BÀI 4 LÀM MỊN DẦN TỪNG BƯỚC TỪ THUẬT TOÁN ĐẾN CHƯƠNG TRÌNH MÁY TÍNH

Học xong bài này, em sẽ:

- Giải thích được sơ bộ phương pháp làm mịn dần trong lập trình.
- Biết được mã giả là gì và sử dụng được mã giả làm mịn dần một số thuật toán đơn giản.

Khi lập trình giải bài toán theo một thuật toán đã cho, em sẽ bắt đầu như thế nào? Theo em cách làm như thế có đúng phương pháp không?

1 Mã giả và mô tả thuật toán bằng mã giả

Mã giả thường được sử dụng trong sách giáo khoa, giáo trình hay các bài nghiên cứu để mô tả thuật toán. Mã giả có thể mô tả thuật toán theo cách ngắn gọn mà vẫn làm rõ ý tưởng chính của thuật toán và đảm bảo sự chính xác. Mã giả độc lập với ngôn ngữ lập trình, môi trường lập trình thực hiện thuật toán. Mã giả là cách mô tả thuật toán rất gần với văn bản mã lệnh chương trình. Một mô tả thuật toán bằng mã giả thậm chí có thể coi như chương trình khung.

Không có một quy ước thống nhất chính thức nào về các "từ khoá", về các kí hiệu được sử dụng trong mã giả. Mã giả phỏng theo các mẫu câu lệnh rẽ nhánh, câu lệnh lặp của ngôn ngữ lập trình bậc cao; sử dụng các kí hiệu toán học, các dấu phép toán quen thuộc hay các kí hiệu gợi tả khác, dễ hiểu với nhiều người. Các kí hiệu được chọn và quy ước rõ để loại bỏ sự nhầm lẫn do khác biệt cách dùng giữa các ngôn ngữ lập trình. Có thể dùng thêm một số từ ngắn gọn sau khi đã định nghĩa rõ ràng.

Quy ước cụ thể khi viết mã giả

Ta sẽ ưu tiên dùng một số yếu tố của ngôn ngữ lập trình Python trong các bài học khi mô tả thuật toán bằng mã giả. Dưới đây là một số quy ước:

– Lời chú thích bắt đầu bằng dấu "#" cho đến hết dòng.

– Cấu trúc rẽ nhánh (phép lựa chọn) dùng mẫu câu lệnh `if...else`.

– Cấu trúc lặp (phép lặp):

+ Số lần lặp biết trước: Phỏng theo mẫu lệnh `for` của Python nhưng mô tả danh sách giá trị theo kiểu toán học. Ví dụ: `for` *biến* `in` $\{ i \mid i \text{ chẵn}, j + 1 \le i \le n - 1 \}$: ...

+ Số lần lặp chưa biết trước: Phỏng theo mẫu lệnh `while` của Python. Ví dụ: `while` *điều kiện* : ...

– Sử dụng các mức thụt lùi đầu dòng để đánh dấu kết thúc dãy lệnh tuần tự trong mỗi nhánh rẽ của phép lựa chọn hay trong thân vòng lặp của phép lặp.

– Các phép toán gồm:

+ Phép toán số học, phép so sánh.

Ví dụ: $+, -, *, /, >, <, =, \neq, \geq, \leq, \ldots$

+ Phép gán dùng dấu mũi tên trái.

Ví dụ: $x \leftarrow 5$ nghĩa là gán x nhận giá trị bằng 5. Không viết "$x = 5$" vì nó có nghĩa là phép so sánh x có bằng 5 hay không, cho kết quả là "đúng" (`True`) hoặc "sai" (`False`).

– Một số thành phần khác:

Các lời gọi hàm thư viện hay hàm do người lập trình định nghĩa có thể mô tả ngắn gọn bằng cách viết toán học. Ví dụ: `min` $\{a_j \mid j+1 \leq i \leq n-1\}$.

Có thể định nghĩa thêm các kí hiệu phép toán để chỉ một việc cụ thể nào đó. Ví dụ: Khi mô tả các thuật toán sắp xếp, người ta thường viết phép đổi chỗ hai phần tử x, y trong dãy số một cách ngắn gọn là `swap` (x, y).

2 Làm mịn dần các bước mô tả thuật toán

Mô tả thuật toán bằng liệt kê các bước còn chứa nhiều cụm từ của ngôn ngữ tự nhiên, mỗi cụm từ nêu một việc phải làm. Để lập trình thực hiện thuật toán, cần làm chi tiết dần từng bước. Theo em, đây có phải là "chia để trị" hay không? Vì sao?

Làm mịn dần các bước mô tả thuật toán là để tiến gần hơn đến các câu lệnh của ngôn ngữ lập trình. Ở đây lựa chọn sử dụng mã giả để trình bày, vì nó ngắn gọn, dễ hiểu và không phụ thuộc vào ngôn ngữ lập trình.

Cách thức chung: Chuyển các cụm từ mô tả một "việc cần làm" thành các đoạn mã giả, tiến gần hơn một bước đến các câu lệnh của chương trình chi tiết. Sau đây là các ví dụ minh hoạ.

Ví dụ 1. Thuật toán kiểm tra một số n là số nguyên tố.

– Đầu vào: một số nguyên dương n.

– Đầu ra: Nếu n là số nguyên tố trả về `True`, ngược lại trả về `False`.

Thuật toán khởi đầu đơn giản nhất là làm theo định nghĩa số nguyên tố.

Bước 1. Nếu $n = 1$ thì n không là số nguyên tố;

Bước 2. Nếu $n = 2$ thì n là số nguyên tố;

Bước 3. Nếu $n > 2$ thì kiểm tra tính nguyên tố của n, trả kết quả kiểm tra `True`/`False`;

Bước 4. Kết thúc.

Nhận thấy Bước 1 và Bước 2 đã chi tiết và đơn giản nên có thể chuyển thành câu lệnh Python dễ dàng. Riêng Bước 3 "Kiểm tra tính nguyên tố của n với $n > 2$" cần được chi tiết và "làm mịn dần" lần lượt theo từng nhận xét sau:

Nhận xét 1: Nếu $n > 2$ thì với k ($2 \le k < n$) là số nguyên dương bất kì mà n chia hết cho k thì n không là số nguyên tố. Một cách để chi tiết cho Bước 3 như sau:

– Với k nào đó thoả mãn $2 \le k < n$:

Nếu n chia hết cho k thì n không là số nguyên tố.

– Các số k ($2 \le k < n$) được biểu diễn qua hàm `range` bằng câu lệnh: `range(2,n)`.

Hình 1 minh hoạ mã giả và các câu lệnh Python kiểm tra số nguyên tố sau khi chi tiết Bước 3 theo Nhận xét 1.

Mã giả		Chương trình
if $n = 1$:	1	`if (n == 1):`
Trả về ***False***	2	`    return False`
if $n = 2$:	3	`if (n == 2):`
Trả về ***True***	4	`    return True`
for k ***in*** $\{k \mid 2 \le k \le n-1\}$:	5	`for k in range(2, n):`
if n chia hết cho k:	6	`    if (n%k == 0):`
Trả về ***False***	7	`        return False`
Trả về ***True***	8	`return True`

Hình 1. Mã giả và các câu lệnh Python sau khi chi tiết Bước 3 theo Nhận xét 1

Nhận xét 2: n chia hết cho k nghĩa là $n = k.m$. Như vậy, hoặc $k \le \sqrt{n}$ hoặc $m \le \sqrt{n}$. Khi đó, ta chỉ cần kiểm tra với k không lớn hơn $\sqrt{n}$. Một cách khác để chi tiết cho Bước 3 như sau:

– Với k nào đó thoả mãn $2 \le k \le \sqrt{n}$:

Nếu n chia hết cho k thì n không là số nguyên tố.

– Các số k ($2 \le k \le \sqrt{n}$) được biểu diễn qua hàm `range` bằng câu lệnh: `range(2,int(math.sqrt(n))+1)`.

Hình 2 minh hoạ mã giả và các câu lệnh Python kiểm tra số nguyên tố ở Bước 3 sau khi chi tiết theo Nhận xét 2.

Mã giả		Chương trình
for k ***in*** $\{k \mid 2 \le k \le \sqrt{n}\}$:	5	`for k in range(2,int (math.sqrt(n))+1):`
if n chia hết cho k:	6	`    if (n%k == 0):`
Trả về ***False***	7	`        return False`

Hình 2. Mã giả và các câu lệnh Python sau khi chi tiết Bước 3 theo Nhận xét 2

Nhận xét 3: Số chẵn lớn hơn 2 không là số nguyên tố. Như vậy chỉ cần kiểm tra với k lẻ và không lớn hơn $\sqrt{n}$. Một cách khác để chi tiết hơn cho Bước 3 như sau:

– Nếu n chẵn và $n > 2$ thì n không là số nguyên tố.

– Trái lại, kiểm tra n chia hết cho k với k lẻ và $3 \le k \le \sqrt{n}$.

– Các số k lẻ ($3 \le k \le \sqrt{n}$) được biểu diễn qua hàm `range` bằng câu lệnh: `range(3,int(math.sqrt(n))+1,2)`.

Hình 3 minh hoạ mã giả và các câu lệnh Python kiểm tra số nguyên tố ở Bước 3 sau khi chi tiết theo Nhận xét 3.

Mã giả		Chương trình
if $n > 2$ và n chẵn:	5	`if n>2 and n%2 == 0:`
Trả về ***False***	6	`  return False`
else:	7	`else:`
for k ***in*** $\{k \mid k$ lẻ, $3 \le k \le \sqrt{n}\}$:	8	`  for k in range(3,int (math.sqrt(n))+1,2):`
if n chia hết cho k:	9	`    if (n%k == 0):`
Trả về ***False***	10	`      return False`

Hình 3. Mã giả và các câu lệnh Python sau khi chi tiết Bước 3 theo Nhận xét 3

Ví dụ 2. Bài toán sàng số nguyên tố.

Cho trước số tự nhiên n, hãy sàng lọc chỉ giữ lại những số là số nguyên tố trong dãy $\{0, 1, 2, \ldots, n\}$.

– Đầu vào: một số nguyên dương n.

– Đầu ra: in ra danh sách tương ứng đánh dấu `True` (là số nguyên tố) hay `False` (là hợp số).

Thuật toán thô: Đục bỏ dần các số $m > 2$ là bội số của 2, 3, 4, 5,... cho đến khi hết các bội số thì còn lại các số nguyên tố.

Bước 1. Tạo danh sách *prime* gồm $n + 1$ giá trị logic `True`;

Bước 2. Với $m > 2$, kiểm tra nếu m là một bội số của k ($k < m$) thì gán *prime*[m] = `False`;

Bước 3. Gán *prime*[0] = `False`; *prime*[1] = `False`;

Một cách chi tiết Bước 2 như sau:

– Bắt đầu với $m = 3$;

– ***Lặp khi*** $m \le n$:

Nếu với k nào đó ($2 \le k \le m - 1$) mà m chia hết cho k thì m không là số nguyên tố;

$m \leftarrow m + 1$

Hết lặp

Mã giả và các câu lệnh Python tương ứng cho Bước 2 có trong *Hình 4*.

Mã giả		Chương trình
$m \leftarrow 3$	1	`m = 3`
while $m \leq n$:	2	`while (m <= n):`
for k ***in*** $\{k \mid 2 \leq k \leq m-1\}$:	3	`for k in range(2,m):`
if $m \% k = 0$:	4	`if m % k == 0:`
prime$[m] \leftarrow$ ***False***	5	`prime[m] = False`
$m \leftarrow m+1$	6	`m += 1`

Hình 4. Mã giả và các câu lệnh Python chi tiết Bước 2

Thuật toán sàng Eratosthenes: Sàng Eratosthenes là một thuật toán cổ để tìm tất cả các số nguyên tố nhỏ hơn hay bằng n. Thuật toán được cho là tìm ra bởi nhà toán học Hy Lạp trước Công nguyên tên là Eratosthenes.Thuật toán kiểm tra số m là hợp số theo cách hiệu quả hơn.

Ý tưởng: Đục bỏ dần các số không nguyên tố bằng cách đánh dấu "là hợp số" (không phải số nguyên tố) mỗi khi biết số đó là bội số của một số nguyên tố.

3 Thực hành

a) Đọc mã lệnh của thuật toán Eratosthenes cho ở *Hình 5* và mô tả liệt kê các bước của thuật toán và bằng mã giả.

File Edit Format Run Options Window Help

```
def SieveOfEratosthenes(n):
    #Tạo mảng biến Boolean "prime[0..n]"; gán trị ban đầu tất cả là True
    #Kết cục prime[i] sẽ là False nếu i không là số nguyên tố
    #Còn lại là số nguyên tố
    prime = [True for i in range(n + 1)]
    p = 2
    while (p * p <= n):
        #Nếu prime[p] không bị sửa, p là nguyên tố
        if  prime[p]:
            #Đục bỏ các bội số của p
            for i in range(p * p, n + 1, p):
                prime[i] = False
        p += 1
    prime[0]= False
    prime[1]= False
    return prime
```

Hình 5. Mã lệnh của thuật toán Eratosthenes

b) Em hãy viết chương trình thực hiện sàng số nguyên tố sử dụng thuật toán thô và sử dụng thuật toán Eratosthenes. Sau đó chạy thử và so sánh kết quả.

Câu 1. Hãy nêu một điều kiện sàng khác cho bài toán sàng số: In ra danh sách các số nguyên dương nhỏ hơn n và thoả mãn điều kiện sàng mới.

Gợi ý: Ví dụ "không là số chính phương".

Câu 2. Viết mô tả mã giả cho thuật toán tương ứng với Câu 1.

Câu 1. Hãy cho biết cách viết các dấu phép toán số học, phép so sánh bằng mã giả.

Câu 2. Hãy cho biết cách viết phép gán bằng mã giả; dấu bằng "=" có ý nghĩa gì trong mã giả.

Câu 3. Cho câu lệnh lặp bằng mã giả như ở hình bên. Hãy diễn giải ý nghĩa và cho biết kết quả là gì nếu bắt đầu ta có j nhận giá trị 5 và n nhận giá trị 15.

```
for i in { i | i chẵn, j + 1 ≤ i ≤ n − 1 }:
    In ra i
```

Tóm tắt bài học

- Mã giả là một cách mô tả thuật toán độc lập với ngôn ngữ lập trình và tạo thuận lợi cho việc chuyển thuật toán thành chương trình máy tính.
- Từ mô tả thuật toán bằng liệt kê các bước, chuyển dần những cụm từ mô tả một công việc thành mã giả bằng cách làm chi tiết từng bước cách thực hiện công việc đó.
- Chuyển câu lệnh mã giả thành mã lệnh của ngôn ngữ lập trình để có văn bản chương trình.

BÀI TÌM HIỂU THÊM

KIỂM TRA TÍNH NGUYÊN TỐ THEO CÁCH NGẪU NHIÊN

Bài toán kiểm tra tính nguyên tố của một số nguyên dương rất lớn trở nên đặc biệt quan trọng khi các hệ mật mã khoá công khai ra đời.

Bên cạnh các phương pháp cho kết quả chính xác, có các phương pháp kiểm tra theo xác suất, dùng các thuật toán ngẫu nhiên. Sau một loạt lần kiểm tra, nếu không tìm được bằng chứng chứng tỏ n là hợp số thì ta kết luận n là số nguyên tố. Xác suất kết luận sai càng nhỏ khi số lần kiểm tra được tăng thêm càng nhiều.

Ví dụ, các phép kiểm tra tính nguyên tố theo cách ngẫu nhiên:

- Phép kiểm tra Fermat (ít được sử dụng).
- Phép Kiểm tra Miller-Rabin.
- Phép kiểm tra Solovay-Strassen.