Тренинг по информатике: “разбор полетов”

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

Д.Ю. Усенков, ст. н. с. Института информатизации образования Российской академии образования, Москва

И для 9-х, и для 11-х классов время с поздней осени по раннюю весну — это своего рода “сезон катастроф”, именуемых тренировочными и диагностическими работами в формате ОГЭ и ЕГЭ ☺. Впрочем, и польза от таких тренировок весьма существенна: они не только дают возможность проверить знания школьников, но и подсказывают учителю, “куда двигаться дальше” в плане подготовки детей к настоящему экзамену — на чем заострить внимание, если какие-то уже разобранные ранее задачи “не удаются”, а какие вновь встретившиеся задания — изучить подробно. А заодно и отточить мастерство в решении задач, искать все более оптимальные методы решения, более легкие и экономные по времени, которого на экзамене, как потом выясняется, так мало…

Ниже представлены некоторые методические выводы и примеры решения некоторых заданий ноябрьской тренировочной работы по информатике для 11-х классов, которые вызвали у школьников трудности. Правда, поскольку МИОО (разработчик заданий) запрещает публикацию своих материалов, разбирать мы будем решение своих собственных заданий, аналогичных по смыслу тренировочным.

1. Снова Фано

Задачи на пресловутое “условие Фано”, определяющее системы кодов, обеспечивающие однозначную расшифровку закодированных сообщений, давно уже “прописались” в экзаменационных задачах. Не стал исключением и данный тренинг.

Для кодирования последовательности символов, состоящей из букв

К, И, Н, О, используется неравномерный двоичный код, удовлетворяющий условию Фано. При этом для буквы К использован код 0, а для буквы И — код 11. Требуется определить наименьшую возможную суммарную длину всех кодовых слов указанных букв.

1) 8; 3) 10;
2) 9; 4) 11.

Решение

Напомним, в чем заключается условие Фано.

> Закодированное сообщение можно однозначно декодировать **с начала**, если **никакое кодовое слово не является началом другого кодового слова.**

Для проверки на соответствие кодов условию Фано нужно попарно сравнивать между собой коды по следующим правилам:

• когда длина обоих сравниваемых кодов совпадает, проверяется равенство этих кодов: если один код совпадает с другим, то такая пара кодов неправильна (не удовлетворяет условию Фано);

• когда длина сравниваемых кодов различна, более короткий код записывается под более длинным с выравниванием обоих кодов по левому краю: если все знаки более короткого кода совпадают с соответствующими знаками в начале более длинного кода, то такая пара кодов неправильна (не удовлетворяет условию Фано).

Опираясь на эти правила, будем подбирать коды для оставшихся букв Н и О. Начнем с буквы Н и будем перебирать возможные двоичные числа с возрастающей длиной (ведь нам важно получить наименьшую возможную суммарную длину кодов!).

Код буквы К	Код буквы И	Предполагаемый код буквы Н	Комментарий
0	11	1	Нельзя, так как совпадает с началом кода буквы И
		00	Нельзя — код буквы К совпадает с его началом
		01	Нельзя — код буквы К совпадает с его началом
		10	Допустимый код (не совпадает с двузначным кодом буквы И, а код буквы К не совпадает с его началом)

Итак, можно предположить, что первый код найден. Но посмотрим — удастся ли при этом найти код для оставшейся четвертой буквы О. При этом можно сразу отбросить те коды, которые не подошли для буквы Н, — ведь код буквы О должен удовлетворять тем же требованиям при сравнении с кодами букв К и И.

Код буквы К	Код буквы И	Код буквы Н	Предполагаемый код буквы О	Комментарий
0	11	10	11	Нельзя — совпадает с кодом буквы И
			000, 001, 010, 011	Нельзя — код буквы К совпадает с его началом
			100, 101	Нельзя — код буквы Н совпадает с его началом
			110, 111	Нельзя — код буквы Н совпадает с его началом

Очевидно, и дальше с увеличением числа разрядов в предполагаемом коде буквы О будет сохраняться та же тенденция: ни один код не пригоден. Как же быть?

Причина этой прискорбной ситуации в том, что, выбрав для буквы Н код 10, мы “закрыли” для себя возможность дальнейшего расширения нашей кодовой системы. Поэтому вместо кода 10 нам придется выбрать для буквы Н более длинный код, — например, 100.

А теперь повторим попытку поиска кода для буквы О:

Код буквы К	Код буквы И	Код буквы Н	Предполагаемый код буквы О	Комментарий
0	11	100	101	Допустимый код (не совпадает с трехзначным кодом буквы Н, а коды букв К и И не совпадают с его началом)

Таким образом, решение найдено. Выпишем коды всех четырех букв:

К	И	Н	О
0	11	100	101

Подсчитаем суммарную длину этих кодов (в знаках): 1 + 2 + 3 + 3 = 9.

Ответ: 2).

2. Крупинки истинности ☺

Для таблицы истинности функции F известны значения нескольких ячеек:

$x1$	$x2$	$x3$	$x4$	F
	1	0		1
0	1			1
1	0			0
	0		1	0

Каким выражением может быть F?

1) $x1 \wedge x2 \wedge x3 \wedge \neg x4$
2) $x1 \vee x2 \vee x3 \vee \neg x4$
3) $\neg x1 \wedge x2 \wedge \neg x3 \wedge x4$
4) $\neg x1 \vee x2 \vee \neg x3 \vee x4$

Решение

Смысл решения — по очереди "примерять" каждый из предложенных вариантов ответа к таблице истинности и смотреть: если при указанных в ней значениях переменных хоть в какой-нибудь строке ***не получается*** требуемое значение F, то данное значение не подходит. Если же для всех строк таблицы истинности и всех имеющихся в ней значений переменных требуемое значение F получается, то мы считаем, что такое логическое выражение *может* соответствовать этой таблице и, соответственно, является решением задачи.

Но можно ускорить решение. Вспомним: для логической операции И "критическим" значением является ноль: если ***хотя бы одна*** переменная равна нулю, то и все выражение тоже равно нулю. Аналогично, для логической операции ИЛИ "критическим" значением является единица: если ***хотя бы одна*** переменная равна единице, то и все выражение тоже равно единице. (Разумеется, нужно не забывать и заменять ноль на единицу и единицу на ноль при операции отрицания.)

Поэтому нам достаточно при проверке вариантов ответа проверять только часть строк таблицы истинности:

• если логическое выражение составлено из операций И, то прежде всего проверяем строки таблицы истинности, в которых $F = 1$: если хоть в одной ячейке, соответствующей "обычной" переменной, записан 0, а в ячейке, соответствующей переменной с НЕ, записана 1, то такой вариант ответа можно отбросить сразу;

• наоборот, если логическое выражение составлено из операций ИЛИ, то прежде всего проверяем строки таблицы истинности, в которых $F = 0$: если хоть в одной ячейке, соответствующей "обычной" переменной, записана 1, а в ячейке, соответствующей переменной с НЕ, записан 0, то такой вариант ответа тоже можно отбросить сразу.

Итак, предполагаемая "кандидатура" на правильный ответ — выражение $\neg x1 \wedge x2 \wedge \neg x3 \wedge x4$. Проверяем его и для оставшихся строк с $F = 0$, убеждаемся, что им оно тоже соответствует (за счет того, что $x2 = 0$).

Итого ответ может быть в наихудшем случае найден за 10 "проверок" вместо 16 при обычном решении.

Ответ: 3).

3. Сколько единиц?

Определите наименьшее пятизначное восьмеричное число, двоичная запись которого содержит три единицы. (В ответе записать только само число, без индекса системы счисления.)

Решение

На первый взгляд кажется, что все просто: чем правее в двоичном числе записаны единицы, тем это число меньше[1]. Значит, наименьшим является число 111_2.

Так-то оно так, — но нам нужно, чтобы число было в восьмеричном формате пятизначным. А двоичное число 111, как легко видеть, соответствует восьмеричной семерке, т.е. является однозначным.

Как же найти наименьшее пятизначное восьмеричное число?

Единственный выход — оставляя как можно больше единичек справа, "отодвигать" только одну единицу влево, записывая после нее нули до тех пор, пока не получится такое двоичное число, которое после перевода в восьмеричную систему счисления будет иметь пять цифр. Причем такое число будет наименьшим из возможных, потому что если бы мы перемещали хоть одну из оставшихся справа единиц левее, то еще увеличивали бы наше число.

А сколько нулей дописывать? Вспомним, что каждой восьмеричной цифре соответствуют *три* двоичных цифры. Значит, нужно "оттаскивать" единицу влево, пока она только-только не "вылезет" в пятую триаду (будет в ней самым младшим битом), — см. схему на с. 23.

№	Выражение	Операция	Проверяем строки таблицы	Комментарий	Вывод
1)	$x1 \wedge x2 \wedge x3 \wedge \neg x4$	И	с $F = 1$	$x3 = 0$ (строка 1)	Не годится
2)	$x1 \vee x2 \vee x3 \vee \neg x4$	ИЛИ	с $F = 0$	$x1 = 1$ (строка 3)	Не годится
3)	$\neg x1 \wedge x2 \wedge \neg x3 \wedge x4$	И	с $F = 1$	все соответствует	
4)	$\neg x1 \vee x2 \vee \neg x3 \vee x4$	ИЛИ	с $F = 0$	$x4 = 1$ (строка 4)	Не годится

[1] Вообще в любой системе счисления в наименьшем числе чем больше цифра, тем правее она должна стоять в записи числа (т.е. попадать в самые младшие разряды). И наоборот, в наибольшем числе самые большие цифры должны быть записаны ближе к левому краю (в старших разрядах). — **Прим. авт.**

$$\underbrace{1}_{1}\ \underbrace{000}_{0}\ \underbrace{000}_{0}\ \underbrace{000}_{0}\ \underbrace{011_2}_{3_8}$$

Ответ: 10003.

Примечание. Некоторые школьники допустили ошибку при верной общей идее решения — они сместили левую единицу так, чтобы получить слева целую триаду из трех двоичных разрядов (“100”), забыв, что можно дополнить триаду незначащими нулями слева. В результате у них получилось восьмеричное число 40003, которое не является минимально возможным.

4. “Эх, дороги…”

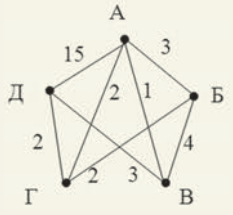
Между городами А, Б, В, Г, Д построены дороги, протяженность которых указана в таблице. Отсутствие числа в ячейке означает, что прямой дороги между соответствующими городами нет.

	А	Б	В	Г	Д
А		3	1	2	15
Б	3		4	2	
В	1	4			3
Г	2	2			2
Д	15		3	2	

Найти длину кратчайшего пути между пунктами А и Д, проходящего через пункт В, если передвигаться можно только по указанным дорогам.

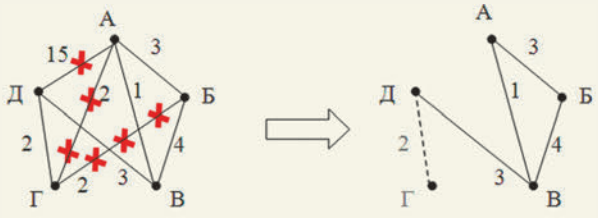
Решение

Как обычно, строим по таблице граф дорог:



Дальше нужно искать все возможные пути из А в Д, отбрасывая те, которые не проходят через город В, подсчитывать их длину и искать путь с минимальным значением его суммарной длины.

Можно ли упростить и ускорить решение? Да! Можно сразу исключить из графа пути “в обход” города В:



Получили совсем простой граф (тем более что в город Г теперь вообще ни одна дорога из А не идет, так что путь ГД тоже можно исключить).

В этом графе — только два возможных пути:

- АБВД с длиной 3 + 4 + 3 = 10,
- АВД с длиной 1 + 3 = 4.

Причем то, что кратчайший из них — это путь АВД, видно “невооруженным глазом”. Его длина 4 — и есть ответ.

Ответ: 4.

5. “Однорукий бандит”

Обычно так называют игровые автоматы, при помощи которых владельцы казино очищают карманы слишком азартных игроков. Но числовой автомат, о котором идет речь в задаче, наверное, тоже представляется ученикам не в самом радужном свете ☺.

Автомат получает на вход трехзначное число. По этому числу формируется новое число по следующим правилам.

1. Складываются первая и вторая, а затем — вторая и третья цифры исходного числа.

2. Полученные два числа записываются друг за другом подряд в порядке возрастания.

Пример. Исходное число: 176. Полученные числа: 1 + 7 = 8, 7 + 6 = 13. Результат: 813.

Найдите наибольшее исходное число, для которого автомат выдаст результат 815.

Решение

Сначала определим, как можно “разрезать” результирующее число на два, вычисленных автоматом.

Вспомним, что сумма двух десятичных цифр (которые могут быть равны от 0 до 9) может равняться от 0 (0 + 0) до 18 (9 + 9).

Если пытаться разделить число 815 как 81 и 5, то первое из этих чисел не соответствует этому возможному диапазону значений сумм цифр (да к тому же тогда числа записаны не по возрастанию). Поэтому остается только один вариант: число 815 как 8 и 15.

Теперь посмотрим, какие цифры могут давать такие суммы:

- 8 = 0 + 8, 1 + 7, 2 + 6, 3 + 5, 4 + 4;
- 15 = 9 + 6, 8 + 7.

А теперь — самое главное. По условию, одна сумма — это сумма первой и второй цифр исходного числа, а вторая — это сумма второй и третьей цифр. Вторая цифра, таким образом, должна повторяться в обеих суммах.

Ищем такие две суммы в обеих “подборках” (для числа 8 и для числа 15).

Это пары: 0+8 и 8+7; 1+7 и 8+7; 9+6 и 2+6.

Им соответствуют числа (учитывая, что ноль не может быть записан самым первым — тогда он был бы незначащим, а число — двузначным): 780, 178, 871, 962, 269 (везде повторяющаяся цифра стоит в середине).

Остается найти среди них наибольшее. Очевидно, это число 962.

Ответ: 962.

6. “Формула-1” в Excel

Дан фрагмент электронной таблицы. Из ячейки C3 в одну из ячеек столбца D была скопирована

формула. После копирования значение этой формулы стало равным 100.

В какую ячейку была скопирована формула? В ответе надо записать только номер строки, в которой находится эта ячейка.

	A	B	C	D
1	5	13		
2	6	12		
3	7	11	=$B3*A$4	
4	8	10		

Решение

Сначала определим, как меняется формула при ее копировании в ячейки столбца D (с учетом абсолютной адресации):

	C	D
1		=$B1*B$4
2		=$B2*B$4
3	=$B3*A$4	=$B3*B$4
4		=$B4*B$4

Остается только вычислить произведения и сверить их с требуемым. Впрочем, и так сразу видно, что 100 — это 10*10, а подходящая формула — только в ячейке D4 (=$B4*B$4).

Ответ: 4.

Примечание. Важно обратить внимание школьников на то, что нужно внимательно читать условие задачи — в том числе о том, как надо записать ответ. Запись ответа в виде “D4” при автоматической проверке ответов будет признана за ошибку!

7. Оцифровка аудио

Выполнена квадро (четырехканальная) звукозапись с частотой дискретизации 32 кГц и 16-битным разрешением. В результате получен файл размером 38 Мбайт, причем сжатие данных не производилось. Требуется приблизительно оценить, сколько времени (в минутах) производилась запись. В качестве ответа нужно указать ближайшее к полученному времени записи целое число минут.

Решение

Такие задачи школьникам уже знакомы. Их решение сводится к записи одного-единственного уравнения, — нужно только внимательно записать в него все составляющие:

- количество каналов записи — 4;
- частота — 32 000 колебаний в секунду;
- разрешение — 16 бит;
- длительность — неизвестна, и мы обозначим ее как t;
- получаемый размер файла равен 38 Мбайт = $= 38 \times 2^{23}$ бит.

Главное — ничего не забыть; обязательно преобразовать все величины к одним и тем же размерностям и правильно выполнить вычисления.

$$4 \cdot 32\,000 \cdot 16 \cdot t = 38 \cdot 2^{23},$$

откуда $t = 155{,}648$. Это — в секундах. А у нас требуют указать длительность в минутах. Значит, полученное значение надо обязательно разделить на 60. А также (согласно условию задачи) — округлить полученное дробное значение ***до ближайшего целого***:

$t = 155{,}648/60 \approx 2{,}594 \approx 3$ минуты.

Ответ: 3.

8. Parole, parole, parole[2]…

Сколько “слов” длины 7 символов, начинающихся с английской буквы, можно составить из букв S, И, R, П, Q? Каждая буква может входить в “слово” несколько раз, а сами получаемые “слова” не обязательно должны быть осмысленными.

Решение

Для начала определим, сколько “слов” можно составить из указанных букв ***без учета*** той самой первой буквы, на которую накладывается особое условие (“только английская”), — т.е. количество “слов” длиной 6 символов.

Сделать это легче всего, если сопоставить каждой букве “свою” цифру, например: S = 0, И = 1, R = 2, П = 3, Q = 4 (порядок цифр в сущности не важен). Тогда нетрудно понять, что мы получили аналогию с числами в соответствующей системе счисления: “сколько различных 6-разрядных чисел можно получить в пятеричной системе счисления”. А это уже легко: количество таких чисел равно n^m, где n — основание системы счисления, а m — количество разрядов. В нашем случае, очевидно, получается $5^6 =$ $= 15\,625$ чисел (“слов”).

А теперь вернемся к первой букве, которая должна быть только английской. Английских букв у нас три. И для ***каждой*** из них возможно 15 625 слов. Значит, общее количество слов будет равно $3 \cdot 15\,625 = 46\,875$.

Ответ: 46 875.

Примечание. А что будет, если в условии будет сказано, что две первые буквы должны быть только английскими?

Тогда решение будет таким:

- кроме этих букв, остаются пять букв любых, тогда таких пятисимвольных “слов” будет $5^5 = 3125$;
- две первые буквы — только английские, а таких букв у нас три; значит, речь идет о количестве двузначных чисел в троичной системе: $3^2 = 9$;
- для каждого из этих девяти вариантов начала “слова” может быть 3125 вариантов продолжения, — значит, всего таких слов будет $9 \cdot 3125 =$ $= 28\,125$.

[2] “Слова, слова, слова…” (из популярной французской песни).

9. “Эх, раз, еще раз!”…

Имеется рекурсивный алгоритм *F*:

```
procedure F(n: integer): integer;
 begin
  if n > 3 then
    F := F(n - 1) + F(n - 2) + F(n - 3)
  else
    F := 2;
 end;
```

Чему равно значение, вычисленное при вызове этой процедуры в виде $F(6)$?

Решение

Вспомним: рекурсия — это вызов функции или процедуры “самой из себя”. При этом в каждом новом вызове вычисления внутри функции / процедуры производятся уже с соответствующим значением “внутренней” (формальной) переменной. Кроме того, в рекурсивной функции / процедуре обязательно должны быть и “конечные” условия, когда значение переменной определяется однозначно, без новых рекурсивных вызовов. В нашем случае это условие:

```
if n > 3 then
  …
else
  F := 2;
```

Чтобы не запутаться и не забыть ни один рекурсивный вызов, удобнее всего вычертить схему этих вызовов в виде дерева.

1) Записываем самый первый вызов, для которого $n = 6$:

$F(6) = F(6-1) + F(6-2) + F(6-3) =$
$= F(5) + F(4) + F(3).$

2) Рекурсивный вызов $F(5)$ расписываем аналогично:

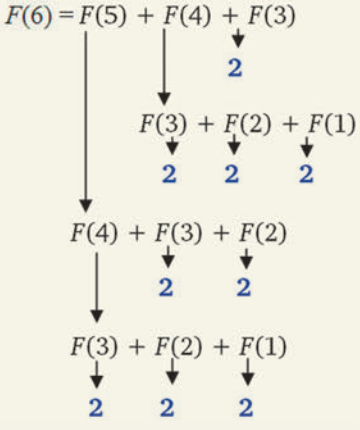
$F(5) = F(5-1) + F(5-2) + F(5-3) =$
$= F(4) + F(3) + F(2).$

И точно так же расписываем вызов $F(4)$:

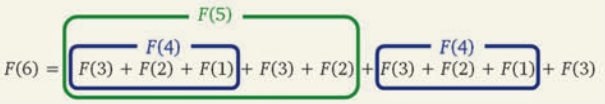
$F(4) = F(4-1) + F(4-2) + F(4-3) =$
$= F(3) + F(2) + F(1).$

А вот значения $F(3)$, $F(2)$ и $F(1)$ у нас — “конечные”: согласно “конечному” условию рекурсии, все они равны 2.

3) Тогда дерево вызовов будет следующим:



4) Теперь можно расписать рекурсивные вычисления в виде простой суммы, постепенно “расшифровывая” каждое значение:



Теперь уже можно заменить вызовы $F(1)$, $F(2)$, $F(3)$ известными нам числовыми значениями:

$F(6) = 2 + 2 + 2 + 2 + 2 + 2 +$
$+ 2 + 2 + 2 = 9 \cdot 2 = 18.$

Ответ: 18.

10. “Призрак Оперы”… а также Firefox, Chrome и прочих

В сетях TCP/IP *маска сети* — это двоичное число, меньшее 2^{32}; в маске сначала (в старших разрядах) записаны единицы, а затем с некоторого бита — нули. Маска определяет, какая часть IP-адреса относится к адресу подсети, а какая — к адресу конкретного компьютера (узла) в этой сети. Маска записывается по тем же правилам, что и IP-адрес, — в виде четырех десятичных чисел, каждое из которых соответствует одному байту и отделяется от других точкой. Адрес сети получается в результате применения поразрядной конъюнкции к заданному IP-адресу узла и маске.

Для узла с IP-адресом 167.57.252.220 адрес сети равен 167.48.0.0. Чему равен второй по счету слева байт маски? Ответ нужно записать в виде десятичного числа.

Решение

Обычно требовалось найти адрес сети по известному IP-адресу и маске. Здесь же нам, наоборот, требуется подобрать маску для известных IP-адреса и адреса сети.

Кроме того, заметим: первый байт адреса сети совпадает с первым байтом IP-адреса, — значит, первый байт маски равен 11111111_2; третий и четвертый же байты адреса сети нулевые, значит, им соответствуют и нулевые байты маски. Второй же по счету байт маски — самый “интересный”, он должен содержать как единицы, так и нули. Именно поэтому в задаче требуется определить не всю маску, а только этот “ключевой” второй байт.

1) Переводим оба “ключевых” числа в двоичную систему счисления:

- $57_{10} = 111001_2$;
- $48_{10} = 110000_2$.

2) Записываем поразрядную конъюнкцию, в которой второй операнд неизвестен:

```
  111001
& ??????
--------
  110000
```

3) Сопоставляем первый операнд и результат:

- первый бит в обоих случаях равен 1, значит, соответствующий бит маски тоже равен 1;

• со вторым битом ситуация та же, значит, второй бит маски тоже равен 1;

• а вот третий бит в IP-адресе равен 1, а в адресе сети уже равен нулю, следовательно, соответствующий третий бит маски должен быть нулевым.

А вот теперь — самое интересное! Четвертый бит IP-адреса, как и четвертый бит адреса сети, равен нулю. Чему равен тогда бит в маске? Ведь он может быть равен как 1, так и 0, — в обоих случаях конъюнкция с нулем дает ноль.

И вот тут-то надо вспомнить, что в маске сначала до какого-то разряда идут только единицы, а начиная с этого разряда — только нули! Поэтому если третий бит маски (как мы однозначно определили) равен нулю, то ***все последующие*** биты правее этого нуля тоже должны быть только нулями! Поэтому маска получится такой: 110000_2, или 48_{10}.

Вроде бы все правильно. Но ответ в этой задаче совсем другой! В чем же мы (намеренно), а также и почти все учащиеся, решавшие эту задачу, ошиблись? А вот в чем.

Мы "забыли", что каждое из четырех чисел, записанных в маске через точку, должно соответствовать одному байту, т.е. восьми битам. А у нас при переводе десятичных чисел 57 и 48 получились 6-битовые двоичные числа. Следовательно, их оба нужно обязательно дополнить до 8-битовых двумя незначащими нулями слева!

Тогда и запись поразрядной конъюнкции должна иметь вид:

```
  00111001
& ????????
  00110000
```

Теперь рассуждения о значениях битов маски тоже нужно начать с точно определяемых значений 3-го и 4-го слева битов: раз соответствующие биты IP-адреса и адреса сети оба равны 1, то и соответствующие биты маски тоже равны 1.

Далее для 5-го слева бита значение бита маски тоже определяется однозначно: оно должно быть равно 0.

А теперь, вспомнив свойство маски, что в ней сначала до некоторого разряда записаны только единицы, а после этого разряда — только нули, определяем, что до найденных нами единичных битов тоже должны быть единицы, а после найденного нами нуля — нули. То есть маска будет такой: $11110000_2 = 240_{10}$. Это и есть правильный ответ.

Ответ: 240.

11. Четыре черненьких чумазеньких чертенка…

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

Помните, как на физике решали задачу о движении тела, брошенного под заданным углом к горизонту ("задача о пушечном выстреле")? Тогда мы составляли два отдельных уравнения движения по координате x и координате y и рассматривали их в виде системы из двух уравнений. В нынешней задаче нужно сделать то же самое, обозначив неизвестные нам значения переменными: k — количество повторений, x и y — неизвестные смещения в первой повторяемой команде. При этом каждое уравнение приравнивается нулю, так как Чертежник после выполнения всех перемещений вернулся в исходную точку.

1) Перемещение Чертежника по координате x:

$$-2 + k \cdot (x - 2) - 19 = 0.$$

2) Перемещение Чертежника по координате y:

$$3 + k \cdot (y - 1) - 18 = 0.$$

3) Объединяем оба уравнения в систему

$$\begin{cases} -2 + k \cdot (x - 2) - 19 = 0, \\ 3 + k \cdot (y - 1) - 18 = 0; \end{cases} \qquad \begin{cases} k \cdot (x - 2) = 21, \\ k \cdot (y - 1) = 15. \end{cases}$$

4) На первый взгляд данная система нерешаема: два уравнения и три неизвестных. Но обратим внимание — в обоих случаях слева от знака равенства стоят произведения, один из сомножителей в которых один и тот же (k).

Разложим числа справа от знаков равенства на простые сомножители (благо это можно сделать одним-единственным способом): $21 = 3 \cdot 7$; $15 = 3 \cdot 5$.

Заметим: в обоих получившихся произведениях тоже повторяется один и тот же сомножитель — 3. Значит, это и есть k.

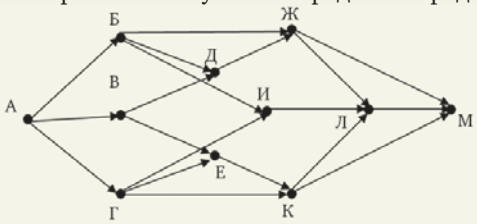
Определив, что $k = 3$, можно при желании (или при необходимости, если это потребуют в условии

задачи) вычислить и значения x и y. Нам же требуется только это значение k.

Ответ: 3.

12. “Эх, дороги” - 2

Дана схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, И, К, Л, М. По каждой дороге можно двигаться только в направлении, указанном стрелкой. Сколько возможно различных путей из города А в город Л?



Решение

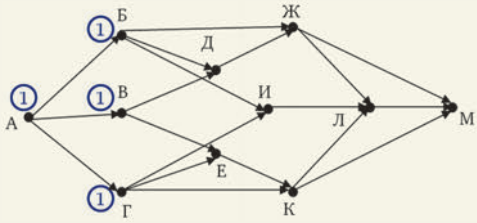
Раньше такие задачи мы решали, строя дерево путей. Но для такого сложного графа (большого количества городов и дорог) вычертить дерево путей очень сложно, оно будет слишком громоздким. Возможен ли другой способ?

Да, возможен. И именно его продемонстрировала в своих работах ученица 11-го “А” класса школы № 1360 Анастасия Ремизова: количество путей, ведущих в каждый город, можно вычислять напрямую.

1) Возле города А записываем единицу. Это — некое “стартовое” значение, поскольку уж один-то путь из А в М есть всегда.

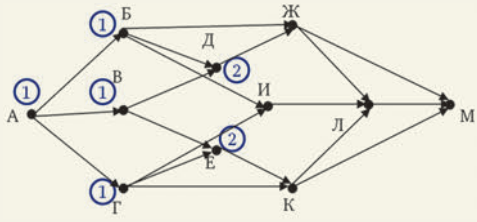
2) Смотрим город Б. В этот узел графа входит только одна стрелка, которая идет от узла А со значением 1. Поэтому возле узла Б тоже записываем единицу.

3) То же с городами (узлами) В и Г — в них по единственной входящей стрелке “переносится” все та же единица из узла А.



4) Смотрим город Д. В него входят две стрелки. Одна идет из узла Б и “несет с собой” оттуда единицу. Вторая же аналогичным способом переносит единицу по стрелке из узла В. Итого в узле Д в сумме получаем значение 2 (1+1).

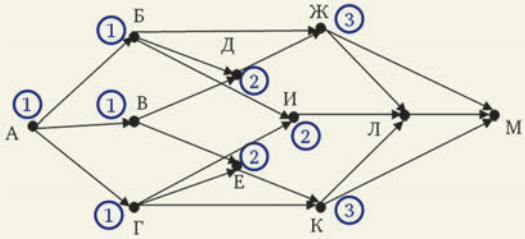
5) То же самое получаем и для узла Е, куда по соответствующим двум стрелкам “приходят” единицы из узлов В и Г.



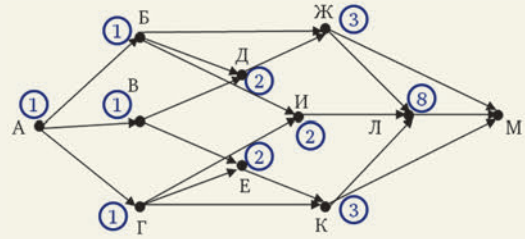
6) В узел Ж тоже входят две стрелки. Одна (из узла Б) “приносит” туда единицу. А вторая (из узла Д) “приносит” уже двойку. Итого в сумме получается 3 (1+2).

7) Для узла К история та же — рядом с ним тоже записываем 3 (1 по стрелке из узла Г плюс 2 по стрелке из узла Е).

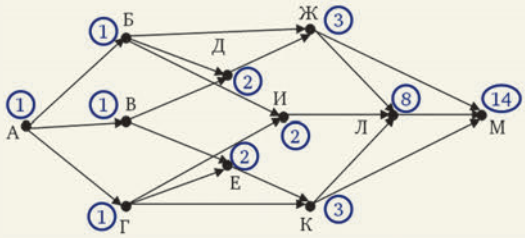
8) Для узла же И две стрелки “принесут” с собой из узлов Б и Г по единице каждая, итого в сумме получаем 2.



9) А вот теперь переходим к самому сложному узлу — Л. В него входят три стрелки. Первая, из узла Ж, “переносит” в Л тройку. Вторая, из узла К, “переносит” тоже тройку. И, наконец, третья, из узла И, “переносит” в Л двойку. В сумме же для Л получаем значение 8 (3+2+3).



10) Остается только узел М. В него приходят тоже три стрелки. Стрелка из узла Ж “приносит” в М тройку. Стрелка из узла К “приносит” в М тоже тройку. А стрелка из узла Л приносит в М восьмерку. В сумме для М получаем 14 (3+3+8).



Объяснение этого решения — гораздо более долгое, чем само решение. Расписать для каждого города соответствующие значения удается буквально за полминуты. Что позволяет сэкономить немало (по сравнению с построением дерева) дефицитного времени для решения других задач.

Ответ: 14.

13. “Ищут пожарные, ищет милиция…”

В языке запросов поискового сервера для обозначения логической операции “ИЛИ” используется символ “|”, а для логической операции “И” — символ “&”.

В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.

Запрос	Кол-во найденных страниц, тыс.
Паскаль & (*Си* & *Бейсик* | *Кобол*)	350
Паскаль & *Кобол*	187
Паскаль & *Си* & *Бейсик* & *Кобол*	48

Какое количество страниц (в тысячах) будет найдено по запросу

Паскаль & *Си* & *Бейсик* ?

Считаем, что все запросы выполнялись почти одновременно и набор страниц, содержащих искомые слова, за это время не изменялся.

Решение

"Традиционное" решение такой задачи — построение кругов Эйлера. Но здесь — целых четыре переменных, а числовых данных явно недостаточно, чтобы выполнить полное решение. Как же быть?

Обратим внимание на запрос *Паскаль* & & (*Си* & *Бейсик* | *Кобол*) и раскроем скобки в этом логическом выражении: *Паскаль* & *Си* & *Бейсик* | *Паскаль* & *Кобол*.

А теперь посмотрим на запрос *Паскаль* & *Си* & & *Бейсик* & *Кобол*. Правда, похоже? И там и там слева стоит "*Паскаль* & *Си* & *Бейсик*". А если сравнить со вторым запросом — *Паскаль* & *Кобол*, то мы увидим: в первом запросе справа от "|" записано "*Паскаль* & *Кобол*" (как во втором запросе), а в третьем запросе правая часть — только "*Кобол*". Но ведь существует "правило поглощения": $A \,\&\, B \,\&\, A = A \,\&\, B$ (повторяющийся операнд A можно не повторять). А значит, верно и обратное: любой операнд можно повторить (записав через тот же знак операции) сколько угодно раз, не меняя значение логического выражения!

Воспользуемся этим и повторим в третьем запросе слово *Паскаль*: *Паскаль* & *Си* & *Бейсик* & *Паскаль* & & *Кобол*. Значение выражения не изменилось, но зато его теперь можно представить в виде: (*Паскаль* & *Си* & *Бейсик*) & (*Паскаль* & *Кобол*).

Итак, четыре наших запроса (включая искомый) теперь выглядят так:

Паскаль & *Си* & *Бейсик* | *Паскаль* & *Кобол*

Паскаль & *Кобол*

(*Паскаль* & *Си* & *Бейсик*) & (*Паскаль* & *Кобол*)

Паскаль & *Си* & *Бейсик*.

Совершенно очевидно, что можно выполнить "макроподстановку":

A = *Паскаль* & *Си* & *Бейсик*,

B = *Паскаль* & *Кобол*.

Тогда наши запросы примут вид:

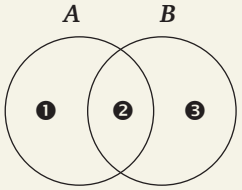
A | ***B*** (350 тыс. стр.),

B (187 тыс. стр.),

A & ***B*** (48 тыс. стр.),

A — искомый.

Теперь мы получили задачу всего с двумя логическими переменными (и, соответственно, с двумя кругами Эйлера):



A | ***B***	❶ + ❷ + ❸	350 тыс. стр.
B	❷ + ❸	187 тыс. стр.
A & ***B***	❷	48 тыс. стр.
A	❶ + ❷	*Требуется найти*

Составляем уравнения:

$$\left.\begin{array}{l} ❶ + ❷ + ❸ = 350; \\ ❷ + ❸ = 187; \\ ❷ = 48. \end{array}\right\} \Rightarrow \left.\begin{array}{l} ❶ + 48 + ❸ = 350; \\ 48 + ❸ = 187; \end{array}\right\} \Rightarrow$$

$$\Rightarrow \left.\begin{array}{l} ❶ + ❸ = 302; \\ ❸ = 139; \end{array}\right\} \Rightarrow ❶ + 139 = 302;$$

Отсюда легко вычислить, что ❶ = 163, а ❶ + ❷ = = 163 + 48 = 211.

Следовательно, по запросу *Паскаль* & *Си* & *Бейсик* будет найдено 211 тыс. стр.

Ответ: 211.

14. Вместо отрезков

Элементами множества X являются натуральные числа. Известно, что выражение

$(n \in \{2,3,6,15,26,30\}) \rightarrow$

$\rightarrow (((n \in \{1,3,6,12,15,35\}) \wedge \neg(n \in X)) \rightarrow$

$\rightarrow \neg(n \in \{2,6,15,26,30\}))$

истинно (принимает значение 1) при любом значении n. Требуется определить наименьшее возможное значение произведения элементов множества X.

Решение

Ранее в ЕГЭ были включены похожие задачи, в которых речь шла о принадлежности x некоторым отрезкам. Теперь же вместо отрезков — дискретные множества. Насколько решение такой "дискретной" задачи аналогично уже знакомому учащимся решению задач с отрезками? Посмотрим…

1) Для упрощения решения заменим предложенные множества чисел буквами:

- $\{2,3,6,15,26,30\} = A$;
- $\{1,3,6,12,15,35\} = B$.

Тогда исходное уравнение можно записать так:

$(n \in A) \rightarrow (((n \in B) \wedge \neg(n \in X)) \rightarrow \neg(n \in A)) = 1$.

2) Точно так же, как мы это делали в задачах с отрезками, заменим утверждения о принадлежности n какому-либо множеству логическими переменными:

- $(n \in A) = a$;
- $(n \in B) = b$;
- $(n \in X) = x$.

Получим выражение в виде:

$$a \to ((b \wedge \neg x) \to \neg a) = 1.$$

3) Теперь попробуем упростить это выражение и заменить операцию следования на "ИЛИ" по правилу $A \to B = \neg A \vee B$:

$a \to (\neg(b \wedge \neg x) \vee \neg a) = 1; \Rightarrow$

$\Rightarrow \neg a \vee (\neg(b \wedge \neg x) \vee \neg a) = 1.$

Применим операцию отрицания к содержимому скобок $(b \wedge \neg x)$:

$\neg a \vee (\neg b \vee x) \vee \neg a) = 1.$

Теперь все операции равноценны и скобки можно убрать:

$\neg a \vee \neg b \vee x \vee \neg a = 1.$

Повторяющийся аргумент $\neg a$ можно "поглотить":

$\neg a \vee \neg b \vee x = 1.$

4) А теперь вернемся к исходной записи переменных — аргументов, учитывая, что операция "НЕ" в данном случае означает непринадлежность тому или иному множеству:

$(n \notin A) \vee (n \notin B) \vee (n \in X) = 1$

или

$(n \notin \{2,3,6,15,26,30\}) \vee$

$\vee (n \notin \{1,3,6,12,15,35\}) \vee (n \in X) = 1.$

5) Для решения задачи (определения элементов множества X) "в лоб" нужно последовательно перебирать различные натуральные значения n. Впрочем, достаточно будет только начать это делать — искомую закономерность можно будет уловить уже на первых шагах.

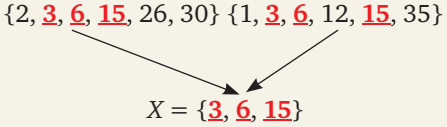
• $n = 1$. Используется логическая операция ИЛИ, значит, для получения результата "ИСТИНА" достаточно истинности хотя бы одного из аргументов. Принадлежит ли число 1 первому множеству ($\{2,3,6,15,26,30\}$)? Нет. Значит, условие $(n \notin \{2,3,6,15,26,30\})$ истинно. Этого достаточно для получения общего результата "ИСТИНА", поэтому значения всех остальных аргументов нам безразличны.

• $n = 2$. Это число принадлежит первому множеству $\{2,3,6,15,26,30\}$, значит, соответствующее условие ложно. Тогда смотрим второй аргумент: $(n \in \{1,3,6,12,15,35\})$. Этому множеству число 2 не принадлежит, значит, данное условие истинно, и этого нам достаточно.

• $n = 3$. Замечаем, что это число принадлежит и первому множеству ($\{2,3,6,15,26,30\}$), и второму ($\{1,3,6,12,15,35\}$). Следовательно, ложны и первое, и второе условия. Что в этом случае может обеспечить истинность результата? Правильно: только истинность третьего условия! Оно записано в виде: $(n \in X)$. Значит, чтобы это условие было истинным, надо включить в состав множества X число 3.

В принципе, чтобы понять закономерность, этих трех примеров достаточно. Мы видим, что если какое-либо натуральное число n не входит хотя бы в одно из двух заданных нам множеств, то этого уже достаточно. И только если n имеется в обоих заданных множествах, это число n требуется включить в искомое множество X: ведь нам требуется в итоге получить наименьшее возможное произведение его элементов, а значит, включать в множество X какие-то "ненужные" числа нам не следует.

6) Теперь легко, сравнивая между собой элементы первого и второго множеств, выписать все множество X:



$X = \{3, 6, 15\}$

7) Вычисляем произведение элементов множества X:

$3 \cdot 6 \cdot 15 = 270.$

Ответ: 270.

15. Еще одна сумма элементов массива

В программе описан одномерный целочисленный массив. Ниже представлен фрагмент программы, обрабатывающей этот массив:

```
s := 0;
n := 8;
for i := 0 to n - 4 do begin
  s := s + A[i] - A[i + 3]
end;
```

До выполнения этого фрагмента в массиве находились четырехзначные нечетные натуральные числа. Какое наименьшее значение может иметь переменная s после выполнения данной программы?

Решение

Такая задача нам уже встречалась в предыдущем тренинге. Здесь условие лишь немного усложнено, но принцип решения — тот же.

1) Расписываем вычисление суммы в виде одной строки, помня, что значение i меняется в цикле от 0 до 6 (т.е. до 10 – 4):

$s = 0 + A[0] - A[3] + A[1] - A[4] + A[2] - A[5] +$
$+ A[3] - A[6] + A[4] - A[7] + A[5] - A[8] +$
$+ A[6] - A[9].$

2) Теперь сокращаем одинаковые переменные (элементы массива), записанные со знаками "плюс" и "минус":

$s = A[0] - A[3] + A[1] - A[4] + A[2] - A[5] +$
$+ A[3] - A[6] + A[4] - A[7] + A[5] - A[8] +$
$+ A[6] - A[9].$

Получаем:

$s = A[0] + A[1] + A[2] - A[7] - A[8] - A[9].$

3) Получаемая сумма (разность) должна, по условию, иметь наименьшее значение. А сами элементы массива — это четырехзначные нечетные натуральные числа, т.е. числа в диапазоне от 1001 до 9999. Следовательно, те элементы массива, которые записаны в выражении со знаком "плюс",

должны быть наименьшими из возможных, а элементы массива со знаком “минус” должны быть наибольшими из возможных. Значит, нужно выбрать их следующими: $A[0] = A[1] = A[2] = 1001$, а $A[7] = A[8] = A[9] = 9999$.

4) Вычисляем значение s при этих значениях элементов массива:

$s = 1001 + 1001 + 1001 - 9999 - 9999 - 9999 =$
$= 3 \cdot (1001 - 9999) = -26\,994$.

Ответ: –26 994.

16. И снова числа

Получив на вход число x, программа печатает два числа — a и b. Определите наименьшее из чисел, при вводе которого алгоритм печатает сначала 2, а потом 357.

```
var x, a, b: integer;
 begin
  readln(x);
  a := 0; b := 0;
  while x > 0 do
   begin
    a := a + 1;
    b := b + (x mod 1000);
    x := x div 1000;
   end;
  writeln(a); write(b);
end.
```

Решение

И такая задача тоже уже встречалась. Но в этом ее варианте есть небольшой нюанс, который существенно меняет решение.

1) Проанализировав алгоритм, видим, что переменная a — это счетчик проходов цикла. А в переменной b накапливается сумма… чего? Раньше в такой задаче имелись операторы

```
b := b + (x mod 10);
x := x div 10;
```

и это означало вычисление суммы цифр. Теперь же вместо константы 10 записана константа 1000, а значит, из числа выделяются сразу ***тройки*** цифр.

2) Поскольку в качестве значения переменной a выводится число 2, цикл выполнялся два раза. Следовательно, исходное число могло состоять из четырех, пяти или шести цифр, — только тогда мы получаем из него две триады цифр, одна из которых (левая), возможно, неполная. А поскольку нам требуется, чтобы число было наименьшим, левая “триада” должна состоять из одной цифры.

3) Обозначим неизвестные цифры исходного числа как a, b, c и d. Тогда первый проход цикла отделит цифры bcd, т.е. число $100 \cdot b + 10 \cdot c + d$. А второй проход цикла добавит к сумме однозначное число (цифру) a.

4) Какой может быть эта единственная цифра a, чтобы число было наименьшим? Очевидно, $a = 1$. Тогда, зная, что итоговая сумма равна 357, сразу получаем, что $bcd = 356$. А все первоначальное число x тогда равно 1356.

Ответ: 1356.

17. Опять логика

Сколько существует различных наборов значений логических переменных $a1, a2, \ldots, a5, b1, b2, \ldots, b5, c1, c2, \ldots, c6$, которые удовлетворяют перечисленным уравнениям?

$(a1 \to a2) \wedge (a2 \to a3) \wedge (a3 \to a4) \wedge (a4 \to a5) = 1$

$(b1 \to b2) \wedge (b2 \to b3) \wedge (b3 \to b4) \wedge (b4 \to b5) = 1$

$(c1 \to c2) \wedge (c2 \to c3) \wedge (c3 \to c4) \wedge (c4 \to c5) = 1$

$a1 \wedge b2 \wedge c3 = 0$.

В качестве ответа нужно указать количество таких наборов переменных.

Решение

1) Анализируем первое уравнение:

$(a1 \to a2) \wedge (a2 \to a3) \wedge (a3 \to a4) \wedge (a4 \to a5) = 1$.

Начинаем составлять таблицу возможных значений переменных $a1, a2, \ldots, a5$. При этом учитываем, что при использовании логической операции “И” результат 1 обеспечивается только единичными значениями всех ее аргументов, а также помним, что операция следования истинна в трех случаях: $1 \to 1$, $0 \to 1$ и $0 \to 0$. Поэтому если в каждом следовании первая переменная равна единице, то из нее может следовать только единица, а из нуля может следовать как единица, так и ноль. В результате получаем следующую таблицу значений переменных:

$a1$	$a2$	$a3$	$a4$	$a5$
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

2) Второе уравнение полностью аналогично первому, только имена переменных — другие. Поэтому таблица значений переменных будет такая же. И то же самое можно сказать про третье уравнение. Получаем:

$b1$	$b2$	$b3$	$b4$	$b5$
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

$c1$	$c2$	$c3$	$c4$	$c5$
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

3) Анализируем четвертое уравнение. Оно является “*ключевым*”:

$$a1 \wedge b2 \wedge c3 = 0.$$

Строим для него таблицу истинности:

$a1$	$b2$	$c3$	Результат
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Из нее нам требуются все строки, кроме последней.

4) Подсчитаем теперь количество значений переменных, соответствующие таблице истинности четвертого уравнения. Для этого в таблицах значений переменных первых трех уравнений выделяем (например, цветом фона) столбцы, соответствующие переменным $a1$, $b2$ и $c3$:

$a1$	$a2$	$a3$	$a4$	$a5$
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

$b1$	$b2$	$b3$	$b4$	$b5$
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

$c1$	$c2$	$c3$	$c4$	$c5$
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

И теперь для каждой строки таблицы значений четвертого уравнения перемножаем друг на друга количество нулей и единиц соответственно, содержащихся в выделенном столбце для каждой из переменных.

Например, в первой строке у нас записаны значения:

$a1$	$b2$	$c3$
0	0	0

Поэтому из трех предыдущих таблиц мы:

• из столбца для $a1$ получаем количество **нулей**, равное 5;

• из столбца для $b2$ получаем количество **нулей**, равное 4;

• из столбца для $c3$ получаем количество **нулей**, равное 3,

и записываем в сводную таблицу произведение: $5 \cdot 4 \cdot 3 = 60$.

Для второй строки имеем значения:

$a1$	$b2$	$c3$
0	0	1

Поэтому из трех предыдущих таблиц:

• из столбца для $a1$ получаем количество **нулей**, равное 5;

• из столбца для $b2$ получаем количество **нулей**, равное 4;

• а вот из столбца для $c3$ надо определить уже количество не нулей, а **единиц**, которое, правда, тоже равно 3.

В результате в сводную таблицу записывается такое же произведение: $5 \cdot 4 \cdot 3 = 60$.

И так поступаем для каждой строки сводной таблицы: по заданным в ней значениям $a1$, $b2$ и $c3$ определяем, надо ли в соответствующих столбцах искать количества нулей или же единиц, определяем эти количества и перемножаем их. А потом суммируем все полученные в каждой строке сводной таблицы произведения. В результате получаем следующую таблицу:

$a1$	$b2$	$c3$	Кол-во наборов переменных
0	0	0	$5 \cdot 4 \cdot 3 = 60$
0	0	1	$5 \cdot 4 \cdot 3 = 60$
0	1	0	$5 \cdot 2 \cdot 3 = 30$
0	1	1	$5 \cdot 2 \cdot 3 = 30$
1	0	0	$1 \cdot 4 \cdot 3 = 12$
1	0	1	$1 \cdot 4 \cdot 3 = 12$
1	1	0	$1 \cdot 2 \cdot 3 = 6$
Итого:			**$60 + 60 + 30 + 30 + 12 + 12 + 6 = 210$**

Ответ: 210.

Что же касается заданий третьей группы (в предыдущих версиях ЕГЭ — “задачи группы С”), то первые две из них не вызвали у учащихся особых трудностей. Первая задача (на поиск ошибки в программе) и вторая (написание программы) вполне решаемы при условии достаточной подготовки школьников по программированию. Третью задачу (на теорию игр “в камешки”) наши ученики тоже решили практически все. А вот четвертая задача на разработку программы для обработки наборов числовых данных, оптимальной по времени выполнения и затратам памяти, оказалась, конечно, по силам только двум наиболее сильным ученикам. Но это и понятно, задачи “С4” — это настоящий “высший пилотаж” программирования. А потому они заслуживают отдельного разговора, к которому авторы предполагают вернуться в одном из последующих номеров журнала.