МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

В.Л. Арлазаров
И.Б. Мамай

Спортивное программирование

МЕТОДИЧЕСКОЕ ПОСОБИЕ ПО ПОДГОТОВКЕ К ОЛИМПИАДАМ ШКОЛЬНИКОВ

7–11-й классы

Рекомендовано учёным советом института ИТАСУ

МИСиС

Москва 2017

УДК 004.42
А82

Арлазаров В.Л.

А82 Спортивное программирование : метод. пособие по подготовке к олимпиадам школьников. 7–11-й классы / В.Л. Арлазаров, И.Б. Мамай. – М. : Изд. Дом МИСиС, 2017. – 28 с.

Целью данного пособия является ознакомление читателя с задачами, которые предлагались в прошлые годы на заочном и очном турах открытой олимпиады школьников по программированию НИТУ «МИСиС» и Cognitive Technologies, а также помощь в подготовке к предстоящим олимпиадам.

Пособие предназначено для школьников 7–11-х классов. Также оно может быть полезно студентам, учителям информатики и преподавателям.

УДК 004.42

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В настоящее время олимпиады по спортивному программированию набирают все большую популярность. Участие в таких олимпиадах позволяет расширять знания по математике, алгоритмам, языкам программирования, а также совершенствоваться в написании эффективных программ в условиях ограниченного времени.

С 2012 года коллектив преподавателей МИСиС ежегодно проводит открытую олимпиаду школьников по программированию НИТУ «МИСиС» и Cognitive Technologies. Данная олимпиада позволяет не только проверить свои силы и потренироваться в решении задач, но и получить льготы при поступлении в высшие учебные заведения. Олимпиада рассчитана на учащихся 11-х классов, но принимать в ней участие могут ученики 7–11-х классов. Вне зависимости от класса, все участники олимпиады решают общий набор задач.

Целью данного пособия является ознакомление читателя с задачами, которые предлагались в прошлые годы на заочном и очном турах открытой олимпиады школьников по программированию НИТУ «МИСиС» и Cognitive Technologies, а также помощь в подготовке к предстоящим олимпиадам.

Данное пособие содержит не только условия задач, предлагавшихся на олимпиадах прошлых лет, но и подробные решения всех приведенных задач, а также реализацию решений на языке C++, что позволяет читателю научиться решать новые типы задач, использовать новые для себя приемы программирования, а также углубить свои знания по математике и алгоритмам.

Пособие предназначено для школьников 7–11-х классов. Также оно может быть полезно студентам, учителям информатики и преподавателям.

Авторы выражают благодарность за помощь в подготовке пособия Д.А. Крохиной, А.А. Овчинкину, Н.Е. Притуле, С.В. Савинову, Ю.С. Чернышовой.

Задача 1
(Очный тур IV Открытой олимпиады школьников по программированию НИТУ «МИСиС» и Cognitive Technologies)

«Раскраска»
Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Афанасий – автор детских раскрасок. Недавно он решил, что изготовлять раскраски будет по совершенно новой технологии. А именно изначально он отмечает на листе бумаги n точек и соединяет эти точки отрезками так, чтобы никакие два отрезка не пересекались и из каждой точки выходило не более трех отрезков. В результате лист бумаги распадается на фигуры. Дети раскрашивают каждую образовавшуюся фигуру в уникальный цвет. Раскраска считается тем лучше, чем больше цветов потребуется детям, чтобы её раскрасить. Помогите Афанасию выбрать такое расположение n точек и отрезков между ними, чтобы количество цветов было максимально.

По заданному числу n выведите такой плоский граф, в котором:

- n вершин;
- степень каждой вершины не превышает трех;
- число граней максимально.

Входные данные

Одно натуральное число n, $1 \le n \le 1000$.

Выходные данные

В первых n строках выведите по два целых числа a_i, b_i ($|a_i| \le 10^9, |b_i| \le 10^9$), разделенных пробелом. В i-й строке – координаты i-й вершины графа ($1 \le i \le n$). Никакие две пары координат не должны совпадать.

В следующей строке выведите целое число m ($0 \le m \le 10000$) – число рёбер графа.

В следующих m строках выведите по два натуральных числа u_i, v_i ($1 \le u_i$, $v_i \le n$, $u_i \ne v_i$), разделенных пробелом, которые задают отрезок, соединяющий вершины u_i и v_i. Каждый отрезок должен быть упомянут ровно один раз. Из каждой точки должно выходить не более трех отрезков, любые два различных отрезка не должны иметь общих точек, кроме своих концов.

Пример

стандартный ввод	стандартный вывод
2	-1000000000 -1000000000 1000000000 1000000000 0
7	0 0 4 0 0 2 4 2 1 1 3 1 2 1 10 1 2 1 3 1 5 2 4 2 6 3 4 3 5 4 6 5 7 6 7

Решение

Для решения данной задачи необходимо воспользоваться формулой Эйлера для плоских графов: В + Г – Р = К + 1, где В – количество вершин, Г – количество граней (требуемое число фигур), Р – количество рёбер, К – количество компонент связности. Из формулы Эйлера получаем, что для максимизации числа фигур нужно максимизировать сумму числа рёбер и числа компонент связности.

Если рассмотреть случаи, когда n мало, несложно заметить, что оптимальная стратегия – это разбивать граф на компоненты связности размера 4, кроме, быть может, одной размера не больше 7.

Если же при рассмотрении малых значений n не получается заметить закономерность, задачу можно решить методом динамического программирования. Пусть dp[n] – оптимальная сумма числа рёбер и числа компонент связности на допустимом графе из n вершин. Нужно рассмотреть два случая.

1. В графе одна компонента связности. В этом случае нужно рассмотреть случаи малых значений n и заметить, что, начиная с восьми вершин, выгодней перейти к двум или более компонентам связности. Для значений n от 1 до 7 значения dp[n] можно найти вручную.

2. В графе не менее двух компонент связности. Пусть k – размер одной из компонент связности. В таком случае наш граф распадается на два графа, один из которых состоит из k вершин, а второй из $n - k$ вершин. Ответы для этих двух графов уже посчитаны в dp[k] и dp[$n - k$]. Следовательно, чтобы найти значение dp[n], нужно перебрать все значения k от 1 до $n - 1$ и выбрать максимальное значение суммы dp[k] + dp[$n - k$].

После того как найден способ оптимального разбиения графа на компоненты связности, а также размеры этих компонент, остается только вывести искомый ответ. Так как размеры компонент не будут превосходить семи, необходимо предварительно найти вид оптимальных компонент размера от 1 до 7.

Решение на языке C++

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

int main() {
    int n; cin>> n;

    int f = max(0, (n / 4) - ((n % 4) != 0 && (n % 4 < 3)));
    int l = n - 4 * f;
    int x = 0;
    int y = 0;
    int max = 900000000;
    for (int i = 0; i < f; ++i) {
        if (x > max) {
            y += 3;
            x = 0;
        }
        cout << x << ' ' << y << endl;
        cout << x + 2 << ' ' << y << endl;
        cout << x + 1 << ' ' << y + 1 << endl;
        cout << x + 1 << ' ' << y + 2 << endl;
        x += 3;
    }
    int r = 6 * f;
```

```
switch (l) {
case 1:
    cout << "-1 -1\n";
    break;
case 2:
    cout << "-1 -1\n-2 -2\n";
    break;
case 3:
    cout << "-1 -1\n-2 -2\n-3 -1\n";
    r += 3;
    break;
case 5:
    cout << "-2 -1\n-1 -2\n-2 -3\n-2 -2\n-3 -2\n";
    r += 7;
    break;
case 6:
   cout << "-1 -1\n-2 -2\n-2 -3\n-3 -2\n-3 -3\n-4 -1\n";
    r += 9;
    break;
default:
    break;
}

cout << r << endl;

for (int i = 0; i < f; ++i) {
    cout << 4 * i + 1 << ' ' << 4 * i + 2 << endl;
    cout << 4 * i + 1 << ' ' << 4 * i + 3 << endl;
    cout << 4 * i + 1 << ' ' << 4 * i + 4 << endl;
    cout << 4 * i + 2 << ' ' << 4 * i + 3 << endl;
    cout << 4 * i + 2 << ' ' << 4 * i + 4 << endl;
    cout << 4 * i + 4 << ' ' << 4 * i + 3 << endl;
}
switch (l) {
case 3:
    cout << 4 * f + 1 << ' ' << 4 * f + 2 << endl;
    cout << 4 * f + 3 << ' ' << 4 * f + 2 << endl;
    cout << 4 * f + 1 << ' ' << 4 * f + 3 << endl;
    break;
case 5:
    cout << 4 * f + 1 << ' ' << 4 * f + 2 << endl;
    cout << 4 * f + 1 << ' ' << 4 * f + 4 << endl;
    cout << 4 * f + 1 << ' ' << 4 * f + 5 << endl;
    cout << 4 * f + 3 << ' ' << 4 * f + 2 << endl;
    cout << 4 * f + 4 << ' ' << 4 * f + 2 << endl;
    cout << 4 * f + 3 << ' ' << 4 * f + 5 << endl;
    cout << 4 * f + 4 << ' ' << 4 * f + 3 << endl;
    break;
```

```
    case 6:
        cout << 4 * f + 1 << ' ' << 4 * f + 2 << endl;
        cout << 4 * f + 1 << ' ' << 4 * f + 3 << endl;
        cout << 4 * f + 1 << ' ' << 4 * f + 6 << endl;
        cout << 4 * f + 3 << ' ' << 4 * f + 2 << endl;
        cout << 4 * f + 4 << ' ' << 4 * f + 2 << endl;
        cout << 4 * f + 3 << ' ' << 4 * f + 5 << endl;
        cout << 4 * f + 4 << ' ' << 4 * f + 5 << endl;
        cout << 4 * f + 4 << ' ' << 4 * f + 6 << endl;
        cout << 4 * f + 5 << ' ' << 4 * f + 6 << endl;
        break;
    default:
        break;
    }

    return 0;
}
```

Задача 2
(Очный тур IV Открытой олимпиады школьников по программированию НИТУ «МИСиС» и Cognitive Technologies)

«Игра Васи»
Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Студент «МИСиСа» Вася любит придумывать разные игры. В частности, однажды он придумал новую уникальную игру. Игра Васи происходит на прямоугольном поле размера $M \times N$ клеток. Будем считать, что левая верхняя клетка имеет координаты (1, 1). В каждой клетке изначально написано некоторое число. В игру играет Q человек. Каждый человек за один ход может выбрать некоторый прямоугольник, стороны которого параллельны сторонам поля, и прибавить ко всем его клеткам некоторое целое число A.

Затем по результирующему полю в конце игры Вася производит начисление очков (каким образом оно происходит, пока не решил и сам Вася). Тем не менее Вася просит Вас помочь по начальной конфигурации поля и ходам игроков рассчитать результирующее игровое поле.

Входные данные

В первой строке записано единственное целое число $1 \le Q \le 10^5$ – количество игроков.

В следующих Q строках записаны ходы игроков. Каждая из строк содержит пять целых чисел: Y_1, X_1, Y_2, X_2, A, где (Y_1, X_1) – строка и столбец верхнего левого угла выбранного игроком прямоугольника, а (Y_2, X_2) – строка и столбец нижнего правого угла выбранного игроком прямоугольника. $1 \le X_1 \le X_2 \le N$, $1 \le Y_1 \le Y_2 \le M$. $-100 \le A \le 100$ – число, которое должно быть прибавлено в каждой клетке выбранного прямоугольника.

В $(Q + 2)$-й строке записаны два числа M и N, $5 \le M$, $N \le 1000$ – высота и ширина поля. В следующих M строках записаны через пробел по N чисел – элементы матрицы. Все элементы матрицы – целые числа, по модулю не превышающие 100.

Выходные данные

Выведите ровно M строк, в каждой строке по N целых чисел через пробел – значения элементов матрицы после ходов всех Q игроков.

Пример

стандартный ввод	стандартный вывод
2 2 4 6 8 1 4 5 9 9 3 10 10 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 5 1 1 0 0 0 0 0 0 0 1 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 8 0 0 0 0 0 0 0 1 1 1 9 1 0 0 0 0 0 1 1 1 1 9 0 0 0 0 0 1 4 4 4 7 3 0 0 0 0 1 4 4 4 9 4 1 0 0 0 1 4 4 4 5 5 2 0 0 0 0 3 3 3 3 3 0 0 0 0 0 3 3 3 3 3 0 0 0 0 0 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0

Решение

Чтобы решить данную задачу, необходимо научиться выполнять все запросы за $O(1)$. Чтобы этого добиться, будем хранить не само поле, а такую таблицу ans, что при вычислении ее частичных сумм получим наше поле.

Чтобы выполнить запрос вида Y_1, X_1, Y_2, X_2, A необходимо выполнить следующие изменения в таблице:

$\text{ans}[X_1][Y_1] \mathrel{+}= A$
$\text{ans}[X_1][Y_2 + 1] \mathrel{-}= A$
$\text{ans}[X_2 + 1][Y_1] \mathrel{-}= A$
$\text{ans}[X_2 + 1][Y_2 + 1] \mathrel{+}= A$

После вычисления всех запросов необходимо перейти от таблицы ans к таблице ее частичных сумм. То есть мы хотим, чтобы после этого преобразования в ячейке ans[i][j] было записано число, равное сумме всех значений ans[x][y] при $1 \le x \le i$, $1 \le y \le j$. Для этого необходимо сначала выполнить вычисление частичных сумм по одной из координат, а затем по другой. В приведенном ниже решении на языке C++ эта операция делается двумя двойными циклами.

Решение на языке C++

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

void solve(int q,
           vector<pair<int, int> >& l_top,
           vector<pair<int, int> >& r_bot,
           vector<int>& addition,
           int m,
           int n,
           vector<vector<int> >& bias,
           vector<vector<int> >& ans) {

    ans.assign(m + 1, vector<int>(n + 1, 0));

    for (int i = 0; i < q; ++i) {
    ans[l_top[i].first][l_top[i].second] += addition[i];
    ans[l_top[i].first][r_bot[i].second + 1] -= addition[i];
    ans[r_bot[i].first + 1][l_top[i].second] -= addition[i];
    ans[r_bot[i].first + 1][r_bot[i].second + 1] += addition[i];
    }
    for (int i = 0; i < m; ++i) {
        for (int j = 1; j < n; ++j) {
            ans[i][j] += ans[i][j - 1];
        }
    }
    for (int j = 0; j < n; ++j) {
        for (int i = 1; i < m; ++i) {
            ans[i][j] += ans[i - 1][j];
```

```
        }
    }
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            ans[i][j] += bias[i][j];
        }
    }
}

int main() {
    int q = 0;
    vector<pair<int, int> > l_top;
    vector<pair<int, int> > r_bot;
    vector<int> addition;
    int m = 0;
    int n = 0;
    vector<vector<int> > bias;
    vector<vector<int> > ans;

    ::scanf("%d", &q);
    l_top.resize(q);
    r_bot.resize(q);
    addition.resize(q);

    for (int i = 0; i < q; ++i) {
        ::scanf("%d %d %d %d %d", &l_top[i].first, &l_top[i].second,
            &r_bot[i].first, &r_bot[i].second, &addition[i]);
        --l_top[i].first;
        --l_top[i].second;
        --r_bot[i].first;
        --r_bot[i].second;
    }

    ::scanf("%d %d", &m, &n);
    bias.assign(m, vector<int>(n, 0));

    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            ::scanf("%d", &bias[i][j]);
        }
    }

    solve(q, l_top, r_bot, addition, m, n, bias, ans);
```

```
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            if (j) ::printf(" ");
            ::printf("%d", ans[i][j]);
        }
        ::printf("\n");
    }

    return 0;
}
```

Задача 3
(Очный тур IV Открытой олимпиады школьников по программированию НИТУ «МИСиС» и Cognitive Technologies)

«Игра Пети»
Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Студент «МИСиС» Петя, так же как и Вася, любит придумывать разные игры. Вдохновившись творением Васи, он придумал свою новую уникальную игру.

Игра Пети также происходит на прямоугольном поле размера $M \times N$ клеток. Будем считать, что левая верхняя клетка имеет координаты (1, 1). Значения во всех клетках изначально равны нулю. В игру играет Q человек. Каждый человек за один ход может выбрать некоторый прямоугольник, стороны которого параллельны сторонам поля, и прибавить ко всем его клеткам некоторое целое число A.

Теперь Петю очень интересует, какое максимальное значение элемента получилось на результирующем поле. Помогите ему в этом.

Входные данные

В первой строке записано единственное целое число $1 \le Q \le 10^4$ – количество игроков.

В следующих Q строках записаны ходы игроков. Каждая из строк содержит пять целых чисел: Y_1, X_1, Y_2, X_2, A, где (Y_1, X_1) – строка и

столбец верхнего левого угла выбранного игроком прямоугольника, а (Y_2, X_2) – строка и столбец нижнего правого угла выбранного игроком прямоугольника. $1 \le X_1 \le X_2 \le N$, $1 \le Y_1 \le Y_2 \le M$. $-100 \le A \le 100$ – число, которое должно быть прибавлено в каждой клетке выбранного прямоугольника.

В $(Q + 2)$-й строке записаны два числа M и N, $5 \le M \le 10^9$, $5 \le N \le 4000$ – высота и ширина поля.

Выходные данные

Выведите ровно одно целое число – максимальное значение элемента матрицы после ходов всех Q игроков.

Пример

стандартный ввод	стандартный вывод
2 2 4 6 8 1 4 5 9 9 3 10 10	4

Решение

Так как количество игроков $Q \le 10^4$ не слишком велико по сравнению с высотой поля, нам выгодно сделать сжатие координат по оси y. После этого размеры поля не будут превосходить $10^4 \times 4000$. Будем последовательно идти по координате y, вычисляя значения элементов в текущей строке. Для этого каждый прямоугольник нужно представить как два события: строка, где он начинается, и строка, где он заканчивается. Все эти события отсортируем, чтобы можно было последовательно их обрабатывать. При обработке очередной строки нам необходимо учесть все прямоугольники, которые начинаются или заканчиваются в этой строке. Для текущей строки каждый прямоугольник вносит вклад в виде прибавления некоторого числа на отрезке. Чтобы выполнить все эти прибавления, необходимо отсортировать концы всех отрезков и последовательным проходом заполнить все изменения в текущей строке.

Для решения задачи мы выполняем сортировку по всем прямоугольникам, а также проход по всем ячейкам сжатой таблицы. Таким образом, асимптотика данного решения

$$O(Q \cdot \log(Q) + N \cdot Q).$$

Решение на языке C++

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

void solve(int q,
           vector<pair<int, int> >& l_top,
           vector<pair<int, int> >& r_bot,
           vector<int>& addition,
           int m,
           int n,
           int& ans) {

    ans = -1e9;
    vector<pair<long long, int> > events;

    l_top.push_back(make_pair(0, 0));
    r_bot.push_back(make_pair(m - 1, n - 1));
    addition.push_back(0);
    for (int query = 0; query <= q; ++query) {
    for (int col = l_top[query].second;
    col <= r_bot[query].second; ++col) {
            events.push_back(make_pair(
                1ll * col * m + l_top[query].first,
                addition[query]));
            events.push_back(make_pair(
                1ll * col * m + r_bot[query].first + 1,
                -addition[query]));
        }
    }

    sort(events.begin(), events.end());

    int cur = 0;
    for (size_t i = 0; i < events.size(); ++i) {
        if (i > 0 && events[i].first != events[i - 1].first) {
            ans = max(ans, cur);
        }
        cur += events[i].second;
    }
}

int main() {
    int q = 0;
    vector<pair<int, int> > l_top;
    vector<pair<int, int> > r_bot;
    vector<int> addition;
    int m = 0;
```

```
    int n = 0;
    int ans = 0;

    ::scanf("%d", &q);
    l_top.resize(q);
    r_bot.resize(q);
    addition.resize(q);

    for (int i = 0; i < q; ++i) {
        ::scanf(«%d %d %d %d %d»,
            &l_top[i].first, &l_top[i].second,
            &r_bot[i].first, &r_bot[i].second, &addition[i]);
        --l_top[i].first;
        --l_top[i].second;
        --r_bot[i].first;
        --r_bot[i].second;
    }

    ::scanf("%d %d", &m, &n);
    solve(q, l_top, r_bot, addition, m, n, ans);
    ::printf("%d\n", ans);

    return 0;
}
```

Задача 4
(Очный тур IV Открытой олимпиады школьников по программированию НИТУ «МИСиС» и Cognitive Technologies)

«Странная функция»
Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 64 мегабайта

Студент Иннокентий обучает младшую сестру Машеньку программированию. Недавно он показал ей такую функцию:

```
C++ code:
long long F(long long a, long long b) {
  return (b ? F(a ^ b, (a & b) << 1) : a);
}

Java code:
public static long F(long a, long b) {
  return ((b != 0) ? F (a ^ b, (a & b) << 1) : a);
}

Pascal code:
function F(a : int64; b : int64) : int64;
begin
  if b <> 0 then F := F(a xor b, (a and b) shl 1) else F := a;
end;
```

Затем он написал два натуральных числа a и b ($1 \le a < b \le 2 \cdot 10^9$) и попросил вычислить значение выражения $F(a, F(a+1, \ldots F(b-1, b)\ldots))$. Например, для чисел 5 и 6 искомое выражение имеет вид $F(5, 6)$, а для чисел 14 и 17 искомое выражение имеет вид $F(14, F(15, F(16, 17)))$.

К удивлению брата, Машенька легко справилась с этой задачей. Попробуйте и Вы.

Входные данные

В единственной строке через пробел записаны два целых числа a и b ($1 \le a < b \le 2 \cdot 10^9$).

Выходные данные

Выведите единственное целое число – ответ на задачу.

Пример

стандартный ввод	стандартный вывод
14 17	62

Решение

Чтобы решить данную задачу, необходимо разобраться, по какой формуле действует функция $F(a, b)$. Самый простой способ разобраться с этим вопросом – это вычислить значения данной функции при малых a и b, например от 1 до 10. Посмотрев на эти значения, легко заметить, что $F(a, b) = a + b$.

Использовав явный вид функции $F(a, b)$, можно сделать следующие вычисления:

$$F(a, F(a+1, \ldots F(b-1, b)\ldots)) = a + F(a+1, \ldots F(b-1, b)\ldots) = \ldots$$
$$\ldots = a + (a+1) + \ldots + b = (a+b) * (b-a+1) / 2.$$

Решение на языке C++

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

int main() {
    long long a, b;
    cin >> a >> b;
    cout << ((a + b) * (b - a + 1) / 2) << endl;
    return 0;
}
```

Задача 5
(Очный тур IV Открытой олимпиады школьников по программированию НИТУ «МИСиС» и Cognitive Technologies)

«Доменная печь 2»
Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 64 мегабайта

Однажды студенту Пафнутию перед сдачей зачета по металлургии приснился страшный сон. Он оказался в комнате очень странной формы, внутри которой стояла огромная доменная печь. Проснувшись, он начал вспоминать детали сна. В частности, он вспомнил, что одна стена комнаты имела вид параболы с вершиной в точке $(0, H)$ и эта парабола пересекала ось O_X в точках $(\pm T, 0)$, а вторая стена представляла собой отрезок, соединяющий точки $(T, 0)$ и $(-T, 0)$.

Теперь ему интересно, доменную печь какого максимального размера можно разместить внутри данной комнаты. Доменную печь Пафнутий для простоты считает кругом. Пафнутий справился с задачей, попробуйте и Вы.

Входные данные

В первой строке записано число T, во второй – число H, $0 \le T$, $H \le 20000$. Оба числа даны с шестью знаками после запятой.

Выходные данные

Выведите единственное число – максимально возможный радиус R доменной печи, с абсолютной погрешностью не более 10^{-6}.

Пример

стандартный ввод	стандартный вывод
10.000000 10.000000	5.000000

Решение

В данной задаче нужно рассмотреть два случая. Если $t < h$, то окружность касается параболы в двух точках, симметричных относительно оси O_Y. В противном случае окружность касается параболы в одной точке, лежащей на оси O_Y.

В первом случае нужно записать условие касания окружности и параболы, из которого вытекает следующая формула:

$$R = t - t^2 / (2\,h).$$

Во втором случае формула, очевидно, следующая:

$$R = h / 2.$$

Решение на языке C++

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

double solve(double h, double t) {
    if (t < h)
        return t - t * t / (2.0 * h);
    else
        return h / 2.0;
}

int main() {
    double t = 0.0;
    double h = 0.0;
    scanf("%lf\n", &t);
    scanf("%lf\n", &h);
    printf("%.6lf\n", solve(h, t));

    return 0;
}
```

Задача 6
(Очный тур IV Открытой олимпиады школьников по программированию НИТУ «МИСиС» и Cognitive Technologies)

«Недели искусства»
Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 64 мегабайта

Не так давно в «МИСиС» прошли недели, посвященные живописи и искусству. В частности, в течение данных недель студентам были представлены разные картины.

Каждая картина оценивается своей красотой n. Красота картины определяется по формуле $n = x^2 + y^3$, где x – натуральное число, характеризующее художественные достоинства картины, а y – стоимость картины в тысячах долларов (также натуральное число). Две картины, у которых совпадают оба параметра x и y, считаются одинаковыми.

Теперь студентам «МИСиС» интересно, какое существует максимальное количество различных картин, имеющих красоту n.

Входные данные

В единственной строке записано целое число n, $1 \le n \le 10^{18}$.

Выходные данные

Выведите единственное число – количество различных картин с красотой n.

Пример

стандартный ввод	стандартный вывод
1	0
17	2

Решение

Чтобы решить данную задачу, достаточно перебрать все значения y, такие что $1 \le y^3 \le n$. Так как $1 \le n \le 10^{18}$, то все подходящие значения y не превосходят 10^6. Для каждого значения y остается проверить, является ли разность $n - y^3$ полным квадратом. Ответом на задачу будет количество таких y, для которых разность $n - y^3$ является полным квадратом.

Чтобы проверить, является ли число x полным квадратом, нужно извлечь из него корень и округлить до целого, а затем полученный результат возвести в квадрат. Если мы снова получили число x, значит, оно является полным квадратом.

Решение на языке C++

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

long long n, a, b, ans;

int main() {
    cin >> n;
    ans = 0;
    for (long long y = 1; y * y * y < n; ++y) {
        a = n - y * y * y;
        b = sqrt(a + 0.1);
        if (a == b * b)
            ++ans;
    }
    cout << ans;

    return 0;
}
```

Задача 7
(Заочный тур IV Открытой олимпиады школьников по программированию НИТУ «МИСиС» и Cognitive Technologies)

«Доменная печь»
Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 64 мегабайта

Доменная печь в сечении является окружностью. На этой окружности в трех различных точках A, B, C расположены датчики. В процессе работы печи датчик, который расположен в точке A, излучает сигнал в трех направлениях. В трех точках окружности, в которые

приходят три сигнала от датчика A, расположены датчики A_1, A_2, A_3. Чтобы вся система датчиков работала правильно, необходимо, чтобы отрезки AA_1, AA_2 и AA_3 совпадали соответственно с биссектрисой, медианой и высотой треугольника ABC.

Рабочие уже установили датчики A_1, A_2, A_3. Ваша задача определить, в какие места необходимо установить датчики A, B, C, считая для определенности, что $AB < AC$.

Входные данные

В первой строке записаны два вещественных числа – координаты точки A_1, во второй строке записаны два вещественных числа – координаты точки A_2, в третьей строке записаны два вещественных числа – координаты точки A_3.

Все числа по модулю не превышают 10.

Выходные данные

Ваша программа должна вывести шесть вещественных чисел по 2 в строке – координаты точек A, B и C с точностью не менее 6 знаков после запятой.

Гарантируется, что решение существует и единственно.

Пример

стандартный ввод	стандартный вывод
1.63491171093880 -1.00305137207223 1.59781562645256 -1.05477872118798 1.76397022837593 -0.02275460746696	0.658368 0.722983 1.816076 -0.518506 1.253514 -1.352538
0.54122777315611 -1.50219813041616 0.90070161920117 -1.06129849816222 -0.98452399209240 -1.54215134647501	-1.446397 -0.950981 -0.687445 -1.696801 1.027293 -0.357101
-0.42477291643007 -1.00173742128497 -0.32871280973019 -1.19786841497190 -0.48116583081017 -0.75747287316698	1.394399 -0.074823 0.123382 0.289635 0.825216 -1.638639

Решение

Для начала заметим, что для существования однозначного решения необходимо, чтобы данные три точки A_1, A_2, A_3 попарно не совпадали. Так как они лежат на той же окружности, что и искомый треугольник ABC, то найдем эту окружность: её центр является точкой пересечения серединных перпендикуляров.

Далее, так как биссектриса AA_1 делит дугу BC окружности пополам, то радиус OA_3, проведенный к точке пересечения биссектрисы и окружности, будет проходить через середину отрезка BC, и перпенди-

кулярен ему. Высота AA_3 также перпендикулярна BC, а значит, можно найти точку A как пересечение окружности и прямой, параллельной OA_1 и проходящей через A_3.

Теперь найдем две оставшиеся вершины треугольника ABC. Проведем медиану AA_2 и пересечем ее с прямой OA_1. Так как обе эти прямые проходят через середину BC, то точка их пересечения и является серединой BC. Затем проведем прямую BC как перпендикулярную к OA_1 и проходящую через найденную ранее середину отрезка BC. Пересечения этой прямой с окружностью дает две оставшиеся вершины треугольника ABC. Остается лишь указать, какая из них B, а какая – C. Это делается из условия $AB < AC$.

Решение на языке C++

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

struct Vector {
    double x, y;
    Vector() {}
    Vector(double ax, double ay): x(ax), y(ay) {}
    Vector operator+(const Vector &v) const {
        return Vector(x + v.x, y + v.y); }
    Vector operator-() const { return Vector(-x, -y); }
    double len() const { return sqrt(x * x + y * y); }
    Vector normalize() const {
        double l = len();
        return Vector(x / l, y / l);
    }
    double dot(const Vector &v) const { return x * v.x + y * v.y;
}
    double cross(const Vector &v) const { return x * v.y - y * v.x;
}
};

struct Point {
    double x, y;
    Point() {}
    Point(double ax, double ay): x(ax), y(ay) {}
    Point operator+(const Vector &v) const {
        return Point(x + v.x, y + v.y); }
    Point operator-(const Vector &v) const {
        return Point(x - v.x, y - v.y); }
    bool operator==(const Point &p) const {
        return fabs(x - p.x) < 1e-8 && fabs(y - p.y) < 1e-8; }
```

```
    bool operator!=(const Point &p) const { return !(*this == p);
}
    double distTo(const Point &p) const { return to(p).len(); }
    Vector to(const Point &p) const { return Vector(p.x - x, p.y - y);
}
    static Point ZERO;
};

Point Point::ZERO(0, 0);

struct Circle {
    Point O;
    double r;
    Circle(Point center, double ar): O(center), r(ar) {}
    Circle operator+(const Vector &v) const { return Circle(O + v, r);
}
};

struct Line {
    Point A, B;
    double a, b, c; // ax+by+c=0
    Vector v; // direction vector from A to B
    static Line twoPoints(Point A, Point B) {
        Line res;
        res.A = A;
        res.B = B;
        res.v = A.to(B).normalize();
        res.calcAbc();
        return res;
    }
    static Line pointDirection(Point p, Vector v) {
        Line res;
        res.A = p;
        res.B = p + v;
        res.v = v.normalize();
        res.calcAbc();
        return res;
    }
    double len() const { return A.distTo(B); }
    Point intersect(const Line &l) const {
       double d = a * l.b - b * l.a;
       return Point(-(c * l.b - b * l.c) / d, -(a * l.c - c * l.a) / d);
    }
    Point getCenter() const {
         return Point((A.x + B.x) / 2.0, (A.y + B.y) / 2.0); }
    Line getPerpendicular(Point p) const {
        return Line::pointDirection(p, Vector(a, b));
```

```
    }
    Line getCenterPerpendicular() const {
        return getPerpendicular(getCenter()); }
    pair<Point, Point> intersect(const Circle &circ) const {
        pair<Point, Point> res;
        if (circ.O != Point::ZERO) {
            Vector transp = circ.O.to(Point::ZERO);
            pair<Point, Point> tmp = Line::twoPoints(A + transp,
                B + transp).intersect(circ + transp);
            res.first = tmp.first - transp;
            res.second = tmp.second - transp;
        } else {
            double x0 = -a*c/(a*a+b*b);
            double y0 = -b*c/(a*a+b*b);
            double d = circ.r * circ.r - c*c/(a*a+b*b);
            double mult = sqrt(d / (a*a+b*b));
            res.first  = Point(x0 + b * mult, y0 - a * mult);
            res.second = Point(x0 - b * mult, y0 + a * mult);
        }
        return res;
    }
    Point intersect(const Circle &circ, const Point &known) const
{
        pair<Point, Point> pts = intersect(circ);
        Point res = pts.first;
        if (res == known)
            res = pts.second;
        return res;
    }
protected:
    void calcAbc() {
        a = -v.y;
        b = v.x;
        c = 0 - (a * A.x + b * A.y);
    }
};

void solve(Point bis, Point med, Point hei, Point &A, Point &B,
Point &C) {
    Line medBis, bisHei, heiMed;
    medBis = Line::twoPoints(med, bis);
    bisHei = Line::twoPoints(bis, hei);
    heiMed = Line::twoPoints(hei, med);
    Line medBisPerp = medBis.getCenterPerpendicular();
    Line bisHeiPerp = bisHei.getCenterPerpendicular();
    Point O = medBisPerp.intersect(bisHeiPerp);
    Line centBis = Line::twoPoints(O, bis);
```

```
    Circle circle(O, centBis.len());
    Line height = Line::pointDirection(hei, -centBis.v);
    A = height.intersect(circle, hei);
    Line Amed = Line::twoPoints(A, med);
    Point centerOfBC = Amed.intersect(centBis);
    Line BCraw = centBis.getPerpendicular(centerOfBC);
    pair<Point, Point> BandC = BCraw.intersect(circle);
    B = BandC.first;
    C = BandC.second;
    if (A.distTo(B) > A.distTo(C))
        swap(B, C);
}

int main() {
    double medx, medy, bisx, bisy, heix, heiy;
    scanf("%lf%lf%lf%lf%lf%lf",
        &bisx, &bisy, &medx, &medy, &heix, &heiy);
    Point A, B, C;
    solve(Point(bisx, bisy), Point(medx, medy),
        Point(heix, heiy), A, B, C);
    printf("%.6f %.6f\n%.6f %.6f\n%.6f %.6f\n",
        A.x, A.y, B.x, B.y, C.x, C.y);

    return 0;
}
```

ЗАКЛЮЧЕНИЕ

В данном пособии ко всем задачам приведены решения, а также реализация решений на языке C++. Читателям после ознакомления с решением задачи рекомендуется самостоятельно реализовать решение на компьютере, а также изучить код авторского решения.

Для более углублённого изучения материала рекомендуется вычислять асимптотику решений приведённых задач, а также стараться придумать более быстрое решение, если такое существует, или доказать оптимальность авторского решения.

Учебное издание

Владимир Львович Арлазаров, член-корреспондент РАН, профессор
Игорь Борисович Мамай, кандидат физико-математических наук

СПОРТИВНОЕ ПРОГРАММИРОВАНИЕ

Методическое пособие по подготовке к олимпиадам школьников 7–11-й классы

Редактор *В.И. Ченцова*

Компьютерная верстка *И.Г. Иваньшина*

Подписано в печать 07.04.17	Бумага офсетная	
Формат 60 × 90 $^{1}/_{16}$	Печать офсетная	Уч.-изд. л. 1,5
	Тираж 250 экз.	Заказ

Национальный исследовательский
технологический университет «МИСиС»,
119049, Москва, Ленинский пр-т, 4

Издательский Дом МИСиС,
119049, Москва, Ленинский пр-т, 4
Тел. (495) 638-45-22

Отпечатано в типографии Издательского Дома МИСиС,
119049, Москва, Ленинский пр-т, 4
Тел. (499) 236-76-17, тел./факс (499) 236-76-35