Всероссийская олимпиада школьников по информатике
Вологодская область
II (муниципальный) этап
2018-2019 учебный год
7 - 8 классы
Методические рекомендации по разбору задач

Задача 1. Робот (100 баллов)

Задача решается из очевидных соображений, но требует аккуратности при подсчёте ответа на 3-й вопрос.

Верные ответы: 8 С3 25 2.

Задача 2. Слова (100 баллов)

Ответы на первые два вопроса можно получить, просто выписав и сосчитав все подходящие слова.

Для ответа на третий вопрос заметим, что искомое количество – это количество слов длины 1, длины 2, … , длины m, количества которых равны 2^1, 2^2, …, 2^m. Ответом будет $2^1 + 2^2 + \ldots + 2^m$. Его можно сосчитать непосредственно по этой формуле, либо преобразовать к более простой форме: $2^{m+1}-2$.

Ответ на 4-й вопрос можно получить так. Вначале добавим к ответу 2 – это количество слов длин 1. Теперь пусть c_aa[i], c_ax[i], c_xa[i], c_xx[i] – количества слов длины *i*, которые кончаются на соответствующую пару букв. Заметим, что при переходе к длине *i* следующие значения можно получить из предыдущих:

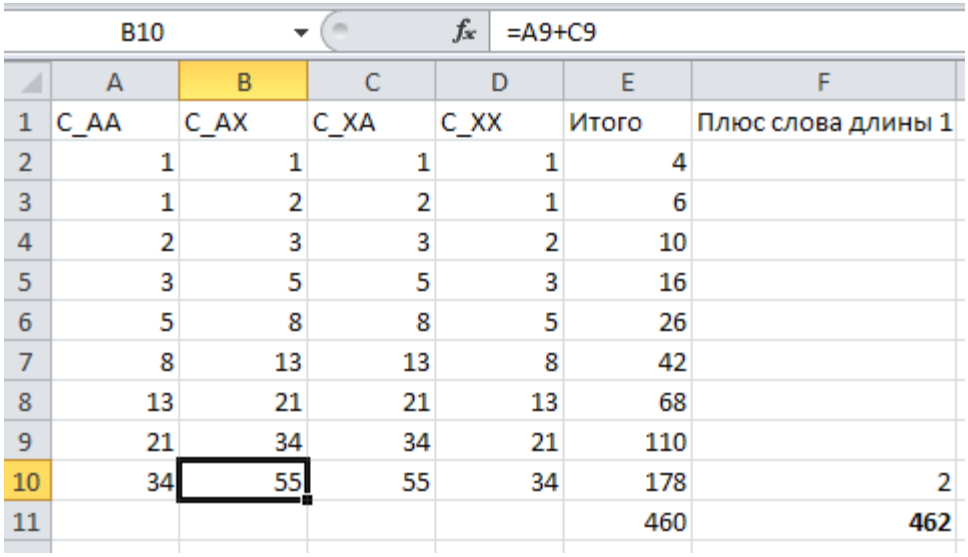
c_aa[i] = c_xa[i-1]

c_ax[i] = c_aa[i-1] + c_xa[i-1]

c_xa[i] = c_ax[i-1] + c_xx[i-1]

c_xx[i] = c_ax[i-1]

Последовательно сосчитаем данные значения для i = 3, …, m и сложим все значения – это и будет ответ. Разумеется, нет необходимости делать это вручную – можно например, воспользоваться программой для работы с электронными таблицами (MS Excel или аналогичной) или написать свою программу. Пример, как выглядит расчёт в Excel:



B10 =A9+C9

	A	B	C	D	E	F
1	C_AA	C_AX	C_XA	C_XX	Итого	Плюс слова длины 1
2	1	1	1	1	4	
3	1	2	2	1	6	
4	2	3	3	2	10	
5	3	5	5	3	16	
6	5	8	8	5	26	
7	8	13	13	8	42	
8	13	21	21	13	68	
9	21	34	34	21	110	
10	34	55	55	34	178	2
11					460	**462**

Ответом будет число 462. Заметим, что описанный способ решения является частным случаем метода динамического программирования.

Интересно отметить, что столбцы в таблице – это последовательные числа Фибоначчи. Тогда ответ на 4-й вопрос после несложных преобразований можно также найти по формуле $2\cdot(f_5+f_7+f_9+2\cdot f_{11})+2 = 2\cdot(5+13+34+2\cdot 89)+2 = 462$.

Верные ответы на все вопросы: 30 22 2046 462.

Задача 3. Алгоритм (100 баллов)

Ответить на вопросы задачи можно двумя путями. Первый способ – проанализировать данный алгоритм и понять, как именно он работает.

Второй способ – переписать алгоритм в виде процедуры на любом языке программирования, вызывать её в цикле с разными значениями *x* и получить ответы экспериментально.

Верные ответы: 3 16 999 288.

Задача 4. Каждый третий (100 баллов)

Первую подзадачу можно решить «в лоб» – выполняя ровно то, что написано в условии. При этом допускается неэффективная реализация – например, каждый раз удалять символ со сдвигом всей оставшейся части строки, что приведёт к квадратичному времени работы.

Вторую подзадачу необходимо решать более эффективно, чем за квадрат. Простейший способ это сделать – не удалять символы в исходной строке, а добавлять оставшиеся символы в конец новой строки. Разумеется, нужно, чтобы операция добавления в конец строки работала за O(1): например, на языке Java следует вместо класса *String* использовать класс *StringBuilder* с функцией *append*.

Пример решения на C++:

```
#include <bits/stdc++.h>

int main() {
  std::string s;
  std::cin >> s;
  while (s.length() > 2) {
    std::string t;
    int len = s.length();
    for(int i = 2; i < len; i += 3)
      s[i] = '$'; // помечаем для удаления
    for(int i = 0; i < len; i ++)
      if (s[i] != '$')
        t.push_back(s[i]);
     std::reverse(t.begin(), t.end());
     s.swap(t);
  }
  std::cout << std::min(s[0], s[1]) << std::max(s[0], s[1]);
}
```

Задача 5. Код (100 баллов)

Для решения данной задачи достаточно перебрать 24 возможных перестановки четырёх частей кода и каждый раз проверить, что полученная строка является корректным кодом.

Немного усложняет задачу запрет одинаковых кодов в ответе – необходимо проверять, не был ли очередной код уже выведен ранее.

Пример решения на C++:

```
#include <bits/stdc++.h>
#include <string>
#include <vector>
#include <algorithm>
#include <regex>
#include <set>

int main() {
 std::string v[4];
 std::cin >> v[0] >> v[1] >> v[2] >> v[3];
 std::sort(v, v + 4);
 std::regex code("[0-9][0-9][.][0-9][0-9][.][0-9][0-9][.][0-9][0-9]");
 std::set<std::string> used;
 for(;;) {
    std::string s = v[0] + v[1] + v[2] + v[3];
    if (std::regex_match(s, code) && used.count(s) == 0) {
      std::cout << s << '\n';
```

```
      used.insert(s);
    }
    if (!std::next_permutation(v, v + 4)) break;
}
}
```

В данном решении для уменьшения кода используются возможности современного C++ (регулярные выражения, библиотека STL). Однако, не представляет особой сложности написать решение, зная лишь базовые элементы языка – условия, циклы, массивы и строки.

В материалах олимпиады можно найти также примеры и других решений задач.