Вологодская область
IV Областная олимпиада школьников по информатике
2019-2020 учебный год
9-10 классы
Заключительный тур

Разбор задач

Авторы задач:

Андрианов Игорь Александрович, доцент кафедры АВТ ВоГУ
Метляхина Валентина Станиславовна, учитель ВМЛ
Свердлов Сергей Залманович, доцент кафедры ПМ ВоГУ

Задача 1 – Тир

Пусть Вася *x* раз попал в цель. Значит, он получил M * x дополнительных выстрелов. То есть K+ M * x = N, отсюда:

x = (N – K) / M, если (N – K) делится на M

x = -1, если не делится

Самое сложное: как записать это одной формулой?

Сначала рассмотрим вспомогательную подзадачу: даны натуральные A и B. Если A делится на B, то получить 1, иначе получить 0.

Решение. Если A делится на B, то A / B * B = A. Иначе A / B * B < A.
Тогда A / B * B / A = 1, если A делится на B, и равно 0, если нет.
Правда, при A = 0 будет деление на ноль, но это легко исправить:

(A / B * B + 1) / (A + 1)

Теперь несложно получить ответ для нашей задачи:

*((N - K) / M * M + 1) / (N – K + 1) * ((N - K) / M) + ((N - K) / M * M + 1) / (N – K + 1) -1*

*После упрощения: ((N - K) / M * M + 1) / (N – K + 1) * ((N – K) / M + 1) - 1*

Альтернативное решение:

*(N - K) / M + (((N – K – 1 + M) % M + 1) / M - 1) * ((N - K) / M + 1)*

Задача 2 – Забор

Сосчитаем количество плохих досок и вычтем его из n.
Какие доски становились плохими?
`1 4 7 10 13 16 19 22 25 28 31 34...` (шаг = 3)
`2 6 10 14 18 22 26 30 34 38 42...` (шаг = 4)
`3 10 17 24 31 38 ...` (шаг = 7)

С шагом 3 было испорчено (n-1) / 3+1 досок (где знак '/ ' означает деление нацело)
С шагом 4 было испорчено (n-2) / 4+1 досок
С шагом 7 было испорчено (n-3) / 7+1 досок
Равен ли ответ сумме этих трёх величин? Нет, так как некоторые доски посчитаются больше одного раза (они были испорчены разными способами).

Сколько досок испорчено одновременно первым и вторым способом?
Начиная с 10-й, и с шагом НОК(3,4) = 12, то есть доски №10, 22, 34 и так далее.
Их количество равно (n - 10) / 12 + 1.
Аналогично, вторым и третьим способом: (n - 10) / 28+1
Аналогично, первым и третьим способом: (n - 10) / 21+1

И учтём ещё доски, испорченные всеми тремя способами: (n - 10) / 84 + 1

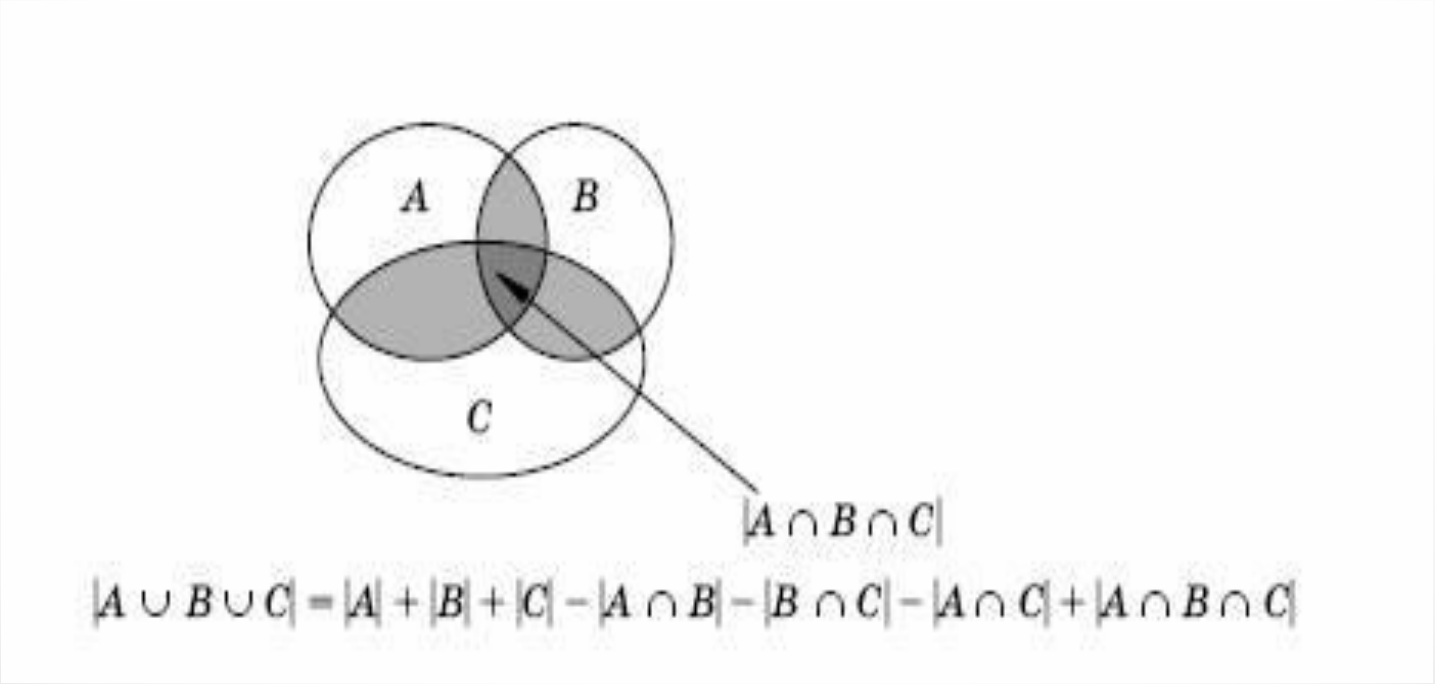
Задача 2 – Забор (продолжение)

По формуле включений-исключений получаем число плохих досок (некоторые +1 и -1 посокращались):

(n-1)/3 + (n-2)/4 + (n-3)/7 – (n-10)/12 - (n-10)/28 - (n-10)/21 + (n-10)/84 + 1

Вычитаем это число из *n* и получаем ответ.

Иллюстрация к формуле включений-исключений:



Задача 3 – Игра

Подзадача 1. Можно написать перебор или попытаться заранее просчитать все варианты вручную.

Подзадача 2. Используем динамическое программирование.
Пусть win[i] = 1, если при i пустых клетках текущий игрок может выиграть, и win[i] = 0, если выигрышных ходов нет.

Заметим, что win[i] = 1 тогда и только тогда, когда соперник проиграет после этого хода, то есть
(win[i-1] = 0) или (win[i-2] = 0) или (win[i-3] = 0) или (win[i-8] = 0).

В цикле заполним массив win до позиции $N * M$ и получим ответ.

Подзадача 3. Возьмём решение предыдущей подзадачи и посмотрим на массив win:
11101111011101111011011110…
Легко видеть, что значения в нём повторяются с периодом 9.
Отсюда легко построить решение, работающее для $N * M$ до 10^{18}.

Задача 3 – Игра (продолжение).

Пример решения на C++ (вместе с исследованием):

```cpp
#include <cstdio>
#include <vector>
#include <cstdlib>

void research() {
    std::vector<bool> wins(100, false);
    for (int i = 1; i < (int)wins.size(); i++) {
        wins[i] =
            (i >= 1 && !wins[i - 1]) ||
            (i >= 2 && !wins[i - 2]) ||
            (i >= 3 && !wins[i - 3]) ||
            (i >= 8 && !wins[i - 8]);
        printf("%d %d\n", i, (int)wins[i]);
    }
    std::exit(0);
}

int main() {
    // research();
    int size1, size2;
    scanf("%d %d", &size1, &size2);
    long long area = 1LL * size1 * size2;
    if (area % 9 == 0 || area % 9 == 4) {
        printf("2");
    } else {
        printf("1");
    }
    return 0;
}
```

Задача 4 - «Разбиение массива»

Подзадача 1. Можно перебрать две точки разбиения и для каждого варианта посчитать суммы подмассивов. Сложность – $O(n^3)$.

Подзадача 2. Заранее сосчитаем суммы для каждого префикса массива ($p[i] = a[1]+a[2]+\ldots+a[i]$). Аналогично сосчитаем суммы для каждого суффикса. Также найдём сумму всех элементов массива.
Тогда сложность предыдущего решения снижается до $O(n^2)$.

Подзадача 3. Вначале сосчитаем сумму S всех элементов. Если она не делится на 3, решения нет.
Пойдём циклом по массиву и будем накапливать сумму пройденных элементов. Если в какой-то момент сумма равна S/3, увеличиваем счётчик cnt. Если сумма равна 2*S/3, добавляем cnt к ответу.
Частный случай здесь – когда массив состоит из одних нулей. Этот случай рассматриваем отдельно.

Задача 4 - Разбиение массива (продолжение)

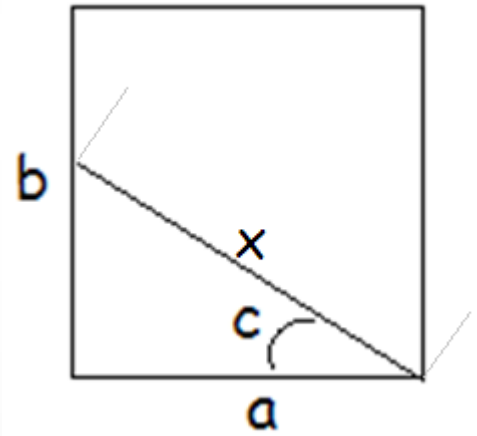
Пример решения на Python 3:

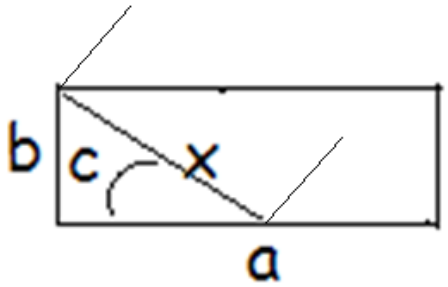
```
n = int(input())
a = list(map(int, input().split()))
s = sum(a)
if s % 3 != 0:
    print(0)
elif s == 0:
    print(n * (n - 1) // 2 - n + 1)
else:
    ans = cnt = p = 0
    for i in range(n):
        p += a[i]
        if p == s // 3:
            cnt += 1
        elif p == s // 3 * 2:
            ans += cnt
    print(ans)
```

Задача 5 – Поворот фотографии

Заметим, что всегда существует решение, когда фотография касается хотя бы трёх сторон рамки (если нет, то фотографию можно либо подвинуть, либо расширить). Пусть фотография касается нижней и левой сторон рамки.

Воспользуемся троичным поиском по стороне фотографии x.
Вначале определим отрезок поиска. Насколько длинной может быть x?
Для «высокой» рамки (рис. слева) xmax = a / cos(c).
Для «низкой» рамки (рис. справа) xmax = b / sin(c).





То есть, отрезок поиска: left = 0, right = min(a / cos(c), b / sin(c))

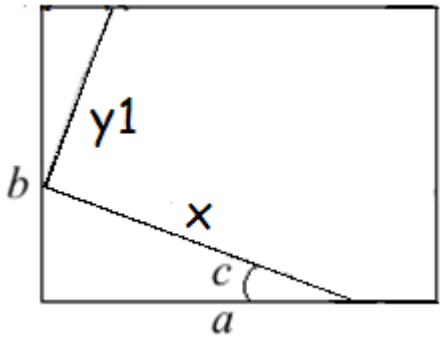
С помощью троичного поиска подберём значение x с нужной точностью (https://neerc.ifmo.ru/wiki/index.php?title=Троичный_поиск)

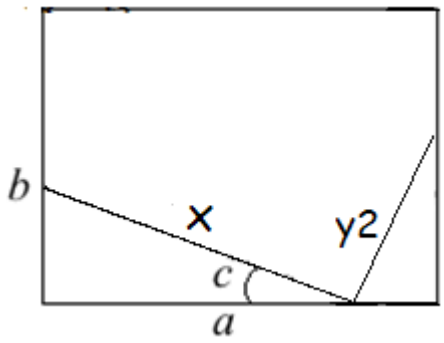
Задача 5 – Поворот фотографии (продолжение)

Осталось понять, как для очередного значения первой стороны фотографии *x* найти вторую сторону *y*.

Для этого попробуем продолжить левую сторону фотографии до пересечения с верхней стороной рамки (рис. слева) – это даст длину y1 = (b – x * sin(c)) / cos(c)).

А также попробуем продолжить правую сторону фотографии до пересечения с правой стороной рамки (рис. справа) – это даст длину y2 = (a – x * cos(c)) / cos(c)).





Тогда y = min(y1, y2), а площадь равна x * y.

У данной задачи существует и аналитическое решение: нужно посмотреть, при каком условии y1 < y2, а когда наоборот. В каждом из случаев находим производную от площади, приравниваем её к нулю и решаем полученное уравнение.

Задача 5 – Поворот фотографии (продолжение)

Пример численного решения на C++:

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

double place(double a, double b, double c, double x)  {
  double d1 = (b - x * sin(c)) / cos(c);
  double d2 = (a - x * cos(c)) / sin(c);
  return x * std::min(d1, d2);
}

int main() {
  double a, b, c;
  std::cin >> a >> b >> c;
  c = c / 180 * acos(-1.0);
  double left = 0, right = std::min(a/cos(c), b/sin(c));
  while (right - left > 1e-9) {
    double x1 = left + (right - left) / 3;
    double x2 = right - (right - left) / 3;
    if (place(a, b, c, x1) > place(a, b, c, x2))
      right = x2;
    else
      left = x1;
  }
  std::cout << std::fixed << std::setprecision(9) << place(a, b, c, left);
}
```

Задача 5 – Поворот фотографии (продолжение)

Пример аналитического решения на Pascal:

```
var
   a,b: real;
   alpha, stepg: real;
   xopt, yopt: real;
   delta, deltax, deltay: real;
   temp: real;

begin
   read(a, b, stepg);
   alpha := stepg * Pi / 180;

   if a > b then begin
      temp := a;
      a := b;
      b := temp;
   end;

   if alpha > pi/2 then
      alpha := alpha - pi/2;

   if sin(2*alpha) >= a/b then begin
      xopt := a / (2 * cos(alpha));
      yopt := a / (2 * sin(alpha));
      end
   else begin
      delta := sqr(sin(alpha)) - sqr(cos(alpha));
      deltax := b * sin(alpha) - a * cos(alpha);
      deltay := a * sin(alpha) - b * cos(alpha);
      xopt := deltax / delta;
      yopt := deltay / delta;
   end;
   writeln(xopt * yopt : 0 : 9);
end.
```

Литература и web-ресурсы для подготовки

1. Дистанционный практикум по программированию ВоГУ: http://avt.vogu35.ru/acm
2. Вологодские олимпиады студентов и школьников: https://olympiads.vogu35.ru/
2. Школа программиста: http://acmp.ru/
3. Алгоритмы на e-maxx: http://e-maxx.ru/algo/
4. Базовые алгоритмы для школьников (учебный курс, видеолекции): http://www.intuit.ru/studies/courses/997/313/info
5. Базовые и "продвинутые" алгоритмы для школьников (учебный курс, видеолекции): http://www.intuit.ru/studies/courses/998/312/info
6. "Продвинутые" алгоритмы для школьников" (учебный курс, видеолекции): http://www.intuit.ru/studies/courses/975/311/info
7. Олимпиадное программирование с нуля на Java: https://vk.com/ol_prog_0
8. Проект "3.5 задачи в неделю“: http://codeforces.com/blog/entry/20066
9. Соревнования по программированию на Codeforces: http://codeforces.com
10. Шень, А. “Программирование: теоремы и задачи”: http://www.e-academy7.narod.ru/COURSES/PROGRAM/LITERATURA/01shen.PDF
11. Меньшиков, Ф. В. Олимпиадные задачи по программированию/ Меньшиков, Федор Владимирович. - Москва: Питер, 2006. - 315 с.
12. Московские олимпиады по информатике / Под ред. Е.В. Андреевой, В. М. Гуровица и В. А. Матюхина—М.: МЦНМО, 2006.— 256 с
13. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен , Ч. Лейзерсон, Р. Ривест, К. Штейн; пер. с англ.; 3-е изд. - Москва: ООО "И.Д. "Вильямс", 2013. - 1328 с.
14. Скиена С.С., Ревилла М.А. Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям. – М.: Кудиц-образ, 2005. – 416 с.

А также многое-многое другое…