Всероссийская олимпиада школьников по информатике
Вологодская область
II (муниципальный) этап
2020-2021 учебный год
7 - 8 классы
Методические рекомендации по разбору задач

Задача 1. Прямоугольник (100 баллов)

Максимальную площадь даёт прямоугольник, как можно более близкий к квадрату. Первую его сторону можно найти как N/4 (напомним, что деление у нас целочисленное).

Вторая сторона иногда может получаться на 1 больше первой. Это случается, когда остаток от деления N на 4 равен 2 или 3. Добавим к N двойку перед делением, и тогда, если (N mod 4) ≥ 2, то результат деления станет на единицу больше – это нам и нужно. Итак, вторая сторона найдётся как (N+2)/4.

Таким образом, площадь прямоугольника найдётся так:

N/4*((N+2)/4)

Внешние скобки здесь важны – без них формула станет неправильной.

Ещё одно возможное решение: (N/4)*(N/2-N/4)

Задача 2. Оформитель текста (100 баллов)

Вопрос 1. Если для каждого предложения 5 вариантов гарнитур, то ответом будет 5^3=**125**.

Вопрос 2**.** Обозначим через $C(n,m)$ число сочетаний из n по m. Гарнитура ‘Segoe Print’ используется ровно 2 раза – это C(4,2) способа. Остальные предложения оформляем любой гарнитурой из четырёх оставшихся:

$C(4,2) \cdot 4^2 = 6 \cdot 16 =$ **96**.

Вопрос 3. Гарнитурой Arial можно оформить одно предложение, два или все три – это $C(3,1)$, $C(3,2)$ и $C(3,3)$. Остальные предложения оформляем любой гарнитурой из четырёх оставшихся:

$C(3,1)\cdot 4^2 + C(3,2)\cdot 4 + C(3,3) = 3\cdot 16 + 3\cdot 4 + 1 =$ **61**.

Вопрос 4. Приведём три различных способа решения.

Решение 1. Посчитаем количество всех вариантов и вычтем варианты с 5 и 4 гарнитурами.

Всего вариантов 5^5.

5 гарнитур: ответом является количество перестановок из 5 элементов, потому что каждое предложение выделяется своей гарнитурой. То есть, 5!

4 гарнитуры: одна гарнитура используется дважды, ещё 3 гарнитуры используются по одному разу. Выбираем две позиции под гарнитуру, используемую дважды – это 5·4/2, далее выбираем гарнитуру, использующуюся дважды (5 вариантов), далее выбираем гарнитуру для первого слева направо предложения, у которого не фиксирована гарнитура (4 варианта) – и для остальных двух (3 и 2). Перемножаем: 5·4/2·5·4·3·2

Итого ответ равен 5^5 - 5! - 5·4/2·5·4·3·2=**1805.**

Решение 2. Посчитаем отдельно случаи, когда используем ровно одну гарнитуру, ровно две и ровно три.

1). Если используем одну гарнитуру, то у нас 5 способов её выбрать.

2). Если используем две гарнитуры, то у нас C(5, 2) = 10 способов выбора пары гарнитур. Теперь как их распределить по предложениям? Для 5 предложений существует 2^5-2=30 способов разбиения на два непустых подмножества. В первое множество ставим первую гарнитуру, во второе - вторую. Получаем $30 \cdot 10 = 300$ способов.

3). Если используем три гарнитуры, то у нас C(5,3) = 10 способов выбора тройки гарнитур. Для 5 предложений существует 3^5 - 3 - 3·(2^5 - 2) = 150 способов разбиения на три непустых подмножества. Пояснение: все варианты разбиения на три подмножества хотя бы с одним пустым можно схематично изобразить так (числа – это размер подмножеств):

0 0 5
0 5 0
5 0 0
0 x y
x 0 y
x y 0

Здесь x и y – какое-то разбиение пяти предложений на два непустых подмножества, это количество равно 2^5-2 (мы его уже использовали во втором пункте).

Итого 150·10 = 1500 способов.

Складываем количества из всех трёх пунктов вместе и получаем ответ: 5 + 300 + 1500 = **1805.**

Решение 3. При наличии навыков программирования проще всего получить ответ, написав программу перебора вариантов. Пример такой программы на языке C++:

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

int n = 5, gmax = 3, cnt = 0;

int used[6], nused = 0;

void bt(int i) {
  if (i == n + 1) {
    cnt++;
    return;
  }
  for (int g = 1; g <= 5; g++) {
    if (nused < gmax || used[g] != 0) {
      if (used[g] == 0) nused++;
      used[g]++;
      bt(i + 1);
      used[g]--;
      if (used[g] == 0) nused--;
    }
  }
}

int main() {
  bt(1);
  std::cout << cnt;
}
```

Все ответы к задаче: **125 96 61 1805**.

Задача 3. Пятнадцать (100 баллов)

Чтобы число делилось на 15, оно должно оканчиваться на 0 или на 5, а сумма всех его цифр должна делиться на 3. Если это не так, выводим -1.

В противном случае сортируем все цифры по убыванию. Если число стало заканчиваться не на ноль, то убираем одну пятёрку со своего места и ставим в конец.

Для получения максимально эффективного и притом простого решения можно написать сортировку подсчётом: создаём массив счётчиков на 10 элементов, проходимся один раз по строке с числом и запоминаем, сколько раз встретилась каждая цифра. Затем идём от 9

до 0 и выводим каждую цифру столько раз, сколько она встречалась (перед этим, возможно, убрав одну пятёрку). Пример решения на Python:

```
s = input()
a = [0] * 10
sum = 0
for c in s:
  d = ord(c) - ord('0')
  a[d] += 1
  sum += d
if sum % 3 != 0 or a[0] == 0 and a[5] == 0:
  print(-1)
else:
  if a[0] == 0:
    a[5] -= 1
  for i in range(9, -1, -1):
    for j in range(a[i]):
      print(i, end='')
  if a[0] == 0:
    print('5')
```

Можно вместо этого использовать и готовую функцию сортировки, которая сейчас имеется в большинстве языков программирования.

Для решения первой подзадачи можно реализовать какой-нибудь свой медленный способ сортировки – например, пузырьком.

Также могут пройти первую подзадачу, но не пройти вторую решения, неэффективно работающие со строками – например, многократно перезаписывающие длинную строку и тому подобное.

Задача 4. Палиндром (100 баллов)

Посчитаем, сколько букв встретилось нечётное число раз. Пусть это число равно *D*. Тогда ответом будет *D* div 2.

Пояснение: нам нужно, чтобы осталось не более одной буквы, которая встречается нечётное число раз. Если, например, нечётное число раз встречались буквы ‘a’, ‘b’, ‘c’, ‘d’,’e’, то ‘a’ меняем на ‘b’, ‘c’ меняем на ‘d’, а букву ‘e’ в результирующем палиндроме ставим в центр. Пример решения на языке C++:

```
#include<bits/stdc++.h>
using namespace std;

int minReplaces(string s) {
```

```
    std::vector<int> cnt(128);
    for(char c : s) {
        cnt[c]++;
    }
    int odd = 0;
    for (int i = 0; i < 128; i++) {
        odd += cnt[i] & 1;
    }
    return odd / 2;
}

int main() {
  string s;
  cin >> s;
  cout << minReplaces(s);
}
```

Задача 5. Вычитания (100 баллов)

Для решения первой подзадачи можно моделировать описанный процесс – то есть, перебирать пару чисел, проверять, не встречалась ли она, и если нет, то добавлять в массив.

Для решения второй задачи можно применить такой же подход, только реализованный более эффективно. Например, можно создать булевский массив *used*, где *used*[*i*] = *true*, если такое число уже встречалось. Тогда такая проверка будет выполняться всего за пару операций. Пример решения второй подзадачи на C++:

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

using namespace std;

int main(){
  int n;
  cin >> n;
  vector<int> a(n);
  vector<bool> used(10001, false);
  for (int i = 0; i < n; i++) {
      cin >> a[i];
      used[a[i]] = true;
  }
  for(;;) {
    bool found = false;
    int n = (int)a.size();
    for (int i = 0; i < n; i++) {
      for (int j = i + 1; j < n; j++) {
        int diff = abs(a[i] - a[j]);
```

```
        if (!used[diff]) {
          used[diff] = true;
          a.push_back(diff);
          found = true;
        }
      }
    }
    if (!found) break;
  }
  cout << a.size();
}
```

Полное решение выглядит так. Заметим, что мы всегда можем получить наибольший общий делитель всех чисел. Взяв первую пару чисел, с помощью алгоритма Евклида (с вычитаниями) мы можем получить их НОД. Взяв его и третье число, мы можем получить НОД уже первых трёх чисел, и так далее.

Раз мы можем получить НОД всех чисел, то далее можем получить и любое число, делящееся на этот НОД и не превышающее максимальное из входных чисел. Никаких других чисел получить нельзя, так как если два числа делятся на НОД, то их разность тоже будет делиться.

Таким образом, ответ – это частное от деления максимального входного числа на НОД всех чисел. Конечно, НОД надо считать эффективно – используя алгоритм Евклида с остатками, а не с вычитаниями. Пример решения на языке C++:

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

using namespace std;

int gcd(int a, int b) {
  while (b != 0) {
    int c = a % b;
    a = b;
    b = c;
  }
  return a;
}

int main() {
  int n;
  cin >> n;
  vector<int> a(n);
  for (int i = 0; i < n; i++) {
      cin >> a[i];
```

```
    }
    int g = a[0];
    for (int i = 1; i < n; i++) {
        g = gcd(g, a[i]);
    }
    int ans = a[0] / g;
    for (int i = 1; i < n; i++){
        ans = max(ans, a[i] / g);
    }
    cout << ans;
}
```