BÀI 27 THỰC HÀNH THIẾT KẾ CHƯƠNG TRÌNH THEO PHƯƠNG PHÁP LÀM MỊN DẦN

SAU BÀI HỌC NÀY EM SẼ:

- Thực hành thiết kế chương trình theo phương pháp làm mịn dần.

Phương pháp làm mịn dần là một trong các cách tiếp cận tổng quát khi giải quyết các bài toán cụ thể. Em có thể sử dụng sơ đồ hình cây để mô tả phương pháp này không?

Nhiệm vụ 1. Kiểm tra hoán vị

Cho trước một dãy n số, các số được kí hiệu A[0], A[1], ..., A[n-1]. Cần thiết kế chương trình kiểm tra xem dãy trên có phải là một hoán vị của dãy số 1, 2, ..., n hay không. Chương trình cần thông báo kết quả là CÓ hoặc KHÔNG.

Hướng dẫn: *Phân tích thiết kế*

a) Tìm hiểu bài toán

Bài toán gốc: Cho trước dãy số A gồm n phần tử, cần kiểm tra xem A có phải là một hoán vị của dãy số 1, 2, .., n hay không.

b) Thiết kế theo phương pháp làm mịn dần

Bước 1. Thiết lập ý tưởng thiết kế ban đầu.

Ý tưởng ban đầu để giải bài toán này khá đơn giản: Sắp xếp A theo thứ tự tăng dần. Sau đó chỉ cần so sánh A có trùng khít với dãy [1, 2, ..., n] hay không. Như vậy sơ đồ khung ban đầu của lời giải sẽ như sau:

```
Sắp xếp dãy A theo thứ tự tăng dần.
Kiểm tra A có phải là dãy 1, 2, ..., n hay không
```

Bước 2. Sắp xếp dãy A

Việc sắp xếp dãy A theo thứ tự tăng dần được mô tả bằng hàm sapxep(A) theo một trong các thuật toán sắp xếp mà chúng ta đã biết. Ví dụ bằng thuật toán sau:

```
def sapxep(A):
    for i in range(len(A)):
        j = i
        while j > 0 and A[j] < A[j-1]:
            A[j],A[j-1] = A[j-1],A[j]
            j = j - 1
```

Bước 3. Với dãy A đã được sắp xếp, kiểm tra A có phải là dãy [1, 2, ..., n]?

Việc kiểm tra này có thể được thực hiện đơn giản như sau:

```
Kiểm tra lần lượt các phần tử của A với các phần tử tương ứng của dãy
1, 2, .., n. Nếu tất cả các so sánh đều bằng nhau thì trả về kết quả True,
ngược lại trả về False.
```

Sử dụng biến kq để trả lại kết quả của việc so sánh A và dãy các số 1, 2, ..., n, , ta có thể viết đoạn chương trình chi tiết thực hiện công việc được mô tả trên như sau:

```
    sapxep(A)
    kq = True
    for i in range(len(A)):
        if A[i] != i+1:
            kq = False
            break
    return kq
```

Kết quả của toàn bộ bước này có thể viết dưới dạng hàm kt_hoanvi(A), trong đó A là dãy số ban đầu. Hàm sẽ trả về True nếu A là hoán vị của [1, 2, ..., n], ngược lại trả về False.

```
def kt_hoanvi(A):
    sapxep(A)
    kq = True
    for i in range(len(A)):
        if A[i] != i+1:
            kq = False
            break
    return kq
```

c) Chương trình hoàn chỉnh

Tổng hợp các bước làm mịn trên chúng ta thu được chương trình hoàn chỉnh cho bài toán. Chương trình hoàn chỉnh sẽ có hai chương trình con là hàm sapxep() và kt_hoanvi().

```
def sapxep(A):
    for i in range(len(A)):
        j = i
        while j > 0 and A[j] < A[j-1]:
            A[j],A[j-1] = A[j-1],A[j]
            j = j - 1

def kt_hoanvi(A):
    sapxep(A)
    kq = True
    for i in range(len(A)):
        if A[i] != i+1:
            kq = False
            break
    return kq

# Chương trình chính
A = [2,1,9,10,8,6,5,2,3,1]
```

```
if kt_hoanvi(A):
    print("CÓ")
else:
    print("KHÔNG")
```

Nhiệm vụ 2. Đếm số lần lặp.

Thiết kế và viết chương trình theo phương pháp làm mịn dần cho bài toán sau: Cho trước dãy số A[0], A[1], ..., A[n-1]. Cần tính được mỗi giá trị của các phần tử của dãy trên được lặp lại bao nhiêu lần trong dãy đó. Kết quả cần được đưa ra dãy B. Như vậy dãy B sẽ có ý nghĩa như sau: B[k] = số lần lặp của phần tử A[k] trong dãy A.

Ví dụ nếu A = [2, 1, 1, 3, 5, 10, 2, 5] thì B = [2, 2, 2, 1, 2, 1, 2, 2].

Hướng dẫn: *Phân tích thiết kế*

a) Tìm hiểu bài toán

Bài toán gốc: cho trước dãy số A có n phần tử. Cần tạo ra một dãy mới là số các lần lặp của các phần tử tương ứng trong A.

b) Thiết kế theo phương pháp làm mịn dần

Bước 1. Thiết lập ý tưởng thiết kế ban đầu.

Theo yêu cầu chúng ta cần xây dựng dãy B có cùng kích thước với dãy A và các phần tử của B liên hệ với dãy A như sau: B[k] = số lần lặp của phần tử A[k] trong dãy A. Do vậy thiết kế lời giải sơ lược ban đầu của bài toán như sau:

```
Thiết lập dãy B rỗng
for i in range(len(A)):
    Tính số lần lặp của A[i] trong dãy A
    Bổ sung giá trị này vào dãy B.
Trả về dãy B
```

Bước 2. Thiết lập dãy B rỗng.

Việc này được thực hiện đơn giản bằng lệnh B = [].

Bước 3. Tính số lần lặp của A[i] trong dãy A.

Công việc này có thể được thực hiện thông qua lời gọi hàm lap(x,A) tính số lần lặp của một giá trị bất kì trong dãy A. Hàm lap(x,A) có thể được viết như sau:

```
def lap(x,A):
    S = 0
    for a in A:
        if a == x:
            S = S + 1
    return S
```

Bước 4. Bổ sung số lần lặp vào dãy B.

Tổng hợp kết quả của các bước trên, chúng ta thu được đoạn chương trình hoàn chỉnh đã nêu ở bước 1 như sau:

```
    B = []
    for i in range(len(A)):
        B.append(lap(A[i],A))
    return B
```

Bước 5. Trả về dãy B cần tìm của bài toán.

Để hoàn thiện toàn bộ chương trình chúng ta sẽ thiết lập hàm tinh_lap(A) mô tả đoạn chương trình đã nêu trong bước 4. Hàm tinh_lap(A) trả về dãy B cần tìm:

```
def tinh_lap(A):
    B = []
    for a in A:
        B.append(lap(a,A))
    return B
```

c) Chương trình hoàn chỉnh

Tới đây việc thiết kế theo phương pháp làm mịn dần kết thúc. Chương trình hoàn chỉnh được viết như sau:

```
def lap(x,A):
    S = 0
    for a in A:
        if a == x:
            S = S + 1
    return S

def tinh_lap(A):
    B = []
    for a in A:
        B.append(lap(a,A))
    return B
# Chương trình chính
A = [2, 1, 1, 3, 5, 10, 2, 5, 2]
B = tinh_lap(A)
print(B)
```


LUYỆN TẬP

1. Thiết kế thuật toán cho nhiệm vụ 1 với ý tưởng khác như sau: Dãy A là một hoán vị của dãy các số từ 1 đến n khi và chỉ khi dãy A có độ dài n và mọi số i từ 1 đến n đều nằm trong A.
2. Trong Nhiệm vụ 2, nếu dãy A đã được sắp xếp theo thứ tự tăng dần thì có thể cải tiến thuật toán tốt hơn được không?

VẬN DỤNG

1. Cho dãy số A = A[0], A[1], ..., A[n – 1]. Thiết kế và viết chương trình kiểm tra trong dãy A có hai phần tử nào trùng nhau hay không. Cần đưa ra câu trả lời là "có" hay "không". Yêu cầu đưa ra quy trình thiết kế theo phương pháp làm mịn dần.
2. Xâu kí tự được gọi là đối xứng nếu thay đổi thứ tự ngược lại các kí tự của xâu thì vẫn nhận được dãy ban đầu. Ví dụ xâu "abcdcba" là đối xứng, còn xâu "1011" không là đối xứng.

   Thiết kế và viết chương trình kiểm tra một xâu kí tự cho trước có là đối xứng hay không. Yêu cầu đưa ra quy trình thiết kế theo phương pháp làm mịn dần.