Farey Sequences

*Nhà địa chất học người Anh John Farey (1766 − 1826) mô tả dãy Farey(n) là dãy các phân số tối giản 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:*

*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 với tử số *t* và mẫu số *m* biến thiên trong khoảng từ 1 đến *n*.
* Pha 2. Bạn sắp tăng các phân số:

Bạn biểu diễn mỗi phân số t/m dưới dạng danh sách hai phần tử:

[t,m]

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

f = [[0, 1], [1, 1]]

Bạn tạo phân số tối giản qua hàm sau đây:

def Reduced(t, m):

d = Gcd(t, m)

return [t // d, 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

Reduced(12,20) = [3,5] # d = Gcd(12,20) = 4

Mỗi khi tạo thêm một phân số mới x = [t,m] bạn kiểm tra, nếu x chưa có trong f thì bạn nạp vào f.

def Farey(n):

# Phase 1

f = [[0, 1], [1, 1]]

for m in range(1, n + 1):

for t in range(1, m):

x = Reduced(t, m)

if x not in f:

f.append(x)

# Phase 2

return sorted(f, key=lambda x: x[0] / x[1])

Để thực hiện Pha 2 ta gọi hàm sorted. Chú ý rằng có hai hàm sắp xếp dãy d là sort và sorted, cụ thể là

d.sort()

d = sorted(d)

Ta đã biết hàm sort() là hàm tự thân, nghĩa là sau khi gọi d.sort() thì dữ liệu trong d sẽ được sắp lại.

Hàm sorted nhận vào các tham biến: bản thân danh sách d và chỉ định về phương thức so sánh. Trong trường hợp này ta mô tả phương thức xác định khóa so sánh bằng hàm tại chỗ lambda với ý nghĩa như sau:

Mỗi phân số x sẽ được chuyển thành một số thực x[0] / x[1] và phép so sánh trong hàm sorted sẽ thực hiện trên các giá trị thực này.

Ví dụ

[3, 5] : 0.6 # 3 / 5 = 0.6

[1, 2] : 0.5 # 1 / 2 = 0.5

Khi đó hàm sorted sẽ thay phép so sánh hai phân số [3,5] và [1,2] bằng phép so sánh hai số thực 0.6 và 0.5.

Program

# Farey Sequences

# Ver. 1

def Gcd(a, b):

while b:

r = a % b

a, b = b, r

return a

def Reduced(t, m):

d = Gcd(t, m)

return [t // d, m // d]

def Farey(n):

# Phase 1

f = [[0, 1], [1, 1]]

for m in range(1, n + 1):

for t in range(1, m):

x = Reduced(t, m)

if x not in f:

f.append(x)

# Phase 2

return sorted(f, key=lambda x: x[0] / x[1])

# Application

n = 15

p = Farey(n)

print('Farey of ', n, ': Total', len(p), ' fraction(s)')

for ps in p:

print(ps[0], '/', ps[1]

Result

Farey of 5 : Total 11 fraction(s)

0 / 1

1 / 5

1 / 4

1 / 3

2 / 5

1 / 2

3 / 5

2 / 3

3 / 4

4 / 5

1 / 1

Cách thứ hai

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

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
* Bạn tạo ra phân số trung bình có mẫu số 2 và đặt vào giữa chúng:
* 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:
* 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 tại dòng # (1) được thực hiện theo tham chiếu, nghĩa là a và b trỏ đến cùng một miền nhớ trong RAM.

Program

# Farey Sequences

# Ver. 2. Phan so trung binh (Middle Fraction)

def Farey(n):

a = [[0, 1], [1, 1]]

for m in range(2, n + 1):

b = [a[0]]

for i in range(1,len(a)):

if a[i][1] + a[i - 1][1] == m: # mau + mau

b.append([a[i][0] + a[i - 1][0], m])

b.append(a[i])

a = b # (1)

return a

# Application

n = 5

p = Farey(n)

print('Farey of ', n, ': Total', len(p), ' fraction(s)')

for ps in p:

print(ps[0], '/', ps[1])

Result

Farey of 5 : Total 11 fraction(s)

0 / 1

1 / 5

1 / 4

1 / 3

2 / 5

1 / 2

3 / 5

2 / 3

3 / 4

4 / 5

1 / 1

Nếu bạn lúng túng trong việc coding chương trình trên thì bạn có thể viết hàm MedFrac(x,y) cho ra phân số trung bình của hai phân số x và y theo quy tắc tử cộng tử, mẫu cộng mẫu như sau:

TU, MAU = 0, 1

def MedFrac(x, y):

return [x[TU] + y[TU], x[MAU] + y[MAU]]

Program

# Farey Sequences

# Ver. 2B. Phan so trung binh (Median Fraction)

TU, MAU = 0, 1

def MedFrac(x, y):

return [x[TU] + y[TU], x[MAU] + y[MAU]]

def Farey(n):

a = [[0, 1], [1, 1]]

for m in range(2, n + 1):

b = [a[0]]

for i in range(1, len(a)):

f = MedFrac(a[i], a[i - 1])

if f[MAU] == m: # mau + mau

b.append(f)

b.append(a[i])

a = b

return a

# Application

n = 5

p = Farey(n)

print('Farey of ', n, ': Total', len(p), ' fraction(s)')

for ps in p:

print(ps[0], '/', ps[1])

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.

Program

# Farey Sequences

# Ver. 3

TU, MAU, N = 0, 1, 0

# v = (mau1 + n) div mau2

# tu3 = v\*tu2 – tu1, mau3 = v\*mau2 – mau1

def NextFrac(p1, p2):

v = (p1[MAU] + N) // p2[MAU]

return [v \* p2[TU] - p1[TU], v \* p2[MAU] - p1[MAU]]

def Farey(n):

global N

if n < 1:

return []

N = n

f = [[0, 1], [1, N]]

while f[-1] != [1, 1]:

f.append(NextFrac(f[-2], f[-1]))

return f

def Go():

if input(' ? ') == '.':

exit(0)

# Application

for n in range(10):

p = Farey(n)

print('Farey of ', n, ': Total', len(p), ' fraction(s)')

for ps in p:

print(ps[0], '/', ps[1])

Go()