BÀI 23 KIỂM THỬ VÀ ĐÁNH GIÁ CHƯƠNG TRÌNH

SAU BÀI HỌC NÀY EM SẼ:

- Biết được vai trò của kiểm thử làm tăng độ tin cậy của chương trình nhưng chưa chứng minh được tính đúng của chương trình.
- Biết được các tiêu chí đánh giá hiệu quả và tính đúng của chương trình.

Trong các bài trước em đã học cách thiết kế thuật toán cho một số bài toán như bài toán tìm kiếm, bài toán sắp xếp và thiết lập chương trình thực hiện thuật toán đó. Một bài toán có thể có nhiều thuật toán khác nhau và do đó có thể có nhiều chương trình khác nhau cùng giải quyết một bài toán. Hãy thảo luận và trả lời các câu hỏi sau:

– Làm thế nào để biết trong các thuật toán giải cùng một bài toán thì thuật toán nào là tốt nhất?

– Có những tiêu chí nào để đánh giá tính "tối ưu" của một thuật toán?

1. VAI TRÒ CỦA KIỂM THỬ CHƯƠNG TRÌNH

Hoạt động 1 Tìm hiểu ý nghĩa của kiểm thử chương trình

Ở lớp 10, em đã học một số phương pháp kiểm thử chương trình. Em hãy thảo luận với các bạn về các phương pháp kiểm thử sau, nêu ý nghĩa của chúng trong việc đánh giá độ tin cậy và chứng minh tính đúng của chương trình:

1. Tạo các bộ dữ liệu kiểm thử (test) để kiểm tra dữ liệu đầu ra có chính xác hay không.
2. Thiết lập điểm dừng hoặc cho chương trình chạy theo từng lệnh để kiểm tra và tìm ra lỗi (bug) của chương trình.
3. Thực hiện in dữ liệu trung gian trong quá trình kiểm thử để tìm ra lỗi của chương trình (nếu có).

– Phương pháp 1 sử dụng các bộ dữ liệu kiểm thử để kiểm tra tính đúng của chương trình. Nếu phát hiện lỗi không chính xác của dữ liệu đầu ra thì kết luận ngay thuật toán và chương trình không đúng. Nếu với các bộ dữ liệu kiểm thử, dữ liệu đầu ra đều đúng thì điều đó chưa chứng minh được thuật toán hay chương trình đúng. Kết quả đó chỉ làm tăng khả năng đúng của chương trình.

– Mục đích của các phương pháp 2 và 3 là tìm ra lỗi và sửa lỗi của chương trình. Như vậy, với việc tìm ra và sửa lỗi chứng tỏ chương trình trở nên tin cậy hơn, tốt hơn. Tuy nhiên, điều đó không chứng minh được tính đúng của thuật toán và chương trình.

Một thuật toán nếu được thiết kế đúng thì với mọi bộ dữ liệu đầu vào sẽ nhận được bộ dữ liệu đầu ra đúng tương ứng. Các phương pháp kiểm thử không có tính năng chứng minh được tính đúng của một thuật toán.

Kiểm thử sẽ giúp tăng độ tin cậy của chương trình nhưng chưa chứng minh được tính đúng của thuật toán và chương trình.

1. Giả sử em thiết lập chương trình giải bài toán nào đó. Em đã kiểm thử với 10 bộ dữ liệu và tất cả các kết quả đều đúng. Khi đó có thể kết luận chương trình đó đúng hay chưa?
2. Giả sử một chương trình kiểm thử với 10 bộ dữ liệu cho kết quả 9 lần đúng, 1 lần sai. Chương trình đó là sai hay đúng?

2. KIỂM TRA TÍNH ĐÚNG CỦA CHƯƠNG TRÌNH

Hoạt động 2 **Tìm hiểu cách kiểm tra tính đúng của chương trình**

Quan sát chương trình mô tả thuật toán sắp xếp chèn. Hãy thảo luận và đưa ra các lập luận để kiểm tra tính đúng của thuật toán sắp xếp chèn.

Chương trình thực hiện thuật toán sắp xếp chèn để sắp xếp dãy A cho trước theo thứ tự tăng dần như sau:

```
1 def InsertionSort(A,n):
2   for i in range(1,n):
3       value = A[i]
4       k = i-1
5       # di chuyển các phần tử A[k] > value qua phải
6       while k >= 0 and A[k] > value:
7         A[k+1] = A[k]
8         k = k - 1
9       A[k+1] = value
```

Trao đổi 1. Cài đặt chương trình sắp xếp chèn, nhập một số bộ dữ liệu đầu vào bất kì cho dãy A và kiểm tra kết quả đầu ra. Ta sẽ nhận thấy các kết quả đầu ra của chương trình đều đúng là dãy A được sắp xếp theo thứ tự tăng dần. Từ đó kết luận thuật toán sắp xếp chèn của chương trình trên là đúng.

Trao đổi 2. Giả sử dãy A ban đầu là A[0], A[1], ..., A[n–1] với n là độ dài dãy A. Dòng lệnh 2 là một vòng lặp gồm n – 1 bước, mỗi bước gán i chạy từ 1 đến n – 1. Với mỗi i (từ 1 đến n – 1), các dòng lệnh từ 3 đến 9 thực hiện việc "chèn" phần tử A[i] vào một vị trí trong dãy con phía trước A[0], A[1], ..., A[i – 1]. Giá trị A[i] sẽ được chèn vào vị trí giữa hai phần tử A[k] và A[k+1] nếu $A[k] \le A[i] < A[k+1]$. Do đó các dòng lệnh từ 3 đến 9 sẽ chèn phần tử A[i] vào đúng vị trí ở dãy đầu ra được sắp xếp tăng dần. Từ đó suy ra đến bước cuối cùng dãy A đã được sắp xếp đúng theo thứ tự.

Trao đổi 3. Trong khoa học máy tính, việc kiểm tra tính đúng của thuật toán đều phải chứng minh bằng lôgic toán học. Một phương pháp chứng minh thường dùng là *bất biến vòng lặp*. Ví dụ với thuật toán sắp xếp chèn ở trên cần chứng minh mệnh đề sau bằng bất biến vòng lặp:

Mệnh đề: Sau mỗi bước lặp i của vòng lặp, dãy con A[0], A[1], ..., A[i – 1], A[i] đã được sắp xếp đúng theo thứ tự tăng dần.

Nếu mệnh đề trên được chứng minh bằng toán học thì áp dụng với i = n – 1 ta sẽ kết luận thuật toán sắp xếp chèn là đúng.

Qua các trao đổi trên ta có nhận xét:

– Trong trao đổi 1, việc sử dụng các bộ dữ liệu kiểm thử chưa chứng minh được tính đúng của thuật toán và chương trình. Tuy nhiên, nếu thử được càng nhiều bộ dữ liệu kiểm thử thì độ tin cậy của chương trình càng cao.

– Trao đổi 2 là một suy luận lôgic vì nó bắt nguồn từ ý tưởng chính của thuật toán sắp xếp chèn. Đây là một cách lập luận đơn giản thường được sử dụng để chứng minh tính đúng của thuật toán.

– Trao đổi 3 cho biết việc chứng minh tính đúng của chương trình thường bằng lập luận toán học, sử dụng phương pháp quy nạp toán học. Đây là cách tốt nhất để chứng minh tính đúng của một thuật toán.

> Tính đúng của thuật toán cần được chứng minh bằng lập luận toán học. Sử dụng các bộ dữ liệu kiểm thử có thể làm tăng độ tin cậy của chương trình nhưng chưa chứng minh được tính đúng của thuật toán.

1. Chương trình sau giải bài toán: Yêu cầu nhập số tự nhiên n và tính tổng 1 + 2 + ... + n.

```
n = int(input("Nhập số tự nhiên n: "))
S = 0
for i in range(n+1):
    S = S + i
print(S)
```

Chương trình trên có đúng không?

2. Chương trình sau giải bài toán đếm số các ước số thực sự của số tự nhiên n.

```
def dem(n):
    count = 0
    k = 2
    while k < n:
        if n%k == 0:
            count = count + 1
        k = k + 1
    return count
```

Chương trình trên là đúng hay sai?

3. ĐÁNH GIÁ HIỆU QUẢ CHƯƠNG TRÌNH

Hoạt động 3 **Những tiêu chí đánh giá tính hiệu quả của chương trình**

Thảo luận về các tiêu chí đánh giá tính hiệu quả của thuật toán hay chương trình giải một bài toán.

1. Tiêu chí quan trọng nhất là thời gian chạy chương trình phải nhanh, không cần quan tâm đến không gian bộ nhớ sử dụng của chương trình.
2. Tiêu chí tiết kiệm bộ nhớ là quan trọng nhất, sau đó mới đến thời gian chạy chương trình.
3. Các tiêu chí 1 và 2 không quan trọng mà quan trọng là chương trình được viết một cách đơn giản, rõ ràng, dễ hiểu và áp dụng.

Hiệu quả hay tính tối ưu của chương trình thường được xem xét trên cơ sở *đánh giá độ phức tạp tính toán* (computational complexity) là lượng tài nguyên (amounts of resources) cần thiết để thực hiện chương trình. Hai loại độ phức tạp tính toán phổ biến nhất đó là:

– **Độ phức tạp thời gian** (time complexity) được xác định là thời gian thực hiện chương trình/thuật toán. Thời gian này phụ thuộc vào khối lượng của dữ liệu cần phải lưu trữ trong quá trình thực hiện chương trình/thuật toán, *đặc biệt liên quan tới các bước giải quyết một vấn đề cụ thể* đưa ra trong chương trình/thuật toán.

– **Độ phức tạp không gian** (space complexity) được xác định là tài nguyên của máy tính trong đó có phần bộ nhớ được sử dụng để thực hiện chương trình.

Một chương trình/thuật toán là ***hiệu quả*** (efficient) nếu độ phức tạp của thuật toán này là thấp, nghĩa là *tốn ít thời gian* và *tốn ít bộ nhớ cần thiết* để thực hiện chương trình/thuật toán.

Ngoài ra để đánh giá hiệu quả chương trình đôi khi người ta còn quan tâm tới các tiêu chí như tính dễ hiểu, rõ ràng, ngắn gọn, dễ bảo trì, dễ cài đặt,… của chương trình.

Độ phức tạp thời gian thường bị ảnh hưởng bởi số lần thực hiện các phép toán/câu lệnh có trong chương trình/thuật toán. Với các bài toán kĩ thuật, thiết kế, nghiên cứu khoa học đòi hỏi khối lượng tính toán rất lớn thì việc thiết kế các chương trình/thuật toán có độ phức tạp thời gian thấp luôn là mối quan tâm hàng đầu. Còn đối với những bài toán chạy liên tục như các dịch vụ trực tuyến thì độ tin cậy của phần mềm, tính dễ cài đặt, dễ duy trì, dễ sử dụng lại có vai trò quan trọng.

Trong phạm vi kiến thức phổ thông, ta sẽ chỉ quan tâm tới độ phức tạp thời gian của chương trình/thuật toán.

Trong các bài học sau em sẽ làm quen với việc *xác định độ phức tạp thời gian dựa trên cơ sở ước lượng thời gian thực hiện các bước (các câu lệnh)* trong chương trình/thuật toán.

Chương trình sau cho em biết một cách đơn giản đo thời gian chạy của chương trình. Hàm chính mô tả thuật toán sắp xếp chèn là **InsertionSort(A)** với A là dãy số đầu vào. Hàm **perf_counter()** đo nhịp xung CPU tại thời điểm thực hiện, đơn vị tính thời gian là giây.

Chương trình sau tính được thời gian chạy của hàm **InsertionSort(A)** hay chính là thời gian chạy của thuật toán sắp xếp chèn. Hàm tính thời gian tại thời điểm trước và sau khi chạy hàm chính tại các dòng 10, 12. Lệnh tại dòng 13 sẽ in ra thời gian chạy của hàm chính **InsertionSort(A)**.

```
1  from time import perf_counter
2  def InsertionSort(A):
3      n = len(A)
4      for i in range(1,n):
5          value = A[i]
6          j = i-1
7          while j >= 0 and A[j] > value:
8              A[j+1] = A[j]
9              j = j -1
10         A[j+1] = value
11 A = [3,0,1,10,7,9,5]
```

```
t1 = perf_counter()
InsertionSort(A)
t2 = perf_counter()
print(t2-t1)
```

Tính hiệu quả của chương trình/thuật toán được xem xét trên cơ sở đánh giá độ phức tạp tính toán. Độ phức tạp tính toán quan trọng nhất là độ phức tạp thời gian có liên quan trực tiếp tới các câu lệnh được thực hiện trong chương trình/thuật toán.

Hai tiêu chí đánh giá độ phức tạp tính toán quan trọng nhất là gì?

LUYỆN TẬP

1. Hãy xây dựng các bộ dữ liệu kiểm thử để tìm lỗi cho chương trình tính n! với n là một số nguyên dương nhập từ bàn phím.

```
n = int(input("Nhập số n: "))
if n > 0:
    giaithua=1
    for i in range(1,n+1):
        giaithua = giaithua*i
    print(n,"giai thừa bằng:",giaithua)
```

2. Xét hàm mô tả thuật toán tính tổng các số chẵn của một dãy số cho trước.

```
def tongchan(A):
    S = 0
    for i in range(len(A)):
        if A[i] % 2 == 0:
            S = S + A[i]
    return S
```

Tìm hai bộ dữ liệu đầu vào có cùng kích thước của thuật toán trên nhưng có thời gian chạy khác nhau.

VẬN DỤNG

1. Cho dãy các số A = [3, 1, 0, 10, 13, 16, 9, 7, 5, 11].

a) Viết chương trình mô tả thuật toán tìm kiếm phần tử C = 9 của dãy trên. Tính thời gian chính xác thực hiện công việc tìm kiếm này.

b) Giả sử dãy A ở trên đã được sắp xếp theo thứ tự tăng dần: A = [0, 1, 3, 5, 7, 9, 10, 11, 13, 16]. Viết chương trình tìm kiếm nhị phân để tìm kiếm phần tử C = 9, đo thời gian thực hiện thuật toán. So sánh với kết quả tìm kiếm ở câu a.

2. Viết ba chương trình mô phỏng các thuật toán sắp xếp chèn, sắp xếp chọn và sắp xếp nổi bọt mà em đã biết. Cho biết thời gian thực tế thực hiện các chương trình trên với bộ dữ liệu đầu vào là dãy A = [3, 1, 0, 10, 13, 16, 9, 7, 5, 11].