Last update: October 16, 2023 Original

Continued fractions

Continued fraction is a representation of a real number as a specific convergent sequence of rational numbers. They are useful in competitive programming because they are easy to compute and can be efficiently used to find the best possible rational approximation of the underlying real number (among all numbers whose denominator doesn't exceed a given value).

Besides that, continued fractions are closely related to Euclidean algorithm which makes them useful in a bunch of number-theoretical problems.

Continued fraction representation

Definition

Let $a_0, a_1, \dots, a_k \in \mathbb{Z}$ and $a_1, a_2, \dots, a_k \geq 1$. Then the expression

$$r = a_0 + \frac{1}{a_1 + \frac{1}{\dots + \frac{1}{a_k}}},$$

is called the **continued fraction representation** of the rational number r and is denoted shortly as $r = [a_0; a_1, a_2, \dots, a_k]$.

Example

Let $r = \frac{5}{3}$. There are two ways to represent it as a continued fraction:

$$\begin{align} r = [1;1,1,1] &= 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1}}},\\ r = [1;1,2] &= 1 + \frac{1}{1 + \frac{1}{2}}. \end{align}$$

It can be proven that any rational number can be represented as a continued fraction in exactly 2 ways:

$$r = [a_0; a_1, \dots, a_k, 1] = [a_0; a_1, \dots, a_k + 1].$$

Moreover, the length k of such continued fraction is estimated as $k = O(\log \min(p, q))$ for $r = \frac{p}{q}$.

The reasoning behind this will be clear once we delve into the details of the continued fraction construction.

Definition

Let $a_0, a_1, a_2, \dots$ be an integer sequence such that $a_1, a_2, \dots \geq 1$. Let $r_k = [a_0; a_1, \dots, a_k]$. Then the expression

$$r = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \dots}} = \lim_{k \to \infty} r_k.$$

is called the **continued fraction representation** of the irrational number r and is denoted shortly as $r = [a_0; a_1, a_2, \dots]$.

Note that for $r = [a_0; a_1, \dots]$ and integer k, it holds that $r + k = [a_0 + k; a_1, \dots]$.

Another important observation is that $\frac{1}{r} = [0; a_0, a_1, \dots]$ when $a_0 > 0$ and $\frac{1}{r} = [a_1; a_2, \dots]$ when $a_0 = 0$.

Definition

In the definition above, rational numbers $r_0, r_1, r_2, \dots$ are called the **convergents** of r.

Correspondingly, individual $r_k = [a_0; a_1, \dots, a_k] = \frac{p_k}{q_k}$ is called the k-th **convergent** of r.

Example

Consider $r = [1; 1, 1, 1, \dots]$. It can be proven by induction that $r_k = \frac{F_{k+2}}{F_{k+1}}$, where F_k is the Fibonacci sequence defined as $F_0 = 0$, $F_1 = 1$ and $F_k = F_{k-1} + F_{k-2}$. From the Binet's formula, it is known that

$$r_k = \frac{\phi^{k+2} - \psi^{k+2}}{\phi^{k+1} - \psi^{k+1}},$$

where $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$ is the golden ratio and $\psi = \frac{1-\sqrt{5}}{2} = -\frac{1}{\phi} \approx -0.618$. Thus,

$$r = 1 + \frac{1}{1 + \frac{1}{1 + \dots}} = \lim_{k \to \infty} r_k = \phi = \frac{1+\sqrt{5}}{2}.$$

Note that in this specific case, an alternative way to find r would be to solve the equation

$$r = 1 + \frac{1}{r} \implies r^2 = r + 1.$$

Definition

Let $r_k = [a_0; a_1, \dots, a_{k-1}, a_k]$. The numbers $[a_0; a_1, \dots, a_{k-1}, t]$ for $1 \leq t \leq a_k$ are called **semiconvergents**.

We will typically refer to (semi)convergents that are greater than r as **upper** (semi)convergents and to those that are less than r as **lower** (semi)convergents.

Definition

Complementary to convergents, we define the **complete quotients** as $s_k = [a_k; a_{k+1}, a_{k+2}, \dots]$.

Correspondingly, we will call an individual s_k the k-th complete quotient of r.

From the definitions above, one can conclude that $s_k \geq 1$ for $k \geq 1$.

Treating $[a_0; a_1, \dots, a_k]$ as a formal algebraic expression and allowing arbitrary real numbers instead of a_i, we obtain

$$r = [a_0; a_1, \dots, a_{k-1}, s_k].$$

In particular, $r = [s_0] = s_0$. On the other hand, we can express s_k as

$$s_k = [a_k; s_{k+1}] = a_k + \frac{1}{s_{k+1}},$$

meaning that we can compute $a_k = \lfloor s_k \rfloor$ and $s_{k+1} = (s_k - a_k)^{-1}$ from s_k.

The sequence $a_0, a_1, \dots$ is well-defined unless $s_k = a_k$ which only happens when r is a rational number.

Thus the continued fraction representation is uniquely defined for any irrational number r.

Implementation

In the code snippets we will mostly assume finite continued fractions.

From s_k, the transition to s_{k+1} looks like

$$s_k = \lfloor s_k \rfloor + \frac{1}{s_{k+1}}.$$

From this expression, the next complete quotient s_{k+1} is obtained as

$$s_{k+1} = (s_k - \lfloor s_k \rfloor)^{-1}.$$

For $s_k = \frac{p}{q}$ it means that

$$s_{k+1} = \left(\frac{p}{q} - \left\lfloor \frac{p}{q} \right\rfloor\right)^{-1} = \frac{q}{p - q \cdot \lfloor \frac{p}{q} \rfloor} = \frac{q}{p \bmod q}.$$

Thus, the computation of a continued fraction representation for $r = \frac{p}{q}$ follows the steps of the Euclidean algorithm for p and q.

From this also follows that $\gcd(p_k, q_k) = 1$ for $\frac{p_k}{q_k} = [a_0; a_1, \dots, a_k]$. Hence, convergents are always irreducible.

C++

```cpp
auto fraction(int p, int q) {
    vector<int> a;
    while(q) {
        a.push_back(p / q);
        tie(p, q) = make_pair(q, p % q);
    }
    return a;
}
```

Python

```python
def fraction(p, q):
    a = []
    while q:
        a.append(p // q)
        p, q = q, p % q
    return a
```

Key results

To provide some motivation for further study of continued fraction, we give some key facts now.

Recurrence

For the convergents $r_k = \frac{p_k}{q_k}$, the following recurrence stands, allowing their fast computation:

$$\frac{p_k}{q_k} = \frac{a_k p_{k-1} + p_{k-2}}{a_k q_{k-1} + q_{k-2}},$$

where $\frac{p_{-1}}{q_{-1}} = \frac{1}{0}$ and $\frac{p_{-2}}{q_{-2}} = \frac{0}{1}$.

Deviations

The deviation of $r_k = \frac{p_k}{q_k}$ from r can be generally estimated as

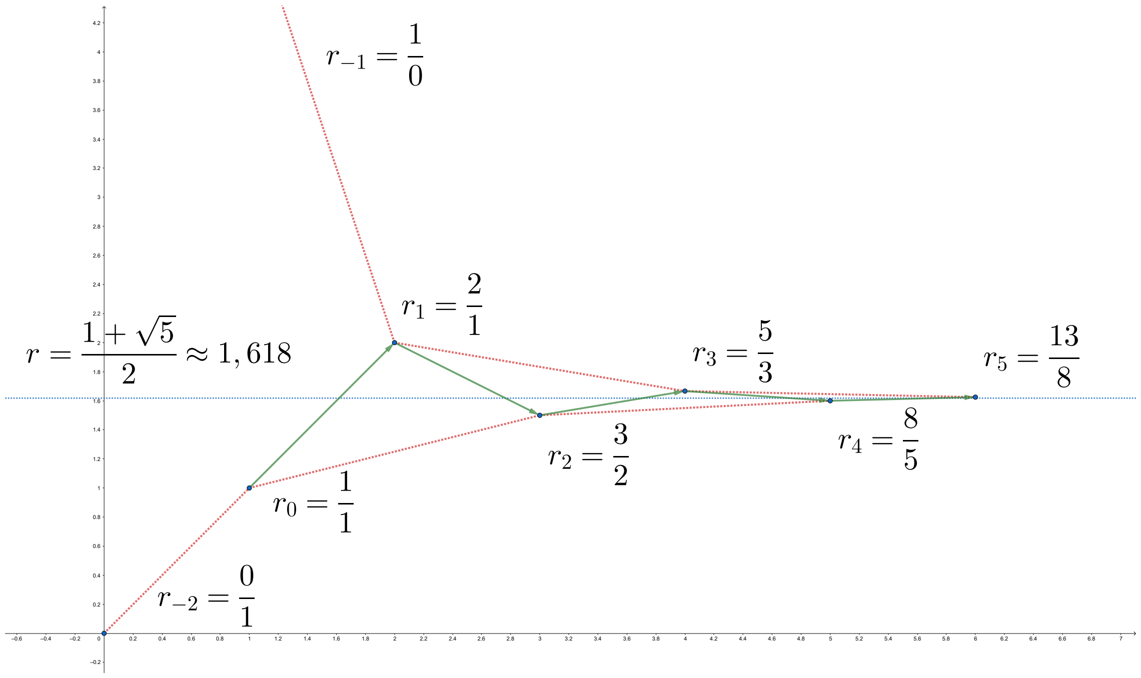
$$\left|\frac{p_k}{q_k} - r\right| \leq \frac{1}{q_k q_{k+1}} \leq \frac{1}{q_k^2}.$$

Multiplying both sides with q_k, we obtain alternate estimation:

$$|p_k - q_k r| \leq \frac{1}{q_{k+1}}.$$

From the recurrence above it follows that q_k grows at least as fast as Fibonacci numbers.

On the picture below you may see the visualization of how convergents r_k approach $r = \frac{1+\sqrt{5}}{2}$:



$r = \frac{1+\sqrt{5}}{2}$ is depicted by blue dotted line. Odd convergents approach it from above and even convergents approach it from below.

Lattice hulls

Consider convex hulls of points above and below the line $y = rx$.

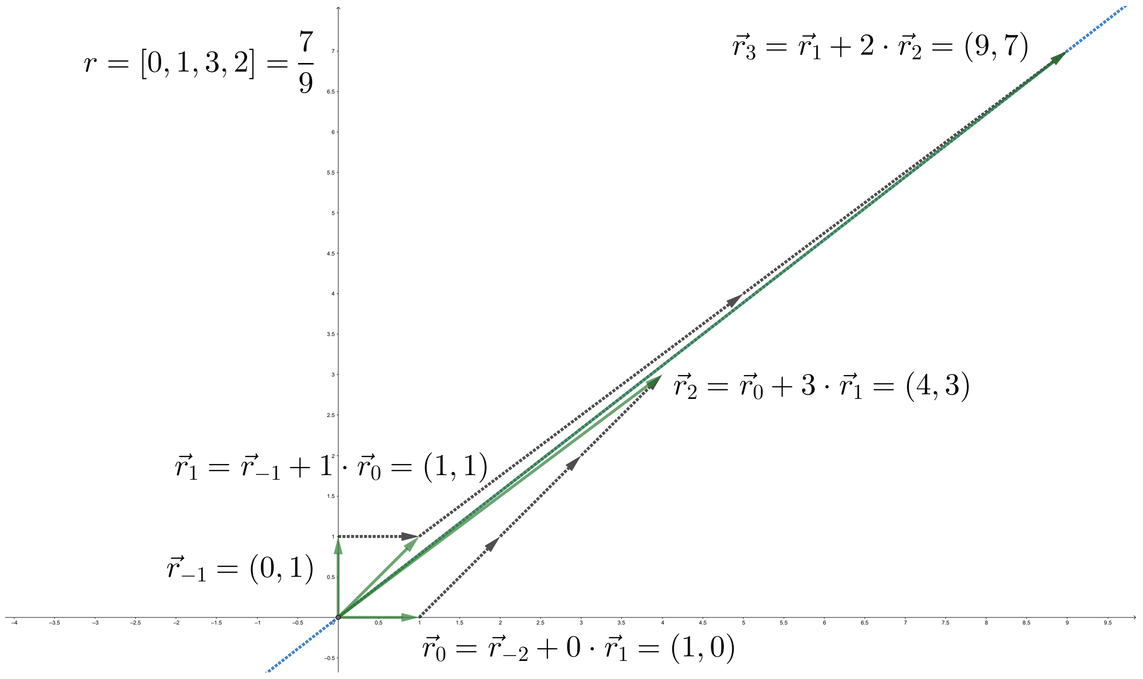
Odd convergents $(q_k; p_k)$ are the vertices of the upper hull, while the even convergents $(q_k; p_k)$ are the vertices of the bottom hull.

All integers vertices on the hulls are obtained as $(q; p)$ such that

$$\frac{p}{q} = \frac{tp_{k-1} + p_{k-2}}{tq_{k-1} + q_{k-2}}$$

for integer $0 \leq t \leq a_k$. In other words, the set of lattice points on the hulls corresponds to the set of semiconvergents.

On the picture below, you may see the convergents and semiconvergents (intermediate gray points) of $r = \frac{9}{7}$.



$r = [0, 1, 3, 2] = \frac{7}{9}$
$\vec{r}_3 = \vec{r}_1 + 2 \cdot \vec{r}_2 = (9, 7)$
$\vec{r}_2 = \vec{r}_0 + 3 \cdot \vec{r}_1 = (4, 3)$
$\vec{r}_1 = \vec{r}_{-1} + 1 \cdot \vec{r}_0 = (1, 1)$
$\vec{r}_{-1} = (0, 1)$
$\vec{r}_0 = \vec{r}_{-2} + 0 \cdot \vec{r}_1 = (1, 0)$

Best approximations

Let $\frac{p}{q}$ be the fraction to minimize $\left|r - \frac{p}{q}\right|$ subject to $q \leq x$ for some x.

Then $\frac{p}{q}$ is a semiconvergent of r.

The last fact allows to find the best rational approximations of r by checking its semiconvergents.

Below you will find the further explanation and a bit of intuition and interpretation for these facts.

Convergents

Let's take a closer look at the convergents that were defined earlier. For $r = [a_0, a_1, a_2, \dots]$, its convergents are

$$\begin{gathered} r_0 = [a_0], \\ r_1 = [a_0, a_1], \\ \dots, \\ r_k = [a_0, a_1, \dots, a_k]. \end{gathered}$$

Convergents are the core concept of continued fractions, so it is important to study their properties.

For the number r, its k-th convergent $r_k = \frac{p_k}{q_k}$ can be computed as

$$r_k = \frac{P_k(a_0, a_1, \dots, a_k)}{P_{k-1}(a_1, \dots, a_k)} = \frac{a_k p_{k-1} + p_{k-2}}{a_k q_{k-1} + q_{k-2}},$$

where $P_k(a_0, \dots, a_k)$ is the continuant, a multivariate polynomial defined as

$$P_k(x_0, x_1, \dots, x_k) = \det \begin{bmatrix} x_k & 1 & 0 & \dots & 0 \\ -1 & x_{k-1} & 1 & \dots & 0 \\ 0 & -1 & x_2 & . & \vdots \\ \vdots & \vdots & . & \ddots & 1 \\ 0 & 0 & \dots & -1 & x_0 \end{bmatrix}.$$

Thus, r_k is a weighted mediant of r_{k-1} and r_{k-2}.

For consistency, two additional convergents $r_{-1} = \frac{1}{0}$ and $r_{-2} = \frac{0}{1}$ are defined.

Detailed explanation

The numerator and the denominator of r_k can be seen as multivariate polynomials of $a_0, a_1, \dots, a_k$:

$$r_k = \frac{P_k(a_0, a_1, \dots, a_k)}{Q_k(a_0, a_1, \dots, a_k)}.$$

From the definition of convergents,

$$r_k = a_0 + \frac{1}{[a_1; a_2, \dots, a_k]} = a_0 + \frac{Q_{k-1}(a_1, \dots, a_k)}{P_{k-1}(a_1, \dots, a_k)} = \frac{a_0 P_{k-1}(a_1, \dots, a_k) + Q_{k-1}(a_1, \dots, a_k)}{P_{k-1}(a_1, \dots, a_k)}.$$

From this follows $Q_k(a_0, \dots, a_k) = P_{k-1}(a_1, \dots, a_k)$. This yields the relation

$$P_k(a_0, \dots, a_k) = a_0 P_{k-1}(a_1, \dots, a_k) + P_{k-2}(a_2, \dots, a_k).$$

Initially, $r_0 = \frac{a_0}{1}$ and $r_1 = \frac{a_0 a_1 + 1}{a_1}$, thus

$$\begin{aligned} P_0(a_0) &= a_0, \\ P_1(a_0, a_1) &= a_0 a_1 + 1. \end{aligned}$$

For consistency, it is convenient to define $P_{-1} = 1$ and $P_{-2} = 0$ and formally say that $r_{-1} = \frac{1}{0}$ and $r_{-2} = \frac{0}{1}$.

From numerical analysis, it is known that the determinant of an arbitrary tridiagonal matrix

$$T_k = \det \begin{bmatrix} a_0 & b_0 & 0 & \dots & 0 \\ c_0 & a_1 & b_1 & \dots & 0 \\ 0 & c_1 & a_2 & . & \vdots \\ \vdots & \vdots & . & \ddots & c_{k-1} \\ 0 & 0 & \dots & b_{k-1} & a_k \end{bmatrix}$$

can be computed recursively as $T_k = a_k T_{k-1} - b_{k-1} c_{k-1} T_{k-2}$. Comparing it to P_k, we get a direct expression

$$P_k = \det \begin{bmatrix} x_k & 1 & 0 & \dots & 0 \\ -1 & x_{k-1} & 1 & \dots & 0 \\ 0 & -1 & x_2 & . & \vdots \\ \vdots & \vdots & . & \ddots & 1 \\ 0 & 0 & \dots & -1 & x_0 \end{bmatrix}.$$

This polynomial is also known as the continuant due to its close relation with continued fraction. The continuant won't change if the sequence on the main diagonal is reversed. This yields an alternative formula to compute it:

$$P_k(a_0, \dots, a_k) = a_k P_{k-1}(a_0, \dots, a_{k-1}) + P_{k-2}(a_0, \dots, a_{k-2}).$$

Implementation

We will compute the convergents as a pair of sequences $p_{-2}, p_{-1}, p_0, p_1, \dots, p_k$ and $q_{-2}, q_{-1}, q_0, q_1, \dots, q_k$:

C++

```cpp
auto convergents(vector<int> a) {
    vector<int> p = {0, 1};
    vector<int> q = {1, 0};
    for(auto it: a) {
        p.push_back(p[p.size() - 1] * it + p[p.size() - 2]);
        q.push_back(q[q.size() - 1] * it + q[q.size() - 2]);
    }
    return make_pair(p, q);
}
```

Python

```python
def convergents(a):
    p = [0, 1]
    q = [1, 0]
    for it in a:
        p.append(p[-1]*it + p[-2])
        q.append(q[-1]*it + q[-2])
    return p, q
```

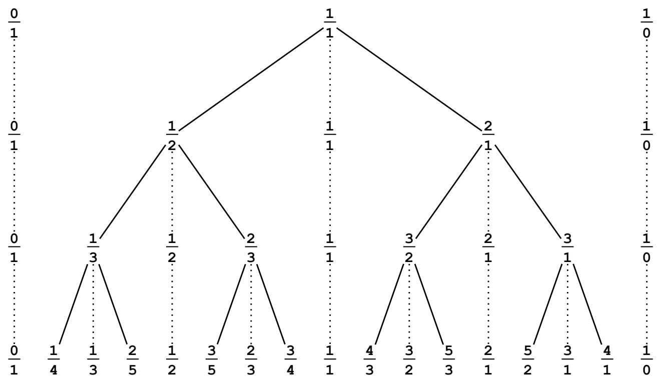
Trees of continued fractions

There are two major ways to unite all possible continued fractions into useful tree structures.

Stern-Brocot tree

The Stern-Brocot tree is a binary search tree that contains all distinct positive rational numbers.

The tree generally looks as follows:



The image by Aaron Rotenberg is licensed under CC BY-SA 3.0

Fractions $\frac{0}{1}$ and $\frac{1}{0}$ are "virtually" kept on the left and right sides of the tree correspondingly.

Then the fraction in a node is a mediant $\frac{a+c}{b+d}$ of two fractions $\frac{a}{b}$ and $\frac{c}{d}$ above it.

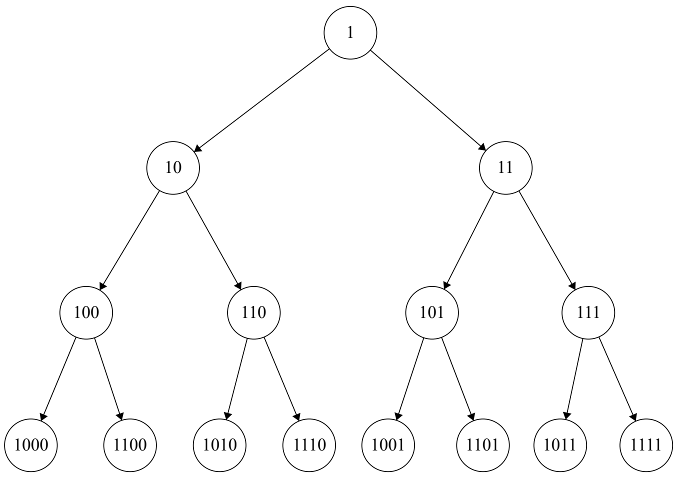
The recurrence $\frac{p_k}{q_k} = \frac{a_k p_{k-1} + p_{k-2}}{a_k q_{k-1} + q_{k-2}}$ means that the continued fraction representation encodes the path to $\frac{p_k}{q_k}$ in the tree. To find $[a_0; a_1, \dots, a_k, 1]$, one has to make a_0 moves to the right, a_1 moves to the left, a_2 moves to the right and so on up to a_k.

The parent of $[a_0; a_1, \dots, a_k, 1]$ then is the fraction obtained by taking one step back in the last used direction.

In other words, it is $[a_0; a_1, \dots, a_k - 1, 1]$ when $a_k > 1$ and $[a_0; a_1, \dots, a_{k-1}, 1]$ when $a_k = 1$.

Thus the children of $[a_0; a_1, \dots, a_k, 1]$ are $[a_0; a_1, \dots, a_k + 1, 1]$ and $[a_0; a_1, \dots, a_k, 1, 1]$.

Let's index the Stern-Brocot tree. The root vertex is assigned an index 1. Then for a vertex v, the index of its left child is assigned by changing the leading bit of v from 1 to 10 and for the right child, it's assigned by changing the leading bit from 1 to 11:



In this indexing, the continued fraction representation of a rational number specifies the run-length encoding of its binary index.

For $\frac{5}{2} = [2; 2] = [2; 1, 1]$, its index is 1011_2 and its run-length encoding, considering bits in the ascending order, is $[2; 1, 1]$.

Another example is $\frac{2}{5} = [0; 2, 2] = [0; 2, 1, 1]$, which has index 1100_2 and its run-length encoding is, indeed, $[0; 2, 2]$.

It is worth noting that the Stern-Brocot tree is, in fact, a treap. That is, it is a binary search tree by $\frac{p}{q}$, but it is a heap by both p and q.

Comparing continued fractions

You're given $A = [a_0; a_1, \dots, a_n]$ and $B = [b_0; b_1, \dots, b_m]$. Which fraction is smaller?

Solution

Assume for now that A and B are irrational and their continued fraction representations denote an infinite descent in the Stern-Brocot tree.

As we already mentioned, in this representation a_0 denotes the number of right turns in the descent, a_1 denotes the number of consequent left turns and so on. Therefore, when we compare a_k and b_k, if $a_k = b_k$ we should just move on to comparing a_{k+1} and b_{k+1}. Otherwise, if we're at right descents, we should check if $a_k < b_k$ and if we're at left descents, we should check if $a_k > b_k$ to tell whether $A < B$.

In other words, for irrational A and B it would be $A < B$ if and only if $(a_0, -a_1, a_2, -a_3, \dots) < (b_0, -b_1, b_2, -b_3, \dots)$ with lexicographical comparison.

Now, formally using ∞ as an element of continued fraction representation it is possible to emulate irrational numbers $A - \varepsilon$ and $A + \varepsilon$, that is, elements that are smaller (greater) than A, but greater (smaller) than any other real number. Specifically, for $A = [a_0; a_1, \dots, a_n]$, one of these two elements can be emulated as $[a_0; a_1, \dots, a_n, \infty]$ and the other can be emulated as $[a_0; a_1, \dots, a_n - 1, 1, \infty]$.

Which one corresponds to $A - \varepsilon$ and which one to $A + \varepsilon$ can be determined by the parity of n or by comparing them as irrational numbers.

Python

```python
# check if a < b assuming that a[-1] = b[-1] = infty and a != b
def less(a, b):
    a = [(-1)**i*a[i] for i in range(len(a))]
    b = [(-1)**i*b[i] for i in range(len(b))]
    return a < b

# [a0; a1, ..., ak] -> [a0, a1, ..., ak-1, 1]
def expand(a):
    if a: # empty a = inf
        a[-1] -= 1
        a.append(1)
    return a

# return a-eps, a+eps
def pm_eps(a):
    b = expand(a.copy())
    a.append(float('inf'))
    b.append(float('inf'))
    return (a, b) if less(a, b) else (b, a)
```

Best inner point

You're given $\frac{0}{1} \leq \frac{p_0}{q_0} < \frac{p_1}{q_1} \leq \frac{1}{0}$. Find the rational number $\frac{p}{q}$ such that $(q; p)$ is lexicographically smallest and $\frac{p_0}{q_0} < \frac{p}{q} < \frac{p_1}{q_1}$.

Solution

In terms of the Stern-Brocot tree it means that we need to find the LCA of $\frac{p_0}{q_0}$ and $\frac{p_1}{q_1}$. Due to the connection between Stern-Brocot tree and continued fraction, this LCA would roughly correspond to the largest common prefix of continued fraction representations for $\frac{p_0}{q_0}$ and $\frac{p_1}{q_1}$.

So, if $\frac{p_0}{q_0} = [a_0; a_1, \dots, a_{k-1}, a_k, \dots]$ and $\frac{p_1}{q_1} = [a_0; a_1, \dots, a_{k-1}, b_k, \dots]$ are irrational numbers, the LCA is $[a_0; a_1, \dots, \min(a_k, b_k) + 1]$.

For rational r_0 and r_1, one of them could be the LCA itself which would require us to casework it. To simplify the solution for rational r_0 and r_1, it is possible to use continued fraction representation of $r_0 + \varepsilon$ and $r_1 - \varepsilon$ which was derived in the previous problem.

Python

```python
# finds lexicographically smallest (q, p)
# such that p0/q0 < p/q < p1/q1
def middle(p0, q0, p1, q1):
    a0 = pm_eps(fraction(p0, q0))[1]
    a1 = pm_eps(fraction(p1, q1))[0]
    a = []
    for i in range(min(len(a0), len(a1))):
        a.append(min(a0[i], a1[i]))
        if a0[i] != a1[i]:
            break
    a[-1] += 1
    p, q = convergents(a)
    return p[-1], q[-1]
```

GCJ 2019, Round 2 - New Elements: Part 2

You're given N positive integer pairs (C_i, J_i). You need to find a positive integer pair (x, y) such that $C_i x + J_i y$ is a strictly increasing sequence.

Among such pairs, find the lexicographically minimum one.

Solution

Rephrasing the statement, $A_i x + B_i y$ must be positive for all i, where $A_i = C_i - C_{i-1}$ and $B_i = J_i - J_{i-1}$.

Among such equations we have four significant groups for $A_i x + B_i y > 0$:

1. $A_i, B_i > 0$ can be ignored since we're looking for $x, y > 0$.
2. $A_i, B_i \leq 0$ would provide "IMPOSSIBLE" as an answer.
3. $A_i > 0$, $B_i \leq 0$. Such constraints are equivalent to $\frac{y}{x} < \frac{A_i}{-B_i}$.
4. $A_i \leq 0$, $B_i > 0$. Such constraints are equivalent to $\frac{y}{x} > \frac{-A_i}{B_i}$.

Let $\frac{p_0}{q_0}$ be the largest $\frac{-A_i}{B_i}$ from the fourth group and $\frac{p_1}{q_1}$ be the smallest $\frac{A_i}{-B_i}$ from the third group.

The problem is now, given $\frac{p_0}{q_0} < \frac{p_1}{q_1}$, find a fraction $\frac{p}{q}$ such that $(q; p)$ is lexicographically smallest and $\frac{p_0}{q_0} < \frac{p}{q} < \frac{p_1}{q_1}$.

Python

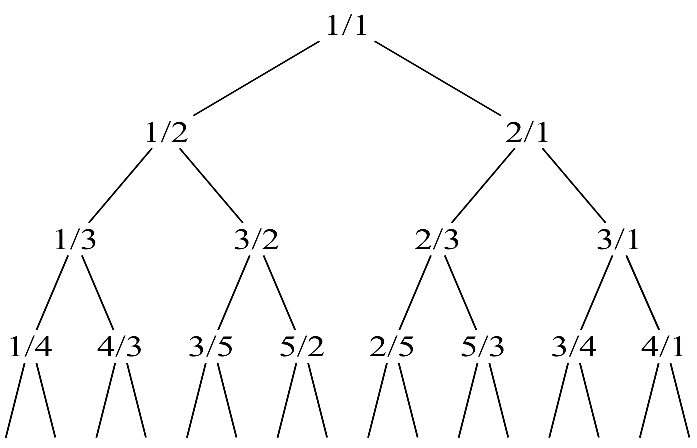
```python
def solve():
    n = int(input())
    C = [0] * n
    J = [0] * n
    # p0/q0 < y/x < p1/q1
    p0, q0 = 0, 1
    p1, q1 = 1, 0
    fail = False
    for i in range(n):
        C[i], J[i] = map(int, input().split())
        if i > 0:
            A = C[i] - C[i-1]
            B = J[i] - J[i-1]
            if A <= 0 and B <= 0:
                fail = True
            elif B > 0 and A < 0: # y/x > (-A)/B if B > 0
                if (-A)*q0 > p0*B:
                    p0, q0 = -A, B
            elif B < 0 and A > 0: # y/x < A/(-B) if B < 0
                if A*q1 < p1*(-B):
                    p1, q1 = A, -B
    if p0*q1 >= p1*q0 or fail:
        return 'IMPOSSIBLE'

    p, q = middle(p0, q0, p1, q1)
    return str(q) + ' ' + str(p)
```

Calkin-Wilf tree

A somewhat simpler way to organize continued fractions in a binary tree is the Calkin-Wilf tree.

The tree generally looks like this:



The image by Olli Niemitalo, Proz is licensed under CC0 1.0

In the root of the tree, the number $\frac{1}{1}$ is located. Then, for the vertex with a number $\frac{p}{q}$, its children are $\frac{p}{p+q}$ and $\frac{p+q}{q}$.

Unlike the Stern-Brocot tree, the Calkin-Wilf tree is not a binary *search* tree, so it can't be used to perform rational binary search.

In the Calkin-Wilf tree, the direct parent of a fraction $\frac{p}{q}$ is $\frac{p-q}{q}$ when $p > q$ and $\frac{p}{q-p}$ otherwise.

For the Stern-Brocot tree, we used the recurrence for convergents. To draw the connection between the continued fraction and the Calkin-Wilf tree, we should recall the recurrence for complete quotients. If $s_k = \frac{p}{q}$, then $s_{k+1} = \frac{q}{p \mod q} = \frac{q}{p-\lfloor p/q \rfloor \cdot q}$.

On the other hand, if we repeatedly go from $s_k = \frac{p}{q}$ to its parent in the Calkin-Wilf tree when $p > q$, we will end up in $\frac{p \mod q}{q} = \frac{1}{s_{k+1}}$. If we continue doing so, we will end up in s_{k+2}, then $\frac{1}{s_{k+3}}$ and so on. From this we can deduce that:

1. When $a_0 > 0$, the direct parent of $[a_0; a_1, \dots, a_k]$ in the Calkin-Wilf tree is $\frac{p-q}{q} = [a_0 - 1; a_1, \dots, a_k]$.
2. When $a_0 = 0$ and $a_1 > 1$, its direct parent is $\frac{p}{q-p} = [0; a_1 - 1, a_2, \dots, a_k]$.
3. And when $a_0 = 0$ and $a_1 = 1$, its direct parent is $\frac{p}{q-p} = [a_2; a_3, \dots, a_k]$.

Correspondingly, children of $\frac{p}{q} = [a_0; a_1, \dots, a_k]$ are

1. $\frac{p+q}{q} = 1 + \frac{p}{q}$, which is $[a_0 + 1; a_1, \dots, a_k]$,
2. $\frac{p}{p+q} = \frac{1}{1+\frac{q}{p}}$, which is $[0, 1, a_0, a_1, \dots, a_k]$ for $a_0 > 0$ and $[0, a_1 + 1, a_2, \dots, a_k]$ for $a_0 = 0$.

Noteworthy, if we enumerate vertices of the Calkin-Wilf tree in the breadth-first search order (that is, the root has a number 1, and the children of the vertex v have indices $2v$ and $2v + 1$ correspondingly), the index of the rational number in the Calkin-Wilf tree would be the same as in the Stern-Brocot tree.

Thus, numbers on the same levels of the Stern-Brocot tree and the Calkin-Wilf tree are the same, but their ordering differs through the bit-reversal permutation.

Convergence

For the number r and its k-th convergent $r_k = \frac{p_k}{q_k}$ the following formula stands:

$$r_k = a_0 + \sum_{i=1}^{k} \frac{(-1)^{i-1}}{q_i q_{i-1}}.$$

In particular, it means that

$$r_k - r_{k-1} = \frac{(-1)^{k-1}}{q_k q_{k-1}}$$

and

$$p_k q_{k-1} - p_{k-1} q_k = (-1)^{k-1}.$$

From this we can conclude that

$$\left| r - \frac{p_k}{q_k} \right| \leq \frac{1}{q_{k+1} q_k} \leq \frac{1}{q_k^2}.$$

The latter inequality is due to the fact that r_k and r_{k+1} are generally located on different sides of r, thus

$$|r - r_k| = |r_k - r_{k+1}| - |r - r_{k+1}| \leq |r_k - r_{k+1}|.$$

Detailed explanation

To estimate $|r - r_k|$, we start by estimating the difference between adjacent convergents. By definition,

$$\frac{p_k}{q_k} - \frac{p_{k-1}}{q_{k-1}} = \frac{p_k q_{k-1} - p_{k-1} q_k}{q_k q_{k-1}}.$$

Replacing p_k and q_k in the numerator with their recurrences, we get

$$\begin{aligned} p_k q_{k-1} - p_{k-1} q_k &= (a_k p_{k-1} + p_{k-2}) q_{k-1} - p_{k-1}(a_k q_{k-1} + q_{k-2}) \\ &= p_{k-2} q_{k-1} - p_{k-1} q_{k-2}, \end{aligned}$$

thus the numerator of $r_k - r_{k-1}$ is always the negated numerator of $r_{k-1} - r_{k-2}$. It, in turn, equals to 1 for

$$r_1 - r_0 = \left(a_0 + \frac{1}{a_1}\right) - a_0 = \frac{1}{a_1},$$

thus

$$r_k - r_{k-1} = \frac{(-1)^{k-1}}{q_k q_{k-1}}.$$

This yields an alternative representation of r_k as a partial sum of infinite series:

$$r_k = (r_k - r_{k-1}) + \dots + (r_1 - r_0) + r_0 = a_0 + \sum_{i=1}^{k} \frac{(-1)^{i-1}}{q_i q_{i-1}}.$$

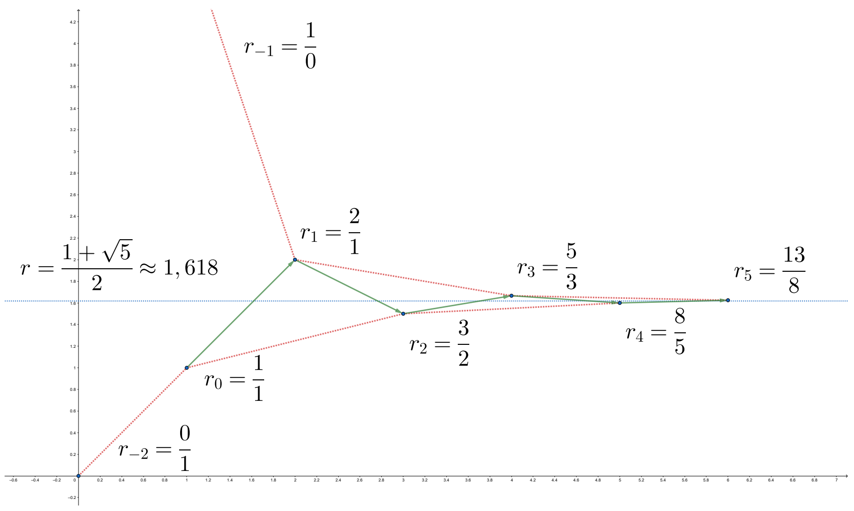
From the recurrent relation it follows that q_k monotonously increases at least as fast as Fibonacci numbers, thus

$$r = \lim_{k \to \infty} r_k = a_0 + \sum_{i=1}^{\infty} \frac{(-1)^{i-1}}{q_i q_{i-1}}$$

is always well-defined, as the underlying series always converge. Noteworthy, the residual series

$$r - r_k = \sum_{i=k+1}^{\infty} \frac{(-1)^{i-1}}{q_i q_{i-1}}$$

has the same sign as $(-1)^k$ due to how fast $q_i q_{i-1}$ decreases. Hence even-indexed r_k approach r from below while odd-indexed r_k approach it from above:



Convergents of $r = \phi = \frac{1+\sqrt{5}}{2} = [1; 1, 1, \dots]$ and their distance from r.

From this picture we can see that

$$|r - r_k| = |r_k - r_{k+1}| - |r - r_{k+1}| \leq |r_k - r_{k+1}|,$$

thus the distance between r and r_k is never larger than the distance between r_k and r_{k+1}:

$$\left|r - \frac{p_k}{q_k}\right| \leq \frac{1}{q_k q_{k+1}} \leq \frac{1}{q_k^2}.$$

Extended Euclidean?

You're given $A, B, C \in \mathbb{Z}$. Find $x, y \in \mathbb{Z}$ such that $Ax + By = C$.

Solution

Although this problem is typically solved with the extended Euclidean algorithm, there is a simple and straightforward solution with continued fractions.

Let $\frac{A}{B} = [a_0; a_1, \dots, a_k]$. It was proved above that $p_k q_{k-1} - p_{k-1} q_k = (-1)^{k-1}$. Substituting p_k and q_k with A and B, we get

$$Aq_{k-1} - Bp_{k-1} = (-1)^{k-1} g,$$

where $g = \gcd(A, B)$. If C is divisible by g, then the solution is $x = (-1)^{k-1} \frac{C}{g} q_{k-1}$ and $y = (-1)^k \frac{C}{g} p_{k-1}$.

Python

```python
# return (x, y) such that Ax+By=C
# assumes that such (x, y) exists
def dio(A, B, C):
    p, q = convergents(fraction(A, B))
    C //= A // p[-1] # divide by gcd(A, B)
    t = (-1) if len(p) % 2 else 1
    return t*C*q[-2], -t*C*p[-2]
```

Linear fractional transformations

Another important concept for continued fractions are the so-called linear fractional transformations.

Definition

A **linear fractional transformation** is a function $f : \mathbb{R} \to \mathbb{R}$ such that $f(x) = \frac{ax+b}{cx+d}$ for some $a, b, c, d \in \mathbb{R}$.

A composition $(L_0 \circ L_1)(x) = L_0(L_1(x))$ of linear fractional transforms $L_0(x) = \frac{a_0 x + b_0}{c_0 x + d_0}$ and $L_1(x) = \frac{a_1 x + b_1}{c_1 x + d_1}$ is itself a linear fractional transform:

$$\frac{a_0 \frac{a_1 x + b_1}{c_1 x + d_1} + b_0}{c_0 \frac{a_1 x + b_1}{c_1 x + d_1} + d_0} = \frac{a_0(a_1 x + b_1) + b_0(c_1 x + d_1)}{c_0(a_1 x + b_1) + d_0(c_1 x + d_1)} = \frac{(a_0 a_1 + b_0 c_1)x + (a_0 b_1 + b_0 d_1)}{(c_0 a_1 + d_0 c_1)x + (c_0 b_1 + d_0 d_1)}.$$

Inverse of a linear fractional transform, is also a linear fractional transform:

$$y = \frac{ax+b}{cx+d} \iff y(cx+d) = ax+b \iff x = -\frac{dy-b}{cy-a}.$$

DMOPC '19 Contest 7 P4 - Bob and Continued Fractions

You're given an array of positive integers $a_1, \dots, a_n$. You need to answer m queries. Each query is to compute $[a_l; a_{l+1}, \dots, a_r]$.

Solution

We can solve this problem with the segment tree if we're able to concatenate continued fractions.

It's generally true that $[a_0; a_1, \dots, a_k, b_0, b_1, \dots, b_k] = [a_0; a_1, \dots, a_k, [b_1; b_2, \dots, b_k]]$.

Let's denote $L_k(x) = [a_k; x] = a_k + \frac{1}{x} = \frac{a_k \cdot x + 1}{1 \cdot x + 0}$. Note that $L_k(\infty) = a_k$. In this notion, it holds that

$$[a_0; a_1, \dots, a_k, x] = [a_0; [a_1; [\dots; [a_k; x]]]] = (L_0 \circ L_1 \circ \dots \circ L_k)(x) = \frac{p_k x + p_{k-1}}{q_k x + q_{k-1}}.$$

Thus, the problem boils down to the computation of

$$(L_l \circ L_{l+1} \circ \dots \circ L_r)(\infty).$$

Composition of transforms is associative, so it's possible to compute in each node of a segment tree the composition of transforms in its subtree.

Linear fractional transformation of a continued fraction

Let $L(x) = \frac{ax+b}{cx+d}$. Compute the continued fraction representation $[b_0; b_1, \dots, b_m]$ of $L(A)$ for $A = [a_0; a_1, \dots, a_n]$.

This allows to compute $A + \frac{p}{q} = \frac{qA+p}{q}$ *and* $A \cdot \frac{p}{q} = \frac{pA}{q}$ *for any* $\frac{p}{q}$.

Solution

As we noted above, $[a_0; a_1, \dots, a_k] = (L_{a_0} \circ L_{a_1} \circ \dots \circ L_{a_k})(\infty)$, hence $L([a_0; a_1, \dots, a_k]) = (L \circ L_{a_0} \circ L_{a_1} \circ \dots L_{a_k})(\infty)$.

Hence, by consequentially adding L_{a_0}, L_{a_1} and so on we would be able to compute

$$(L \circ L_{a_0} \circ \dots \circ L_{a_k})(x) = L\left(\frac{p_k x + p_{k-1}}{q_k x + q_{k-1}}\right) = \frac{a_k x + b_k}{c_k x + d_k}.$$

Since $L(x)$ is invertible, it is also monotonous in x. Therefore, for any $x \geq 0$ it holds that $L(\frac{p_k x + p_{k-1}}{q_k x + q_{k-1}})$ is between $L(\frac{p_k}{q_k}) = \frac{a_k}{c_k}$ and $L(\frac{p_{k-1}}{q_{k-1}}) = \frac{b_k}{d_k}$.

Moreover, for $x = [a_{k+1}; \dots, a_n]$ it is equal to $L(A)$. Hence, $b_0 = \lfloor L(A) \rfloor$ is between $\lfloor L(\frac{p_k}{q_k}) \rfloor$ and $\lfloor L(\frac{p_{k-1}}{q_{k-1}}) \rfloor$. When they're equal, they're also equal to b_0.

Note that $L(A) = (L_{b_0} \circ L_{b_1} \circ \dots \circ L_{b_m})(\infty)$. Knowing b_0, we can compose $L_{b_0}^{-1}$ with the current transform and continue adding $L_{a_{k+1}}$, $L_{a_{k+2}}$ and so on, looking for new floors to agree, from which we would be able to deduce b_1 and so on until we recover all values of $[b_0; b_1, \dots, b_m]$.

Continued fraction arithmetics

Let $A = [a_0; a_1, \dots, a_n]$ and $B = [b_0; b_1, \dots, b_m]$. Compute the continued fraction representations of $A + B$ and $A \cdot B$.

Solution

Idea here is similar to the previous problem, but instead of $L(x) = \frac{ax+b}{cx+d}$ you should consider bilinear fractional transform $L(x, y) = \frac{axy+bx+cy+d}{exy+fx+gy+h}$.

Rather than $L(x) \mapsto L(L_{a_k}(x))$ you would change your current transform as $L(x, y) \mapsto L(L_{a_k}(x), y)$ or $L(x, y) \mapsto L(x, L_{b_k}(y))$.

Then, you check if $\lfloor \frac{a}{e} \rfloor = \lfloor \frac{b}{f} \rfloor = \lfloor \frac{c}{g} \rfloor = \lfloor \frac{d}{h} \rfloor$ and if they all agree, you use this value as c_k in the resulting fraction and change the transform as

$$L(x, y) \mapsto \frac{1}{L(x, y) - c_k}.$$

Definition

A continued fraction $x = [a_0; a_1, \dots]$ is said to be **periodic** if $x = [a_0; a_1, \dots, a_k, x]$ for some k.

A continued fraction $x = [a_0; a_1, \dots]$ is said to be **eventually periodic** if $x = [a_0; a_1, \dots, a_k, y]$, where y is periodic.

For $x = [1; 1, 1, \dots]$ it holds that $x = 1 + \frac{1}{x}$, thus $x^2 = x + 1$. There is a generic connection between periodic continued fractions and quadratic equations. Consider the following equation:

$$x = [a_0; a_1, \dots, a_k, x].$$

On one hand, this equation means that the continued fraction representation of x is periodic with the period $k+1$.

On the other hand, using the formula for convergents, this equation means that

$$x = \frac{p_k x + p_{k-1}}{q_k x + q_{k-1}}.$$

That is, x is a linear fractional transformation of itself. It follows from the equation that x is a root of the second degree equation:

$$q_k x^2 + (q_{k-1} - p_k)x - p_{k-1} = 0.$$

Similar reasoning stands for continued fractions that are eventually periodic, that is $x = [a_0; a_1, \dots, a_k, y]$ for $y = [b_0; b_1, \dots, b_k, y]$. Indeed, from first equation we derive that $x = L_0(y)$ and from second equation that $y = L_1(y)$, where L_0 and L_1 are linear fractional transformations. Therefore,

$$x = (L_0 \circ L_1)(y) = (L_0 \circ L_1 \circ L_0^{-1})(x).$$

One can further prove (and it was first done by Lagrange) that for arbitrary quadratic equation $ax^2 + bx + c = 0$ with integer coefficients, its solution x is an eventually periodic continued fraction.

Quadratic irrationality

Find the continued fraction of $\alpha = \frac{x + y\sqrt{n}}{z}$ where $x, y, z, n \in \mathbb{Z}$ and $n > 0$ is not a perfect square.

Solution

For the k-th complete quotient s_k of the number it generally holds that

$$\alpha = [a_0; a_1, \dots, a_{k-1}, s_k] = \frac{s_k p_{k-1} + p_{k-2}}{s_k q_{k-1} + q_{k-2}}.$$

Therefore,

$$s_k = -\frac{\alpha q_{k-1} - p_{k-1}}{\alpha q_k - p_k} = -\frac{q_{k-1} y \sqrt{n} + (x q_{k-1} - z p_{k-1})}{q_k y \sqrt{n} + (x q_k - z p_k)}.$$

Multiplying the numerator and denominator by $(xq_k - zp_k) - q_k y \sqrt{n}$, we'll get rid of $\sqrt{n}$ in the denominator, thus the complete quotients are of form

$$s_k = \frac{x_k + y_k \sqrt{n}}{z_k}.$$

Let's find s_{k+1}, assuming that s_k is known.

First of all, $a_k = \lfloor s_k \rfloor = \left\lfloor \frac{x_k + y_k \lfloor \sqrt{n} \rfloor}{z_k} \right\rfloor$. Then,

$$s_{k+1} = \frac{1}{s_k - a_k} = \frac{z_k}{(x_k - z_k a_k) + y_k \sqrt{n}} = \frac{z_k (x_k - y_k a_k) - y_k z_k \sqrt{n}}{(x_k - y_k a_k)^2 - y_k^2 n}.$$

Thus, if we denote $t_k = x_k - y_k a_k$, it will hold that

$$\begin{aligned} x_{k+1} &= & z_k t_k, \\ y_{k+1} &= & -y_k z_k, \\ z_{k+1} &= & t_k^2 - y_k^2 n. \end{aligned}$$

Nice thing about such representation is that if we reduce $x_{k+1}, y_{k+1}, z_{k+1}$ by their greatest common divisor, the result would be unique. Therefore, we may use it to check whether the current state has already been repeated and also to check where was the previous index that had this state.

Below is the code to compute the continued fraction representation for $\alpha = \sqrt{n}$:

Python

```python
# compute the continued fraction of sqrt(n)
def sqrt(n):
    n0 = math.floor(math.sqrt(n))
    x, y, z = 1, 0, 1
    a = []
    def step(x, y, z):
        a.append((x * n0 + y) // z)
        t = y - a[-1]*z
        x, y, z = -z*x, z*t, t**2 - n*x**2
        g = math.gcd(x, math.gcd(y, z))
        return x // g, y // g, z // g

    used = dict()
    for i in range(n):
        used[x, y, z] = i
        x, y, z = step(x, y, z)
        if (x, y, z) in used:
            return a
```

Using the same `step` function but different initial x, y and z it is possible to compute it for arbitrary $\frac{x+y\sqrt{n}}{z}$.

Tavrida NU Akai Contest - Continued Fraction

You're given x and k, x is not a perfect square. Let $\sqrt{x} = [a_0; a_1, \dots]$, find $\frac{p_k}{q_k} = [a_0; a_1, \dots, a_k]$ for $0 \le k \le 10^9$.

Solution

After computing the period of $\sqrt{x}$, it is possible to compute a_k using binary exponentiation on the linear fractional transformation induced by the continued fraction representation. To find the resulting transformation, you compress the period of size T into a single transformation and repeat it $\lfloor \frac{k-1}{T} \rfloor$ times, after which you manually combine it with the remaining transformations.

Python

```python
x, k = map(int, input().split())

mod = 10**9+7

# compose (A[0]*x + A[1]) / (A[2]*x + A[3]) and (B[0]*x + B[1]) / (B[2]*x + B[3])
def combine(A, B):
    return [t % mod for t in [A[0]*B[0]+A[1]*B[2], A[0]*B[1]+A[1]*B[3], A[2]*B[0]+A[3]*B[2], A[2]*B[1]+A[3]*B[3]]]

A = [1, 0, 0, 1] # (x + 0) / (0*x + 1) = x

a = sqrt(x)

T = len(a) - 1 # period of a

# apply ak + 1/x = (ak*x+1)/(1x+0) to (Ax + B) / (Cx + D)
for i in reversed(range(1, len(a))):
    A = combine([a[i], 1, 1, 0], A)

def bpow(A, n):
    return [1, 0, 0, 1] if not n else combine(A, bpow(A, n-1)) if n % 2 else bpow(combine(A, A), n // 2)


C = (0, 1, 0, 0) # = 1 / 0
while k % T:
    i = k % T
    C = combine([a[i], 1, 1, 0], C)
    k -= 1

C = combine(bpow(A, k // T), C)
C = combine([a[0], 1, 1, 0], C)
print(str(C[1]) + '/' + str(C[3]))
```

Geometric interpretation

Let $\vec r_k = (q_k; p_k)$ for the convergent $r_k = \frac{p_k}{q_k}$. Then, the following recurrence holds:

$$\vec r_k = a_k \vec r_{k-1} + \vec r_{k-2}.$$

Let $\vec r = (1; r)$. Then, each vector $(x; y)$ corresponds to the number that is equal to its slope coefficient $\frac{y}{x}$.

With the notion of pseudoscalar product $(x_1; y_1) \times (x_2; y_2) = x_1 y_2 - x_2 y_1$, it can be shown (see the explanation below) that

$$s_k = -\frac{\vec r_{k-2} \times \vec r}{\vec r_{k-1} \times \vec r} = \left|\frac{\vec r_{k-2} \times \vec r}{\vec r_{k-1} \times \vec r}\right|.$$

The last equation is due to the fact that r_{k-1} and r_{k-2} lie on the different sides of r, thus pseudoscalar products of $\vec r_{k-1}$ and $\vec r_{k-2}$ with $\vec r$ have distinct signs. With $a_k = \lfloor s_k \rfloor$ in mind, formula for $\vec r_k$ now looks like

$$\vec r_k = \vec r_{k-2} + \left\lfloor \left| \frac{\vec r \times \vec r_{k-2}}{\vec r \times \vec r_{k-1}} \right| \right\rfloor \vec r_{k-1}.$$

Note that $\vec r_k \times r = (q; p) \times (1; r) = qr - p$, thus

$$a_k = \left\lfloor \left| \frac{q_{k-1} r - p_{k-1}}{q_{k-2} r - p_{k-2}} \right| \right\rfloor.$$

Explanation

As we have already noted, $a_k = \lfloor s_k \rfloor$, where $s_k = [a_k; a_{k+1}, a_{k+2}, \dots]$. On the other hand, from the convergent recurrence we derive that

$$r = [a_0; a_1, \dots, a_{k-1}, s_k] = \frac{s_k p_{k-1} + p_{k-2}}{s_k q_{k-1} + q_{k-2}}.$$

In vector form, it rewrites as

$$\vec r \parallel s_k \vec r_{k-1} + \vec r_{k-2},$$

meaning that $\vec r$ and $s_k \vec r_{k-1} + \vec r_{k-2}$ are collinear (that is, have the same slope coefficient). Taking the pseudoscalar product of both parts with $\vec r$, we get

$$0 = s_k (\vec r_{k-1} \times \vec r) + (\vec r_{k-2} \times \vec r),$$

which yields the final formula

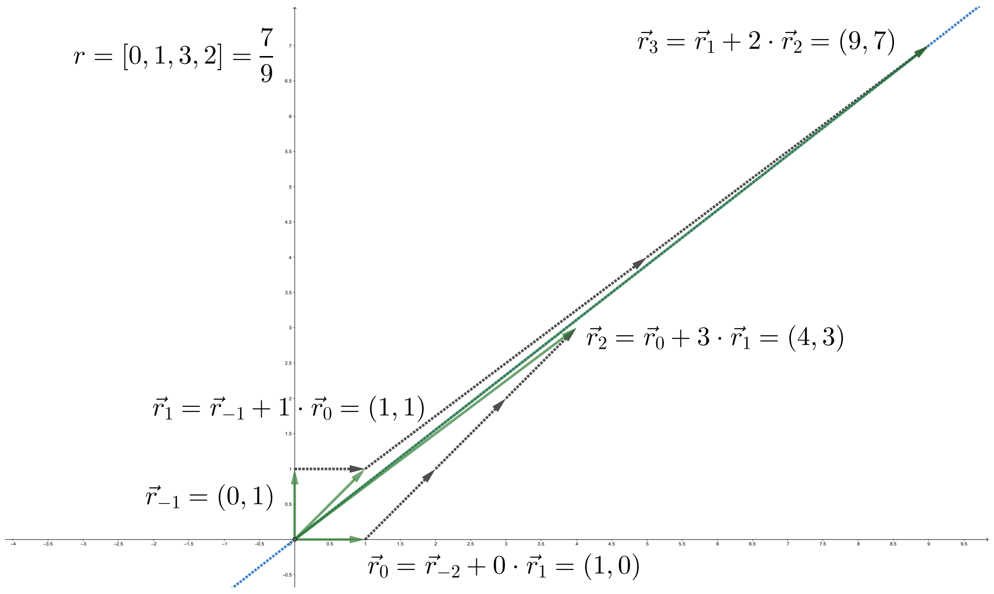
$$s_k = -\frac{\vec r_{k-2} \times \vec r}{\vec r_{k-1} \times \vec r}.$$

Nose stretching algorithm

Each time you add $\vec r_{k-1}$ to the vector $\vec p$, the value of $\vec p \times \vec r$ is increased by $\vec r_{k-1} \times \vec r$.

Thus, $a_k = \lfloor s_k \rfloor$ is the maximum integer number of $\vec r_{k-1}$ vectors that can be added to $\vec r_{k-2}$ without changing the sign of the cross product with $\vec r$.

In other words, a_k is the maximum integer number of times you can add $\vec r_{k-1}$ to $\vec r_{k-2}$ without crossing the line defined by $\vec r$:



$r = [0, 1, 3, 2] = \frac{7}{9}$
$\vec r_3 = \vec r_1 + 2 \cdot \vec r_2 = (9, 7)$
$\vec r_2 = \vec r_0 + 3 \cdot \vec r_1 = (4, 3)$
$\vec r_1 = \vec r_{-1} + 1 \cdot \vec r_0 = (1, 1)$
$\vec r_{-1} = (0, 1)$
$\vec r_0 = \vec r_{-2} + 0 \cdot \vec r_1 = (1, 0)$

Convergents of $r = \frac{7}{9} = [0; 1, 3, 2]$. Semiconvergents correspond to intermediate points between gray arrows.

On the picture above, $\vec r_2 = (4; 3)$ is obtained by repeatedly adding $\vec r_1 = (1; 1)$ to $\vec r_0 = (1; 0)$.

When it is not possible to further add $\vec r_1$ to $\vec r_0$ without crossing the $y = rx$ line, we go to the other side and repeatedly add $\vec r_2$ to $\vec r_1$ to obtain $\vec r_3 = (9; 7)$.

This procedure generates exponentially longer vectors, that approach the line.

For this property, the procedure of generating consequent convergent vectors was dubbed the **nose stretching algorithm** by Boris Delaunay.

If we look on the triangle drawn on points $\vec r_{k-2}$, $\vec r_k$ and $\vec 0$ we will notice that its doubled area is

$$|\vec r_{k-2} \times \vec r_k| = |\vec r_{k-2} \times (\vec r_{k-2} + a_k \vec r_{k-1})| = a_k |\vec r_{k-2} \times \vec r_{k-1}| = a_k.$$

Combined with the Pick's theorem, it means that there are no lattice points strictly inside the triangle and the only lattice points on its border are $\vec 0$ and $\vec r_{k-2} + t \cdot \vec r_{k-1}$ for all integer t such that $0 \leq t \leq a_k$. When joined for all possible k it means that there are no integer points in the space between polygons formed by even-indexed and odd-indexed convergent vectors.

This, in turn, means that $\vec r_k$ with odd coefficients form a convex hull of lattice points with $x \geq 0$ above the line $y = rx$, while $\vec r_k$ with even coefficients form a convex hull of lattice points with $x > 0$ below the line $y = rx$.

Definition

These polygons are also known as **Klein polygons**, named after Felix Klein who first suggested this geometric interpretation to the continued fractions.

Problem examples

Now that the most important facts and concepts were introduced, it is time to delve into specific problem examples.

Convex hull under the line

Find the convex hull of lattice points $(x; y)$ such that $0 \leq x \leq N$ and $0 \leq y \leq rx$ for $r = [a_0; a_1, \dots, a_k] = \frac{p_k}{q_k}$.

Solution

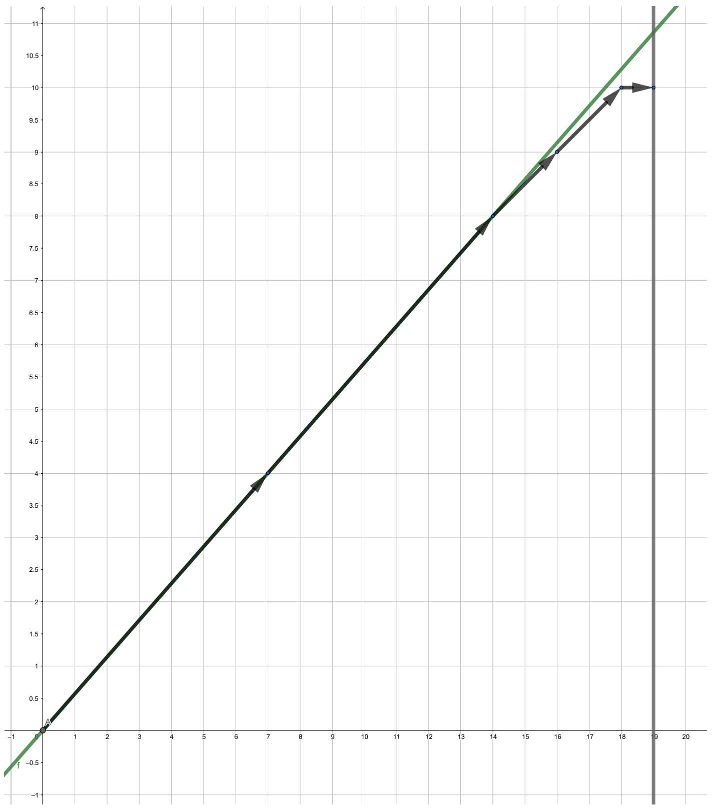
If we were considering the unbounded set $0 \leq x$, the upper convex hull would be given by the line $y = rx$ itself.

However, with additional constraint $x \leq N$ we'd need to eventually deviate from the line to maintain proper convex hull.

Let $t = \lfloor \frac{N}{q_k} \rfloor$, then first t lattice points on the hull after $(0;0)$ are $\alpha \cdot (q_k; p_k)$ for integer $1 \leq \alpha \leq t$.

However $(t+1)(q_k; p_k)$ can't be next lattice point since $(t+1)q_k$ is greater than N.

To get to the next lattice points in the hull, we should get to the point $(x; y)$ which diverges from $y = rx$ by the smallest margin, while maintaining $x \leq N$.



Convex hull of lattice points under $y = \frac{4}{7}x$ for $0 \leq x \leq 19$ consists of points
$(0;0), (7;4), (14;8), (16;9), (18;10), (19;10)$.

Let $(x; y)$ be the last current point in the convex hull. Then the next point $(x'; y')$ is such that $x' \leq N$ and $(x'; y') - (x; y) = (\Delta x; \Delta y)$ is as close to the line $y = rx$ as possible. In other words, $(\Delta x; \Delta y)$ maximizes $r\Delta x - \Delta y$ subject to $\Delta x \leq N - x$ and $\Delta y \leq r\Delta x$.

Points like that lie on the convex hull of lattice points below $y = rx$. In other words, $(\Delta x; \Delta y)$ must be a lower semiconvergent of r.

That being said, $(\Delta x; \Delta y)$ is of form $(q_{i-1}; p_{i-1}) + t \cdot (q_i; p_i)$ for some odd number i and $0 \leq t < a_i$.

To find such i, we can traverse all possible i starting from the largest one and use $t = \lfloor \frac{N - x - q_{i-1}}{q_i} \rfloor$ for i such that $N - x - q_{i-1} \geq 0$.

With $(\Delta x; \Delta y) = (q_{i-1}; p_{i-1}) + t \cdot (q_i; p_i)$, the condition $\Delta y \leq r\Delta x$ would be preserved by semiconvergent properties.

And $t < a_i$ would hold because we already exhausted semiconvergents obtained from $i + 2$, hence $x + q_{i-1} + a_i q_i = x + q_{i+1}$ is greater than N.

Now that we may add $(\Delta x; \Delta y)$, to $(x; y)$ for $k = \lfloor \frac{N - x}{\Delta x} \rfloor$ times before we exceed N, after which we would try the next semiconvergent.

C++

```cpp
// returns [ah, ph, qh] such that points r[i]=(ph[i], qh[i]) constitute upper convex hull
// of lattice points on 0 <= x <= N and 0 <= y <= r * x, where r = [a0; a1, a2, ...]
// and there are ah[i]-1 integer points on the segment between r[i] and r[i+1]
auto hull(auto a, int N) {
    auto [p, q] = convergents(a);
    int t = N / q.back();
    vector ah = {t};
    vector ph = {0, t*p.back()};
    vector qh = {0, t*q.back()};

    for(int i = q.size() - 1; i >= 0; i--) {
        if(i % 2) {
            while(qh.back() + q[i - 1] <= N) {
                t = (N - qh.back() - q[i - 1]) / q[i];
                int dp = p[i - 1] + t * p[i];
                int dq = q[i - 1] + t * q[i];
                int k = (N - qh.back()) / dq;
                ah.push_back(k);
                ph.push_back(ph.back() + k * dp);
                qh.push_back(qh.back() + k * dq);
            }
        }
    }
    return make_tuple(ah, ph, qh);
}
```

Python

```python
# returns [ah, ph, qh] such that points r[i]=(ph[i], qh[i]) constitute upper convex hull
# of lattice points on 0 <= x <= N and 0 <= y <= r * x, where r = [a0; a1, a2, ...]
# and there are ah[i]-1 integer points on the segment between r[i] and r[i+1]
def hull(a, N):
    p, q = convergents(a)
    t = N // q[-1]
    ah = [t]
    ph = [0, t*p[-1]]
    qh = [0, t*q[-1]]
    for i in reversed(range(len(q))):
        if i % 2 == 1:
            while qh[-1] + q[i-1] <= N:
                t = (N - qh[-1] - q[i-1]) // q[i]
                dp = p[i-1] + t*p[i]
                dq = q[i-1] + t*q[i]
                k = (N - qh[-1]) // dq
                ah.append(k)
                ph.append(ph[-1] + k * dp)
                qh.append(qh[-1] + k * dq)
    return ah, ph, qh
```

Timus - Crime and Punishment

You're given integer numbers A, B and N. Find $x \geq 0$ and $y \geq 0$ such that $Ax + By \leq N$ and $Ax + By$ is the maximum possible.

Solution

In this problem it holds that $1 \le A, B, N \le 2 \cdot 10^9$, so it can be solved in $O(\sqrt{N})$. However, there is $O(\log N)$ solution with continued fractions.

For our convenience, we will invert the direction of x by doing a substitution $x \mapsto \lfloor \frac{N}{A} \rfloor - x$, so that now we need to find the point $(x; y)$ such that $0 \le x \le \lfloor \frac{N}{A} \rfloor$, $By - Ax \le N \mod A$ and $By - Ax$ is the maximum possible. Optimal y for each x has a value of $\lfloor \frac{Ax + (N \bmod A)}{B} \rfloor$.

To treat it more generically, we will write a function that finds the best point on $0 \le x \le N$ and $y = \lfloor \frac{Ax+B}{C} \rfloor$.

Core solution idea in this problem essentially repeats the previous problem, but instead of using lower semiconvergents to diverge from line, you use upper semiconvergents to get closer to the line without crossing it and without violating $x \le N$. Unfortunately, unlike the previous problem, you need to make sure that you don't cross the $y = \frac{Ax+B}{C}$ line while getting closer to it, so you should keep it in mind when calculating semiconvergent's coefficient t.

Python

```python
# (x, y) such that y = (A*x+B) // C,
# Cy - Ax is max and 0 <= x <= N.
def closest(A, B, C, N):
    # y <= (A*x + B)/C <=> diff(x, y) <= B
    def diff(x, y):
        return C*y-A*x
    a = fraction(A, C)
    p, q = convergents(a)
    ph = [B // C]
    qh = [0]
    for i in range(2, len(q) - 1):
        if i % 2 == 0:
            while diff(qh[-1] + q[i+1], ph[-1] + p[i+1]) <= B:
                t = 1 + (diff(qh[-1] + q[i-1], ph[-1] + p[i-1]) - B - 1) // abs(diff(q[i], p[i]))
                dp = p[i-1] + t*p[i]
                dq = q[i-1] + t*q[i]
                k = (N - qh[-1]) // dq
                if k == 0:
                    return qh[-1], ph[-1]
                if diff(dq, dp) != 0:
                    k = min(k, (B - diff(qh[-1], ph[-1])) // diff(dq, dp))
                qh.append(qh[-1] + k*dq)
                ph.append(ph[-1] + k*dp)
    return qh[-1], ph[-1]

def solve(A, B, N):
    x, y = closest(A, N % A, B, N // A)
    return N // A - x, y
```

June Challenge 2017 - Euler Sum

Compute $\sum\limits_{x=1}^{N} \lfloor ex \rfloor$, where $e = [2; 1, 2, 1, 1, 4, 1, 1, 6, 1, \dots, 1, 2n, 1, \dots]$ is the Euler's number and $N \le 10^{4000}$.

Solution

This sum is equal to the number of lattice point $(x; y)$ such that $1 \leq x \leq N$ and $1 \leq y \leq ex$.

After constructing the convex hull of the points below $y = ex$, this number can be computed using Pick's theorem:

C++

```cpp
// sum floor(k * x) for k in [1, N] and x = [a0; a1, a2, ...]
int sum_floor(auto a, int N) {
    N++;
    auto [ah, ph, qh] = hull(a, N);

    // The number of lattice points within a vertical right trapezoid
    // on points (0; 0) - (0; y1) - (dx; y2) - (dx; 0) that has
    // a+1 integer points on the segment (0; y1) - (dx; y2).
    auto picks = [](int y1, int y2, int dx, int a) {
        int b = y1 + y2 + a + dx;
        int A = (y1 + y2) * dx;
        return (A - b + 2) / 2 + b - (y2 + 1);
    };

    int ans = 0;
    for(size_t i = 1; i < qh.size(); i++) {
        ans += picks(ph[i - 1], ph[i], qh[i] - qh[i - 1], ah[i - 1]);
    }
    return ans - N;
}
```

Python

```python
# sum floor(k * x) for k in [1, N] and x = [a0; a1, a2, ...]
def sum_floor(a, N):
    N += 1
    ah, ph, qh = hull(a, N)

    # The number of lattice points within a vertical right trapezoid
    # on points (0; 0) - (0; y1) - (dx; y2) - (dx; 0) that has
    # a+1 integer points on the segment (0; y1) - (dx; y2).
    def picks(y1, y2, dx, a):
        b = y1 + y2 + a + dx
        A = (y1 + y2) * dx
        return (A - b + 2) // 2 + b - (y2 + 1)

    ans = 0
    for i in range(1, len(qh)):
        ans += picks(ph[i-1], ph[i], qh[i]-qh[i-1], ah[i-1])
    return ans - N
```

NAIPC 2019 - It's a Mod, Mod, Mod, Mod World

Given p, q and n, compute $\sum\limits_{i=1}^{n} [p \cdot i \bmod q]$.

Solution

This problem reduces to the previous one if you note that $a \bmod b = a - \lfloor \frac{a}{b} \rfloor b$. With this fact, the sum reduces to

$$\sum_{i=1}^{n} \left(p \cdot i - \left\lfloor \frac{p \cdot i}{q} \right\rfloor q \right) = \frac{pn(n+1)}{2} - q \sum_{i=1}^{n} \left\lfloor \frac{p \cdot i}{q} \right\rfloor.$$

However, summing up $\lfloor rx \rfloor$ for x from 1 to N is something that we're capable of from the previous problem.

C++

```
void solve(int p, int q, int N) {
    cout << p * N * (N + 1) / 2 - q * sum_floor(fraction(p, q), N) << "\n";
}
```

Python

```
def solve(p, q, N):
    return p * N * (N + 1) // 2 - q * sum_floor(fraction(p, q), N)
```

Library Checker - Sum of Floor of Linear

Given N, M, A and B, compute $\sum\limits_{i=0}^{N-1} \lfloor \frac{A \cdot i + B}{M} \rfloor$.

Solution

This is the most technically troublesome problem so far.

It is possible to use the same approach and construct the full convex hull of points below the line $y = \frac{Ax+B}{M}$.

We already know how to solve it for $B = 0$. Moreover, we already know how to construct this convex hull up to the closest lattice point to this line on $[0, N-1]$ segment (this is done in the "Crime and Punishment" problem above.

Now we should note that once we reached the closest point to the line, we can just assume that the line in fact passes through the closest point, as there are no other lattice points on $[0, N-1]$ in between the actual line and the line moved slightly below to pass through the closest point.

That being said, to construct the full convex hull below the line $y = \frac{Ax+B}{M}$ on $[0, N-1]$, we can construct it up to the closest point to the line on $[0, N-1]$ and then continue as if the line passes through this point, reusing algorithm for constructing convex hull with $B = 0$:

Python

```python
# hull of lattice (x, y) such that C*y <= A*x+B
def hull(A, B, C, N):
    def diff(x, y):
        return C*y-A*x
    a = fraction(A, C)
    p, q = convergents(a)
    ah = []
    ph = [B // C]
    qh = [0]

    def insert(dq, dp):
        k = (N - qh[-1]) // dq
        if diff(dq, dp) > 0:
            k = min(k, (B - diff(qh[-1], ph[-1])) // diff(dq, dp))
        ah.append(k)
        qh.append(qh[-1] + k*dq)
        ph.append(ph[-1] + k*dp)

    for i in range(1, len(q) - 1):
        if i % 2 == 0:
            while diff(qh[-1] + q[i+1], ph[-1] + p[i+1]) <= B:
                t = (B - diff(qh[-1] + q[i+1], ph[-1] + p[i+1])) // abs(diff(q[i], p[i]))
                dp = p[i+1] - t*p[i]
                dq = q[i+1] - t*q[i]
                if dq < 0 or qh[-1] + dq > N:
                    break
                insert(dq, dp)

    insert(q[-1], p[-1])

    for i in reversed(range(len(q))):
        if i % 2 == 1:
            while qh[-1] + q[i-1] <= N:
                t = (N - qh[-1] - q[i-1]) // q[i]
                dp = p[i-1] + t*p[i]
                dq = q[i-1] + t*q[i]
                insert(dq, dp)
    return ah, ph, qh
```

OKC 2 - From Modular to Rational

There is a rational number $\frac{p}{q}$ such that $1 \leq p, q \leq 10^9$. You may ask the value of pq^{-1} modulo $m \sim 10^9$ for several prime numbers m. Recover $\frac{p}{q}$.

Equivalent formulation: Find x that delivers the minimum of $Ax \mod M$ for $1 \leq x \leq N$.

Solution

Due to Chinese remainder theorem, asking the result modulo several prime numbers is the same as asking it modulo their product. Due to this, without loss of generality we'll assume that we know the remainder modulo sufficiently large number m.

There could be several possible solutions (p, q) to $p \equiv qr \pmod m$ for a given remainder r. However, if (p_1, q_1) and (p_2, q_2) are both the solutions then it also holds that $p_1 q_2 \equiv p_2 q_1 \pmod m$. Assuming that $\frac{p_1}{q_1} \neq \frac{p_2}{q_2}$ it means that $|p_1 q_2 - p_2 q_1|$ is at least m.

In the statement we were told that $1 \leq p, q \leq 10^9$, so if both p_1, q_1 and p_2, q_2 are at most 10^9, then the difference is at most 10^{18}. For $m > 10^{18}$ it means that the solution $\frac{p}{q}$ with $1 \leq p, q \leq 10^9$ is unique, as a rational number.

So, the problem boils down, given r modulo m, to finding any q such that $1 \leq q \leq 10^9$ and $qr \mod m \leq 10^9$.

This is effectively the same as finding q that delivers the minimum possible $qr \bmod m$ for $1 \leq q \leq 10^9$.

For $qr = km + b$ it means that we need to find a pair (q, m) such that $1 \leq q \leq 10^9$ and $qr - km \geq 0$ is the minimum possible.

Since m is constant, we can divide by it and further restate it as find q such that $1 \leq q \leq 10^9$ and $\frac{r}{m} q - k \geq 0$ is the minimum possible.

In terms of continued fractions it means that $\frac{k}{q}$ is the best diophantine approximation to $\frac{r}{m}$ and it is sufficient to only check lower semiconvergents of $\frac{r}{m}$.

Python

```python
# find Q that minimizes Q*r mod m for 1 <= k <= n < m
def mod_min(r, n, m):
    a = fraction(r, m)
    p, q = convergents(a)
    for i in range(2, len(q)):
        if i % 2 == 1 and (i + 1 == len(q) or q[i+1] > n):
            t = (n - q[i-1]) // q[i]
            return q[i-1] + t*q[i]
```

Practice problems

- UVa OJ - Continued Fractions
- ProjectEuler+ #64: Odd period square roots
- Codeforces Round #184 (Div. 2) - Continued Fractions
- Codeforces Round #201 (Div. 1) - Doodle Jump
- Codeforces Round #325 (Div. 1) - Alice, Bob, Oranges and Apples
- POJ Founder Monthly Contest 2008.03.16 - A Modular Arithmetic Challenge
- 2019 Multi-University Training Contest 5 - fraction
- SnackDown 2019 Elimination Round - Election Bait
- Code Jam 2019 round 2 - Continued Fraction

Contributors:
adamant-pwn (99.91%) jatingaur18 (0.09%)