Две задачи ЕГЭ: более удобное решение

О.Б. Богомолова,
д. п. н., учитель информатики и математики
ГБОУ СОШ № 1360,
Восточный округ
г. Москвы,

Д.Ю. Усенков,
Москва,

Арсений Парфенов,
ученик 11-го класса
ГБОУ СОШ № 1360,
Восточный округ г. Москвы

Многие задачи Единого госэкзамена можно решать разными способами. Одни из этих способов более универсальны, а другие при их “жесткой” привязке к конкретной разновидности задач позволяют решать их проще, быстрее или более наглядно. Нередко такие “оптимизированные” способы решения предлагают сами учащиеся. В этой статье читателям предлагаются новые методы решения двух заданий ЕГЭ, предложенные Арсением Парфеновым (учеником 11-го “А” класса школы № 1360 г. Москвы).

Рекурсии

Задача 1. Имеется следующий рекурсивный алгоритм:

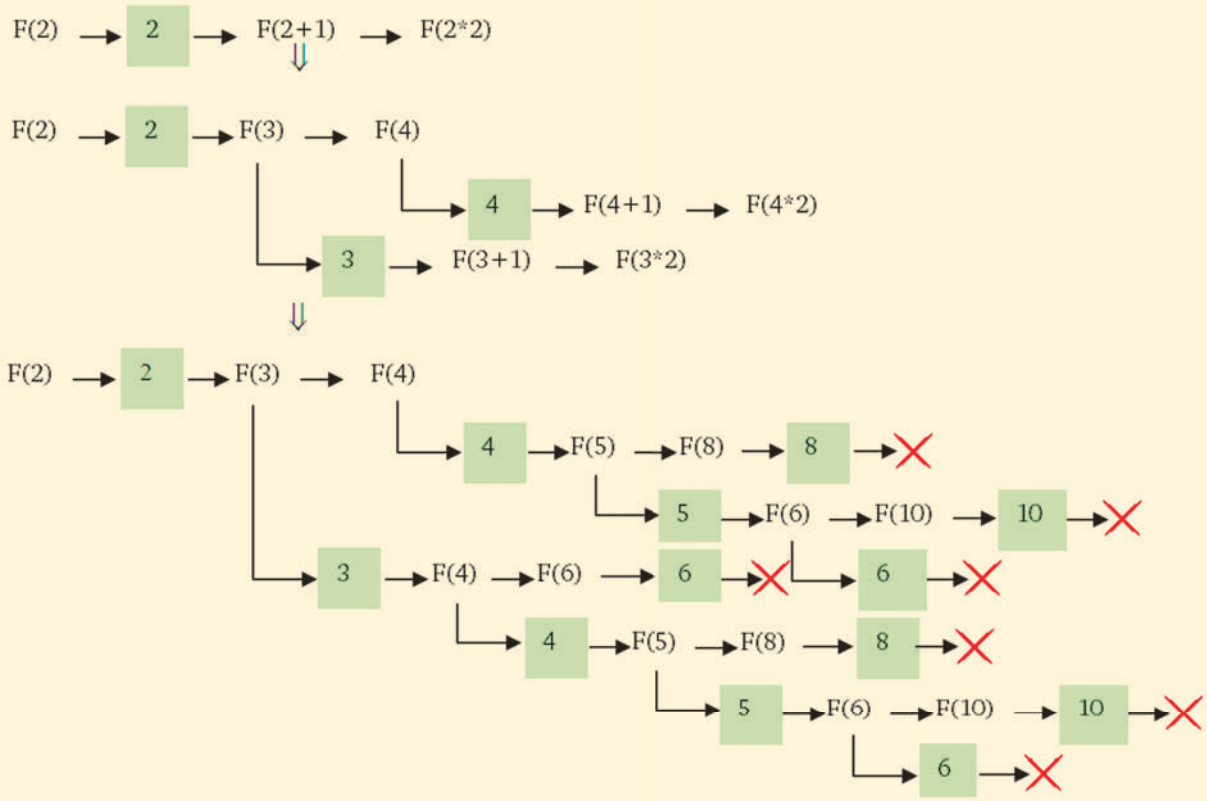
```
procedure F(n: integer);
begin
 writeln(n);
 if n < 6 then begin
   F(n + 1);
   F(n * 2)
 end
end;
```

Чему равна сумма чисел, выведенных на экран при вызове F(2)?

Решение

Подобные задачи уже встречались в заданиях ЕГЭ (хотя эта — несколько усложнена по сравнению с уже привычными). Наиболее универсальным способом их решения является построение “дерева вызовов” функции с последующим подсчетом суммы чисел на концах его “ветвей” (см., например, нашу статью «Тренинг по информатике: “разбор полетов”» в № 2/2015). Например, для этой задачи такая схема “по шагам” представлена на с. 35.

А далее остается просуммировать числа, которые выводятся на экран при работе данного алгоритма (и которые в нашем “дереве” выделены зе-



леным фоном). Эта сумма равна 2 + 4 + 8 + 5 + +10 + 6 + 3 + 6 + 4 + 8 + 5 + 10 + 6 = 77.

Нетрудно видеть, что даже для такой несложной задачи "дерево вызовов" получается довольно "развесистое", и к тому же при его рисовании требуется изрядная доля внимательности, чтобы ничего не упустить (а особенно — не забыть, что даже если очередное значение *n* уже не удовлетворяет условию в операторе `if`, значение *n* все равно выводится на экран, так как соответствующий оператор располагается до условного оператора) и просуммировать все полученные числа. И, как показала практика, с рисованием таких "деревьев" справляются далеко не все учащиеся.

Что можно предложить взамен?

Метод решения, предложенный Арсением Парфеновым, можно назвать "алгебраическим". Он несколько "искусственный" и работает только для таких задач, в которых в рекурсивной функции сначала стоит оператор `if` с условием "*n* меньше некоторого граничного значения", а в ветви `then` этого `if` записаны рекурсивные вызовы функции *F* с аргументом, изменяющимся в бóльшую сторону, причем оператор вывода значения *n* стоит перед `if` и может также иметься в ветви `then`.

Рассмотрим этот метод решения на примере задачи, только что разобранной "классическим" способом с построением "дерева".

В отличие от "дерева вызовов" здесь мы решаем задачу "наоборот", в порядке, противоположном нормальной работе рекурсивного алгоритма. Начинаем запись с последнего натурального числа, которое еще удовлетворяет условию в операторе `if`. В данном случае это условие имеет вид $n < 6$, поэтому начинаем запись с числа 5.

При этом цветом выделяем число, выводимое на экран.

Для записи вызовов функции *F*() в строку записываем или ранее вычисленное значение для этой функции (если оно есть), или просто число, получаемое в качестве аргумента функции.

В последней записи должно быть то число, которое задано в вызове (в данном случае — число 2).

n	Запись
5	F(5) = **5** + F(5 + 1) + F(5 * 2) = **5** + F(6) + F(10)= = **5** + 6 + 10 = 21
4	F(4) = **4** + F(4 + 1) + F(4 * 2) = **4** + F(5) + F(8) = = **4** + 21 + 8 = 33
3	F(3) = **3** + F(3 + 1) + F(3 * 2) = **3** + F(4) + F(6) = **3** + 33 + 6 = 42
2	F(2) = **2** + F(2 + 1) + F(2 * 2) = **2** + F(3) + F(4) = **2** + 42 + 33 = **77**
Итог:	F(2) = 77

Как видим, ответ получается тот же самый.

Еще немного усложним задание:

Задача 2. Имеется следующий рекурсивный алгоритм:

```
procedure F(n: integer);
begin
 writeln(n);
 if n < 7 then begin
   writeln(n);
   F(n + 1);
   F(n + 2);
   F(n * 2)
 end
end;
```

Чему равна сумма чисел, выведенных на экран при вызове F(2)?

Решение

Основное усложнение здесь — в том, что в алгоритме есть два оператора вывода: один — вне `if`, второй — в его ветви `then`. Однако принцип формирования записей остается тот же, хотя к таблице добавляется еще одна отдельная графа: теперь в "основной" части таблицы мы записываем расчеты для "внутренних" операторов (в ветви `then` оператора `if`), а затем в отдельной графе записываем расчеты для отдельного оператора вывода вне `if`.

1. Часть внутри `if`. Условие имеет вид $n < 7$, поэтому начинаем запись с числа 6. Цветом выделяем число, выводимое на экран. Для записи вызовов функции $F()$ в строку записываем или ранее вычисленное значение для этой функции (если оно есть), или просто число, получаемое в качестве аргумента функции. В последней записи должно быть число, которое задано в вызове (в данном случае — число 2).

2. Часть вне `if`. В каждой строке записи сначала записывается само число, которое выводится на экран внешним оператором вывода. Затем мы смотрим на соответствующую этому числу "основную" запись $F()$: если в ней задействованы ранее вычисленные значения $F(n)$, то к выведенному на экран числу прибавляются значения, вычисленные нами в "дополнительной" графе для этих $F(n)$.

В итоге нужно сложить значение, полученное в последней записи "основной" графы, и значение, полученное в последней записи "дополнительной" графы.

n	Внутри if	Вне if
6	F(6) = **6** + F(6 + 1) + F(6 + 2) + F(6 * 2) = = **6** + F(7) + F(8) + F(12) = **6** + 7 + 8 + 12 = 33	**6**
5	F(5) = **5** + F(5 + 1) + F(5 + 2) + F(5 * 2) = = **5** + F(6) + F(7) + F(10) = **5** + 33 + 7 + 10 = 55	**5** + 6 = 11 (шестерка "пришла" сюда через вызов F(6))
4	F(4) = **4** + F(4 + 1) + F(4 + 2) + F(4 * 2) = = **4** + F(5) + F(6) + F(8) = **4** + 55 + 33 + 8 = 100	**4** + 11 + 6 = 21 (число 11 "пришло" через вызов F(5), число 6 — через вызов F(6))
3	F(3) = **3** + F(3 + 1) + F(3 + 2) + F(3 * 2) = = **3** + F(4) + F(5) + F(6) = = **3** + 100 + 55 + 33 = 191	**3** + 21 + 11 + 6 = 41 (число 21 — через вызов F(4), 11 — через вызов F(5), 6 — через вызов F(6))
2	F(2) = **2** + F(2 + 1) + F(2 + 2) + F(2 * 2) = = **2** + F(3) + F(4) + F(4) = **2** + 191 + 100 + 100 = = 393	**2** + 41 + 21 + 21 = 85 (число 41 — через вызов F(3), 21 — через вызов F(4))
Итог:	F(2) = 393 + 85 = **478**.	

На первый взгляд объяснение кажется сложным. Но реально решать такую задачу показанным выше способом гораздо быстрее и проще, чем при помощи "дерева".

Черчение

Задача 3. Исполнитель Чертежник перемещается по координатной плоскости. Основная команда Чертежника — **сместиться на (*a*, *b*)**, где *a*, *b* — целые числа. Она перемещает Чертежника из точки с координатами (*x*, *y*) в точку с координатами (*x* + *a*, *y* + *b*). Например, из точки с координатами (3, 5) команда **сместиться на (–2, 1)** переместит Чертежника в точку (1, 6).

Цикл

```
ПОВТОРИ число РАЗ
  последовательность команд
КОНЕЦ ПОВТОРИ
```

означает, что заданная последовательность команд будет выполнена указанное количество раз (значение должно быть натуральным).

Чертежник выполнял алгоритм, в котором количество повторений и оба смещения в первой из повторяемых команд неизвестны:

```
НАЧАЛО
  сместиться на (-2, 3)
ПОВТОРИ … РАЗ
  сместиться на (…, …)
  сместиться на (-2, -1)
КОНЕЦ ПОВТОРИ
  сместиться на (-19, -18)
КОНЕЦ
```

При этом, выполнив этот алгоритм, Чертежник вернулся в исходную точку. Какое наибольшее количество повторений могло быть указано в конструкции “ПОВТОРИ … РАЗ”?

Решение

Такие задачи мы тоже уже рассматривали в одной из предыдущих статей. Полное их решение заключается в составлении системы двух уравнений по координатам *x* и *y* с дальнейшим их анализом:

$$\begin{cases} -2 + k \cdot (x - 2) - 19 = 0, \\ 3 + k \cdot (y - 1) - 18 = 0; \end{cases}$$

$$\begin{cases} k \cdot (x - 2) = 21, \\ k \cdot (y - 1) = 15. \end{cases}$$

В обоих случаях слева от знака равенства стоят произведения, один из сомножителей в которых один и тот же (*k*). Разложив числа справа от знаков равенства на простые сомножители ($21 = 3 \cdot 7$; $15 = 3 \cdot 5$), замечаем, что в обоих получившихся произведениях тоже повторяется один и тот же сомножитель — 3. Значит, это и есть *k*.

Для таких задач можно упростить и ускорить решение, исключив составление уравнений.

Для этого нужно, не обращая внимание на команды внутри цикла ПОВТОРИ, отдельно сложить только пары чисел — значений координат в командах перемещений до и после цикла:

```
сместиться на (-2, 3)
…
сместиться на (-19, -18)
```

Получаем:

- по *X*: –2 + (–19) = –21;
- по *Y*: 3 + (–18) = –15.

Полученные значения сумм берем по модулю, получаем числа 21 и 15.

Их можно разложить на простые множители так:

$$21 = 3 \cdot 7,$$
$$15 = 3 \cdot 5.$$

Повторяется в них число 3. Это и есть значение количества повторений цикла.

Задача 4. Исполнитель Чертежник перемещается по координатной плоскости. Основная команда Чертежника — **сместиться на (*a*, *b*)**, где *a*, *b* — целые числа. Она перемещает Чертежника из точки с координатами (*x*, *y*) в точку с координатами (*x* + *a*, *y* + *b*).

Чертежник выполнял алгоритм, в котором количество повторений и оба смещения в первой из повторяемых команд неизвестны:

```
НАЧАЛО
  сместиться на (-2, -5)
ПОВТОРИ … РАЗ
  сместиться на (…, …)
  сместиться на (-2, -1)
КОНЕЦ ПОВТОРИ
  сместиться на (22, 15)
КОНЕЦ
```

При этом, выполнив этот алгоритм, Чертежник вернулся в исходную точку. Какое наибольшее количество повторений могло быть указано в конструкции “ПОВТОРИ … РАЗ”?

Решение

Как и в предыдущем случае, берем только первую и последнюю пары значений смещения по *X* и *Y*: (–2, –5) и (22, 15).

Вычисляем:

- по *X*: –2 + 22 = 20,
- по *Y*: –5 + 15 = 10.

Полученные числа можно разложить на множители так: $20 = 2 \cdot 2 \cdot 5$, $10 = 2 \cdot 5$.

В этих разложениях повторяются двойка и пятерка. Значит, количество повторений цикла может быть равно 2, 5 или 10. Нам же, по условию, нужно определить наибольшее возможное количество повторений цикла. Следовательно, ответом является число 10.

Ответ: 10.