specifikacii-kodifikatory/2022/izm_ege_2022.pdf
6. https://doc.fipi.ru/ege/demoversii-specifikacii-kodifikatory/2021/inf-ege-2021.zip

Статью к публикации подготовил
Муржин Дмитрий Викторович
Преподаватель подготовительных курсов
факультета ВМК МГУ им. М. В. Ломоносова

Новости Новости Новости Новости

Физики МГУ обучили нейросеть предсказывать полноту набора измерений в квантовой томографии

Международная группа ученых, включающая сотрудников Центра квантовых технологий физического факультета МГУ имени М.В.Ломоносова, обучила свёрточную нейронную сеть предсказывать, является ли набор измерений в квантовой томографии информационно полным или нет. Это позволяет реконструировать квантовые состояния за меньшее количество измерений по сравнению с традиционными подходами, не используя при этом априорных предположений о возможном типе состояния. Также исследователи добавили вторую нейронную сеть, чтобы определять точность реконструкции без явного проведения томографии состояния. Статья по результатам работы опубликована в New Journal of Physics.

Методы сжатого считывания (compressive sensing) квантовых состояний, не использующие априорную информацию, опираются на процедуру определения полноты набора проведённых измерений. В момент, когда набор становится полным, возможно однозначное восстановление неизвестного исследуемого состояния по результатам измерений. До недавнего времени определение полноты измерений сводилось к серии задач оптимизации, решения которых нужно было вычислять по ходу эксперимента. Замена оптимизационного алгоритма предварительно обученной нейронной сетью позволяет существенно снизить сложность вычислений (например, для состояний размерности 64 продемонстрировано ускорение на 3 порядка), что ведёт к кратному уменьшению затрачиваемого время при проведении эксперимента. Соответственно, повышается точность томографии при наличии флуктуаций исследуемого состояния во времени, и результаты меньше подвержены эффекту накопления шумов.

Представленные свёрточные нейронные сети проверялись как в эксперименте с многофотонными поляризационными состояниями, так и для пространственных состояний света большой размерности.

Источник: https://www.msu.ru/science/main_themes/