Открытая личная олимпиада по программированию
Зимний тур 2018
12 декабря 2018

A. Around the problems

Автор: Баев А.Ж.

Сгруппируем периоды по парам, чтобы найти суммарное количество минут, которое прошло от начала олимпиады. Если p нечетное, то количество минут от начала олимпиады равно

$$t = \left[\frac{p-1}{2}\right] * (n_1 + n_2) + m.$$

Если p четное, то количество минут от начала олимпиады равно

$$t = \left[\frac{p-1}{2}\right] * (n_1 + n_2) + n_1 + m.$$

Ответ зависит от остатка при делении t на 7. Если t делится на 7, то ответ G, иначе это ответ $(t-1) \bmod 7$-я задача.

Асимптотика по времени $O(1)$.

```
#include <iostream>

int main() {
    int n1, n2, p, m, sum;
    char answer;

    std::cin >> n1 >> n2 >> p >> m;
    sum = (p - 1) / 2 * (n1 + n2);
    if (p % 2 == 0)
        sum += n1;
    sum += m;
    answer = 'A' + (sum - 1) % 7;

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

B. Be lazy

Автор: Абдикалыков А.К.

Найдем расстояния до ближайших к числам p и q элементов: $mp = \min_i |a_i - p|$ и $mq = \min_i |a_i - q|$. Ответом будет $\min(mp + mq, |p - q|)$.

Асимптотика по времени $O(n)$.

```
#include <iostream>
#include <algorithm>

using namespace std;

int find(int n, int a[], int floor) {
    int dist = 2e9;
    if (floor <= a[0])
        return a[0] - floor;
    if (floor >= a[n - 1])
        return floor - a[n - 1];
    for (int i = 0; i + 1 < n; i++)
        if (a[i] <= floor && floor < a[i + 1]) {
            int current = min(floor - a[i],
```

```
                                a[i + 1] - floor);
            dist = min(dist, current);
        }
    return dist;
}

int main() {
    int n, p, q;
    int a[100000];
    cin >> p >> q >> n;
    for (int i = 0; i < n; i++)
        cin >> a[i];
    int nolift = abs(p - q);
    int plift = find(n, a, p);
    int qlift = find(n, a, q);
    cout << min(nolift, plift + qlift) << endl;
    return 0;
}
```

C. Calculator

Автор: Жусупов Али

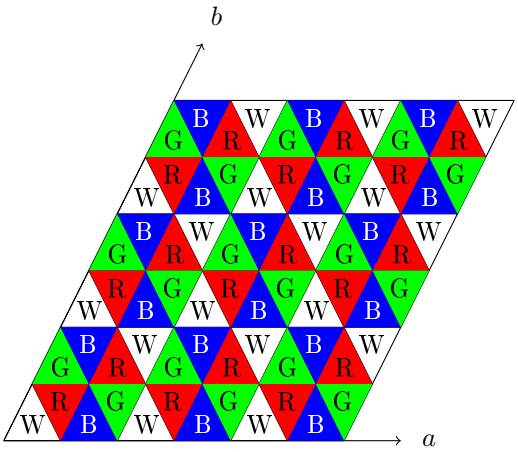
Переберём все числа k от 0 до $2^n - 1$, где n — количество символов данной строки. Число k не подходит, если хотя бы в одной позиции i в строке s будут символ отличный от вопроса (то есть 0 или 1) и он не будет совпадать с i-м битом числа k. Выбор соответствующего бита удобны сделать через битовый сдвиг числа.

Асимптотика по времени $O(2^n \cdot n)$.

```
#include <iostream>
#include <cstring>

using namespace std;

bool check(char mask[], int n, int k) {
    for (int i = 0; i < n; i++) {
        int bit = (k >> (n - 1 - i)) & 1;
        if (mask[i] != '?' && mask[i] != bit)
            return false;
    }
    return true;
}

int main() {
    char s[20];
    int n;
    cin >> s;
    n = strlen(s);
    for (int i = 0; i < n; i++)
        s[i] -= '0';
    for (int k = 0; k < (1 << n); k++)
        if (check(s, n, k))
            cout << k << '\n';
    return 0;
}
```

D. Deep rolling

Автор: Баев А.Ж.



Если $[a]$ и $[b]$ одной четности, то это либо белый, либо красный цвет. Белый цвет бывает в случае, если $[a]$ — четное и $\{a\}+\{b\}<1$ или если $[a]$ — нечетное и $\{a\}+\{b\}>1$. В противном случае, цвет — красный.

Если $[a]$ и $[b]$ разной четности, то это либо синий, либо зеленый цвет. Зеленый цвет бывает в случае, если $[a]$ — четное и $\{a\}+\{b\}<1$ или если $[a]$ — нечетное и $\{a\}+\{b\}>1$. В противном случае, цвет — зеленый.

Асимптотика по времени $O(1)$.

```
#include <iostream>
using namespace std;
int main() {
    double a, b;
    int aa, bb, answer[2][2] = {{"red", "white"},
                                {"blue", "green"}};
    cin >> a >> b;
    aa = a, bb = b;
    double fracsum = (a - aa) + (b - bb);
    int i = (aa % 2 == bb % 2);
    int j = ((aa % 2 == 0) ^ (fracsum < 1.0));
    cout << answer[i][j] << '\n';
    return 0;
}
```

E. Elementary balance

Автор: Баев А.Ж.

Суффиксные суммы считаем за один проход слева направо.

$$l_i = l_{i-1} + a_i$$

Сумма всего массива s равна сумме префиксной суммы l_k, элемента a_k и суффиксной суммы r_k.

$$s = \sum_{i=1}^{k-1} a_i + a_k + \sum_{i=k+1}^{n} a_i = l_k + a_k + r_k$$

Значит префиксные суммы можно выразить через суффиксные.

Ответ $ans = \min_i |l_i - r_i|$ — минимум модуля их разности. Несложно заметить, что префиксные и суффиксные суммы можно не хранить в массивах, а вычислять «на лету».

Асимптотика по времени $O(n)$.

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main() {
```

```
    int n;
    long long a[100000], sum = 0, left = 0, right = 0;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> a[i];
        sum += a[i];
    }
    right = sum;
    left = 0;
    for (int i = 0; i < n; i++) {
        right -= a[i];
        long long value = llabs(right - left);
        if (ans == -1 || ans > value)
            ans = value;
        left += a[i];
    }
    long long ans = -1;
    right = sum;
    left = 0;
    for (int i = 0; i < n; i++) {
        right -= a[i];
        long long value = llabs(right - left);
        if (ans == value)
            cout << i + 1 << ' ';
        left += a[i];
    }
    cout << '\n';
    return 0;
}
```

F. Full overlapping

Предложил: Жусупов Али

Сохраним все левые концы в один массив l, все правые концы — в массив r. Отсортируем оба массива по возрастанию. Заведем переменную-счетчик c для вычисления количества наложений. Теперь пройдемся сканирующей прямой по объединению значений в обоих массивов слева направо. Для этого модифицируем алгоритм слияния двух отсортированных массивов через два указателя i и j. Если текущий правый конец l_i не меньше текущего левого конца r_j, то указатель i передвигаем к следующему правому концу и уменьшаем счетчик c. В противном случае, указатель j передвигаем к следующему левому концу и увеличиваем счетчик c. Максимальное значение c за всё время — ответ на задачу. Обратите внимание, если $l_i = r_j$, то необходимо сдвигать указатель j.

Асимптотика по времени $O(n)$.

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    int n, c = 0, answer = 0, i, j;
    long long l[100000], r[100000], scanline = 0;
    cin >> n;
    for (int i = 0; i < n; i++)
        cin >> l[i] >> r[i];
    sort(l, l + n);
    sort(r, r + n);
    while (i < n && j < n) {
        scanline = min(l[i], r[j]);
        while (j < n && r[j] == scanline)
            j++, c--;
        while (i < n && l[i] == scanline)
            i++, c++;
```

```
        if (answer < c)
            answer = c;
    }
    cout << answer << '\n';
    return 0;
}
```

G. Galaxy number

Автор: Абдикалыков А.К.

Пусть m — искомое число. Тогда $m - 10^{41} < k$ и $m \bmod k \equiv 0$. То есть $m = 10^{41} + r$, где r остаток, дополняющий остаток при делении 10^{41} на k до 0 или k. Таким образом:

$$m = 10^{41} + (k - r) \bmod k$$

где $r = 10^{41} \bmod k$, который можно вычислить наивным алгоритмом, не забывая вычислить остаток при делении на k после каждого умножения.

Стоит обратить внимание, что наличие встроенной длинной арифметики на python позволяет очень написать ответ прямой формулой.

Асимптотика по времени $O(1)$.

```
#include <iostream>
#include <algorithm>
#include <iomanip>

using namespace std;

int main() {
    int len = 42;
    long long k, decpow = 1;
    cin >> k;
    for (int i = 0; i < len - 1; i++)
        decpow = decpow * 10 % k;
    long long m = (k - decpow) % k;
    cout << 1 << setw(len - 1) << setfill('0') << m;
    return 0;
}
```