**Algorithm Design Methods**

|  |  |
| --- | --- |
| A person in a suit standing outside  Description automatically generated | Lecturer  PGS TSKH Nguyễn Xuân Huy  Viện Công nghệ Thông tin  Viện Hàn lâm Khoa học  và Công nghệ Việt nam  [nxhuy564@gmail.com](mailto:nxhuy564@gmail.com)  0903203800  Connection: Zalo |

Nguyễn Xuân Huy – ID: **0903203800** - Pass: **huynx**

1. Mon July 8 2024: Introduction

2. Thu July 11 2024: Arithmetics, Next

3. Mon July 15 2024: Backtracking

4. Thu July 18 2024: Dynamic Programming

5. Mon July 22 2024: Divide and Conquer

6. Thu July 25 2024: Primes

7. Mon July 29 2024: Stack, Queue, Dequeue

8. Thu July Aug 1 2024: Graph

9. Mon Aug 5 2024: Robot

10. Thu Aug 8 2024:

**Leson 6: Primes**

6. Thu July 25 2024

***Little problems***

## P1. Numeral Systems

Viết các hàm

Str(x, b) return dạng string của số x trong hệ đếm b = 2..35.

Str(235, 10) = “235”

Str(235, 2) = “11101011”

Str(235, 3) = “22201”

Str(235, 16) = “eb

Int(s, b) return số nguyên từ string s trong hệ đếm b = 2..35.

Int('235', 10) = 235

Int('11101011', 2) = 235

Int('2201', 3) = 235

Int('eb', 16) = 235

Thuật toán

Lấy phải – ghép trái: Lần lươt lấy từng chữ số của *x* ghép sang bên trái của string kết quả. Hai phép toán mod và div phải thực hiện trong hệ đếm *b*. Hệ đếm *b* dùng *b* chữ số từ 0 đến *b*-1. Nếu *b* > 10 ta dùng thêm các chữ số *a* = 10, *b* = 11, *c* = 12, ...

Độ phức tạp

Số chữ số của đầu vào.

Chương trình

# Numeral Systems

import random

NUM = "0123456789" # base = 2..10

ABC = "abcdefghijklmnopqrstuvwxyz" # base = 11..36

DIGIT = NUM + ABC # base = 2..36

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def Str(x, base = 10): # ngầm định là hệ đếm 10

if x == 0: return "0"

sign = '' if x > 0 else '-' # ghi nhận dấu

x = abs(x)

y = ''

while x > 0:

x, r = divmod(x, base)

y = DIGIT[r] + y

return sign + y

# chuyển đổi số str sang int

def Int(s, base = 10): # ngầm định là hệ đếm 10

s.strip()

if s == '0': return 0

sign = 1

if s[0] == '-':

sign = -1

s = s[1:]

print(s)

x = 0

for c in s:

x = x\*base + DIGIT.find(c)

return sign\*x

def Run():

while True:

x = random.randrange(0, 100000)

b = random.randrange(2, 17)

s = Str(x, b)

y = Int(s, b)

print(x," -> ", Str(x, b) , '->', y, (x==y), ': Base', b)

Go()

# APPLICATION

Run()

print(' T h e E n d')

## P2. Palindrome

Số nguyên x được gọi là palindrome (đối xứng) nếu đọc xuôi và ngược đều cho cùng một giá trị. Viết hàm Pal(x) kiểm tra số x có phải là palindrome? x có tối đa 18 chữ số.

# Palindrom

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def Rev(x): # Lật số x

sign = 1 if x >= 0 else -1

x = abs(x)

y = 0

while x > 0:

y = y\*10 + (x % 10)

x //= 10

return sign\*y

def Pal(x):

return x == Rev(x)

def Run():

for x in range(10, 25, 3):

print(x, " Pal ? ", Pal(x))

d = [1221, 12021, 120021, 1231, 12345678087654321, 0]

for x in d:

print(x, " Pal ? ", Pal(x))

y = -x

print(y, " Pal ? ", Pal(y))

Run()

print('T h e E n d')

Output

10 Pal ? False

13 Pal ? False

16 Pal ? False

19 Pal ? False

22 Pal ? True

1221 Pal ? True

-1221 Pal ? True

12021 Pal ? True

-12021 Pal ? True

120021 Pal ? True

-120021 Pal ? True

1231 Pal ? False

-1231 Pal ? False

12345678087654321 Pal ? True

-12345678087654321 Pal ? True

0 Pal ? True

0 Pal ? True

T h e E n d

## P3. Bi-palindrome

Số tự nhiên x được gọi là song đối xứng a, b nếu x là palindrome đồng thời trong cả hai hệ đếm a và b. Cho hai hệ đếm a và b. Liệt kê các số dưới 1M (một triệu) song đối xứng trong a và b.

Thuật toán

*BiPal*(*x,a,b*) *khi* và chỉ khi *Pal(x,a*) và *Pal(x,b*) = *x.*

*Pal*(*x,a*) *khi* và chỉ khi *x* = *Rev*(*x, a*).

Tóm lại ta cần viết các hàm sau:

* Str(x, a): chuyển số x trong hệ đếm a sang string để hiển thị
* Rev(x, a): Lật số x trong hệ đếm a
* Pal(x, a): kiểm tra số x là palindrome trong hệ đếm a.

Bạn cần lưu ý:

* Hàm Str chỉ dùng khi hiển thị số x.

Độ phức tạp

Số chữ số của input.

Chương trình

# BiPalindrome

NUM = "0123456789"

ABC = "abcdefghijklmnopqrstuvwxyz" # base = 11..36

DIGIT = NUM + ABC # base = 2..36

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

# Chỉ dùng khi hiển thị

def Str(x, base):

if x == 0: return "0"

sign = "" if x > 0 else "-"

x = abs(x)

y = ""

while x > 0:

y = DIGIT[x % base] + y

x //= base

return sign + y

# Số lật trong hệ base

def Rev(x, base):

if x == 0: return 0

sign = 1 if x > 0 else -1

x, y = abs(x), 0

while x > 0:

y = y\*base + (x % base)

x //= base

return sign\*y

def Bipal(x, a, b):

if x == Rev(x,a): return x == Rev(x,b)

else: return False

def Run():

a, b = 10, 16 # hệ 10 và 16

limit = 1000000

print('Bi-Palindromic numbers between 1..',limit)

d = 0

for x in range(0, limit):

if Bipal(x, a, b):

d += 1

print(str(d) + '. ', Str(x,a), Str(x,b))

print('Total ', d)

# APPLICATION

Run()

print(' T h e E n d')

Output

Bi-Palindromic numbers between 1.. 1000000

1. 0 0

2. 1 1

3. 2 2

4. 3 3

5. 4 4

6. 5 5

7. 6 6

8. 7 7

9. 8 8

10. 9 9

11. 11 b

12. 353 161

...

16. 1991 7c7

17. 3003 bbb

18. 39593 9aa9

...

28. 845548 ce6ec

Total 28

T h e E n d

## P4. Height of Numbers

Độ cao của một số tự nhiên là tổng các chữ số của số đó, ký hiệu là H(x). Trong các số dưới 1M có bao nhiêu số có độ cao h cho trước?

Phương án 1

Duyệt các số từ 1..1000000 đếm các số có độ cao h.

Chương trình

# Độ cao của số tự nhiên

from time import time

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def H(x):

h = 0

while x > 0:

h += x % 10

x //= 10

return h

def RH(x): # Recursive

return 0 if x == 0 else RH(x//10)+x%10

# Số lượng các số < n, H(x) == h

def Run(n, h):

d = 0

for x in range(0, n):

if H(x) == h: d += 1

# if RH(x) == h: d += 1 # comppare with H(x) ?

print('\n Total: ', d) # 21582

# APPLICATION

t = time()

Run(1000000, 17) # 21582

print(time()-t)

print(' T h e E n d')

Độ phức tạp

Khoảng 10 triệu lần lặp, thời gian chừng 1.2 giây.

Chú thích

Có thể tính độ cao của một số qua hàm sau:

H(x) = sum([int(c) for c in str(x)]

Trong đó

str(x) là dạng string của số nguyên x: str(102) = ‘102‘

int(c) chuyển đổi ký tự c sang số nguyên: int(‘3‘) = 3

[int(c) for c in str(x)] là danh sách các chữ số của x:

[int(c) for c in str(102)] = [1, 0, 2]

sum(d) = tổng các số trong danh sách d

Phương án 2 (cải tiến)

Pha 1. Tính trước các độ cao H(x), x = 1..limit ghi vào một danh sách d: d[x] = H(x) theo hệ thức sau:

H(x) = H(x // 10) + (x % 10)

Khởi trị: d[x] = x, x = 1..9

d[x] = d[x // 10] + (x % 10), x, i = 10..limit

Ví dụ

H(123) = H(12)+3

Pha 2. Số lượng các số x: 1..limit có độ cao h khi đó sẽ là

d.count(h)

Chương trình

# Độ cao của số tự nhiên

# Tính trước mọi độ cao

from time import time

def AllH(limit):

d = bytearray([0]\*limit)

for x in range(10): d[x] = x

for x in range(10, limit): d[x] = d[x//10]+(x % 10)

return d

# APPLICATION

t = time()

n = 1000000

d = AllH(n)

h = [17, 30]

print('Số lượng các số 1..', n, ' có độ cao', h[0], end = ': ')

print(d.count(h[0])) # 21582

print('Số lượng các số 1..', n, ' có độ cao', h[1], end = ': ')

print(d.count(h[1])) # 50877

print(time()-t)

print(' T h e E n d')

Output

Số lượng các số 1.. 1000000 có độ cao 17: 21582

Số lượng các số 1.. 1000000 có độ cao 30: 50877

0.28124189376831055

T h e E n d

## P5. False Multiplication

Một người không hiểu thuật toán nhân nên đã thực hiện phép nhân z = xy như sau. Người đó lấy từng chữ số của số nhân y, nhân với số bị nhân x rồi ghi thành từng dòng thẳng cột với nhau.Tổng các dòng khi đó sẽ là z. Biết y và z, liệu bạn có thể khôi phục phép nhân đúng?

Ví dụ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x |  | 1 | 3 | 5 | 7 |
| y |  |  | × | 1 | 2 |
|  |  | 2 | 7 | 1 | 4 |
|  |  | 1 | 3 | 5 | 7 |
| z |  | 4 | 0 | 7 | 1 |

Thuật toán

Do mỗi chữ số của *y* đều được nhân với *x* và dóng thẳng cột nên kết quả của phép nhân sai sẽ là *z* = *x*H(*y*). Từ đó suy ra *x* = *z* // H(*y*).

Chương trình

# Phép nhân sai

def H(x):

return sum([int(c) for c in str(x)])

def NhanSai(y, z): return (z // H(y), y)

x,y = NhanSai(12, 4071)

print(x,'\*',y, '=', x\*y)

print(' T h e E n d')

Output

1357 \* 12 = 16284

T h e E n d

## P6. Scrolling number

Số cuộn của số tự nhiên x là số S(x) = H(x) mod 9. Cho số tự nhiên x có tối đa 200 chữ số. Hãy cộng thêm cho x một giá trị nhỏ nhất k để x+k là bội số của 9. Giá trị x+k là bao nhiêu?

Input file SCROLL.INP gồm nhiều dòng, mỗi dòng một số, kết thúc là -1.

Kết quả hiển thị trên màn hình.

Ví dụ

|  |  |  |
| --- | --- | --- |
| SCROLL.INP | OUTPUT | Chú thích |
| 385  113519  13580245318  1999  135802458  -1 | 387  113526  13580245323  2007  135802458 | cộng thêm 2  cộng thêm 7  cộng thêm 5  cộng thêm 8  cộng thêm 0 |

Thuật toán

Số *x* là bội của 9 khi và chỉ khi số cuộn của *x* bằng 0. Nếu 1 ≤ S(x) ≤ 8 thì ta cần cộng thêm cho *x* một lượng d = 9-S(x). Lưu ý rằng Python có thể thực hiện các phép toán số học trên các số lớn.

Chương trình

# Scrolling number

# Tổng các chữ số của x mod 9

def S(x):

return sum([int(c) for c in str(x)]) % 9

# Đọc dữ liệu từ input file SCROLL.INP

def ReadInput():

with open('SCROLL.INP') as f:

d = list(map(int,f.read().split()))

d.pop() # bỏ phần tử -1 cuối dãy

return d

def Run():

d = ReadInput()

print(d)

for x in d:

v = S(x) # scrolling number

t = 0 if v == 0 else 9-v # cần thêm t ?

print(x, 'Scroll =',v, ' Cộng thêm ', t, end = ' -> ')

print(x+t)

# APPLICATION

Run()

print(' T h e E n d')

Output

[385, 113519, 13580245318, 1999, 135802458]

385 Scroll = 7 Cong them 2 -> 387

113519 Scroll = 2 Cong them 7 -> 113526

13580245318 Scroll = 4 Cong them 5 -> 13580245323

1999 Scroll = 1 Cong them 8 -> 2007

135802458 Scroll = 0 Cong them 0 -> 135802458

T h e E n d

Độ phức tạp

Số chữ số của input.

## P7. Multiples of 11

Cho số tự nhiên x có tối đa 200 chữ số. Nếu x là bội của 11 thì hiển thị YES; ngược lại, hiển thị NO.

Input file MULT11.INP gồm nhiều dòng, mỗi dòng một số, kết thúc là -1.

Kết quả hiển thị trên màn hình.

Ví dụ

|  |  |
| --- | --- |
| MULT11.INP | OUTPUT |
| 2714943  138369  5353518  1320407  -1 | YES  YES  NO  YES |

Thuật toán

Ta gọi *độ lệch* của số tự nhiên *x* là giá trị D(*x*) = |C-L|, trong đó C là tổng các chữ số ở vị trí chẵn, L là tổng các chữ số ở vị trí lẻ của *x*; |C-L| là giá trị tuyệt đối của hiệu C-L.

Ví dụ

D(138369) = |(1+8+6)-(3+3+9)| = |15-15| = 0

D(5353518) = |(5+5+5+8)-(3+3+1)| = |23-7| = 16

Định lý

*(Dấu hiệu chia hết cho 11)*

*Số tự nhiên x chia hết cho 11 khi và chỉ khi độ lệch D(x) chia hết cho 11.*

Vậy ta chỉ cần tính hàm D11(x) = D(x) mod 11. Khi đó *x* chia hết cho 11 khi và chỉ khi D11(x) = 0.

Ta thêm vào cuối *x* một chữ số 0 để tiện tính toán.

Độ phức tạp

Số chữ số của input.

Chương trình

# Bội số của 11

# Dấu hiệu chia hết cho 11

def D11(x):

a = b = 0

while x > 0:

x, ab = divmod(x, 100)

a += (ab // 10)

b += (ab % 10)

return abs(a-b) % 11

def ReadInput():

with open('MULT11.INP') as f:

d = list(map(int,f.read().split()))

d.pop() # bỏ phần tử -1 cuối dãy

return d

def Run():

p = ReadInput()

print(p)

for x in p:

d = D11(x)

msg = 'YES.' if d == 0 else 'NO.'

print(x, 'D11 =',d, msg)

# APPLICATION

Run()

print(' T h e E n d')

Output

[2714943, 138369, 5353518, 1320407]

2714943 D11 = 0 YES.

138369 D11 = 0 YES.

5353518 D11 = 5 NO.

1320407 D11 = 0 YES.

T h e E n d

Chú thích

Bạn có thể tính độ lệch như sau:

# Dấu hiệu chia hết cho 11

def NewD11(x):

# tách các chữ số của x vào danh sách v, thêm 0 vào cuối dãy

v = [int(c) for c in str(x)] + [0]

C = L = 0 # tổng chẵn, lẻ

for i in range(1, len(v), 2):

C += v[i-1]

L += v[i]

return abs(C-L) % 11

**Leson 5: Divide & Conquer**

5. Mon July 22 2024

***Little problems***

### P1. Arithmetic Progression

*Tìm các số tự nhiên lẻ có ba chữ số. Ba chữ số này, theo trật tự từ trái qua phải*

*tạo thành một cấp số cộng.*

Specification

1. x là số tự nhiên có ba chữ số: x = 100a + 10b + c.
2. x là số lẻ nên chữ số hàng đơn vị c phải là lẻ: c ∈ {1, 3, 5, 7, 9}.
3. Chữ số hàng trăm của x phải khác 0: a = 1..9.
4. Nếu dãy a, b, c lập thành một cấp số cộng thì số đứng giữa b là trung bình cộng của hai số đầu và cuối: b = (a + c)/2 hay 2b = a+c.

Từ đặc tả 4 ta suy ra a+c là số chẵn. Do c lẻ, a+c chẵn nên a lẻ.

Nếu biết a và c ta tính được x = 100a +10(a + c) / 2 + c

= 100a + 5(a + c) + c = 105a + 6c.

Vì chỉ có 5 chữ số lẻ là 1, 3, 5, 7 và 9 nên tổ hợp của a và c sẽ cho ta 25 số.

Algorithm

[105\*a + 6\*c] ; a, c ∈ {1,3,5,7,9}

Program

# Số cấp cộng: Các số tự nhiên lẻ 3 chữ số

# tạo thành cấp cộng

def CapCong(m = 10):

# Hiển thị mỗi dòng m số

chuSoLe = [1,3,5,7,9]

d = 0

for a in chuSoLe:

x = 105\*a

for c in chuSoLe:

print(x + 6\*c, end = ' ')

d += 1

if d % m == 0: print()

print('\n Tổng cộng ', d, ' số')

CapCong(10)

print("\n T H E E N D .")

Output

111 123 135 147 159 321 333 345 357 369

531 543 555 567 579 741 753 765 777 789

951 963 975 987 999

Tổng cộng 25 số

T H E E N D .

Complexity

Khoảng 25 lần lặp.

Comment

1. Dựa vào nhận xét: dãy ba số *a, b, c* tạo thành cấp số cộng khi và chỉ khi *b* là trung bình cộng của a và c, tức là *2b = a + c* ta có thể giải bài toán trên bằng phương pháp vét cạn dùng ba vòng for như sau:

for a = 1 to 9 do

for b = 0 to 9 do

for c = 0 to 9 do

if odd(c) and (2\*b = a+c) then

Ghi nhận số 100\*a+10\*b+c;

Hàm **odd(c)** kiểm tra tính lẻ của số nguyên *c*.

Phương pháp vét cạn đòi hỏi khoảng 10\*10\*10 = 1000 lần tính toán, trong khi chỉ có 25 số, tức là một phần bốn mươi các số thoả mãn điều kiện của đầu bài. Phương pháp mô tả trong chương trình được gọi là *phương pháp sinh*: nó sinh ra đúng 25 số cần tìm.

2.Ta cần ghi nhận *phương pháp sinh*

|  |
| --- |
| ***Phương pháp sinh*** |
| *Thay vì duyệt tìm các đối tượng*  *hãy sinh ra chúng.* |

### P2. Geometric Progression

*Tìm các số tự nhiên có ba chữ số. Ba chữ số này, theo trật tự từ trái qua phải*

*tạo thành một cấp số nhân với công bội là một số tự nhiên khác 0.*

Specification

Chú ý rằng ta chỉ xét các cấp số trên dãy số tự nhiên với công bội *d* là một số nguyên dương. Gọi *x* là số cần tìm, ta có:

1. *x* là số có ba chữ số: *x* = 100*a* + 10*b + c*.

2. *a* = 1..9; *b* = *ad*; 0 < *c* = *bd = add* ≤ 9.

Hệ thức 2 cho phép ta tính giới hạn trên của *d*:



Ta cho *a* biến thiên trong khoảng 1..9 rồi cho công bội *d* biến thiên trong khoảng từ 1 đến .

Với mỗi cặp số *a* và *d* ta tính

*x* = 100*a*+10*ad*+*add* = *a*(100+10*d+dd*)

Tuy nhiên, ta có thể nhẩm tính trước cận trên của *d* thì sẽ đỡ phải gọi các hàm làm tròn *trunc*

và tính căn là những hàm thao tác trên số thực do đó sẽ tốn thời gian.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *a* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* |
| *Cận trên d* | 3 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Bảng trên cho ta biết,

* với *a* = 1 thì *d* biến thiên trong khoảng 1..3
* với *a* = 2 thì *d* biến thiên trong khoảng 2..2
* với *a* = 3..9 thì *d* biến thiên trong khoảng 1..1
* Ký pháp *a..b* cho ta dãy số nguyên từ *a* đến *b*.

Vậy ta sẽ dùng *phương pháp sinh.*

Program

# Số cấp nhân: Các số tự nhiên 3 chữ số

# tạo thành cấp nhân

def CapNhan(m):

# Hiển thị m số trên dòng

d = [0,3,2,1,1,1,1,1,1,1]

n = 0

for a in range(1,10):

for j in range(1,d[a]+1):

print(a\*(100 + 10\*j + j\*j), end = ' ')

n += 1

if n % m == 0: print()

print('\n Tổng cộng ', n, ' số.')

# APPLICATION

CapNhan(10)

print("\n T H E E N D .")

Output

111 124 139 222 248 333 444 555 666 777

888 999

Tổng cộng 12 số.

T H E E N D .

Complexity

Khoảng 12 lần lặp.

### P3. Greatest Common Divisor

*Ước chung lớn nhất của hai số nguyên a và b là số nguyên không âm lớn nhất d*

*mà a và b cùng chia hết. Viết hàm Gcd(a,b) cho ra ước chung lớn nhất của hai số a và b.*

Gcd(28, 20) = 4

Gcd(28, 0) = 28

Gcd(0, 0) = 0 (quy ước)

Properties

*Ký hiệu* (*a,b*) là hàm hai ngôi, cho *ước chung lớn nhất* của hai số tự nhiên *a* và *b*.

Các tính chất của hàm (*a, b*)

* *Tính giao hoán:* (*a, b*) = (*b, a*)
* *Tính lũy đẳng hoặc tính đồng nhất:* (*a, a*) = *a*
* *Tính zero:* (*a,* 0) = *a*
* *Tính đơn vị:* (*a,* 1) = 1
* *Tính trừ vào:* (*a, b*) = (*a-b, b*), nếu *a* ≥ *b*
* *Tính dư:* (*a, b*) = (*a mod b, b*)

Dựa vào các tính chất trên, ta xây dựng ba phương án tính (*a,b*) như sau.

Phương án 1. Cân đối: đặt hai số *a* và *b* lên hai đĩa cân thăng bằng rồi so sành:

* Nếu hai số bằng nhau *a = b*: cho kết quả *a* (theo tính lũy đẳng);
* Nếu *a* > *b*: trừ *b* vào *a*, *a* = *a* – *b* (theo tính trừ vào);
* Nếu *a* < *b*: trừ *a* vào *b*. *b* = *b* – *a* (theo tính trừ vào).

Ví dụ

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | a ? b | Thao tác |
| 28 | 20 | a > b | a = a – b |
| 8 | 20 | a < b | b = b – a |
| 8 | 12 | a < b | b = b – a |
| 8 | 4 | a > b | a = a – b |
| 4 | 4 | a = b | return a |

Để tính Gcd của hai số nguyên nói chung, trước hết cần chuyển các tham biến sang số tự nhiên.

# Phương án 1: Cân bằng

def Gcd1(a,b):

a, b = abs(a), abs(b)

if a == 0: return b

if b == 0: return a

while a != b:

if a > b: a -= b

else: b -= a

return a

Phương án 2 (dùng phép chia dư): Phép trừ liên tiếp *x* = *x* – *y* sẽ được thay bằng phép chia dư: *x* = *x* mod *y*. Thuật toán này mang tên nhà toán học Hy Lạp Cổ đại Euclide.

# Phương án 2: Chia dư (Euclid)

def Gcd2(a, b):

a, b = abs(a), abs(b)

while b:

r = a % b

a, b = b, r

return a

Ví dụ

|  |  |  |
| --- | --- | --- |
| a | b | r = a mod b |
| 28 | 20 | 28 % 20 = 8 |
| 20 | 8 | 20 % 8 = 4 |
| 8 | 4 | 8 % 4 = 0 |
| 4 | 0 | return 4 |

Phương án 3 (đệ quy)

Hàm *f*(*x*) được gọi là hàm đệ quy nếu *f* chứa lời gọi đến chính nó. Để tránh lặp vô hạn ta cần đặt giới hạn *x* cho mỗi lần gọi *f*(*x*).

# Phương án 3: Đệ quy

def Gcd(a, b):

return a if b == 0 else Gcd(b, a % b)

def Gcd3(a, b): return Gcd(abs(a), abs(b))

Chú ý

Bạn nên dùng phương án 2 hoặc 3 vì phương án 1 có thể chạy lâu, ví dụ Gcd1(1000000,1) sẽ pahir thực hiện một triệu phép trừ.

Chương trình

# Ước chung lớn nhất

# Gcd: The greatest common divisor

# Phương án 1: Cân bằng

def Gcd1(a,b):

a, b = abs(a), abs(b)

if a == 0: return b

if b == 0: return a

while a != b:

if a > b: a -= b

else: b -= a

return a

# Phương án 2: Chia dư (Euclid)

def Gcd2(a, b):

a, b = abs(a), abs(b)

while b:

r = a % b

a, b = b, r

return a

# Phương án 3: Đệ quy

def Gcd(a, b):

return a if b == 0 else Gcd(b, a % b)

def Gcd3(a, b): return Gcd(abs(a), abs(b))

def Run():

a = [28, 15, 170]

b = [20, 35, 68]

for x in a:

for y in b:

print(x,y,':',Gcd1(x,y),Gcd2(x,y),Gcd3(x,y))

# APPLICATION

Run()

print(' T H E E N D')

Output

28 20 : 4 4 4

28 35 : 7 7 7

28 68 : 4 4 4

15 20 : 5 5 5

15 35 : 5 5 5

15 68 : 1 1 1

170 20 : 10 10 10

170 35 : 5 5 5

170 68 : 34 34 34

T H E E N D

Độ phức tạp

Số phép chia dư trong thuật toán Euclid, Gcd(a,b), không vượt quà 5 lần chiều dài (số chữ số thập phân)

của số lớn nhất trong hai số a, b. Với các số kiểu int thì giá trị lớn nhất dài khoảng 10 chữ số,

do đó thuật toán Gcd cần tối đa khoảng 10×5 = 50 phép chia dư.

### P4. Greatest Common Divisor of Sequence

*Cho dãy gồm n > 1 số nguyên dương và một số nguyên dương k. Bạn được phép giảm*

*mỗi số trong dãy không quá k đơn vị để nhận được ước chung lớn nhất m. Hãy tìm giá trị lớn nhất của m.*

*Dữ liệu vào text file GCD.INP*

*Dòng đầu tiên hai số n và k*

*Tiếp đến là n số nguyên dương của dãy*

*Dữ liệu ra: hiển thị trên màn hình giá trị max m.*

Ví dụ

|  |  |
| --- | --- |
| GCD.INP | OUTPUT |
| 5 2  45 27 61 12 92 | 5 |

Gcd(45, 27-2=25, 61-1=60, 12-2=10, 92-2=90) = Gcd(45,25,60,10,90)5.

Algorithm

Nhận xét: Gcd của dãy số không lớn hơn số nhỏ nhất của dãy.

Gọi *m* là ước chung lớn nhất của dãy số *a*, ta có *m* ≤ min(*a*). Ta có thuật toán sau:

1. Đọc các phần tử của dãy *a* và tính giá trị *mina* = min(*a*)
2. Duyệt ngược *m* từ *mina* giảm đến 1, nếu mọi phần tử của *a* chía dư cho *m* đều cho số dư ≤ *k* thi *m* là số cần tìm.

Program

# Ước chung lớn nhất của dãy số

# Kiểm tra mọi a[i] mod m <= k ?

def Scan(a, m,k):

for x in a:

if x % m > k: return False

return True

def Run():

with open("GCD.INP") as f:

n, k = list(map(int, f.readline().split()))

print('n =',n,' k =',k)

a = list(map(int, f.readline().split()))

print('a =',a)

mina = min(a)

print('mina =', mina)

for m in range(mina, 0, -1):

if Scan(a, m, k):

print('Result: ', m)

return

# APPLICATION

Run()

print(' T H E E N D')

Output

n = 5 k = 2

a = [45, 27, 61, 12, 92]

mina = 12

Result: 5

T H E E N D

Độ phức tạp

Khoảng *nm* phép chia dư, *m* = min(*a*).

### P5. Least Comm Multiple

Lcm = BCNN = [a, b]

BCNN của hai số tự nhiên a và b là số tj nhiên nhỏ nhát m cùng chia hết cho a vfa b.

Properties

Ký hiệu [*a,b*] là hàm hai ngôi cho *bội chung nhỏ nhất* (Lcm) của hai số tự nhiên *a* và *b*.

Các tính chất của hàm [*a,b*]*:*

* *Tính giao hoán:* [*a,b*] = [*b,a*]
* *Tính lũy đẳng hoặc đồng nhất:* [*a,a*] = *a*
* *Tính đơn vị:* [*a,*1] = *a*
* *Quan hệ với Gcd:* [*a,b*] = (*a div* (*a,b*))\**b*

Từ tính chất cuối, ta suy ra

Lcm(a, b) = (a div Gcd(a,b)) \* b = a\*(b div Gcd(a,b))

Trong biểu thức trên ta làm phép chia trước để tránh tràn số.

# bcnn: The lowest common multiple

def Lcm(a, b):

return a\*(b // Gcd(a,b))

Độ phức tạp

Tương đương độ phức tạp tính Gcd(a,b) nghĩa là số phép chia dư gấp khoảng 5 lần số chữ số của max(a,b).

### P6. GCD and LCM of Sequence

Ví dụ

MGcd(3) = 3

MGcd(30, 27) = 3

MLcm(3) = 3

MLcm(30, 15) = 30

Program

# Pentagonal Number

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def Gcd(a, b):

return a if b == 0 else Gcd(b, a % b)

def Lcm(a, b):

return a\*(b // Gcd(a,b))

def MGcd(d):

a = 0

for x in d: a = Gcd(a, x)

return a

def MLcm(d):

a = 1

for x in d: a = Lcm(a, x)

return a

# APPLICATION

print(MGcd([12, 34, 5, 6, 10]))

print(MLcm([12, 34, 5, 6, 10]))

while True:

s = input('Get numbers or [.] to stop: ')

if s == '.': break

d = list(map(int, s.split()))

print('Gcd = ', MGcd(d))

print('Lcm = ', MLcm(d))

print(' T h e E n d')

### P7. LCM and GCD of Range

Tính BCNN của dãy a..b.

If a % b = 0 then Lcm(a,b) = a

BCNN : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

BCNN : 4, 5, 6, 7, 8, 9, 10 11 12 13 14

RLcm(a, b):

# Pentagonal Number

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def Gcd(a, b):

return a if b == 0 else Gcd(b, a % b)

def Lcm(a, b):

return a\*(b // Gcd(a,b))

def MGcd(d):

a = 0

for x in d: a = Gcd(a, x)

return a

def MLcm(d):

a = 1

for x in d: a = Lcm(a, x)

return a

def RLcm(d, c):

a = 1

for x in range(c//2+1, c+1):

a = Lcm(a, x)

return a

# APPLICATION

while True:

s = input('Get range or [.] to stop: ')

if s == '.': break

d, c = map(int, s.split())

x = RLcm(d,c)

v = list(range(d, c+1))

y = MLcm(v)

print(x, y, (x == y))

print(' T h e E n d')

Gcd: 1 2 3 4 5 6 7 = 1

Gcd 7 8 9 10 11 12 13 14 15 = 1

nguyen to cung nhau (a,b) = 1

Hai so lien ke la ntcn

**Leson 4: Dynamic Programming**

4. Thu July 18 2024

***Some little problems***

### P1. Quadratic equations

Program

# Quadratic Equation

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

# return n, x1, x2

def QE(a, b, c): # ax^2 + bx + c = 0

delta = b\*b - 4\*a\*c

if delta < 0: return 0, 0.0, 0.0

aa = a + a # 2a

if delta == 0:

x = -b // aa

return 1, x, x

# delta > 0

sd = delta\*\*0.5 # can delta

x1 = (-b+sd) // aa

x2 = (-b-sd) // aa

return 2, x1, x2

# APPLICATION

"""

Test

(x+2)^2 = x^2 + 4x + 4 = 0

a b c = 1 4 4

(x-2)(x+3) = x^2 - x - 6 = 0

a b c = 1 -1 -6

"""

while True:

s = input('a b c or [.] to stop : ')

if s == '.': break

a,b,c = map(int, s.split())

n,x1,x2 = QE(a, b, c)

if n == 0: print('No solutions'); continue

if n == 1: print('Duple solutions', x1, x2); continue

print('Two solutions', x1, x2)

print(' T h e E n d')

### P2. Triangular Number

*Với mỗi số tự nhiên n, số T(n) = 1 + 2 + ... + n được gọi là số tam giác. Dưới đây là sáu số tam giác đầu tiên:*

T(0) = 0

T(1) = 1

T(2) = 1 + 2 = 3

T(3) = 1 + 2 + 3 = 6

T(4) = 1 + 2 + 3 + 4 = 10

T(5) = 1 + 2 + 3 + 4 + 5 = 15

*Nhập các số tự nhiên x từ bàn phím và hiển thị YES nếu x là số tam giác; ngược lại hiển thị NO.*

Algorithm

T(n) = n(n+1)//2 (Gauss)

x = T(n) = n(n+1) // 2

2x = n(n+1)

Can(2x) = n ? n\*(n+1) = 2x

Program

# Triangular Numbers

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def T(n): return n\*(n+1)//2

# x = T(n) = n(n+1)// 2

# 2x = n(n+1)

def IsT(x):

if x < 0: return False

xx = x+x

n = int(xx\*\*0.5)

return n\*(n+1) == xx

def Test1(limit):

for n in range(limit):

x = T(n)

print('Is triangular ', x, end = ' ? ')

if (IsT(x)): print(' YES.')

else: print(' NO.')

x += 3

print('Is triangular ', x, end = ' ? ')

if (IsT(x)): print(' YES.')

else: print(' NO.')

def Test2():

while True:

s = input('get a number or [.] to stop: ')

if s == '.': break

if IsT(int(s)): print(' YES.')

else: print(' NO.')

# APPLICATION

Test1(20)

Test2()

print(' T h e E n d')

### P3. Pentagonal Number

*Với mỗi số tự nhiên n, số được gọi là số ngũ giác. Dưới đây là sáu số ngũ giác đầu tiên:*

P(0) = 0

P(1) = 1

P(2) = 5

P(3) = 12

P(4) = 22

P(5) = 35

*Viết chương trình hiển thị mỗi số nguyên không âm cho trước kèm theo thông báo YES nếu số đó là số ngũ giác; ngược lại thì thông báo NO. Nếu gặp số âm thì dừng chương trình.*

Algorithm

Nếu *x = 0* thì *x* là số ngũ giác đầu tiên*.*  Giả sử *x >* 0, khi đó *x* là số ngũ giác thứ *n*, *x* = *P*(*n*) khi và chỉ khi , tức là tam thức bậc hai có ít nhất một nghiệm là số tự nhiên *n*. Ta có , suy ra phải là một số tự nhiên.

Hàm PNum(x) return n nếu x là số ngũ giác thứ n, ngược lại, hàm return -1.

Phương án 1 cần vài phép tính số học.

# Phuong an 1

def PNum1(x):

if x == 0:

return 0

if x < 0:

return -1

n = (1 + int(1+24\*x)\*\*0.5) // 6

return n if P(n) == x else -1

Do *P*(*n*) là dãy tăng chặt nên ta cũng có thể dùng công thức đệ quy để sinh lại dãy *P*.

Ta có,

Vậy

Phương án 2 (cần lần lặp)

# Phuong an 2

def PNum2(x):

if x == 0:

return 0

y, d, n = 0, -2, 0

while y < x:

d += 3

y += d

n += 1

return n if y == x else -1

Program

# Pentagonal Number

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def P(n): return n\*(3\*n-1)//2

# x = P(n) = n(3n-1)// 2

# 2x = n(3n-1): 3n^2 - n - 2x = 0 Quadratic EquationD

# a = 3, b = -1, c = -2x, n1, n2 ? 1 nghiem nguyen

# return n: P(n) = x

def IsP(x):

if x < 0: return -1

if x == 0: return 0

sd = (1+24\*x)\*\*0.5

n = int((1+sd)//6)

return n if P(n) == x else -1

def Test1(limit):

for n in range(limit):

x = P(n)

id = IsP(x)

print('Is pentagonal number ', x, end = ' ? ')

if id != -1: print(' YES (' ,id,')')

else: prit(' NO.')

x += 3

id = IsP(x)

print('Is pentagonal number ', x, end = ' ? ')

if id != -1: print(' YES (' ,id,')')

else: print(' NO.')

def Test2():

while True:

s = input('get a number or [.] to stop: ')

if s == '.': break

x = int(s)

id = IsP(x)

if id != -1: print(' YES (' ,id,')')

else: print(' NO.')

# APPLICATION

Test1(20)

Test2()

print(' T h e E n d')

### P4. Hexagonal Number

*Với mỗi số tự nhiên n, số H được gọi là số lục giác. Dưới đây là sáu số lục giác đầu tiên:*

H(0) = 0

H(1) = 1

H(2) = 6

H(3) = 15

H(4) = 28

H(5) = 45

*Viết chương trình đọc từng số nguyên x từ input file NUM.INP và hiển thị YES nếu x là số lục giác; ngược lại hiển thị NO. Nếu x < 0 thì dừng chương trình*

**Leson 3: Searching**

3. Mon July 15 2024

|  |  |
| --- | --- |
| Newton và cuộc cách mạng trong khoa học | *Những bài toán nhỏ*  *đã dẫn tôi đến*  *những bước tiến lớn*  *Isac Newton* (1642-1727) |

***Các bài toán nhỏ vài dòng lệnh***

*P1. Nhập dãy số nguyên, tính tổng các số này.*

*P2. Nhập dãy số nguyên, tính tổng bình phương các số này*

*P3. Kiểm định định lý: tổng n số tự nhiên lẻ bằng*

*Ví dụ,*

*P4. Nhập dãy số nguyên tùy ý, tách thành hai dãy con chẵn và lẻ.*

*P5. Nhập dãy tùy ý, kiểm tra dãy có là palindrome không ?*

*P6. Nhập dãy số nguyên tùy ý, kiểm tra dáng điệu của dãy?*

*1. đồng nhất : mọi phần tử bằng nhau: 3 3 3 … 3*

*2. tăng chặt: 3 7 10 17….*

*3. đồng biến (tăng): 3 3 4 4 7 10 10 …*

*4. giảm chặt: 17 10 7 4…*

*5. nghịch biến (giảm): 10 10 7 7 4 1 -3*

*0. nhấp nhô: 1 7 2 10 4…*

***Rhythm***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | g | t | b | **r (bin)** |  | b = t = g = 0  for i in range(1,len(d)):  if d[i] == d[i-1]:  b = 1; continue  if d[i] > d[i-1]:  t = 1; continue  else: g = 1  r = (4\*g + 2\*t + b) % 7 % 6 |
| 0 | 0 | 0 | **0** | **?** |
| 0 | 0 | 1 | **1** | **=** |
| 0 | 1 | 0 | **2** | **↑** |
| 0 | 1 | 1 | **3** | **↑=** |
| 1 | 0 | 0 | **4** | **↓** |
| 1 | 0 | 1 | **5** | **↓=** |
| 1 | 1 | 0 | **6 🡪 0** | **~** |
| 1 | 1 | 1 | **7 🡪 0** | **~** |

*Program*

"""

Các bài toán nhỏ vài dòng lệnh

P1. Nhập dãy số nguyên, tính tổng các số này.

P2. Nhập dãy số nguyên, tính tổng bình phương các số này

P3. Kiểm định định lý: tổng n số tự nhiên lẻ bằng n^2;

Ví dụ, 1+3+5+7= 4^2=16.

P4. Nhập dãy số nguyên tùy ý, tách thành hai dãy con chẵn và lẻ.

P5. Nhập dãy tùy ý, kiểm tra dãy có là palindrome không ?

P6. Nhập dãy số nguyên tùy ý, kiểm tra dáng điệu của dãy?

1. đồng nhất : mọi phần tử bằng nhau: 3 3 3 … 3

2. tăng chặt: 3 7 10 17….

3. đồng biến (tăng): 3 3 4 4 7 10 10 …

4. giảm chặt: 17 10 7 4…

5. nghịch biến (giảm): 10 10 7 7 4 1 -3

0. nhấp nhô: 1 7 2 10 4…

"""

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def GetNum(msg = ' Input a number: '):

return int(input(msg))

def Sum(msg = 'Input: '): # P1

return sum(list(map(int, input(msg).split())))

def SumSquare(msg = 'Input: '): # P2 `

'''

d = list(map(int, input(msg).split()))

print(d)

v = list(map(lambda x: x\*x ,d))

print(sum(v))

'''

print(sum(list(map(lambda x: x\*x, \

list(map(int, input(msg).split()))\

))))

def Theorem(maxn): # P3

for n in range(1, maxn):

print('Test for n = ', n, end = ': ')

d = list(range(1, n+1, 2))

m = len(d)

s = sum(d)

print(' sum ', d, ' =', s, end = ': ')

if s == m\*m: print('YES.')

else: print('NO.')

def SplitEvenOdd(msg = 'Input: '): # P4

# d = list(map(int, input(msg).split()))

# print(d)

even, odd = [], []

for x in list(map(int, input(msg).split())):

if x % 2 == 0: even.append(x)

else: odd.append(x)

print(even, odd)

def Palindrome(msg = 'Input: '): # P5

s = input(msg)

print('YES.') if s == s[::-1] else print('NO.')

def Rhythm(msg = 'Input numbers: '): # P6

d = list(map(int, input(msg).split()))

# print(d)

b = t = g = 0

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

if d[i] == d[i-1]: b = 1; continue

if d[i] > d[i-1]: t = 1; continue

else: g = 1

print((4\*g + 2\*t + b) % 7 % 6)

# APPLICATION

# print(Sum('Sum of numbers: '))

# print(SumSquare('Sum square of numbers: '))

# Theorem(20)

# SplitEvenOdd('Input numbers: ')

while True:

# Sum()

# SumSquare()

# Theprem(20)

# SplitEvenOdd()

# Palindrome()

Rhythm()

Go('Press dot key [.] to stop: ')

print(' T h e E n d')

***Blind Searching***

**password** xxxx, x : 0..9

1. Brute Force: tan cong toan dien = vet can

k = 0..9999 ?

for k = 0..9999

Tach k thanh 4 chu so abcd

Kiem tra

2. Biet them thong tin: nam sinh, duoi 24 tuoi

k + y = 2024, k = 2024-y k = 2020…2024 ?

for k = 2020..9999

Tach k thanh 4 chu so abcd

Kiem tra

3. Biet them thong tin: sinh nhat ddmm

dd: 01..31, mm = 01..12 or

mm: 01..12, dd = 01..31

r

for k = 2020..9999

Tach k thanh 4 chu so abcd

Kiem tra

One eye searching: Backtracking: Vet, ko di lai duong da qua

**Leson 2**

2. Thu July 11 2024

Vi sao Next using for i in range(len(s)-2,-2,-1) ?

Go()?

A cartoon of a group of kids

Description automatically generated

'''

Nobita

Doraemon

Suka

Seko

Chaen

'''

def Go(msg = ' ? '):

if input(msg) == '.': exit(0)

def GetNum(msg = ' Input a number: '):

x = int(input(msg))

return x

# APPLICATION

x = GetNum()

print(x)

x = GetNum('Please, input a number: ')

print(x)

x = GetNum('Draemon say: Bạn nào cho tớ một số để tớ làm ảo thuật: ')

print(x)

x = GetNum('Nobita say: Cho tớ một số: ')

print(x)

x = GetNum('Suka say: Cậu nạp một số nhé: ')

print(x)

x = GetNum('Chaen say: Chúng mày nộp cho tao một số ngay: ')

print(x)

x = GetNum('Seko mỏ nhọn say: Tao thèm nhìn thấy một số quá: ')

print(x)

print(' T h e E n d')

**Leson 1: Introduction**

1. Mon July 8 2024

Some recommendations

* Clean Room: One line only + Test
* 5E : five examples