NGUYỄN XUÂN HUY

CLASS IN C++

Hà Nội, 2021

Contents

Farey Sequences

Nhà địa chất học người Anh John Farey (1766 – 1826) mô tả dãy Farey(n) là dãy các phân số tối giản $\frac{t}{m}$, $0 \leq \frac{t}{m} \leq 1$ được sắp tăng, có mẫu số không vượt quá giá trị n cho trước.

Ví dụ

Với n = 5 bạn có dãy Farey gồm 11 phân số như sau:

$$\frac{0}{1}\ \frac{1}{5}\ \frac{1}{4}\ \frac{1}{3}\ \frac{2}{5}\ \frac{1}{2}\ \frac{3}{5}\ \frac{2}{3}\ \frac{3}{4}\ \frac{4}{5}\ \frac{1}{1}$$

Bạn hãy viết dãy Farey tổng quát với giới hạn n là 100.

Algorithm

Bạn có ba cách làm sau đây (xét với ví dụ n = 5):

Cách thứ nhất

Pha 1. Bạn tạo ra toàn bộ các phân số tối giản không trùng lặp $\frac{t}{m}$ với tử số t và mẫu số m biến thiên trong khoảng từ 1 đến n.

$$\frac{0}{1}\ \frac{1}{2}\ \frac{1}{3}\ \frac{2}{3}\ \frac{1}{4}\ \frac{3}{4}\ \frac{1}{5}\ \frac{2}{5}\ \frac{3}{5}\ \frac{4}{5}\ \frac{1}{1}$$

Pha 2. Bạn sắp tăng các phân số:

$$\frac{0}{1}\ \frac{1}{5}\ \frac{1}{4}\ \frac{1}{3}\ \frac{2}{5}\ \frac{1}{2}\ \frac{3}{5}\ \frac{2}{3}\ \frac{3}{4}\ \frac{4}{5}\ \frac{1}{1}$$

Bạn biểu diễn mỗi phân số t/m dưới dạng cawpj hai phần tử: tu va mau.

```
class Frac {
  public:
    int tu;
    int mau;
};
```

Giả sử a là danh sách chứa kết quả khi thực hiện hàm Farey(n). Bạn khởi trịaf với hai phân số nhỏ nhất 0/1 và lớn nhất của dãy 1/1:

```
a[0] = (0,1); a[1] = (1,1);
```

Bạn tạo phân số tối giản qua hàm sau đây:

```
void SetReduce(int t, int m) {
     int d = Gcd(t, m);
    tu = t/d; mau = m/d;
   }
```

Hàm này nhận vào hai giá trị nguyên dương t và m và cho ra phân số tối giản (t/d, m/d) với d là ước chung lớn nhất của t và m.

Ví dụ

```
SetReduce(12,20) = 3/5  // d = Gcd(12,20) = 4
```

Để thực hiện Pha 2 ta gọi hàm sort với phương thức so sánh hai phân số a/b và c/d là tích chéo:

```
a/b < c/d khi và chỉ khi a*d < b*c
```

```
bool Less(Frac a, Frac b) {
   return a.tu * b.mau < a.mau * b.tu;
}
```

Program 1

```
// FAREY.CPP (version1)
#include <iostream>
#include <bits/stdc++.h>
#include <windows.h>

using namespace std;

int Gcd(int a, int b) {
   a = abs(a); b = abs(b);
   int r;
   while (b) {
      r = a % b;
      a = b; b = r;
   }
   return a;
}

class Frac{
  public:
   int tu;
   int mau;

   Frac() {tu = 1; mau = 0;}

    Frac(int t, int m) {
      tu = t; mau = m;
   }

   void Set(int t, int m) {
      tu = t; mau = m;
   }

   void SetReduce(int t, int m) {
      int d = Gcd(t, m);
    tu = t/d; mau = m/d;
   }

   Print(string msg = "") {
      cout << msg;
      cout << tu << " / " << mau;
```

```
   }

   Reduce() {
     int d = Gcd(tu, mau);
     tu /= d;
     mau /= d;
   }

    bool Eq(Frac y) {
       return tu*y.mau == mau*y.tu;
    }

    void operator =(Frac y) {
       tu = y.tu; mau = y.mau;
    }

};

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i) cout << " " << x[i];
}

void Print(Frac x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i)
        x[i].Print("\n");
}

bool Less(Frac a, Frac b) {
   return a.tu * b.mau < a.mau * b.tu;
}

int Farey1(int n) {
  Frac a[200];
  a[0].Set(0,1);
   a[1].Set(1,1);
   int k = 2;
   Print(a, 0, 1, "\n Init a: ");
   for (int m = 2; m <= n; ++m) {
      for (int t = 1; t < m; ++t) {
         a[k++].SetReduce(t,m);
      }
   }
   Print(a, 0, k-1, "\n a: ");
   sort(a, a+k, Less);
   cout << "\n Total " << k;
   Print(a, 0, k-1, "\n Sorted a: ");
   cout << "\n Ignore duplicated: ";
   int j = 0;
   for (int i = 1; i < k; ++i)
      if (!a[i].Eq(a[j])) a[++j] = a[i];
```

```
   k = j+1;
   Print(a, 0, k-1, "\n Result: ");
   cout << "\n Total " << k << " Fraction(s.)";
   return k;
}

void Test() {
   int n = 15;
  Farey1(n);
}

main() {
   Test();
   cout << "\n T h e   E n d";
   return 0;
}
```

Result 1

```
 Init a:
0 / 1
1 / 1
 a:
0 / 1
1 / 1
1 / 2
1 / 3
2 / 3
1 / 4
1 / 2
3 / 4
1 / 5
2 / 5
3 / 5
4 / 5
 Total 12
 Sorted a:
0 / 1
1 / 5
1 / 4
1 / 3
2 / 5
1 / 2
1 / 2
3 / 5
2 / 3
3 / 4
4 / 5
1 / 1
 Ignore duplicated:
 Result:
0 / 1
1 / 5
1 / 4
1 / 3
```

```
2 / 5
1 / 2
3 / 5
2 / 3
3 / 4
4 / 5
1 / 1
 Total 11 Fraction(s.)
```

Cách thứ hai

Đầu tiên bạn làm quen với khái niệm *phân số trung bình* (*middle fraction*).

$$\textit{Từ hai phân số tối giản } \frac{a}{b} \textit{ và } \frac{c}{d}$$

$$\textit{bạn tạo ra phân số trung bình } \frac{a+c}{b+d}$$

Đó chính là quy tắc "tử cộng tử, mẫu cộng mẫu".

Tiếp theo bạn thực hiện các bước sau đây:

Bạn xuất phát từ hai phân số

$$\frac{0}{1} \quad \frac{1}{1}$$

Bạn tạo ra phân số trung bình có mẫu số 2 và đặt vào giữa chúng:

$$\frac{0}{1} \quad \mathbf{\frac{1}{2}} \quad \frac{1}{1}$$

Bạn duyệt dãy trên để tạo ra các phân số trung bình mẫu số 3 và đặt vào giữa mỗi cặp phân số thành phần:

$$\frac{0}{1} \quad \mathbf{\frac{1}{3}} \quad \frac{1}{2} \quad \mathbf{\frac{2}{3}} \quad \frac{1}{1}$$

Bạn thực hiện lại bước trên để tạo ra các phân số trung bình mẫu số 4, mẫu số 5 và đặt vào giữa mỗi cặp...

Trong chương trình dưới đây toán tử gán a = b được thực hiện theo tham chiếu, nghĩa là a và b trỏ đến cùng một miền nhớ trong RAM.

Program 2

```
// FAREY.CPP
#include <iostream>
#include <bits/stdc++.h>
#include <windows.h>

using namespace std;

int Gcd(int a, int b) {
   a = abs(a); b = abs(b);
   int r;
   while (b) {
```

```
    r = a % b;
    a = b; b = r;
  }
  return a;
}

class Frac{
  public:
   int tu;
   int mau;

   Frac() {tu = 0; mau = 1;}

   Frac(int t) {tu = t; mau = 1;}

    Frac(int t, int m) {
      tu = t; mau = m;
   }

   void Set() {
     tu = 0; mau = 1;
   }

    void Set(int t) {
      tu = t; mau = 1;
   }

   void Set(int t, int m) {
     tu = t; mau = m;
   }

   void SetReduce(int t, int m) {
     int d = Gcd(t, m);
    tu = t/d; mau = m/d;
   }

   Print(string msg = "") {
      cout << msg;
      cout << tu << " / " << mau;
   }

   Frac Reduce() {
     int d = Gcd(tu, mau);
     tu /= d;
     mau /= d;
     return *this;
   }

    bool Eq(Frac y) {
       return tu*y.mau == mau*y.tu;
    }

    Frac operator %(Frac y) {
        return Frac(tu + y.tu, mau + y.mau).Reduce();
    }

    void operator =(Frac y) {
       tu = y.tu; mau = y.mau;
```

```
    }

};


void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i) cout << " " << x[i];
}

void Print(Frac x[], int d, int c, string msg = "") {
   cout << msg;
    for (int i = d; i <= c; ++i)
        x[i].Print("\n");
}

int Farey2(int n) {
  int MN = 200;
   Frac *a = new Frac[MN];
    Frac *b = new Frac[MN];
    Frac *c;
    a[0].Set(0); a[1].Set(1);
    int k = 2;
    // a: 0 1 2 ...
    // b: 0
    int j;
    for (int m = 2; m <= n; ++m) {
       b[0] = a[0];
       j = 1;
        for (int i = 1; i < k; ++i) {
           if (a[i-1].mau + a[i].mau == m) {
              b[j++] = a[i-1] % a[i];
           }
           b[j++] = a[i];
        }
        k = j;
        c = a; a = b; b = c;
    }
     Print(a, 0, k-1, "\n Result: ");
     cout << "\n Total " << k << " Fraction(s.)";

}

void Test() {
   int n = 5;
  Farey2(n);
}

main() {
   Test();
   cout << "\n T h e   E n d";
   return 0;
```

```
}
```

Result 2

```
 Result:
0 / 1
1 / 5
1 / 4
1 / 3
2 / 5
1 / 2
3 / 5
2 / 3
3 / 4
4 / 5
1 / 1
 Total 11 Fraction(s.)
 T h e   E n d
```

Cách thứ ba

Ta sử dụng một số tính chất của dãy Farey để tiếp tục cải tiến thuật toán.

```
 Nếu p1, p2 và p3 là ba phân số liên tiếp trong dãy Farey thì
(1) Tử(p2) * Mẫu(p1) - Tử(p1) * Mẫu(p2) = 1
(2) Mẫu(p1) + Mẫu(p2) > n
(3) p3 là phân số trung bình của p1 và p2
(4) Tử(p3) = v*Tử(p2)–Tử(p1), Mẫu(p3) = v*Mẫu(p)–Mẫu(p1),
    v = (Mẫu(p1)+n) div Mẫu(p2).
```

Từ tính chất 4 ta suy ra ngay cách xác định phân số thứ ba thông qua hai phân số sát trước.

Ví dụ

```
Với n = 5 ta khởi trị hai phân số đầu tiên của dãy Farey là:
p1 = 0/1, p2 = 1/5
Ta tính p3
v = (Mẫu(p1)+n) div Mẫu(p2) = (1+5) div 5 = 6 div 5 = 1
Tử(p3) = v * Tử(p2) - Tử(p1) = 1*1 - 0 = 1
Mẫu(p3) = v * Mẫu(p2) - Mẫu(p1) = 1*5 - 1 = 4
Vậy p3 = 1 / 4
Tiếp đến ta tính p4 theo p2 và p3,...
Trong chương trình dưới đây f[-1] là phần tử cuối của danh sách f
chứa các phân số được sinh ra, f[-2] là phân số sát phân số cuối
trong f.
```

Complete Frac Class

```
// Frac.H
#include <iostream>

#ifndef __Frac__H
#define __Frac__H

using namespace std;

int Gcd(int a, int b) {
    a = abs(a); b = abs(b);
    int r;
    while (b) {
          r = a % b;
          a = b; b = r;
    }
    return a;
}

class Frac{
  public:
   int tu;
   int mau;

   Frac() {tu = 0; mau = 1;}

   Frac(int t) {
                tu = t;
                mau = 1;
          }

     Frac(int t, int m) {
        if (m <= 0) Err(1);
          tu = t; mau = m;
   }

   void Err(int e) {
          cerr << "\n ERROR " << e << ": ";
          switch(e) {
                case 1: cerr << " mau <= 0."; break;
                case 2: cerr << " chia cho 0."; break;
          }
          exit(e);
   }

   inline void Set() {
          tu = 0; mau = 1;
   }

     inline void Set(int t) {
          tu = t; mau = 1;
   }

   inline void Set(int t, int m) {
      if (m <= 0) Err(1);
          tu = t; mau = m;
```

```
}

void SetReduce(int t, int m) {
  if (m <= 0) Err(1);
  int d = Gcd(t, m);
  tu = t/d; mau = m/d;
}

inline Print(string msg = "") {
   cout << msg;
   cout << tu << " / " << mau;
}

Frac Reduce() {
 int d = Gcd(tu, mau);
 tu /= d;
 mau /= d;
 return *this;
}

Frac GetReduce() {
 int d = Gcd(tu, mau);
 return Frac(tu / d, mau / d);
}

  Frac operator +(Frac y) {
              return Frac(tu*y.mau + mau*y.tu, mau*y.mau).Reduce();
  }

  Frac operator +(int b) {
              return *this + Frac(b);
  }

  void operator ++() {
              *this = *this + 1;
  }

  void operator +=(Frac y) {
              *this = *this + y;
  }

  Frac operator ~() { // doi dau
    return Frac(-tu, mau);
  }

  Frac operator -(Frac y) {
              return *this + (~y);
  }

  void operator --() {
              *this = *this - 1;
  }

  void operator -=(Frac y) {
              *this = *this - y;
  }

  Frac operator *(Frac y) {
```

```
                return Frac(tu * y.tu, mau * y.mau).Reduce();
    }

    void operator *=(Frac y) {
                *this = *this * y;
    }

    Frac operator !() { // nghich dao
       if (tu == 0) Err(1);
       if (tu > 0)     return Frac(mau, tu);
       else return Frac(-mau, -tu);
    }

    Frac operator /(Frac y) {
                return *this * (!y);
    }

    void operator /=(Frac y) {
                *this = *this / y;
    }

    Frac operator %(Frac y) { // PS trung binh
                return Frac(tu + y.tu, mau + y.mau).Reduce();
    }

    void operator %=(Frac y) {
                *this = *this % y;
    }

    void operator =(Frac y) {
       tu = y.tu; mau = y.mau;
    }

    bool Eq(Frac y) {
       return tu*y.mau == mau*y.tu;
    }

    bool operator ==(Frac y) {
       return tu*y.mau == mau*y.tu;
    }

    bool operator > (Frac y) {
       return tu*y.mau > mau*y.tu;
    }

    bool operator < (Frac y) {
       return tu*y.mau < mau*y.tu;
    }

    bool operator >= (Frac y) {
       return !(*this < y);
    }

    bool operator <= (Frac y) {
       return !(*this > y);
    }
};
```

```
#endif
```

Test Frac Program

```
// FracTest.CPP
#include <iostream>
#include <bits/stdc++.h>
#include <windows.h>
#include "Frac.h"


void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, string msg = "") {
   cout << msg;
     for (int i = d; i <= c; ++i) cout << " " << x[i];
}

void Print(Frac x[], int d, int c, string msg = "") {
   cout << msg;
     for (int i = d; i <= c; ++i)
                 x[i].Print("\n");
}


void Test() {
   srand(time(NULL));
     Frac x, y, z, t, xx;
     while(true) {
                 x.Set(rand() % 100 + 1, rand() % 100 + 1);
                 y.Set(rand() % 100 + 1, rand() % 100 + 1);
                 z.Set(rand() % 100 + 1, rand() % 100 + 1);
                 x.Print("\n x = ");
                 y.Print("\n y = ");
                 z.Print("\n z = ");
                 t = (x + y) * z; // t = x*z + y*z
                 xx = (t-y*z)/z;
                 xx.Print("\n xx = ");
                 if (xx == x) cout << "\n CORRECT.";
                 else cout << "\n ERROR.";
                 Go();
     }
}

main() {
   Test();
   cout << "\n T h e   E n d";
   return 0;
}
```

Result

```
x = 51 / 32
y = 92 / 27
z = 94 / 50
xx = 51 / 32
CORRECT. ?

x = 62 / 40
y = 29 / 92
z = 3 / 9
xx = 31 / 20
CORRECT. ?

x = 93 / 71
y = 60 / 88
z = 65 / 60
xx = 93 / 71
CORRECT. ?

x = 70 / 53
y = 34 / 47
z = 48 / 84
xx = 70 / 53
CORRECT. ?

x = 76 / 50
y = 67 / 52
z = 88 / 95
xx = 38 / 25
CORRECT. .
```

Using Arraylist

```
// FAREY.CPP
#include <iostream>
#include <bits/stdc++.h>
#include <windows.h>
#include <list>
#include "Frac.h"


using namespace std;

typedef list<Frac> List;

void Go() {
   cout << " ? ";
   fflush(stdin);
   if (cin.get() == '.') exit(0);
}

void Print(List d, string msg = "") {
```

```
  cout << msg;
   List :: iterator it;
   for (it = d.begin(); it != d.end(); it++)
        it->Print("\n ");
     cout << "\n Total " << d.size();
}

bool Less(Frac a, Frac b) {
   return a.tu * b.mau < a.mau * b.tu;
}

// Compares two intervals
// according to staring times.
bool CompareFrac(Frac &a, Frac &b) {
    return a < b;
}

 // Function for binary_predicate
bool EqFrac(Frac & a, Frac & b) {
    return a == b;
}

int Farey(int n) {
  List a;
  a.push_back(Frac(0));
   a.push_back(Frac(1));
   Frac x;
   for (int m = 2; m <= n; ++m) {
      for (int t = 1; t < m; ++t) {
         x.SetReduce(t,m);
         a.push_back(x);
      }
   }
   Print(a,"\n a: ");
   Go();
   a.sort(CompareFrac);
   Print(a,"\n Sorted a: ");
   a.unique(EqFrac);
   Print(a,"\n Result: ");
   return a.size();
}

void Test() {
   int n = 5;
  Farey(n);
}

main() {
   Test();
   cout << "\n T h e   E n d";
   return 0;
}
```

Result

```
a:
0 / 1
1 / 1
1 / 2
1 / 3
2 / 3
1 / 4
1 / 2
3 / 4
1 / 5
2 / 5
3 / 5
4 / 5
Total 12 ?

Sorted a:
0 / 1
1 / 5
1 / 4
1 / 3
2 / 5
1 / 2
1 / 2
3 / 5
2 / 3
3 / 4
4 / 5
1 / 1
Total 12
Result:
0 / 1
1 / 5
1 / 4
1 / 3
2 / 5
1 / 2
3 / 5
2 / 3
3 / 4
4 / 5
1 / 1
Total 11
T h e   E n d
```

Collatz Problem

Nhà toán học Đức Lothar Collatz xây dựng dãy số mang tên ông là *dãy Collatz* như sau:

Xuất phát từ số nguyên dương n.

Lặp đến khi n = 1 thao tác sau:

Nếu n là số lẻ thì thay n bằng 3n + 1.

Nếu n chẵn thì thay n bằng n div 2.

Ví dụ

Với n = 3 bạn sẽ nhận được dãy Collatz gồm 8 số như sau:

3, 10, 5, 16, 8, 4, 2, 1.

Lothar Collatz

1910-1990

Ký hiệu

`Len(n) = chiều dài của dãy Collatz(n).`

Example

`Len(1) = 1,`

`Len(3) = 8 (3 → 10 → 5 → 16 → 8 → 4 → 2 → 1)`

Program

```
#include <iostream>
#include <bits/stdc++.h>
#include <windows.h>
#include <list>
#include "Frac.h"


using namespace std;

typedef list<int> List;

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.') exit(0);
}

void Print(List d, string msg = "") {
   cout << msg;
   List :: iterator it;
   for (it = d.begin(); it != d.end(); it++)
           cout << " " << *it;
    cout << "\n Total " << d.size();
}
```

```
int Collatz(int n) {
    cout << "\n Collatz of " << n;
    List c;
    c.push_back(n);
    while (n != 1) {
           if (n % 2 == 0) n /= 2;
           else n = n*3 + 1;
           c.push_back(n);
    }
    Print(c, "\n Result: ");
}


main() {
   Collatz(3);
   Collatz(13);
   cout << "\n T h e   E n d";
   return 0;
}
```

Result

```
Collatz of 3
Result:  3 10 5 16 8 4 2 1
Total 8
Collatz of 13
Result:  13 40 20 10 5 16 8 4 2 1
Total 10
T h e   E n d
```

Improve