Златопольский Дмитрий Михайлович,
Кандидат технических наук, доцент кафедры информатики и прикладной математики Московского городского педагогического университета; организатор и директор Музея истории вычислительной техники.

Методика выполнения задания 27 демонстрационного варианта ЕГЭ по информатике 2020 года

В статье описываются различные варианты программы решения самой сложной задачи по программированию демонстрационного варианта ЕГЭ по информатике и ИКТ 2020 года.

Условие [1]

На вход программы поступает последовательность из n целых положительных чисел. Рассматриваются все пары элементов последовательности a_i и a_j, такие, что $i<j$ и $a_i>a_j$(первый элемент пары больше второго; i и j – порядковые номера чисел в последовательности входных данных). Среди пар, удовлетворяющих этому условию, необходимо найти и напечатать пару с максимальной суммой элементов, которая делится на $m = 120$. Если среди найденных пар максимальную сумму имеют несколько, то можно напечатать любую из них.

Описание входных и выходных данных

В первой строке входных данных задаётся количество чисел n ($2 \le n \le 12\,000$). В каждой из последующих n строк записано одно целое положительное число, не превышающее 10 000.

В качестве результата программа должна напечатать элементы искомой пары. Если таких пар несколько, можно вывести любую из них.

Гарантируется, что хотя бы одна такая пара в последовательности есть.

Пример входных данных:

```
6
60
140
61
100
300
59
```

Пример выходных данных для приведённого выше примера входных данных:

140 100

Пояснение. Из шести заданных чисел можно составить три пары, сумма элементов которых делится на m=120: 60+300, 140+100 и 61+59. Во второй и третьей из этих пар первый элемент больше второго, но во второй паре сумма больше.

Требуется написать эффективную по времени и памяти программу для решения описанной задачи.

Программа считается эффективной по времени, если при одновременном увеличении количества элементов последовательности n и параметра m в k раз время работы программы увеличивается не более чем в k раз.

Программа считается эффективной по памяти, если память, необходимая для хранения всех переменных программы, не превышает 4 килобайта и неувеличивается с ростом n.

Максимальная оценка за правильную (не содержащую синтаксических ошибок и дающую правильный ответ при любых допустимых входных данных) программу, эффективную по времени и памяти, – 4 балла.

Максимальная оценка за правильную программу, возможно, неэффективную по памяти или время выполнения которой существенно зависит отвеличины m, – 3 балла.

Максимальная оценка за правильную программу, не удовлетворяющую требованиям эффективности, – 2 балла.

Вы можете сдать одну программу или две программы решения задачи (например, одна из программ может быть менее эффективна). Если Вы сдадите две программы, то каждая из них будет оцениваться независимо от другой, итоговой станет большая из двух оценок.

Перед текстом программы обязательно кратко опишите алгоритм решения.

Укажите использованный язык программирования и его версию.

Решение

Задача может быть решена так.

Обозначим пару искомых значений:

1) *лев* – расположенное в последовательности раньше;

2) *прав* – расположенное в последовательности позже.

Запишем все n чисел последовательности в массив (имя массива – *мас*), после чего сравним все пары элементов *мас*[i] и *мас*[j], таких, что $i<j$. Если *мас*[i]>*мас*[j], сумма этих значений делится на m = 120 и больше суммы «старых» значений величин *лев* и *прав*, то значения *мас*[i] и *мас*[j] принимаются в качестве новых значений, соответственно, величин *лев* и *прав*.

Соответствующая программа на школьном алгоритмическом языке:

```
цел m
m := 120
алг
нач цел n, лев, прав, i, j, цел таб мас[1 : 1200]
```

```
ввод n
| Ввод чисел последовательности и запись их в массив
нц для i от 1 до n
 ввод мас[i]
кц
| Начальное присваивание значений искомым величинам
лев := 0
прав := 0
| Перебор и проверка всех пар значений элементов массива
нц для i от 1 до n - 1
 нц для j от i + 1 до n
  если мас[i] > мас[j] и mod(мас[i] + мас[j], m) = 0
   то| Сравниваем суммы
    если мас[i] + мас[j] > лев + прав
     то| Встретилась новая пара искомых значений
       | Запоминаем их
       лев := мас[i]
       прав := мас[j]
     все
   все
 кц
кц
| Вывод ответа
вывод лев, " ", прав
кон
```

где mod – функция, возвращающая остаток от деления своего первого аргумента на второй (в других языках программирования для определения остатка используется не функция, а специальная операция).

Такое решение является неэффективным ни по памяти, ни по времени (оценивается в 2 балла).

Опишем более эффективный вариант.

Прежде всего, сделаем важное напоминание, которое будет лежать в основе решения задачи: «Сумма двух чисел делится на m, если сумма остатков этих чисел от деления на m равна m или 0».

Как и ранее, обозначим пару искомых значений:

1) *лев* – расположенный в последовательности раньше;

2) *прав* – расположенный в последовательности позже.

Допустим, что первые 10 чисел последовательности такие:

114 0 120 255 240 76 234720 56 240 7

Пусть очередное число (его имя – *очер*) равно 120. Какие числа среди рассмотренных десяти могут, так сказать, «составить пару» этому числу с точки зрения решения задачи? Так как остаток от деления числа 120 на m равен нулю, то ответ такой: 0, 120, 240, 720 и 240 (поскольку у них соответствующий остаток

также равен нулю). Но среди них нас интересует максимальное из чисел, больших очередного *очер* (мы ищем пару чисел с максимальной суммой элементов). Таким числом является 720. Означает ли это, что встретилась новая пара искомых чисел *лев* и *прав*? Для ответа на этот вопрос следует сравнить сумму чисел 720 и 120 с суммой значений *лев* и *прав* для первых 10 чисел последовательности. Анализ показывает, что нет (было *лев* = 720 и *прав* = 240, то есть сумма была больше).

Если бы очередное число было равно, например, 360, то тогда бы значения *лев* и *прав* изменились бы на 720 и 360, соответственно.

Заметим также, что если очередное число равно, например, 960, то, возможно, оно когда-то станет значением переменной *лев*.

Во всех случаях следует знать максимальное из рассмотренных ранее чисел, кратных m (в нашем случае – 120).

Рассмотрим теперь случай, когда значение *очер* не кратно 120, например, равно 6. При этом остаток от деления на 120 равен 6, и, согласно отмеченному выше правилу кратности суммы двух чисел, следует рассмотреть предшествующие числа, остаток от деления которых на 120 равен 120 – 6 = 114. Таких чисел два – 114 и 234. Оба они больше 6; из них максимальным является 234. Появилась ли новая пара чисел *лев* и *прав*? Сравним суммы. Сумма 234 + 6 = 240 меньше суммы значений «старой» пары (720 + 240 = 960). Значит, текущие значения искомых величин *лев* и *прав* не меняются. А если бы среди первых 10 чисел было, например, число 1334 (остаток от его деления на 120 равен 114):

1334 0 120 255 240 76 234 720 56 240 7

то тогда сумма 1334 + 6 = 1440 была бы больше 960, то есть встретилась бы пара новых значений *лев* = 1334 и *прав* = 6.

Итак, можем сделать вывод, что для решения задачи для каждого очередного числа последовательности *очер*, имеющего остаток от деления на m (120), равный *остат*, надо знать:

если *остат* равен 0

то:

максимальное из уже рассмотренных чисел, остаток от деления которых на m также равен 0.

иначе: максимальное из уже рассмотренных чисел, остаток от деления которых на m равен m – *остат*

Всего нужно знать 120 значений. Для их хранения следует использовать массив из 120 элементов с индексами от 0 до 119 (от 0 до m – 1). В приведенной ниже программе решения задачи на школьном алгоритмическом языке имя этого массива – *макс_числа_с_остатком*. Это значит, что, например, в элементе массива с индексом 13 будет храниться максимальное из уже рассмотренных чисел, остаток от деления которых на m равен 13, и т.п.

Прежде чем представлять программу, заметим, что после обработки очередного числа его, возможно, следует записать в соответствующий элемент массива (если оно больше записанного там значения).

```
цел m
m := 120
алг
нач цел n, очер, лев, прав, остат, i,
цел таб макс_числа_с_остатком[0 : m - 1]
ввод n
| Обнуление элементов массива
нц для i от 0 до m -1
 макс_числа_с_остатком[i] := 0
кц
| Начальное присваивание значений искомым величинам
лев := 0
прав := 0
| Ввод и обработка чисел последовательности
нц для i от 1 до n
 ввод очер
 остат := mod(очер, m)
 если остат= 0
  то | Сравниваем значение очер
     | с соответствующим элементов массива
   если очер < макс_числа_с_остатком[0] | Только в этом случае
    то | Сравниваем суммы
     если макс_числа_с_остатком[0] + очер > лев + прав
      то | Встретилась новая пара искомых значений
         | Запоминаем их
         лев := макс_числа_с_остатком[0]
         прав := очер
     все
   все
 иначе | Остаток не равен 0
  если очер < макс_числа_с_остатком[m-остат]
   |Только в этом случае
   то | Сравниваем суммы
   если макс_числа_с_остатком[m - остат] + очер > лев + прав
     то | Встретилась новая пара искомых значений
        | Запоминаем их
        лев := макс_числа_с_остатком[m - остат]
        прав := очер
    все
  все
все
| После обработки очередного числа может измениться
| значение элемента массива с индексом остат
если очер > макс_числа_с_остатком[остат]
 то
```

```
    макс_числа_с_остатком[остат] := очер
  все
кц
| Вывод ответа
вывод лев, " ", прав
кон
```

Примечания

1. Вместо вложенных условных операторов можно использовать сложные условия:

```
остат = 0 и очер < макс_числа_с_остатком[0] и
      макс_числа_с_остатком[0] + очер > лев + прав
```

и

```
очер < макс_числа_с_остатком[m - остат] и
    макс_числа_с_остатком[m - остат] + очер > лев + прав
```

2. Можно при обработке чисел последовательности не рассматривать два варианта значения *остат* (не использовать полный условный оператор), а объединить их в общий случай.

Это можно сделать двумя способами:

1) заменить значение *остат*, равное нулю, на значение m, и при сравнении использовать элемент массива с индексом $m - остат$:

```
…
| Ввод и обработка чисел последовательности
нц для i от 1 до n
 ввод очер
 остат := mod (очер, m)
 если остат= 0
  то | Меняем
    остат := m
 все
 | Сравниваем
если очер < макс_числа_с_остатком[m - остат]
 | Только в этом случае
 то
 если макс_числа_с_остатком[m - остат] + очер > лев + прав
   то
    лев := макс_числа_с_остатком[m - остат]
    прав := очер
  все
все
```

В этом случае при возможном изменении массива *макс_числа_с_остатком* следует рассмотреть два случая:

```
если остат = m
 то
   | Сравниваем с элементов с индексом 0
   если очер > макс_числа_с_остатком[0]
    то
     макс_числа_с_остатком[0] := очер
   все
 иначе
если очер>макс_числа_с_остатком[m - остат]
 то
  макс_числа_с_остатком[m - остат] := очер
все
```

2) использовать в массиве элемент с индексом mod ($m - остат$, m). Нетрудно убедиться, что этот индекс «обобщает» два индекса:

0 при $остат = 0$ и $m - остат$ в противном случае. Если этот индекс обозначить *общ_инд*, то фрагмент программы, связанный с вводом и обработкой чисел последовательности примет вид:

```
нц для i от 1 до n
 ввод очер
 остат := mod(очер, m)
 общ_инд := mod(m - остат, m)
 | Сравниваем
если очер<макс_числа_с_остатком[общ_инд]
 | Только в этом случае
 то
  если макс_числа_с_остатком[общ_инд] + очер > лев + прав
   то
    лев := макс_числа_с_остатком[общ_инд]
    прав := очер
  все
все
```

В этом случае фрагмент, связанный с возможным изменением массива, не меняется:

```
если очер > макс_числа_с_остатком[остат]
 то
  макс_числа_с_остатком[остат] := очер
все
```

В описанной программе хранится только очередной прочитанный элемент (массив не используется) и информация о максимальных значениях, имеющих различные остатки от деления на m (на их хранение будет потрачено не более $4m$ байт памяти, а на все переменные в целом – менее 4 килобайт). Таким образом, используемая память не зависит от длины последовательности n. Время обработки очередного числа фиксировано, то есть не зависит от длины последовательности и даже от величины m. Поэтому при увеличении длины последовательности в k раз время работы программы увеличивается не более чем в k раз. Таким образом, приведённая выше программа эффективна как по времени, так и по используемой памяти. Напомним, что такое решение оценивается четырьмя баллами.

Возможен также вариант программы, основанный на описанных чуть выше идеях, но в котором все элементы последовательности предварительно записываются в массив. Такое решение эффективно по времени, но неэффективно по памяти. Оно оценивается исходя из максимального балла, равного трем.

Литература

1. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена 2020 года по информатике и ИКТ. *http://fipi.ru/ege-i-gve-11/demoversii-specifikacii-kodifikatory*

Калейдоскоп Калейдоскоп Калейдоскоп

Учёные о науке

* * *

Разгадать загадки Вселенной, познать непознанное призвано научное творчество.

А.Б. Мигдал

* * *

Не меняется в науке со временем лишь одно – но главное. Основной стимул для исследователя – это внутреннее стремление к познанию. Это неизменно...

Ф. Крик

* * *

Для человека, любящего науку, узнавать о новых явлениях и эффектах, знакомиться с научными новостями... – огромное удовольствие, радость. Думаю, что это чувство вполне сравнимо и сопоставимо с теми, которые посещают людей при встрече с произведениями искусства.

В.Л. Гинзбург

* * *

Только после превращения собранных фактов в стройную систему представлений – в теорию – возможно предсказание новых явлений.

А.Б. Мигдал