Busca e Ordenação

Algoritmos de busca: problemas resolvidos

Prof. Edson Alves

2019

Faculdade UnB Gama

Sumário

AtCoder Beginner Contest 111 – Problem B: AtCoder Beginner Contest 111

Problema

Kurohashi has never participated in AtCoder Beginner Contest (ABC).

The next ABC to be held is ABC N (the N-th ABC ever held). Kurohashi wants to make his debut in some ABC x such that all the digits of x in base ten are the same.

What is the earliest ABC where Kurohashi can make his debut?

Entrada e saída

Constraints

- $100 \leq N \leq 999$
- N is an integer.

Input

Input is given from Standard Input in the following format:

$$N$$

Output

If the earliest ABC where Kurohashi can make his debut is ABC n, print n.

Exemplo de entradas e saídas

Exemplo de Entrada	Exemplo de Saída
111	111
112	222
750	777

Solução

- Há apenas 9 possibilidades para a resposta:
  $111, 222, 333, 444, 555, 666, 777, 888$ e 999
- A resposta será o menor dentre entes valores que é maior ou igual a N
- Tal valor pode ser localizado através de uma busca linear simples
- Outra alternativa é utilizar a função `lower_bound()` da STL
- Em ambos casos, a complexidade da solução é constante

Solução $O(1)$

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

using namespace std;

int solve(int N)
{
    vector<int> contests;

    for (int d = 1; d <= 9; ++d)
        contests.push_back(100*d + 10*d + d);

    auto it = lower_bound(contests.begin(), contests.end(), N);

    return *it;
}

```

Solução $O(1)$

```
17 int main()
18 {
19     ios::sync_with_stdio(false);
20
21     int N;
22     cin >> N;
23
24     auto ans = solve(N);
25
26     cout << ans << '\n';
27
28     return 0;
29 }
```

UVA 10077 – The Stern-Brocot Number System

Problema

The Stern-Brocot tree is a beautiful way for constructing the set of all nonnegative fractions $\frac{m}{n}$ where m and n are relatively prime. The idea is to start with two fractions $(\frac{0}{1}, \frac{1}{0})$ and then repeat the following operations as many times as desired:

Insert $\frac{m+m'}{n+n'}$ between two adjacent fractions $\frac{m}{n}$ and $\frac{m'}{n'}$.

For example, the first step gives us one new entry between $\frac{0}{1}$ and $\frac{1}{0}$,

$$\frac{0}{1}, \frac{1}{1}, \frac{1}{0}$$

and the next gives two more:

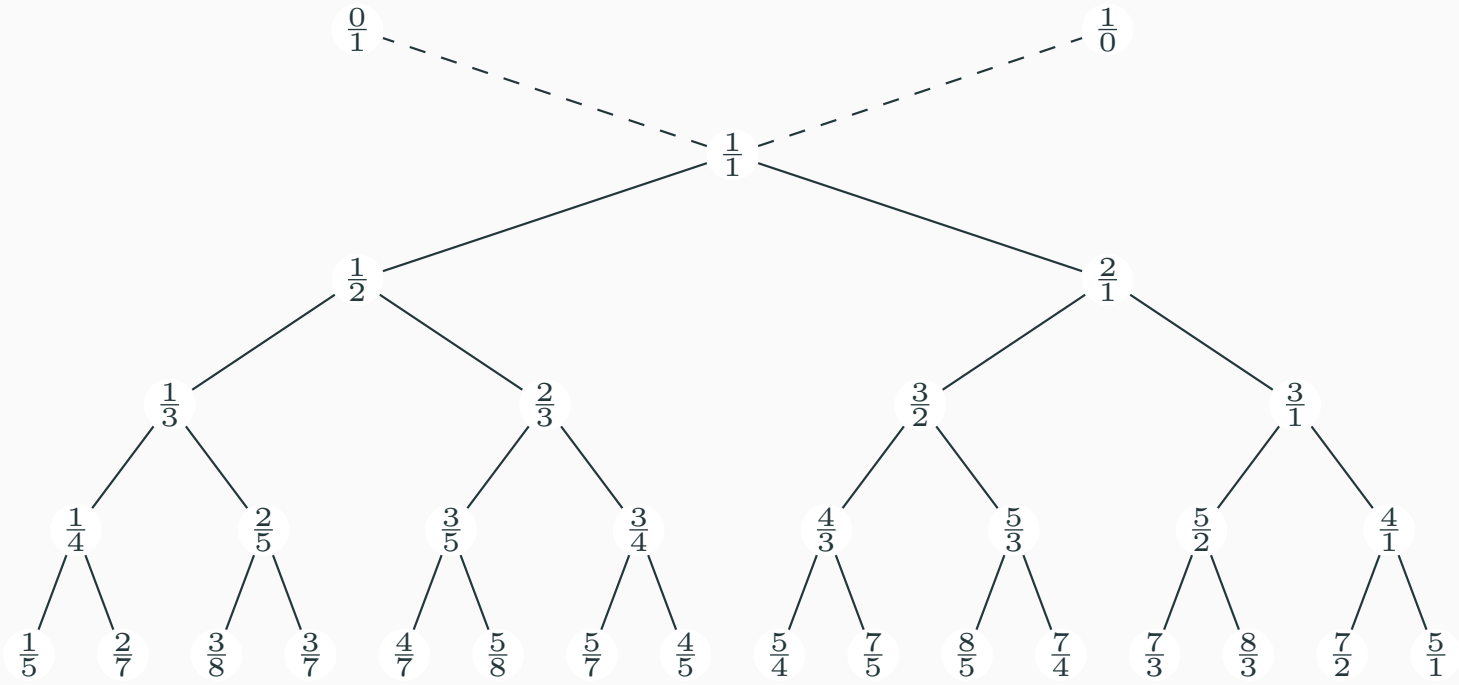
$$\frac{0}{1}, \frac{1}{2}, \frac{1}{1}, \frac{2}{1}, \frac{1}{0}$$

Problema

The next gives four more,

$$\frac{0}{1}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{1}{1}, \frac{3}{2}, \frac{2}{1}, \frac{3}{1}, \frac{1}{0}$$

and then we will get 8, 16, and so on. The entire array can be regarded as an infinite binary tree structure whose top levels look like this:



Problema

The construction preserves order, and we couldn't possibly get the same fraction in two different places.

We can, in fact, regard the Stern-Brocot tree as a number system for representing rational numbers, because each positive, reduced fraction occurs exactly once. Let's use the letters 'L' and 'R' to stand for going down to the left or right branch as we proceed from the root of the tree to a particular fraction; then a string of L's and R's uniquely identifies a place in the tree. For example, LRRL means that we go left from $\frac{1}{1}$ down to $\frac{1}{2}$, then right to $\frac{2}{3}$, then right to $\frac{3}{4}$, then left to $\frac{5}{7}$. We can consider LRRL to be a representation of $\frac{5}{7}$. Every positive fraction gets represented in this way as a unique string of L's and R's.

Problema

Well, actually there's a slight problem: The fraction $\frac{1}{1}$ corresponds to the empty string, and we need a notation for that. Let's agree to call it `I`, because that looks something like 1 and it stands for "identity".

In this problem, given a positive rational fraction, you are expected to represent it in *Stern-Brocot number system*.

Entrada e saída

Input

The input file contains multiple test cases. Each test case consists of a line contains two positive integers m and n where m and n are relatively prime. The input terminates with a test case containing two 1's for m and n, and this case must not be processed.

Output

For each test case in the input file output a line containing the representation of the given fraction in the *Stern-Brocot number system*.

Exemplo de entradas e saídas

Exemplo de Entrada	Exemplo de Saída
5 7 878 323 1 1	LRRL RRLRRLRLLLLRLRRR

Solução

- A árvore apresentada no exemplo e das sequências iniciais permitem a observação que a sequência é crescente
- Mais precisamente, a árvore com raiz $\frac{1}{1}$ é uma árvore binária de busca
- Isto permite o uso da busca binária para determinar, a cada elemento, se é preciso seguir à esquerda ou à direita
- A busca se encerra quando o número a ser representado é atingido
- Deste modo, a solução tem complexidade $O(N \log N)$, onde N é o número máximo de elementos da árvore até que se atinja a fração desejada

Solução $O(N \log N)$

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

using namespace std;
using ll = long long;
using ii = pair<ll, ll>;

const int MAX { 10000010 };

bool operator<(const ii& m, const ii& n)
{
    return m.first * n.second < m.second * n.first;
}

ll gcd(ll a, ll b)
{
    return b ? gcd(b, a % b) : a;
}

string stern_brocot(const ii& n)
{
    ii m(1, 1), L(0, 1), R(1, 0);
```

Solução $O(N \log N)$

```
22
23      ostringstream os;
24
25      while (m != n)
26      {
27          if (n < m)
28          {
29              os << "L";
30              R = m;
31          } else
32          {
33              os << "R";
34              L = m;
35          }
36
37          m = ii(L.first + R.first, L.second + R.second);
38
39          ll d = gcd(m.first, m.second);
40          m.first /= d;
41          m.second /= d;
42      }
```

Solução $O(N \log N)$

```
43
44      return os.str();
45  }
46
47  int main()
48  {
49      int p, q;
50
51      while (cin >> p >> q)
52      {
53          if (p == 1 and q == 1)
54              break;
55
56          auto ans = stern_brocot(ii(p, q));
57
58          cout << ans << '\n';
59      }
60
61      return 0;
62  }
```

Codeforces Round #251 (Div. 2) – Problem D: Devu and his Brother

Problema

Devu and his brother love each other a lot. As they are super geeks, they only like to play with arrays. They are given two arrays a and b by their father. The array a is given to Devu and b to his brother.

As Devu is really a naughty kid, he wants the minimum value of his array a should be at least as much as the maximum value of his brother's array b.

Now you have to help Devu in achieving this condition. You can perform multiple operations on the arrays. In a single operation, you are allowed to decrease or increase any element of any of the arrays by 1. Note that you are allowed to apply the operation on any index of the array multiple times.

You need to find minimum number of operations required to satisfy Devu's condition so that the brothers can play peacefully without fighting.

Entrada e saída

Input

The first line contains two space-separated integers n, m $(1 \leq n, m \leq 10^5)$. The second line will contain n space-separated integers representing content of the array a $(1 \leq a_i \leq 10^9)$. The third line will contain m space-separated integers representing content of the array b $(1 \leq b_i \leq 10^9)$.

Output

You need to output a single integer representing the minimum number of operations needed to satisfy Devu's condition.

Exemplo de entradas e saídas

Sample Input	Sample Output
2 2 2 3 3 5	3
3 2 1 2 3 3 4	4
3 2 4 5 6 1 2	0

Solução com complexidade $O(N \log N)$

- Seja $f(x) = \sum_i \alpha(a_i, x)$, onde $\alpha(a_i, x) = x - a_i$, se $a_i < x$, e $\alpha(a_i, x) = 0$, caso contrário
- De forma semelhante, seja $g(x) = \sum_j \beta(b_j, x)$, onde $\beta(b_j, x) = b_j - x$, se $b_j > x$, e $\beta(b_j, x) = 0$, caso contrário
- Observe que as funções $f(x)$ e $g(x)$ representam os custos para tornar x uma cota inferior ou superior dos vetores a e b, respectivamente
- Assim, o problema consiste em determinar o mínimo da função $h(x) = f(x) + g(x)$
- Observe que $h'(x) = \mu(a, x) - \rho(b, x)$, onde $\mu(c, x)$ é o número de elementos do vetor a menores do que x e $\rho(b, x)$ é o número de elementos do vetor b maiores do que x
- Como a função $\mu(a, x)$ é não-decrescente e a função $\rho(c, x)$ é não-crescente, a $h'(x)$ é não-decrescente
- Isto significa que $h''(x) \geq 0$, ou seja, $h(x)$ é convexa (unimodal), de modo que seu mínimo pode ser obtido através de uma busca ternária

Solução AC com complexidade $O(N \log N)$

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

using namespace std;
using ll = long long;

const ll oo { 1000000010LL };

ll f(ll x, const vector<ll>& as, const vector<ll>& bs)
{
    ll y = 0;

    for (const auto& a : as)
        y += (a < x ? x - a : 0);

    for (const auto& b : bs)
        y += (b > x ? b - x : 0);

    return y;
}

```

Solução AC com complexidade $O(N \log N)$

```
21 ll solve(const vector<ll>& as, const vector<ll>& bs)
22 {
23     // f(x) é convexa: busca ternária
24     ll a = 0, b = oo, ans = 200000000000000010L;
25
26     while (a <= b)
27     {
28         auto m1 = a + (b - a)/3;
29         auto m2 = b - (b - a)/3;
30
31         auto y1 = f(m1, as, bs), y2 = f(m2, as, bs);
32
33         ans = min(ans, y1);
34         ans = min(ans, y2);
35
36         y1 > y2 ? a = m1 + 1 : b = m2 - 1;
37     }
38
39     return ans;
40 }
41
```

Solução AC com complexidade $O(N \log N)$

```
42 int main()
43 {
44     ios::sync_with_stdio(false);
45
46     int N, M;
47     cin >> N >> M;
48
49     vector<ll> as(N), bs(M);
50
51     for (int i = 0; i < N; ++i)
52         cin >> as[i];
53
54     for (int j = 0; j < M; ++j)
55         cin >> bs[j];
56
57     auto ans = solve(as, bs);
58
59     cout << ans << '\n';
60
61     return 0;
62 }
```

Referências

1. AtCoder Beginner Contest 111 – Problem B: AtCoder Beginner Contest 111
2. UVA 10077 – The Stern-Brocot Number System
3. Codeforces Round #251 (Div. 2) – Problem D: Devu and his Brother