ЕГЭ по информатике 2022: задание 26

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

В трех предыдущих статьях [2][3][4] мы рассмотрели решения заданий 16, 17, 24 и 25, представленных в проекте демонстрационного варианта Единого государственного экзамена по информатике и ИКТ 2022 года [1] и предусматривающих самостоятельное написание программы на выбранном языке программирования. Этим, однако, задания на программирование не исчерпываются. В настоящей, четвертой по счету, статье мы продолжим рассмотрение такого типа заданий и подробно разберём предпоследнее задание ЕГЭ по информатике под номером 26.

Ранее, до 2020 года, то есть до перевода ЕГЭ по информатике в компьютерную форму, в демонстрационном варианте Единого государственного экзамена по информатике и ИКТ [5] задание 26 было представлено задачей на теорию игр. После перевода ЕГЭ в компьютерную форму изменился и тип задания 26. Задача на теорию игр сменила нумерацию и была разделена на три отдельных задания (19, 20 и 21 задания ЕГЭ [1]). А новое задание 26 теперь представляет собой ещё одно задание на программирование.

Кроме того, ранее все задания, предусматривающие написание программы на некотором языке программирования, содержали разбор в демоверсии ЕГЭ [5], причём сразу на нескольких языках программирования. Перевод ЕГЭ по информатике в компьютерную форму повлёк не только изменение задания 26, но и внёс изменения в формат демонстрационного варианта: сейчас в таком варианте полностью отсутствует разбор каких-либо заданий, а содержатся только краткие ответы для самопроверки. Настоящей статьёй мы восполним этот недостаток.

Задание 26

Условие

Системный администратор раз в неделю создаёт архив пользовательских файлов. Однако объём диска, куда он помещает архив, может быть меньше, чем суммарный объём архивируемых файлов.

Известно, какой объём занимает файл каждого пользователя.

По заданной информации об объёме файлов пользователей и свободном объёме на архивном диске определите максимальное число пользователей, чьи файлы можно сохранить в архиве,

а также максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Входные данные.

В первой строке входного файла находятся два числа: S – размер свободного места на диске (натуральное число, не превышающее 10 000) и N – количество пользователей (натуральное число, не превышающее 1000). В следующих N строках находятся значения объёмов файлов каждого пользователя (все числа натуральные, не превышающие 100), каждое – в отдельной строке.

Запишите в ответе два числа: сначала наибольшее число пользователей, чьи файлы могут быть помещены в архив, затем – максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Пример входного файла:

100 4
80
30
50
40

При таких исходных данных можно сохранить файлы максимум двух пользователей. Возможные объёмы этих двух файлов – 30 и 40, 30 и 50 или 40 и 50. Наибольший объём файла из перечисленных пар – 50, поэтому ответ для приведённого примера:

2	50

Таб. 1

Решение

В этой задаче нам потребуется массив (список) для хранения размера файлов пользователей и некоторый алгоритм для его сортировки. Количество элементов массива указано в первой строке приложенного файла (вторым числом), поэтому проблем с выбором размера статического массива здесь нет.

Задания 26 и 27 оцениваются вдвое «дороже» всех остальных заданий экзамена [1], что подчёркивает их сложность, поэтому, прежде чем приступить к написанию программы, нужно хорошо продумать сам алгоритм решения такой задачи.

Первая часть

Условие задачи перед нами ставит два вопроса. Начнём с первого. Как добиться того, чтобы на диске были сохранены файлы максимального числа пользователей? Для этого нам нужно отдать приоритет файлам меньшего размера, поэтому будем сохранять файлы пользователей по одному, каждый раз выбирая наименьший из оставшихся.

Сортировка массива по возрастанию (неубыванию) объема файлов пользователей существенно упрощает нам задачу, поэтому после завершения ввода всех данных из файла в массив (a) мы произведем его сортировку. Для этого можно воспользоваться встроенными функциями сортировки либо самостоятельно реализовать один из известных алгоритмов сортировки массива. Мы возьмем для реализации алгоритм сортировки выбором [6] [7], не самый быстрый в общем случае, но достаточно понятный, чтобы продемонстрировать, как реализуется сортировка массива вручную.

После сортировки получить ответ на первый вопрос задачи достаточно просто. Необходимо последовательно «сохранять файлы пользо-

вателей», начиная с самого первого (минимального) элемента массива, одновременно считая суммарный объём сохраненных файлов (sum), пока объём свободного места (s) не закончится. Запоминая индекс последнего сохраненного нами «файла пользователя (j)», мы, после окончания цикла, будем иметь в этой переменной индекс последнего «вошедшего» элемента массива, а значит, сможем по нему определить и общее число вошедших файлов.

Вторая часть

Получить ответ на второй вопрос нам также поможет отсортированный массив. Как при том же количестве «сохраненных файлов» узнать максимальный размер имеющегося файла, который можно сохранить на диске?

Это необязательно элемент с индексом «j», что хорошо иллюстрирует пример из условия задачи (иначе ответ был бы 40, а не 50). Поэтому для сохранения того же количества файлов нам нужно попробовать заменить какой-то ранее «сохраненный» файл одного пользователя на файл другого пользователя большего размера.

Какой файл следует убрать? Для поиска максимального размера нового файла нам нужно убрать из ранее сохраненных тот файл, который даст при его «удалении» больше всего свободного места, а это файл с максимальным размером из ранее сохраненных. Вспомним, что наш массив был отсортирован по возрастанию, поэтому максимальный размер будет иметь последний сохраненный файл. Его индекс (j) мы очень кстати получили в результате решения первой части задачи.

Теперь нам нужно только найти, на какой файл наибольшего размера нам хватит места после того, как мы уберем последний сохраненный в первой части файл/элемент массива (с индексом «j»). Проходя в цикле элементы массива по возрастанию и обновляя каждый раз индекс подошедшего на замену элемента в отдельной переменной (t), мы, после окончания цикла, будем иметь в этой переменной индекс последнего (наибольшего) подошедшего на замену файла. Отметим только, что на вывод нужно отправлять не саму переменную «t», являющуюся индексом нужного элемента, а именно элемент массива «a[t]», т.е. размер этого файла.

Примечание:

Решение обеих частей задачи можно объединить в одном цикле. В этом случае вначале, когда идут «небольшие» по объёму файлы, мы заполняем свободное место и находим индекс последнего вошедшего элемента (j), а после заполнения места мы в том же цикле начинаем проверять оставшиеся «большие» файлы на предмет возможной замены файла с индексом (j), сохраняя и обновляя их индекс (t) каждым новым подходящим элементом.

На случай, если на диск войдут все файлы, и файлов на замену не останется, то для корректного вычисления индекса файла максимального размера (t) мы его будем дополнительно обновлять и в первой части программы одновременно с обновлением индекса (j). Для описанного случая последний вошедший элемент является последним элементом массива, причём массива, отсортированного по возрастанию (неубыванию), последний элемент которого и является максимальным, поэтому последний вошедший элемент и максимальный элемент совпадают.

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

C++ (№26)

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

int main()
{
  ifstream f("26.txt");
  long s, n, i, j, k, t, sum;
  long a[1000];
  f >> s >> n;
  for (i = 0; i < n; i++)
    f >> a[i];
  // сортировка массива по возрастанию
  for (i = 0; i < n - 1; i++)
  {
    k = i;
    for (j = i + 1; j < n; j++)
      if (a[j] < a[k])
        k = j;
    if (k != i)
    {
      t = a[i];
      a[i] = a[k];
      a[k] = t;
    }
  }
  // подсчет числа пользователей,
  // чьи файлы можно сохранить,
  // и поиск наибольшего файла
  sum = 0;
  for (i = 0; i < n; i++)
    // если можем сохранить ещё один файл
    if (sum + a[i] <= s)
    {
      // обновляем суммарный объём
      // файлов, их количество
      // и индекс наибольшего файла
      sum += a[i];
      j = i;
      t = j;
    }
    else
      // если можем заменить наибольший
      // сохраненный файл на новый
      if (sum - a[j] + a[i] <= s)
        // сохраняем индекс последнего
        // подходящего элемента массива
        t = i;
  cout << j + 1 << ' ' << a[t];
  f.close();
  return 0;
}
```

Рис. 1

Алгоритмический язык (№26)

```
алг
нач
  файл f
  цел s, n, i, j, k, t, sum
  целтаб a[1:1000]
  f := открыть на чтение("26.txt")
  ввод f, s, n, нс
  нц для i от 1 до n
    ввод f, a[i], нс
  кц
  | сортировка массива по возрастанию
  нц для i от 1 до n - 1
    k := i
    нц для j от i + 1 до n
      если a[j] < a[k] то
        k := j
      все
    кц
    если k <> i то
      t := a[i]
      a[i] := a[k]
      a[k] := t
    все
  кц
  | подсчет числа пользователей,
  | чьи файлы можно сохранить,
  | и поиск наибольшего файла
  sum := 0
  нц для i от 1 до n
    | если можем сохранить ещё один файл
    если sum + a[i] <= s то
      | обновляем суммарный объём
      | файлов, их количество
      | и индекс наибольшего файла
      sum := sum + a[i];
      j := i;
      t := j
    иначе
      | если можем заменить наибольший
      | сохраненный файл на новый
      если sum - a[j] + a[i] <= s то
        | сохраняем индекс последнего
        | подходящего элемента массива
        t := i
      все
    все
  кц
  вывод j, " ", a[t]
  закрыть(f)
кон
```

Рис. 2

Паскаль (№26)

```
var
  f: text;
  s, n, i, j, k, t, sum: longint;
  a: array [1..1000] of integer;
begin
  assign(f, '26.txt');
  reset(f);
  readln(f, s, n);
  for i := 1 to n do
    readln(f, a[i]);
  {сортировка массива по возрастанию}
  for i := 1 to n - 1 do
  begin
    k := i;
    for j := i + 1 to n do
      if a[j] < a[k] then
        k := j;
    if k <> i then
    begin
      t := a[i];
      a[i] := a[k];
      a[k] := t
    end
  end;
  {подсчет числа пользователей, чьи файлы}
  {можно сохранить, и поиск наибольшего файла}
  sum := 0;
  for i := 1 to n do
    {если можем сохранить ещё один файл}
    if sum + a[i] <= s then
    begin
      {обновляем суммарный объём файлов, их}
      {количество и индекс наибольшего файла}
      sum := sum + a[i];
      j := i;
      t := j
    end
    else
      {если можем заменить наибольший}
      {сохраненный файл на новый}
      if sum - a[j] + a[i] <= s then
        {сохраняем индекс последнего}
        {подходящего элемента массива}
        t := i;
  writeln(j, ' ', a[t]);
  close(f)
end.
```

Рис. 3

Python (№26)

```
f = open("26.txt")
s, n = [int(x) for x in
f.readline().split()]
a = [int(line) for line in f]
# сортировка массива по возрастанию
for i in range(n - 1):
    k = i
    for j in range (i + 1, n):
        if a[j] < a[k]:
            k = j
    if k != i:
        t = a[i]
        a[i] = a[k]
        a[k] = t
# подсчет числа пользователей,
# чьи файлы можно сохранить,
# и поиск наибольшего файла
sum = 0
for i in range(n):
    # если можем сохранить ещё один файл
    if sum + a[i] <= s:
        # обновляем суммарный объём
        # файлов, их количество и
        # индекс наибольшего файла
        sum += a[i]
        j = i
        t = j
    else:
        # если можем заменить наибольший
        # сохраненный файл на новый
        if sum - a[j] + a[i] <= s:
            # сохраняем индекс последнего
            # подходящего элемента массива
            t = i
print(j + 1, a[t])
f.close()
```

Рис. 4

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. Муржин Д.В. "ЕГЭ по информатике 2022: задание 25" // Потенциал (Математика. Физика. Информатика). – 2021. – № 11.
5. http://doc.fipi.ru/ege/demoversii-specifikacii-kodifikatory/2020/inf_ege_2020.zip
6. Кнут Д. Э. Искусство программирования. Том 3. Сортировка и поиск = The Art of Computer Programming. Volume 3. Sorting and Searching / под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). — 2-е изд. — Москва: Вильямс, 2007. — Т. 3. — 832 с. — ISBN 5-8459-0082-1.
7. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ = INTRODUCTION TO ALGORITHMS. — 2-е изд. — М.: «Вильямс», 2006. — С. 1296. — ISBN 5-8459-0857-4.

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