Открытая командная олимпиада по программированию
Весенний тур 2018
19 мая 2018

A. Azat and bookshelf

Автор: Баев А.Ж.

Ответ равен разности объемов двух прямоугольных параллелепипедов:

$$W \cdot H \cdot L - (W - 2D)(H - 2D)(L - D).$$

Асимптотика: $O(1)$.

```
#include <iostream>
using namespace std;
int main() {
    long long d, w, h, l, volume_ext, volume_int;
    cin >> d >> w >> h >> l;
    volume_ext = w * h * l;
    volume_int = (w - 2 * d) * (h - 2 * d) * (l - d);
    cout << volume_ext - volume_int << endl;
    return 0;
}
```

B. Bekarys and khet

Автор: Абдикалыков А.К.

Длина пути луча в каждой клетка равна 1. Значит, ответ — количество клеток, через которые пройдет луч. Для этого можно построить простой автомат, определяющий (x_k, y_k) — текущее положение, v_k — направление на входе в клетку.

Направление меняется следующим образом, если в клетке (x_k, y_k) находится зеркало вида '/' и вида '\' соответственно:

$$v_{k+1} = \begin{cases} UP, & v_k = RIGHT \\ DOWN, & v_k = LEFT \\ RIGHT, & v_k = UP \\ LEFT, & v_k = DOWN \end{cases} \qquad v_{k+1} = \begin{cases} DOWN, & v_k = RIGHT \\ UP, & v_k = LEFT \\ LEFT, & v_k = UP \\ RIGHT, & v_k = DOWN \end{cases}$$

Положение меняется в зависимости от направления естественным образом.

Асимптотика: $O(M \cdot N)$ (так как каждую клетку луч перечесет не более двух раз).

```
#include <iostream>
using namespace std;
int main() {
    long long n, m;
    char field[101][101];
    cin >> n >> m;
    for (int i = 0; i < n; i++)
        cin >> field[i];

    int dx[] = {0, 0, -1, 1};
    int dy[] = {1, -1, 0, 0};
    enum dir{RIGHT, LEFT, UP, DOWN};
    int forward_slash[] = {UP, DOWN, RIGHT, LEFT};
    int backward_slash[] = {DOWN, UP, LEFT, RIGHT};
    int x = 0, y = 0, v = RIGHT;
    int answer = 0;
    while (x >= 0 && x < n && y >= 0 && y < m) {
        if (field[x][y] == '/')
            v = forward_slash[v];
```

```
        if (field[x][y] == '\\')
            v = backward_slash[v];
        x += dx[v];
        y += dy[v];
        answer++;
    }

    cout << answer << endl;
    return 0;
}
```

C. Computer vision

Автор: Абдикалыков А.К.

Обозначим: $dp[m][r]$ — количество m-значных чисел, которые соответствует первым m символам шаблона и дают остаток r при делении на 11.

В случае, если символ $a[m]$ в шаблоне является цифрой, то количество $dp[m][r]$ определяется как количество соответствующих $(m-1)$-значных чисел:

$$dp[m][r_k] = dp[m-1][k],$$

где $r_k = 10k + a[m] \pmod{11}$, для всех $k = 0, \ldots, 10$.

В случае, если символ $a[m]$ является звездочкой, то вместо неё можно подставить любую цифру. Для фиксированного k от 0 до 9, значение $10 * k + r \pmod{11}$ дает все возможные остатки, кроме $10 - k$ (так как никакая цифра не может давать остаток $r = 10$ при делении на 11):

$$dp[m][r] = \sum_{k+r \neq 10, k=0}^{10} dp[m-1][k] = dp[m][r] =$$

$$= \sum_{k=0}^{10} dp[m-1][k] - dp[m-1][10-r],$$

Асимптотика: $O(N)$.

```
#include <iostream>
#include <string>
using namespace std;

int dp[100001][11];
int main() {
    string s;
    cin >> s;
    long long mod = 1000000007LL;
    int n = s.size();
    dp[0][0] = 1;
    for (int m = 1; m <= n; m++) {
        if (s[m - 1] != '*') {
            for (int k = 0; k < 11; k++) {
                int r = (10 * k + s[m - 1] - '0') % 11;
                dp[m][r] = dp[m - 1][k];
            }
        } else {
            long long s = 0;
            for (int k = 0; k < 11; k++)
                s = (s + dp[m - 1][k]) % mod;
            for (int r = 0; r < 11; r++) {
                dp[m][r] = (s + mod - dp[m - 1][10 - r]) % mod;
            }
        }
    }
```

```
    cout << dp[n][0] << endl;
    return 0;
}
```

D. DPK rover

Автор: Баев А.Ж.

При $M = 1$, ответ очевидно будет $D/2$. Пусть M — простое число. Обозначим x — расстояние, после которого первые $(M-1)$ марсоходов отдадут топливо M-му марсоходу. На обратную дорогу они должны оставить себе столько же топлива, сколько потратили. Значит, каждый из них отдаст M-му марсоходу топлива, на котором можно проехать расстояние $(D-2x)$. Итого: $(M-1)(D-2x)$.

У M-го марсохода бак свободен ровно настолько, сколько он уже проехал. Чтобы всё полученное топливо поместилось в его в бак необходимо выполенение:

$$x \geqslant (M-1)(D-2x).$$

Схема движения обратно соответствует и прямой схеме (если произвести все движения в обратном порядке). То есть полученного топлива должно хватать для движения обратно:

$$x \leqslant (M-1)(D-2x).$$

Таким образом, до точки разворота группа проедет:

$$x = \frac{M-1}{2M-1}D.$$

Покажем, что в случае с составным числом $M = A \cdot B$ всегда выгодно делиться на группы. Два деления (на группы размером B и далее на группы размером A) выгоднее, чем одно (на группу размера AB). Для этого проверим неравенство:

$$\frac{A-1}{2A-1}D + \frac{B-1}{2B-1}D \geqslant \frac{AB-1}{2AB-1}D$$
$$\frac{(4AB-1)(B-1)(A-1)}{(2AB-1)(2B-1)(2A-1)} \geqslant 0$$

Это верно с учетом $A > 1$ и $B > 1$.

Для решения задачи в общем виде достаточно получить каноническое разложение числа $M = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ на простые множители. Сделать это можно стандартным алгоритмом разложения на простые множители, проверяя делители до $\sqrt{M}$. Ответ равен:

$$L = D\left(\frac{1}{2} + \sum_{i=1}^{k} \alpha_i \frac{p_i - 1}{2p_i - 1}\right).$$

Асимптотика: $O(\sqrt{M})$.

Решение за $O(M)$ не проходило ограничения по времени.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    long long d, m;
    long double answer = 0.5;
    cin >> d >> m;
    for (long long p = 2; p * p <= m; p++) {
        while (m % p == 0) {
            answer += (p - 1.0) / (2.0 * p - 1.0);
            m /= p;
        }
    }
    if (m != 1) {
        answer += (m - 1.0) / (2.0 * m - 1.0);
    }
    cout << fixed << setprecision(10) << answer * d << endl;
    return 0;
}
```

E. Excellent idea

Автор: Абдикалыков А.К.

Необходимо было вывести k-й символ текста, где k — это число в этом же тексте. Число было во всех тестах на последнем месте. В некоторых тестах числа были больше 10. Во всех тестах числа не превышали длину теста.

Асимптотика: $O(N)$, где N — длина теста.

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    getline(cin, s);
    int index = 0, n = s.size();
    for (int i = 0; i < n; i++)
        if (s[i] >= '0' && s[i] <= '9')
                index = 10 * index + s[i] - '0';
    cout << s[index - 1] << endl;
    return 0;
}
```

F. Forced reduction

Автор: Баев А.Ж.

Решение 1. Дерево представим в виде списка ребер ориентированного графа (от родителя к детям). Для быстрого выделения ребер с одинаковыми буквами у каждой вершины булдем хранить сразу 26 списков (по одному списку для каждой буквы).

Запустим обход в ширину из корня. При этом для каждой вершины, которая изымается из очереди, будет объединять всех её детей с одинаковыми ребрами. Например, у вершины v есть дети $a_1, a_2, \ldots, a_k$, в которые ведут ребра с буквой `a`. В отличии от классического добавления в очередь всех вершин a_i, в очередь добавим только a_1. При этом в список детей вершины a_1 добавим детей из $a_2, a_3, \ldots, a_k$. То же самое исполним для списков $b_i, c_i, \ldots, z_i$.

Каждый ребенок (точнее ребро) будет перенесено не более одного раза. Поэтому асимптотика аккумулировано не превосходит $O(N)$. Поиск ребер с одинаковыми буквами требует $O(1)$ действий, так как у каждой вершины есть отдельный список для каждой буквы. Остается алгоритм обхода в ширину: $O(N)$ действий.

Асиптотика: $O(N)$.

Решение 2 (которое, увы, проходило на олимпиаде). Достаточно промоделировать требуемый процесс при обходе дерева в глубину или в ширину, начиная с корня. При этом поддерживать ответ в какой-либо структуре с элементами в виде строк (без повторов), например `set <string>`.

Такое решение в худшем будет работать на дереве типа «бамбук» (у каждого родителя ровно один ребенок). Длина k-й строки будет равна k. Поиск будет производится за $O(\log k)$ в худшем, с добавлением за $O(k)$ в худшем. Общая сложность обработки $\sum_{k=1}^{N} k = O(N^2)$. Но оптимизиции и кэширование значительно ускоряют процесс копирования строк.

Асиптотика: $O(N^2)$.

```
#include <iostream>
#include <vector>
#include <queue>
using namespace std;

vector < vector < vector <int> > > son;
int main() {
    int n;
    cin >> n;
    son.resize(n + 1);
    for (int from = 1; from <= n; from++)
        son[from].resize(26);
    for (int to = 2; to <= n; to++) {
```

```
        int from;
        char ch;
        cin >> from >> ch;
        son[from][ch - 'a'].push_back(to);
    }

    queue <int> q;
    q.push(1);
    int answer = 0;
    while (!q.empty()) {
        int current = q.front();
        q.pop();
        for (char ch = 0; ch < 26; ch++) {
            if (son[current][ch].size() > 0) {
                int dst = son[current][ch][0];
                for (size_t i = 1;
                     i < son[current][ch].size(); i++) {
                    int src = son[current][ch][i];
                    for (char c = 0; c < 26; c++) {
                        for (size_t j = 0;
                             j < son[src][c].size(); j++) {
                             int sonson = son[src][c][j];
                            son[dst][c].push_back(sonson);
                            }
                    }
                }
                q.push(dst);
                answer++;
            }
        }
    }

    cout << answer << endl;
    return 0;
}
```

G. Giant chandelier

Автор: Баев А.Ж.

Центр тяжести люстры:

$$\vec{m} = \frac{1}{n}\sum_{i=1}^{n}\vec{v}_i.$$

Чтоб понять насколько отклоняется люстра вдоль центра тяжести, найдем длины проекций каждого вектора на вектор $\vec{m}$ через скалярное произведение:

$$p_i = (\vec{v}_i, \vec{e}),$$

где $\vec{e} = \frac{\vec{m}}{|\vec{m}|}$.

Ответ на задачу: $\max p_i - \min p_i$.

Асиптотика: $O(N)$.

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

struct Point {
    double x, y, z;
};
double dot(const Point &a, const Point &b) {
```

```
    return a.x * b.x + a.y * b.y + a.z * b.z;
}

Point lamp[100000];
int main() {
    int n;
    cin >> n;
    Point m = {0, 0, 0};
    for (int i = 0 ; i < n; i++) {
        cin >> lamp[i].x >> lamp[i].y >> lamp[i].z;
        m.x += lamp[i].x; m.y += lamp[i].y; m.z += lamp[i].z;
    }
    double length = sqrt(dot(m, m));
    m.x /= length; m.y /= length; m.z /= length;
    double p_min = 0.0, p_max = 0.0;
    for (int i = 0; i < n; i++) {
        double d = dot(lamp[i], m);
        if (p_min > d)
            p_min = d;
        if (p_max < d)
            p_max = d;
    }
    cout << fixed << setprecision(10) << p_max - p_min << endl;
    return 0;
}
```

H. Hyper numbers

Автор: Абдикалыков А.К.

Заметим, что всегд верно $(n, x) = (n, n - x)$. По определению гипер-чисел, не может быть несколько взаимно простых чисел. Значит, $x = n - x$. Тогда $n = 2x$ и $(n, x) = x$. Опять же по опредению эти числа должны быть взаимно просты, то есть $x = 1$, $n = 2$. Таким образом, достаточно проверить, входит ли число 2 в указанный диапазон.

Асиптотика: $O(1)$.

```
#include <iostream>
using namespace std;
int main() {
    long long L, R, answer;
    cin >> L >> R;
    answer = (L == 2);
    cout << answer << endl;
    return 0;
}
```