Tên bài giảng

Phân tích thiết kế thuật toán lặp và đệ quy

Môn học: **Phân tích thiết kế thuật toán**

Chương: 2

Hệ: Đại học

Giảng viên: TS. Phạm Đình Phong

Email: phongpd@utc.edu.vn

Nội dung bài học

1. **Các yếu tố phân tích, thiết kế thuật toán**
2. **Phân tích thiết kế thuật toán tính tổng**
3. **Phân tích thiết kế thuật toán trên dãy**

Các yếu tố phân tích, thiết kế thuật toán

- Muốn tìm thuật toán để giải quyết bài toán thì
  - Phân tích thuật toán
  - Chọn cấu trúc dữ liệu phù hợp
  - Thiết kế thuật toán
  - Chứng minh tính đúng đắn của thuật toán
  - Chọn ngôn ngữ lập trình
  - Cài đặt thuật toán để tạo ra chương trình
  - Kiểm thử và gỡ lỗi logic để hoàn thiện chương trình

Các yếu tố phân tích, thiết kế thuật toán

- Phân tích thuật toán
  - Khi phân tích, thiết kế thuật toán ta phân tích, thiết kế các yếu tố:
    - Đặc tả bài toán
    - Cách thức tiến hành (ý tưởng giải quyết bài toán)
    - Tính đúng đắn của thuật toán
    - Truy hồi và đệ quy
    - Độ phức tạp của thuật toán

Các yếu tố phân tích, thiết kế thuật toán

- Đặc tả bài toán
  - Phải hiểu được bài toán yêu cầu làm gì
  - Xác định các điều kiện và miền ràng buộc của dữ liệu vào (tiền điều kiện) và dữ liệu ra (hậu điều kiện)
  - Hiểu được các ví dụ mẫu nếu có

Các yếu tố phân tích, thiết kế thuật toán

- Cách thức tiến hành
  - Đặc tả bài toán → những ý tưởng khác nhau để giải quyết bài toán → lựa chọn ý tưởng phù hợp nhất cho bài toán đặt ra → mô hình hóa bởi sơ đồ thuật giải hoặc giả mã, thậm chí mã thật cho thuật toán

Các yếu tố phân tích, thiết kế thuật toán

- Tính đúng đắn của thuật toán
  - Tính đúng đắn của thuật toán lặp được chứng minh thông qua việc xác định bất biến vòng lặp, đó là một mệnh đề logic đúng trước và sau mỗi bước lặp
    - Thiết lập bất biến trước khi vào vòng lặp;
    - Duy trì bất biến trong từng bước trong vòng lặp;
    - Đảm bảo bất biến đúng khi ra khỏi vòng lặp.

Chú ý: trong khi thực hiện vòng lặp đôi khi có thể có những tình huống thoát bất thường khỏi vòng lặp do tìm thấy kết quả hoặc có những bất thường không tìm được kết quả đối với những dữ liệu đầu vào đặc biệt nào đó.

Các yếu tố phân tích, thiết kế thuật toán

- Truy hồi và đệ quy
  - Khi xác định được bất biến vòng lặp ta xây dựng biểu thức toán học truy hồi và chương trình đệ quy tương ứng để cài đặt thuật toán

Các yếu tố phân tích, thiết kế thuật toán

- Độ phức tạp của thuật toán
  - Độ phức tạp thuật toán lặp và đệ quy sẽ cho chúng ta thấy được tốc độ tính toán và so sánh tốt xấu so với các thuật toán khác

Các yếu tố phân tích, thiết kế thuật toán

- Ví dụ: Nhập vào số nguyên không âm n và tính n!
  - Đặc tả bài toán
    - Tiền điều kiện: Dữ liệu vào là số nguyên không âm n
    - Hậu điều kiện: Dữ liệu ra $n! = 1 \times 2 \times ... \times n$ cũng là số nguyên dương

Các yếu tố phân tích, thiết kế thuật toán

- Ví dụ: Nhập vào số nguyên không âm n và tính n!
  - Cách thức tiến hành (ý tưởng thuật toán)
    - Đặt $S_n = n! = 1 \times 2 \times ... \times n$, dễ thấy

$$S_i = i! = 1 \times 2 \times ... \times (i-1) \times i = S_{i-1} \times i$$

và $S_0 = 0! = 1$

Như vậy, để tính n! ta dùng mảng S có n+1 phần tử với S[0] = 1 và sau đó tính lần lượt các giá trị:

S[1] = S[0] * 1; S[2] = S[1] * 2; ...; S[i] = S[i-1] * i; ... ;

S[n] = S[n-1] * n theo các bước sau đây:

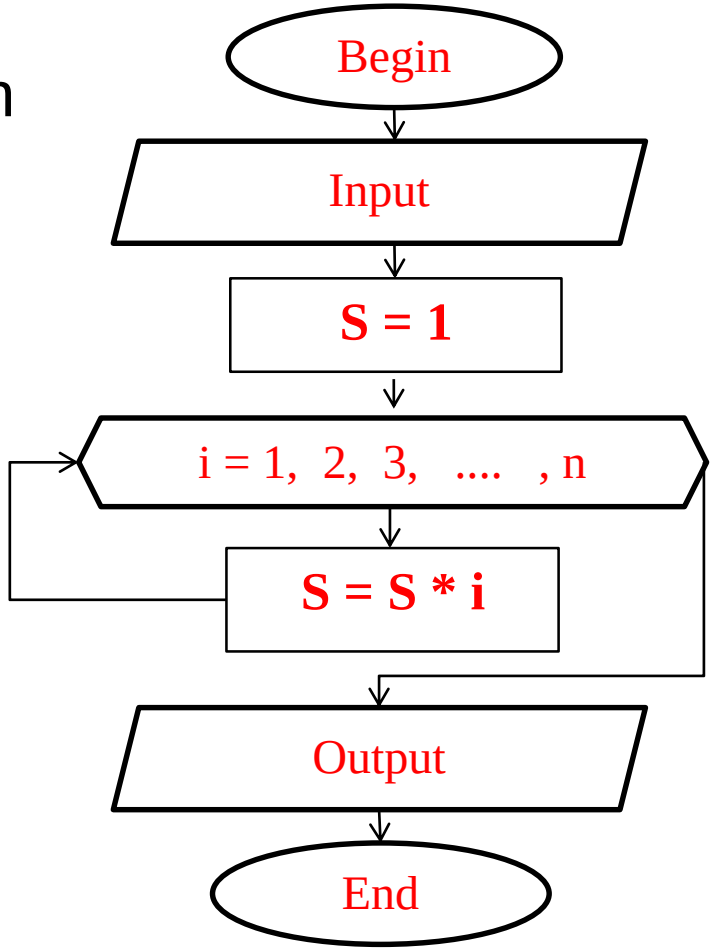
Các yếu tố phân tích, thiết kế thuật toán

- Ví dụ: Nhập vào số nguyên không âm n và tính n!
  - Cách thức tiến hành (ý tưởng thuật toán)
    - Bước 1. Nhập vào giá trị nguyên không âm n.
    - Bước 2: Khởi gán S[0] = 1.
    - Bước 3. Thực hiện vòng lặp i lần lượt từ 1 đến n với mỗi bước lặp ta tính S[i] = S[i-1] * i.
    - Bước 4. Xuất ra giá trị S[n] là kết quả cần tính.

Để tối ưu độ phức tạp không gian của thuật toán, thay vì dùng mảng S[n+1] thì ta dùng một biến S

Các yếu tố phân tích, thiết kế thuật toán

- Ví dụ: Nhập vào số nguyên không âm n và tính n!
  - Cách thức tiến hành



Các yếu tố phân tích, thiết kế thuật toán

- Ví dụ: Nhập vào số nguyên không âm n và tính n!
  - Cách thức tiến hành

```
#include<stdio.h>
int main() {
        int n;
        printf("Nhap vao n = ");
        scanf("%d",&n);
        long long S=1;
        for (int i=1; i<=n; i++)
        S*=i;
        printf("S = %lld",S);
}
```

Các yếu tố phân tích, thiết kế thuật toán

- Tính đúng đắn của thuật toán
  - Bất biến vòng lặp: Mỗi bước thứ i của vòng lặp đều có $S_i = i! = 1 \times 2 \times ... \times i$ với i = 1, …, n

    Có thể biểu diễn bằng mệnh đề logic toán:

    $(1 \le i) \wedge (i \le n) \wedge (S_i = i!)$

    hoặc dưới dạng logic Hoare:

    {S = 1} for (int i=1; i<=n; i++) S*=i {S = n!}

    trong đó, {S = 1} là điều kiện trước khi vào vòng lặp và {S = n!} là điều kiện sau khi kết thúc vòng lặp

Các yếu tố phân tích, thiết kế thuật toán

- Tính đúng đắn của thuật toán
  - Thiết lập bất biến vòng lặp:
    - Trước khi vào vòng lặp S = 1
    - Nếu S nhận giá trị khác thì khi kết thúc vòng lặp sẽ không thu được kết quả là n!

Các yếu tố phân tích, thiết kế thuật toán

- Tính đúng đắn của thuật toán
  - Duy trì bất biến vòng lặp:
    - Tại bước i - 1 thì S = (i-1)!
    - Muốn thực hiện xong bước thứ i ta có S = i! và để duy trì bất biến thì S = S * i

Các yếu tố phân tích, thiết kế thuật toán

- Tính đúng đắn của thuật toán
  - Thoát khỏi vòng lặp:
    - Không có tình huống nào thoát bất thường ra khỏi vòng lặp
    - Chỉ có thoát bình thường khỏi vòng lặp ta thu được sau n bước và S = n! → đảm bảo tính đúng đắn của thuật toán

Các yếu tố phân tích, thiết kế thuật toán

- Truy hồi và đệ quy
  - Theo phân tích bất biến vòng lặp ta có biểu thức truy hồi:

$$S_n = \begin{cases} 1 & Khi \;\; n = 0 \\ S_{n-1} * n & Khi \; n > 0 \end{cases}$$

```
long S(int n)
{
        if (n == 0) return 1;
        return n * S(n-1);
}
```

Các yếu tố phân tích, thiết kế thuật toán

- Độ phức tạp của thuật toán
  - Độ phức tạp của thuật toán lặp là O(n) do chỉ có một vòng lặp n bước
  - Độ phức tạp của thuật toán đệ quy cũng là O(n)

Các yếu tố phân tích, thiết kế thuật toán

- Ví dụ 2: Nhập vào số nguyên dương n > 1, kiểm tra n có là số nguyên tố không?

Phân tích thiết kế thuật toán tính tổng

- Ví dụ: Nhập số thực x và số nguyên dương n tính giá trị biểu thức

$$T = \sqrt{\sin x - \frac{\cos x}{1} + \frac{\cos^3 x}{3} - \frac{\cos^5 x}{5} + \frac{\cos^7 x}{7} - \ldots + (-1)^{n+1} \frac{\cos^{2n+1} x}{2n+1}}$$

Phân tích thiết kế thuật toán tính tổng

- Đặc tả bài toán
  - **Tiền điều kiện**: Dữ liệu vào là số thực x và số nguyên dương n.
  - **Hậu điều kiện**: Dữ liệu ra hoặc là tính được S là một số thực không âm hoặc là không tính được giá trị của biểu thức.

Phân tích thiết kế thuật toán tính tổng

- Cách thức tiến hành (ý tưởng thuật toán)
  - Ta có biểu thức cần tính

$$T = \sqrt{\sin x + \sum_{i=0}^{n} (-1)^{i+1} \frac{\cos^{2i+1} x}{2i+1}} = \sqrt{\sin x - \cos x \sum_{i=0}^{n} (-1)^{i} \frac{\cos^{2i} x}{2i+1}}$$

Đặt $t = -\cos^2 x$ và xét

$$S = \sum_{i=0}^{n} (-1)^{i} \frac{\cos^{2i} x}{2i+1} = \sum_{i=0}^{n} \frac{(-\cos^2 x)^{i}}{2i+1} = \sum_{i=0}^{n} \frac{t^{i}}{2i+1}$$

Khi đó, bài toán đưa về tìm giá trị của biểu thức S → giá trị của *T* là:

$$T = \sqrt{\sin x - \cos x \times S}$$

Phân tích thiết kế thuật toán tính tổng

- Các thức tiến hành (ý tưởng thuật toán)
  - Xét hai biểu thức

$$S_n = \sum_{i=0}^{n} \frac{t^i}{2i+1} \quad \text{và} \quad P_n = t^n$$ thực hiện lần lượt các

bước để tính giá trị của $S = S_n$

$$\begin{cases} P_i = t^i = P_{i-1} \times t \\ S_i = \frac{1}{1} + \frac{t^1}{3} + \frac{t^2}{5} + \ldots + \frac{t^{i-1}}{2(i-1)+1} + \frac{t^i}{2i+1} = S_{i-1} + \frac{P_i}{2i+1} \end{cases}$$

Phân tích thiết kế thuật toán tính tổng

- Các thức tiến hành (ý tưởng thuật toán)
  - Các bước thực hiện

    Bước 1: Nhập vào giá trị thực x và số nguyên dương n.

    Bước 2: Đặt $t = -\cos^2 x$.

    Bước 3: Thực hiện vòng lặp để tính P_n và S_n.

    Bước 3.0: Khởi tạo $\begin{cases} P_0 = t^0 = 1 \\ S_0 = \frac{1}{1} = 1 \end{cases}$

    Bước 3.1: Tính $\begin{cases} P_1 = t^1 = P_0 \times t \\ S_1 = \frac{1}{1} + \frac{t^1}{3} = S_0 + \frac{P_1}{3} \end{cases}$

Phân tích thiết kế thuật toán tính tổng

- Các thức tiến hành (ý tưởng thuật toán)
  - Các bước thực hiện

    …

    Bước 3.i: Tính

$$\begin{cases} P_i = t^i = P_{i-1} \times t \\ S_i = \frac{1}{1} + \frac{t^1}{3} + \frac{t^2}{5} + \ldots + \frac{t^{i-1}}{2(i-1)+1} + \frac{t^i}{2i+1} = S_{i-1} + \frac{P_i}{2i+1} \end{cases}$$

    Bước 3.n: Tính

$$\begin{cases} P_n = t^n = P_{n-1} \times t \\ S_n = \frac{1}{1} + \frac{t^1}{3} + \ldots + \frac{t^{n-1}}{2(n-1)+1} + \frac{t^n}{2n+1} = S_{n-1} + \frac{P_n}{2n+1} \end{cases}$$

Phân tích thiết kế thuật toán tính tổng

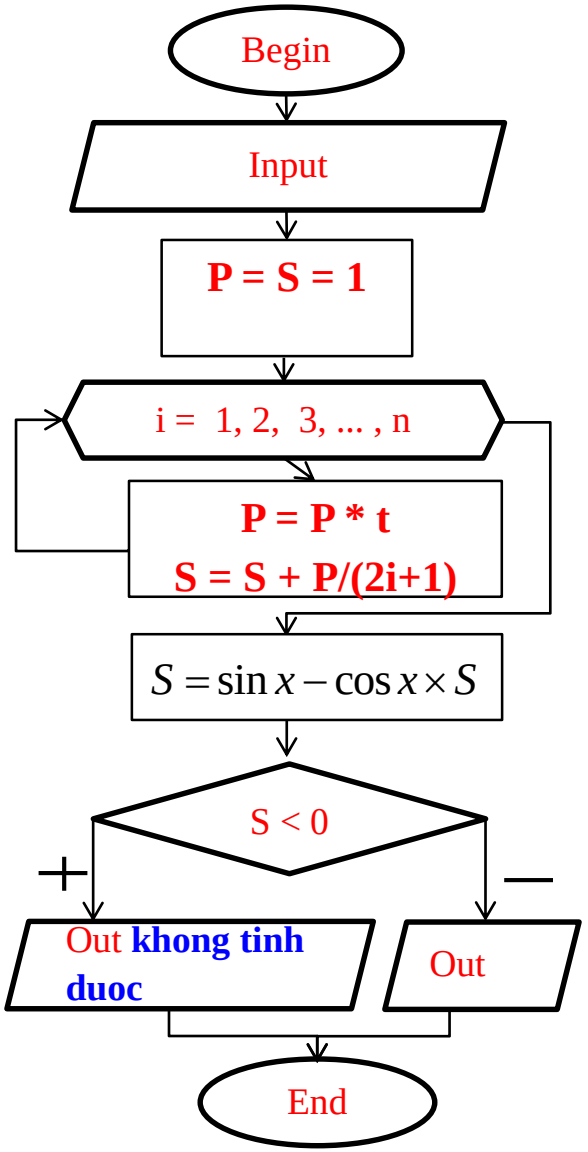
- Các thức tiến hành (ý tưởng thuật toán)
  - Các bước thực hiện

    Bước 4: Tính $S = \sin x - \cos x \times S_n$

    Bước 5: Kiểm tra nếu S < 0 thì không tính được, ngược lại xuất $\sqrt{S}$

Phân tích thiết kế thuật toán tính tổng

- Các thức tiến hành
  - Sơ đồ thuật toán



Phân tích thiết kế thuật toán tính tổng

- Các thức …
  - Chương trình

```
#include<stdio.h>
#include<math.h>
int main() {
        float x,S,P,t;
        int n,i;
        printf("Nhap x = ");
        scanf("%f",&x);
        printf("Nhap n = ");
        scanf("%d",&n);
        t=cos(x);
        t=-t*t;
        P=S=1;
        for (i=1; i<=n; i++) {
                P*=t;
                S+=P/(2*i+1);
        }
        S=sin(x)-cos(x)*S;
        if (S<0)printf("Khong tinh duoc T");
        else printf("Gia tri T = %f",sqrt(S));
}
```

Phân tích thiết kế thuật toán tính tổng

- Tính đúng đắn của thuật toán
  - Bất biến vòng lặp: Mỗi bước thứ i (1 ≤ i ≤ n) của vòng lặp đều có đại lượng bất biến B_i gồm hai thành phần P_i và S_i với công thức như sau:

$$B_i: \begin{cases} P_i = t^i \\ S_i = \frac{1}{1} + \frac{t^1}{3} + \frac{t^2}{5} + \ldots + \frac{t^{i-1}}{2(i-1)+1} + \frac{t^i}{2i+1} \end{cases}$$

biểu diễn bằng mệnh đề logic toán:

$$(1 \le i) \wedge (i \le n) \wedge B_i$$

Phân tích thiết kế thuật toán tính tổng

- Tính đúng đắn của thuật toán
  - Thiết lập bất biến vòng lặp: trước khi vào vòng lặp, khởi tạo bất biến $P_0 = t^0 = 1$ và $S_0 = \frac{1}{1} = 1$

    Nếu khởi tạo P_0 và S_0 khác 1 thì sẽ không đúng

Phân tích thiết kế thuật toán tính tổng

- Tính đúng đắn của thuật toán
  - Duy trì bất biến vòng lặp: khi ở bước thứ i – 1 thì bất biến là:

$$B_{i-1}: \begin{cases} P_{i-1} = t^{i-1} \\ S_i = \frac{1}{1} + \frac{t^1}{3} + \frac{t^2}{5} + \ldots + \frac{t^{i-1}}{2(i-1)+1} \end{cases}$$

Với mong muốn đạt được bất biến ở bước thứ i là:

$$B_i: \begin{cases} P_i = t^i \\ S_i = \frac{1}{1} + \frac{t^1}{3} + \frac{t^2}{5} + \ldots + \frac{t^{i-1}}{2(i-1)+1} + \frac{t^i}{2i+1} \end{cases}$$

ta phải duy trì bất biến bằng cách tính $P_i = P_{i-1} \times t$ và $S_i = S_{i-1} + P_i / (2i+1)$

Phân tích thiết kế thuật toán tính tổng

- Tính đúng đắn của thuật toán
  - Thoát khỏi vòng lặp:
    - Thuật toán không có thoát bất thường
    - Mọi phép tính để duy trì bất biến không gây ra lỗi toán học nào bởi phép chia không có khả năng chia cho 0
    - Thoát bình thường khỏi vòng lặp ta thu được bất biến sau n bước

$$B_n: \begin{cases} P_n = t^n \\ S_n = \frac{1}{1} + \frac{t^1}{3} + \ldots + \frac{t^{n-1}}{2(n-1)+1} + \frac{t^n}{2n+1} \end{cases}$$

Phân tích thiết kế thuật toán tính tổng

- Truy hồi và đệ quy
  - Đại lượng truy hồi B_n gồm hai thành phần P_n và S_n là bất biến vòng lặp được biểu diễn dưới dạng biểu thức truy hồi:

$$B_n: \begin{cases} \begin{cases} P_n = t^n = 1 \\ S_n = \dfrac{1}{1} = 1 \end{cases} & Khi\ n = 0 \\ \begin{cases} P_n = P_{n-1} \times t \\ S_n = S_{n-1} + P_n / (2n+1) \end{cases} & Khi\ n > 0 \end{cases}$$

Phân tích thiết kế thuật toán tính tổng

- Truy hồi và đệ quy
  - Chương trình:

Phân tích thiết kế thuật toán tính tổng

- Độ phức tạp tính toán
  - Với thuật toán lặp với vòng lặp không có thoát bất thường và duyệt tuyến tính gồm n bước nên độ phức tạp O(n)

Phân tích thiết kế thuật toán tính tổng

- Độ phức tạp tính toán
  - Với thuật toán đệ quy với độ phức tạp của thuật toán là T_n

$$T(n) = \begin{cases} c_0 & Khi\ n = 0 \\ T(n-1) + c & Khi\ n > 0 \end{cases}$$

Khai triển theo công thức này ta có

$$\begin{aligned} T(n) &= T(n-1) + c \\ &= T(n-2) + c + c \qquad Thay\ \ T(n-1) = T(n-2) + c \\ &= T(n-2) + 2c \\ &= T(n-3) + 3c \\ &= \ldots \\ &= T(n-n) + nc = T(0) + nc = c_0 + nc \end{aligned}$$

Độ phức tạp của thuật toán đệ quy là O(n)

Phân tích thiết kế thuật toán trên dãy

Tìm phần tử âm lớn nhất trong dãy

- Nhập vào một dãy số nguyên gồm n ($1 \le n \le 10^5$) phần tử $a_1, a_2, \ldots, a_n$ ($-10^6 \le a_i \le 10^6$), hãy tìm giá trị âm lớn nhất trong dãy nếu có

Tìm phần tử âm lớn nhất trong dãy

- Đặc tả bài toán
  - **Tiền điều kiện**: Dữ liệu vào là số nguyên n ($1 \le n \le 10^5$) và dãy số nguyên $a_1, a_2, \ldots, a_n$ ($-10^6 \le a_i \le 10^6$)
  - **Hậu điều kiện**: Dữ liệu ra hoặc dãy không có phần tử âm hoặc là có, trong trường hợp có chỉ ra giá trị âm lớn nhất

Tìm phần tử âm lớn nhất trong dãy

- Cách thức tiến hành
  - Đầu ra cần hai thông tin: dãy có phần tử âm hay không và giá trị âm lớn nhất trong dãy nếu có
  - Để lưu hai thông tin này, ta sử dụng hai đại lượng:
    - Q_i có kiểu bool nhận giá trị true nếu trong dãy a_1, a_2, …, a_i có xuất hiện phần tử âm, ngược lại nhận giá trị false
    - M_i có kiểu nguyên lưu giá trị âm lớn nhất trong dãy a_1, a_2, …, a_i

Tìm phần tử âm lớn nhất trong dãy

- Cách thức tiến hành
  - Các bước

    Bước 1: Nhập số nguyên dương n ($1 \le n \le 10^5$)

    Bước 2: Nhập dãy số nguyên n phần tử $a_1, a_2, \ldots, a_n$ ($-10^6 \le a_i \le 10^6$)

    Bước 3: Thực hiện vòng lặp lần lượt để tính Q_n và M_n

    Bước 3.0: Khởi tạo Q_n = false, $M_n = -\infty$.

    Bước 3.1: Nếu $a_1 \ge 0$, $M_1 = M_0$ và $Q_1 = Q_0$.
    Ngược lại, Q_1 = true và $M_1 = a_1$

    …

Tìm phần tử âm lớn nhất trong dãy

- Cách thức tiến hành
  - Các bước

    …

    Bước 3.i: Nếu $a_i \geq 0$, $M_i = M_{i-1}$ và $Q_i = Q_{i-1}$.

    Ngược lại, $a_i < 0$

    - Tính M_i: nếu Q_{i-1} = true, chứng tỏ $a_1, a_2, \ldots, a_{i-1}$ đã tồn tại phần tử có giá trị âm và M_{i-1} lưu giá trị âm lớn nhất trong dãy đó nên $M_i = \max(M_{i-1}, a_i)$.

    Ngược lại, nếu Q_{i-1} = false, $M_i = a_i$.

    - Tính Q_i: vì $a_i < 0$ nên trong dãy $a_1, a_2, \ldots, a_i$ có ít nhất một phần tử âm nên Q_i = true

    …

Tìm phần tử âm lớn nhất trong dãy

- Cách thức tiến hành
  - Các bước

    …

    Bước 3.n: Nếu $a_n \geq 0$, $M_n = M_{n-1}$ và $Q_n = Q_{n-1}$.

    Ngược lại, $a_n < 0$

    - Tính M_n: nếu Q_{n-1} = true, chứng tỏ $a_1, a_2, \ldots, a_{n-1}$ đã tồn tại phần tử có giá trị âm và M_{n-1} lưu giá trị âm lớn nhất trong dãy đó nên $M_n = \max(M_{n-1}, a_n)$.

    Ngược lại, nếu Q_{n-1} = false, $M_n = a_n$.

    - Tính Q_n: vì $a_n < 0$ nên trong dãy $a_1, a_2, \ldots, a_n$ có ít nhất một phần tử âm nên Q_n = true

    Bước 4: Kết thúc vòng lặp nếu Q_n = true thì giá trị âm lớn nhất là M_n, ngược lại không có phần tử âm.

Tìm phần tử âm lớn nhất trong dãy

- Cách thức tiến hành
  - Lưu đồ thuật toán và chương trình: thay các Q_i và M_i bằng các biến Q và M.

Tìm phần tử âm lớn nhất trong dãy

- Tính đúng đắn của thuật toán
  - Bất biến vòng lặp: Mỗi bước thứ i, bất biến $M_i = 0$ nếu dãy $a_k \geq 0$ với $1 \leq k \leq i$. Ngược lại M_i là chính là phần tử âm lớn nhất trong dãy đó.
  - Thiết lập bất biến vòng lặp: Bắt đầu vòng lặp, khởi tạo giá trị $M_0 = -\infty$ hoặc $M_0 = 0$.

Tìm phần tử âm lớn nhất trong dãy

- Tính đúng đắn của thuật toán
  - Duy trì bất biến vòng lặp: Khi ở bước thứ i -1,

$$M_{i-1} = \begin{cases} 0 & Khi\ \ a_j \geq 0\ \ \forall j \in \{1,...i-1\} \\ \underset{1 \leq j \leq i-1}{Max}(a_j < 0) & Khi\ a_j < 0\ \ \exists j \in \{1,...i-1\} \end{cases}$$

với mong muốn đạt được bất biến ở bước i là:

$$M_i = \begin{cases} 0 & Khi\ \ a_j \geq 0\ \ \forall j \in \{1,...i\} \\ \underset{1 \leq j \leq i}{Max}(a_j < 0) & Khi\ a_j < 0\ \ \exists j \in \{1,...i\} \end{cases}$$

thì phải duy trì bất biến, nếu $a_i \geq 0$ thì $M_i = M_{i-1}$. Ngược lại ($a_i < 0$), nếu $M_{i-1} = 0$ thì a_i là phần tử âm đầu tiên thì $M_i = a_i$, còn không thì $M_i = \max(M_{i-1}, a_i)$.

Tìm phần tử âm lớn nhất trong dãy

- Tính đúng đắn của thuật toán
  - Thoát khỏi vòng lặp: Không có thoát khỏi vòng lặp bất thường. Khi vòng lặp kết thúc ta thu được M_n. Nếu Q_n = false thì không có phần tử nào âm ngược lại thì M_n là phần tử âm lớn nhất

Tìm phần tử âm lớn nhất trong dãy

- Truy hồi và đệ quy
  - M_n có thể được biểu diễn dưới dạng biểu thức truy hồi

$$M_n = \begin{cases} 0 & Khi\ n=0 \\ \begin{cases} M_{n-1} & Khi\ n>0,\ a_n \geq 0 \\ a_n & Khi\ n>0,\ a_n<0,\ M_{n-1}=0 \\ Max(M_{n-1}, a_n) & Khi\ n>0,\ a_n<0,\ M_{n-1}\neq 0 \end{cases} \end{cases}$$

Tìm phần tử âm lớn nhất trong dãy

- Độ phức tạp của thuật toán
  - Thuật toán lặp: O(n)
  - Thuật toán đệ quy: ta có phương trình đệ quy

$$T(n) = \begin{cases} c_0 & Khi\ n = 0 \\ T(n-1) + c & Khi\ n > 0 \end{cases}$$

triển khai

$$\begin{aligned} T(n) &= T(n-1) + c \\ &= T(n-2) + c + c \qquad Thay\ \ T(n-1) = T(n-2) + c \\ &= T(n-2) + 2c \\ &\ldots \\ &= T(n-n) + nc = T(0) + nc = c_0 + nc \end{aligned}$$

→ độ phức tạp: O(n)

TỔNG KẾT