

# COPYRIGHT NOTICE

## THÔNG BÁO BẢN QUYỀN

© 2024 Duc A. Hoang (Hoàng Anh Đức)

### COPYRIGHT (English):

This document is licensed under Creative Commons Attribution-ShareAlike 4.0 International (CC-BY-SA 4.0). You are free to share and adapt this material with appropriate attribution and under the same license.

This document is not up to date and may contain several errors or outdated information.

Last revision date: 2024-10-09

### BẢN QUYỀN (Tiếng Việt):

Tài liệu này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution-ShareAlike 4.0 (CC-BY-SA 4.0). Bạn được tự do chia sẻ và chỉnh sửa tài liệu này với điều kiện ghi nguồn phù hợp và sử dụng cùng loại giấy phép.

Tài liệu này không được cập nhật và có thể chứa nhiều lỗi hoặc thông tin cũ.

Ngày sửa đổi cuối cùng: 2024-10-09



Creative Commons Attribution-ShareAlike 4.0 International

# VNU-HUS MAT3500: Toán rời rạc

## Thuật toán I

Mô tả, chứng minh, đánh giá thuật toán; Tìm kiếm và sắp xếp

Hoàng Anh Đức

Bộ môn Tin học, Khoa Toán-Cơ-Tin học  
Đại học KHTN, ĐHQG Hà Nội  
[hoanganhduc@hus.edu.vn](mailto:hoanganhduc@hus.edu.vn)



# Nội dung



Thuật toán I

Hoàng Anh Đức

## Định nghĩa và một số khái niệm

### Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

### Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

### Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

### Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp



# Định nghĩa và một số khái niệm

Thuật toán I

Hoàng Anh Đức

## 2 Định nghĩa và một số khái niệm

### Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

### Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

### Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

### Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

- Một **thuật toán (algorithm)** là một tập hữu hạn các hướng dẫn cụ thể để thực hiện một nhiệm vụ nào đó
    - cộng hai số tự nhiên biểu diễn dưới dạng số thập phân
    - đăng ký môn học trực tuyến
    - đi từ nhà đến trường
  - Một **chương trình máy tính (computer program)** là
    - một mô tả của thuật toán nào đó
    - sử dụng một ngôn ngữ đủ chuẩn xác để máy tính có thể hiểu
    - cùng với các phép toán mà máy tính đã biết cách thực hiện
- Ta nói rằng thuật toán được **cài đặt (implement)** cụ thể bằng chương trình máy tính
- Khi mở một phần mềm trong máy tính, ta nói rằng **chương trình hoặc thuật toán của nó được chạy hoặc được thực hiện bởi máy tính**
  - Khi có mô tả của một thuật toán, bạn cũng **có thể thực hiện từng bước của thuật toán với giấy và bút**



# Định nghĩa và một số khái niệm

Một số tính chất của một thuật toán

**Đầu vào (Input)** Một thuật toán có các giá trị đầu vào từ một tập đã được xác định trước

**Đầu ra (Output)** Từ mỗi một tập các giá trị đầu vào, một thuật toán sinh ra các giá trị đầu ra. Các giá trị này chính là lời giải cho bài toán

**Tính xác định (Definiteness)** Các bước của một thuật toán cần phải được xác định một cách chính xác

**Tính đúng đắn (Correctness)** Với mỗi tập giá trị đầu vào, một thuật toán cần cho ra kết quả đầu ra đúng

**Tính hữu hạn (Finiteness)** Với mỗi tập giá trị đầu vào, một thuật toán cần cho ra các giá trị đầu ra mong muốn sau một số hữu hạn (có thể là rất lớn) các bước

**Tính hiệu quả (Effectiveness)** Mỗi bước của thuật toán cần được thực hiện một cách chính xác và trong thời gian hữu hạn

**Tính tổng quát (Generality)** Thuật toán phải áp dụng được cho mọi bài toán mong muốn, chứ không phải chỉ với một tập các giá trị đầu vào đặc biệt

Thuật toán I

Hoàng Anh Đức

3 Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp



# Định nghĩa và một số khái niệm

Thuật toán I

Hoàng Anh Đức

## 4 Định nghĩa và một số khái niệm

### Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

### Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

### Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

### Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

- Một *thuật toán* cũng có thể được *mô tả bằng một ngôn ngữ máy tính* (C, Python, Java, v.v...). Tuy nhiên, những mô tả này cần tuân theo các chỉ dẫn cụ thể trong ngôn ngữ máy tính tương ứng. Điều này dẫn đến việc các mô tả theo phương pháp này thường phức tạp và khó hiểu
- Thay vì dùng một ngôn ngữ máy tính cụ thể để mô tả thuật toán, ta sử dụng *ngôn ngữ thông thường*, *giả mã (pseudocode)*, hoặc *sơ đồ khối (flowchart)*
- Một mô tả đầy đủ của một thuật toán bao gồm ba phần
  - (1) *Thuật toán (algorithm)*
    - Mô tả một cách rõ ràng và chính xác nhất có thể
    - Thường kèm theo mô tả ngắn gọn về ý tưởng của thuật toán
  - (2) Một chứng minh về *tính đúng đắn (correctness)* của thuật toán
    - Với mọi tập đầu vào, thuật toán cần cho kết quả đầu ra đúng
  - (3) Một phân tích về *thời gian chạy (running time)* của thuật toán

# Mô tả thuật toán

Một số quy tắc viết giả mã



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

5

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

- **Giả mã (pseudocode)** là một dạng hỗn hợp giữa ngôn ngữ thông thường và ngôn ngữ lập trình
- Một **biến (variable)** được sử dụng để biểu diễn vị trí trong bộ nhớ máy tính để lưu trữ một giá trị. Khi ta nói đến một biến  $X$  nào đó, trên thực tế, chúng ta muốn sử dụng giá trị lưu tại một vị trí nào đó trong bộ nhớ ứng với  $X$
- Một **phép gán (assignment)** thường có dạng  $variable := expression$ , trong đó biểu thức  $expression$  ở vế phải sẽ được tính toán và kết quả tính toán được lưu trữ ở vị trí trong bộ nhớ tương ứng với biến  $variable$
- Trong **cấu trúc điều kiện (conditional statement) if condition then  $S_1$  else  $S_2$** , biểu thức  $condition$  được tính toán và sẽ cho ra giá trị cuối cùng là True (đúng) hoặc False (sai). Nếu True thì đoạn mã  $S_1$  sẽ được thực hiện, còn ngược lại thì  $S_2$  sẽ được thực hiện. Sau khi  $S_1$  hoặc  $S_2$  được thực hiện, các lệnh ngay tiếp sau cấu trúc điều kiện sẽ được thực hiện

# Mô tả thuật toán

Một số quy tắc viết giả mã



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

6

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

- Trong **vòng lặp for (for loop)** **for**  $i = 1$  **to**  $n$  **do**  $S$ , giá trị ban đầu của biến  $i$  được gán bằng 1. Nếu giá trị của  $i$  nhỏ hơn hoặc bằng  $n$ , đoạn mã  $S$  sẽ được thực hiện. Đoạn mã  $S$  có thể sử dụng biến  $i$  hoặc không. Sau mỗi lần thực hiện  $S$  (**lần lặp (iteration)**), biến  $i$  được tăng thêm 1 và được kiểm tra xem giá trị sau khi tăng của  $i$  có phải vẫn nhỏ hơn hoặc bằng  $n$  hay không. Nếu kết quả là True thì vòng lặp được thực hiện thêm một lần nữa với giá trị mới của  $i$ . Ngược lại, các lệnh tiếp theo ngay sau vòng **for** được thực hiện
- Trong **vòng lặp while (while loop)** **while**  $condition$  **do**  $S$ , đoạn mã  $S$  được thực hiện bất kể khi nào giá trị của biểu thức  $condition$  còn đúng (True). Nếu  $condition$  sai ngay khi bắt đầu vòng lặp **while** thì  $S$  không bao giờ được thực hiện
- Mọi thứ nằm sau các dấu  $//$  hoặc nằm giữa  $/*$  và  $*/$  là các nhận xét hoặc chú thích mà chương trình sẽ bỏ qua





## Ví dụ 1 (Mô tả thuật toán bằng ngôn ngữ thông thường)

### ■ Bài toán:

■ **Input:**  $a_1, a_2, \dots, a_n$ : dãy số nguyên

■ **Output:** Giá trị của phần tử lớn nhất trong dãy

### ■ Tìm giá trị của phần tử lớn nhất:

- (1) Gán giá trị của một biến tạm thời  $v$  (phần tử lớn nhất đến thời điểm hiện tại) bằng  $a_1$
- (2) Xét phần tử ngay tiếp theo trong dãy
- (3) Nếu phần tử đó lớn hơn  $v$  thì gán giá trị của  $v$  bằng giá trị của phần tử
- (4) Lặp lại (2) và (3) cho đến khi không còn phần tử nào để xét
- (5) Trả lại giá trị của  $v$

# Mô tả thuật toán

Ví dụ



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

8

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Ví dụ 2 (Thực hiện thuật toán)

- **Input:** Dãy  $a_1 = 7, a_2 = 12, a_3 = 5, a_4 = 16, a_5 = 9$
- **Output:** Giá trị của phần tử lớn nhất trong dãy

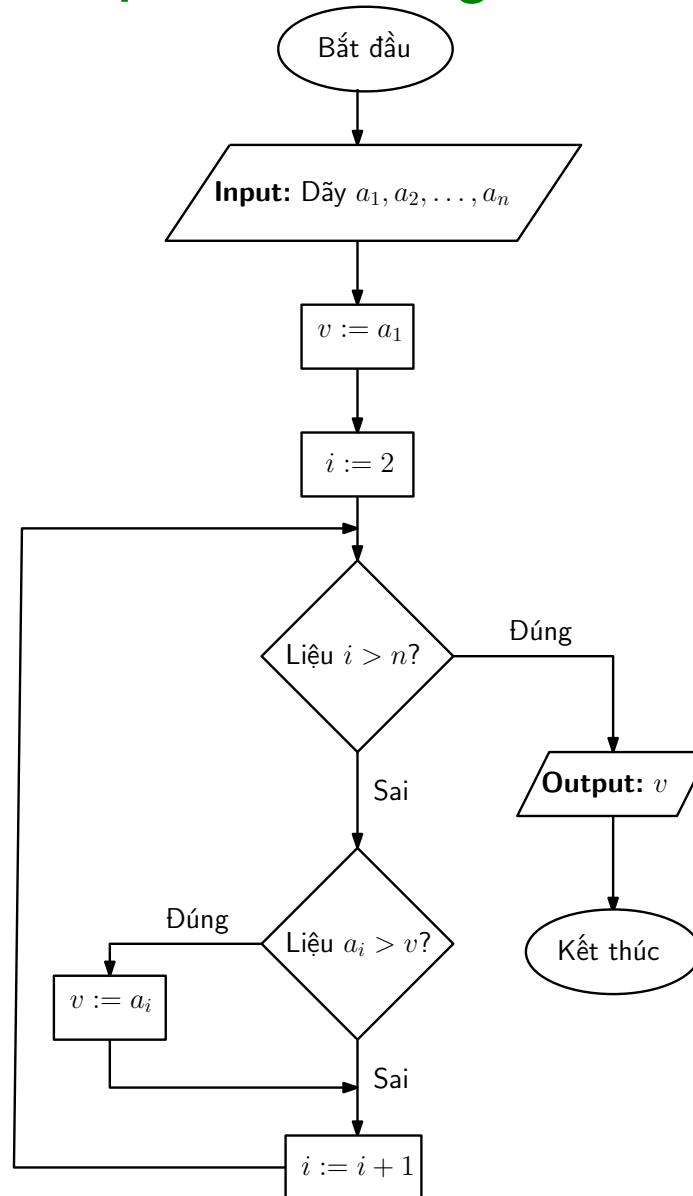
$a_1$	$a_2$	$a_3$	$a_4$	$a_5$
<div><div>7</div></div>	<div><div>12</div></div>	<div><div>5</div></div>	<div><div>16</div></div>	<div><div>9</div></div>
	$i = 2$	$i = 3$	$i = 4$	$i = 5$
$v = 7$	$v = 12$	$v = 12$	$v = 16$	$v = 16$

# Mô tả thuật toán

Ví dụ



## Ví dụ 3 (Mô tả thuật toán bằng sơ đồ khối)



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

9

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Ví dụ 4 (Mô tả thuật toán bằng giả mã)

### Thuật toán 1: Tìm giá trị của phần tử lớn nhất

**Input:**  $a_1, a_2, \dots, a_n$ : dãy số nguyên

**Output:** Giá trị của phần tử lớn nhất trong dãy

```
1   $v := a_1$                                 // phần tử lớn nhất đến hiện tại
2  for  $i := 2$  to  $n$  do                        // lần lượt xét  $a_2, \dots, a_n$ 
3      if  $a_i > v$  then //  $a_i >$  phần tử lớn nhất hiện tại?
4           $v := a_i$                                 // bây giờ  $v$  lớn nhất trong
               $a_1, \dots, a_i$ 
5  return  $v$ 
```

# Chứng minh thuật toán

Bất biến vòng lặp



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

11

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

- Có rất nhiều phương pháp khác nhau để *chứng minh tính đúng đắn của một thuật toán*
- Một trong số đó là sử dụng *bất biến vòng lặp (loop invariant)*—một phương pháp được xây dựng dựa trên phương pháp quy nạp toán học
  - *Vòng lặp (loop): for, while, v.v...*
  - Một *bất biến vòng lặp* là *một phát biểu luôn đúng trước và sau mỗi lần lặp (iteration) của một vòng lặp (loop)*

# Chứng minh thuật toán

## Bất biến vòng lặp



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

12 Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

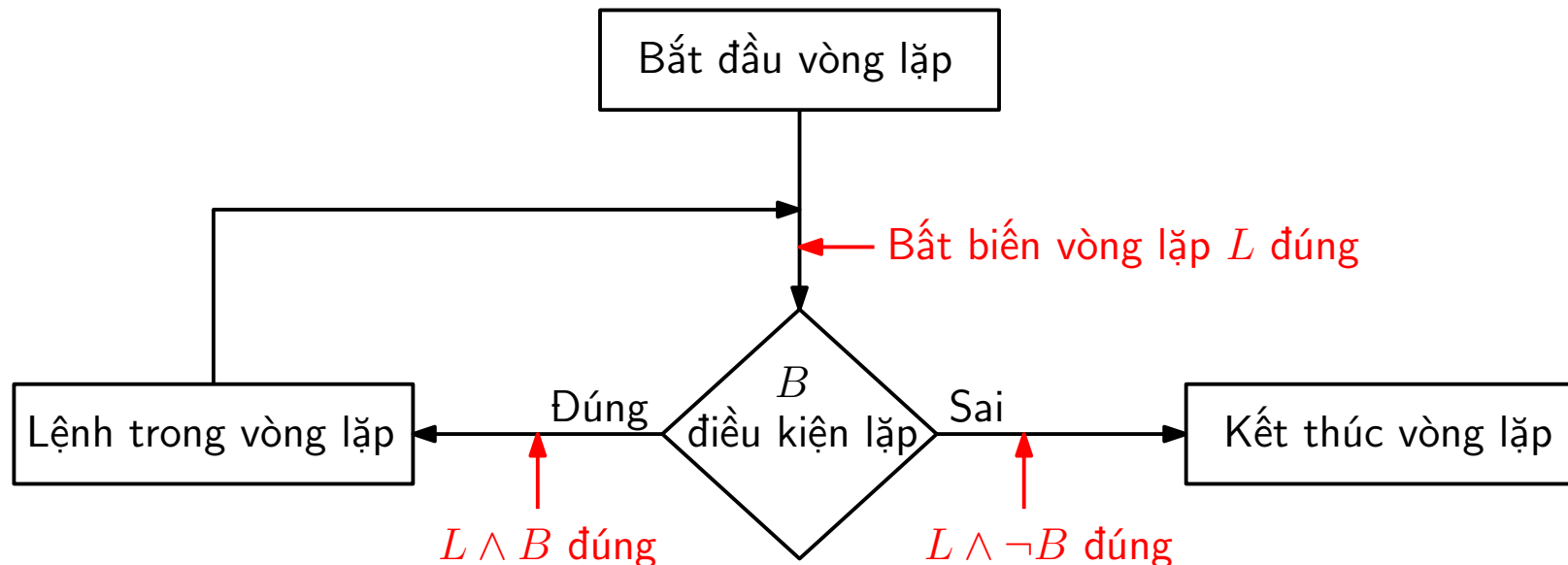
Một số thuật toán sắp xếp

Ta cần chỉ ra ba điều về một bất biến vòng lặp  $L$

**Khởi động (Initialization)**  $L$  đúng trước lần lặp đầu tiên của vòng lặp

**Duy trì (Maintenance)** Nếu  $L$  đúng trước một lần lặp của vòng lặp thì nó cũng đúng trước lần lặp tiếp theo

**Dừng (Termination)** Khi vòng lặp dừng, bất biến vòng lặp cho ta một tính chất hữu ích để chứng minh thuật toán đúng



Hình: Bất biến vòng lặp

# Chứng minh thuật toán

## Bất biến vòng lặp



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

### Ví dụ 5

Một bất biến vòng lặp trong Thuật toán 1 (vòng **for** ở Dòng 2–4) tìm giá trị lớn nhất trong dãy số nguyên  $a_1, \dots, a_n$

$$L := \text{Ở trước lần lặp với biến } i, v = \max\{a_1, a_2, \dots, a_{i-1}\}$$

Gọi  $v_i$  là giá trị của  $v$  trước lần lặp với biến  $i$

- **Khởi động** ( $i = 2$ ): Ta cần chỉ ra rằng trước vòng **for**,  $v_2 = \max\{a_1\} = a_1$ , và điều này hiển nhiên đúng do Thuật toán 1 gán  $v$  bằng  $a_1$  ở Dòng 1
- **Duy trì**: Giả sử  $L$  đúng ở trước lần lặp với  $i = k$  nào đó, nghĩa là  $v_k = \max\{a_1, \dots, a_{k-1}\}$ . Ta chứng minh  $L$  đúng ở trước lần lặp với  $i = k + 1$ , nghĩa là  $v_{k+1} = \max\{a_1, \dots, a_{k-1}, a_k\}$ . Ta xét các trường hợp dựa trên điều kiện ở Dòng 3
  - Nếu  $a_k > v = v_k$  sai, giá trị của  $v$  không thay đổi, và do đó  $v_{k+1} = v_k$ . Ta có  $\max\{a_1, \dots, a_{k-1}, a_k\} = \max\{v_k, a_k\} = v_k$ . Suy ra  $v_{k+1} = \max\{a_1, \dots, a_{k-1}, a_k\}$
  - Nếu  $a_k > v = v_k$  đúng, giá trị của  $v$  được gán bằng  $a_k$ , và do đó  $v_{k+1} = a_k$ . Ta cũng có  $\max\{a_1, \dots, a_{k-1}, a_k\} = \max\{v_k, a_k\} = a_k$ . Suy ra  $v_{k+1} = \max\{a_1, \dots, a_{k-1}, a_k\}$
- **Dừng**: Sau khi kết thúc lần lặp  $i = n$  (hoặc, trước khi bắt đầu lần lặp  $i = n + 1$  mà sẽ không bao giờ được thực hiện),  $v = \max\{a_1, \dots, a_n\}$  và do đó là giá trị lớn nhất của các phần tử trong dãy đầu vào

13

62

# Chứng minh thuật toán

Bất biến vòng lặp



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

14

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Bài tập 1

Một thuật toán tính  $x^n$  với  $x \in \mathbb{R}^+$  và  $n \in \mathbb{N}$  được mô tả như sau

---

**Thuật toán 2:** Tính  $x^n$ .

---

**Input:**  $x$ : số thực dương,  $n$ : số tự nhiên

**Output:** Giá trị của  $x^n$

```
1  answer := 1
2  m := n
3  while m > 0 do
4      answer := answer × x
5      m := m - 1
6  return answer
```

---

Hãy chứng minh phát biểu  $I$  sau là một bất biến vòng lặp cho vòng **while**

Ở trước lần lặp với biến  $m$ ,  $answer = x^{n-m}$



# Độ phức tạp tính toán

## Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

15

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

- Việc *phân tích (analysis) một thuật toán* yêu cầu cung cấp một xấp xỉ về *thời gian (time)* và *bộ nhớ (space)* cần thiết để thực hiện một thuật toán
- *Độ phức tạp (complexity)* của một thuật toán là lượng thời gian và bộ nhớ cần để thực hiện một thuật toán
  - Thường được thể hiện thông qua các hàm của *kích thước của đầu vào (input size)*
- Để đánh giá và so sánh độ phức tạp của các thuật toán khác nhau, ta giới thiệu *ký hiệu O-lớn (big-O notation)* và mô tả cách xấp xỉ độ tăng của các hàm thông qua ký hiệu này và từ đó xấp xỉ độ phức tạp của các thuật toán
- Với các hàm  $f : \mathbb{R} \rightarrow \mathbb{R}$  hoặc  $f : \mathbb{N} \rightarrow \mathbb{R}$ , trong nhiều trường hợp ta cần tìm hiểu xem chúng tăng nhanh đến mức nào
  - So sánh các hàm: Nếu  $f$  *tăng nhanh hơn*  $g$  thì  $f(x) \geq g(x)$  với “giá trị  $x$  đủ lớn”
  - So sánh tính hiệu quả của các thuật toán khác nhau cùng giải quyết một bài toán

# Độ phức tạp tính toán

## Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

16

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

### Ký hiệu $O$ -lớn

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Ta nói rằng  $f$  là  $O(g)$  (đọc là “ $f$  là  $O$ -lớn của  $g$ ” hoặc “ $f$  thuộc lớp  $O(g)$ ”) nếu tồn tại các hằng số  $C$  và  $k$  sao cho  $|f(x)| \leq C|g(x)|$  với mọi  $x > k$

### Ký hiệu $O$ -lớn

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Giả sử  $g(x) \neq 0$  với  $x \in \mathbb{R}$  đủ lớn. Ta nói rằng  $f$  là  $O(g)$  (đọc là “ $f$  là  $O$ -lớn của  $g$ ” hoặc “ $f$  thuộc lớp  $O(g)$ ”) nếu  $\lim_{x \rightarrow \infty} \frac{|f(x)|}{|g(x)|}$  là hữu hạn

- $f$  là  $O(g)$  nếu từ sau điểm  $k$  nào đó, giá trị của hàm  $f$  không vượt quá giá trị của một hằng số nhân với giá trị của hàm  $g$ . Ta cũng nói “ $f$  bị chặn trên bởi  $g$ ”
- Các hằng số  $C$  và  $k$  được gọi là các **bằng chứng (witness)** cho mối liên hệ giữa  $f$  và  $g$ . **Để xác định  $f$  là  $O(g)$ , chỉ cần một cặp bằng chứng là đủ**

# Độ phức tạp tính toán

## Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

17

Độ tăng của các hàm

Định nghĩa và khái niệm

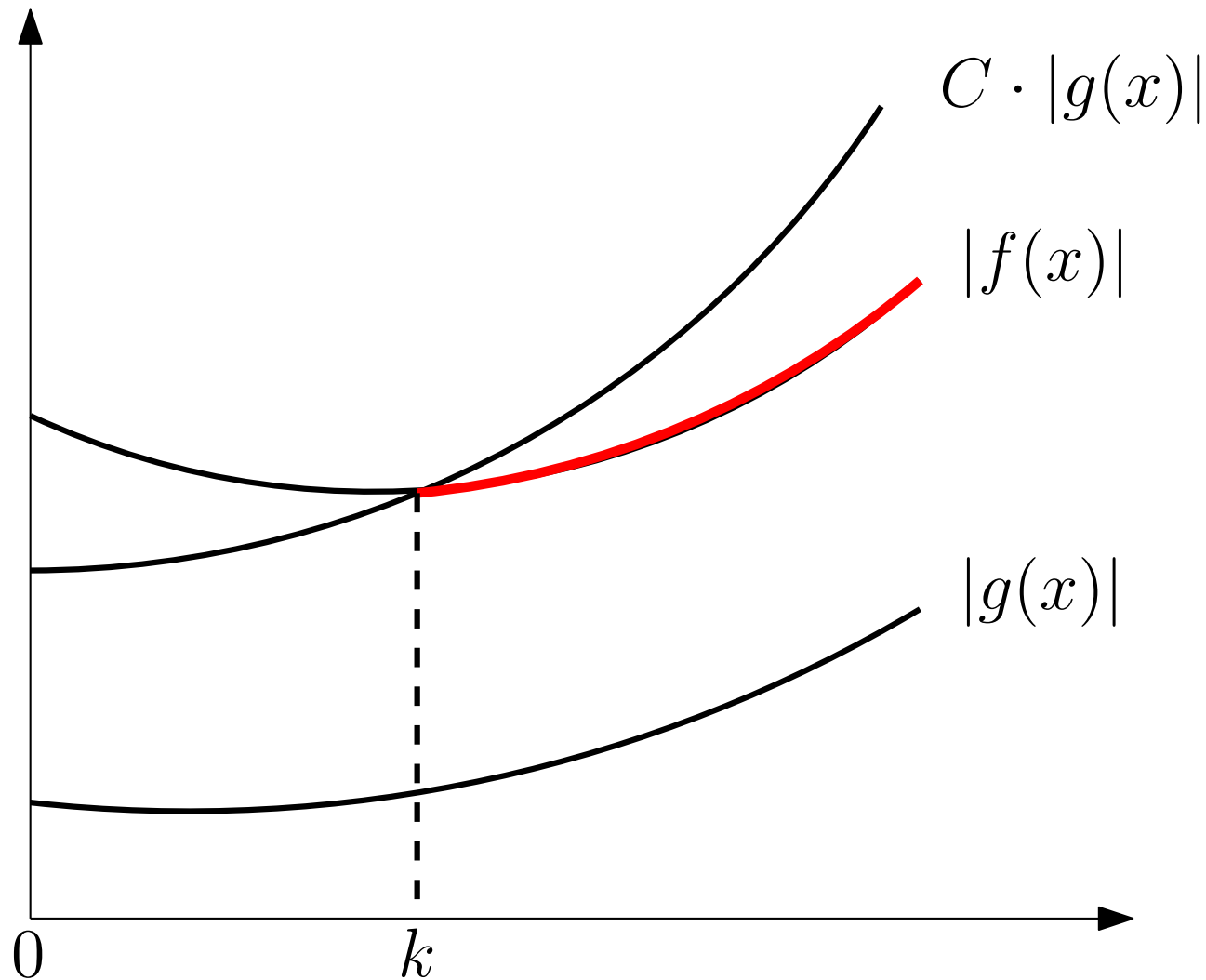
Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp



$f$  là  $O(g)$

# Độ phức tạp tính toán

## Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

18 Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Ví dụ 6

Ta chứng minh hàm  $f : \mathbb{R} \rightarrow \mathbb{R}$  cho bởi  $f(x) = x^2 + 2x + 1$  là  $O(g)$  với  $g(x) = x^2$  (Ta cũng viết  $x^2 + 2x + 1$  là  $O(x^2)$ )

### Cách 1:

- Chú ý rằng khi  $x > 1$ , ta có  $x < x^2$  và  $1 < x^2$
- Do đó với mọi  $x > 1$ , ta có

$$|f(x)| = |x^2 + 2x + 1| \leq |x^2 + 2x^2 + x^2| = 4|x^2|$$

- Ta chọn  $C = 4$  và  $k = 1$

**Cách 2:** Do  $\lim_{x \rightarrow \infty} \frac{|f(x)|}{|g(x)|} = \lim_{x \rightarrow \infty} \frac{|x^2 + 2x + 1|}{|x^2|} = 1$ , ta có  
 $f = O(g)$

# Độ phức tạp tính toán

## Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

19 Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

### Nhắc lại: Quy tắc L'Hospital (L'Hospital's rule)

Nếu  $\lim_{n \rightarrow \infty} f(n) = \infty$  và  $\lim_{n \rightarrow \infty} g(n) = \infty$ , thì  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)}$ , trong đó  $f'(n)$  và  $g'(n)$  lần lượt là đạo hàm của  $f(n)$  và  $g(n)$

# Độ phức tạp tính toán

## Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

20

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Ví dụ 7

Ta chứng minh  $x^2$  là  $O(2^x)$  bằng cách sử dụng định nghĩa  $O$ -lớn theo giới hạn. Ta có

$$\begin{aligned}\lim_{x \rightarrow \infty} \frac{|x^2|}{|2^x|} &= \lim_{x \rightarrow \infty} \frac{x^2}{2^x} \\ &= \lim_{x \rightarrow \infty} \frac{2x}{2^x \cdot \ln 2} \\ &= \frac{2}{\ln 2} \lim_{x \rightarrow \infty} \frac{x}{2^x} \\ &= \frac{2}{\ln 2} \lim_{x \rightarrow \infty} \frac{1}{2^x \cdot \ln 2} \\ &= 0.\end{aligned}$$

Quy tắc L'Hospital

Quy tắc L'Hospital

Do đó,  $x^2 = O(2^x)$

# Độ phức tạp tính toán

Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

21

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Chú ý

Nếu không đề cập gì thêm thì  $\log(n) = \log_2(n)$

## Bài tập 2

*Chứng minh*

- (a)  $7x$  là  $O(x^3)$
- (b)  $x^3$  không là  $O(x^2)$
- (c)  $1 + 2 + \dots + n$  là  $O(n^2)$
- (d)  $n! = 1 \times 2 \times \dots \times n$  là  $O(n^n)$
- (e)  $\log(n!)$  là  $O(n \log n)$
- (f)  $n^3$  là  $O(2^n)$
- (g)  $\log n$  là  $O(n)$
- (h) Với các hằng số  $b > 1$  và  $k > 0$ ,  $\log_b(n^k)$  là  $O(\log n)$

## Bài tập 3

Hãy giải thích một hàm  $f : \mathbb{R} \rightarrow \mathbb{R}$  là  $O(1)$  nghĩa là gì

# Độ phức tạp tính toán

Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

## Bài tập 4

Chứng minh rằng

- (a)  $x^3$  là  $O(x^4)$  nhưng  $x^4$  không là  $O(x^3)$
- (b)  $3x^4 + 1$  là  $O(x^4/2)$  và  $x^4/2$  là  $O(3x^4 + 1)$
- (c)  $x \log x$  là  $O(x^2)$  nhưng  $x^2$  không là  $O(x \log x)$
- (d)  $2^n$  là  $O(3^n)$  nhưng  $3^n$  không là  $O(2^n)$

## Bài tập 5

Chứng minh rằng nếu  $f(x)$  là  $O(x)$  thì  $f(x)$  cũng là  $O(x^2)$

## Bài tập 6

Chứng minh rằng nếu  $f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$  với  $a_0, a_1, \dots, a_n$  là các số thực (nghĩa là,  $f(x)$  là một đa thức bậc  $n$ ) thì  $f$  là  $O(x^n)$

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

22

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp



# Độ phức tạp tính toán

## Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

23

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

- Ký hiệu  $f(n) = O(g(n))$  thường được sử dụng để chỉ  $f(n)$  là  $O(g(n))$ 
  - Ký hiệu này không hoàn toàn chặt chẽ về mặt toán học, do  $f(n)$  là một hàm còn  $O(g(n))$  là một tập hợp các hàm
  - $f(n) = O(g(n))$  trên thực tế nghĩa là  $f(n) \in O(g(n))$ , do đó có thể viết  $n = O(n^2)$  nhưng **không nên viết**  $O(n^2) = n$
- Bạn có thể gặp biểu thức dạng “ $f(n) + O(g(n)) = O(h(n))$ ”
  - Dấu “=” ở đây nghĩa là “ $\subseteq$ ”. Cụ thể, biểu thức trên cần được hiểu là tập hợp  $S$  gồm các hàm  $f(n) + g_1(n)$  với  $g_1(n) \in O(g(n))$  là tập con của tập  $O(h(n))$
- Bạn có thể gặp biểu thức dạng “ $f(n) \leq g(n) + O(h(n))$  với mọi  $n \geq 0$ ” hoặc tương tự
  - Nghĩa là tồn tại  $e(n)$  sao cho (a)  $f(n) \leq g(n) + e(n)$  với mọi  $n \geq 0$  và (b)  $e(n) \in O(h(n))$
- Một số tác giả định nghĩa  $O$ -lớn bằng cách thay điều kiện  $|f(x)| \leq C|g(x)|$  bằng  $0 \leq f(x) \leq C(g(x))$ . (Làm việc với giá trị tuyệt đối và khả năng các hàm  $f(x)$  và  $g(x)$  có thể nhận giá trị âm thường khó hơn là chỉ làm việc với các hàm nhận giá trị dương.) Định nghĩa theo cách này không hoàn toàn chặt chẽ. Ví dụ như hàm  $\log x$  có thể nhận giá trị âm với  $x$  nhỏ

# Độ phức tạp tính toán

Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

24

Độ tăng của các hàm

Định nghĩa và khái niệm

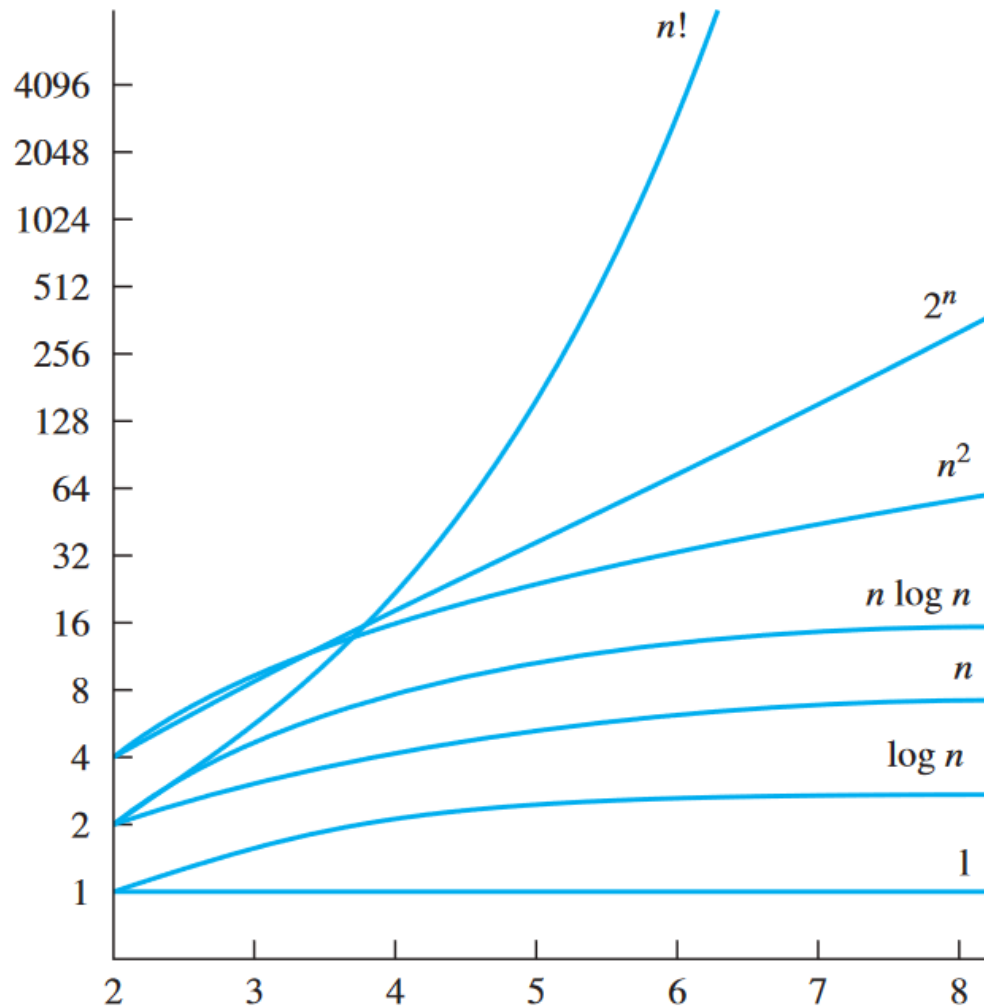
Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp



**Hình:** Độ tăng của một số hàm thường dùng khi đánh giá với ký hiệu  $O$ -lớn [Rosen 2012]

# Độ phức tạp tính toán

## Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

25

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

Một số xấp xỉ hữu ích

- (1) Nếu  $d > c > 1$ , thì  $n^c$  là  $O(n^d)$ , nhưng  $n^d$  không là  $O(n^c)$
- (2) Nếu  $b > 1$  và  $c, d$  là các số dương, thì  $(\log_b n)^c$  là  $O(n^d)$  nhưng  $n^d$  không là  $O((\log_b n)^c)$
- (3) Nếu  $b > 1$  và  $d$  là số dương, thì  $n^d$  là  $O(b^n)$  nhưng  $b^n$  không là  $O(n^d)$
- (4) Nếu  $c > b > 1$ , thì  $b^n$  là  $O(c^n)$  nhưng  $c^n$  không là  $O(b^n)$

Một số tính chất quan trọng

- (a) Nếu  $f_1(x) = O(g_1(x))$  và  $f_2(x) = O(g_2(x))$  thì  $(f_1 + f_2)(x) = O(g(x))$  trong đó  $g(x) = \max\{|g_1(x)|, |g_2(x)|\}$  với mọi  $x \in \mathbb{R}$
- (b) Nếu  $f_1(x) = O(g_1(x))$  và  $f_2(x) = O(g_2(x))$  thì  $(f_1 f_2)(x) = O(g_1(x)g_2(x))$

## Bài tập 7

Ước lượng theo  $O$ -lớn hàm  $f(n) = 3n \log n! + (n^2 + 3) \log n$

# Độ phức tạp tính toán

Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

26

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Ký hiệu $\Omega$ -lớn

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Ta nói rằng  $f$  là  $\Omega(g)$  nếu tồn tại các hằng số  $C > 0$  và  $k$  sao cho  $|f(x)| \geq C|g(x)|$  với mọi  $x > k$

## Ký hiệu $\Omega$ -lớn

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Giả sử  $g(x) \neq 0$  với  $x \in \mathbb{R}$  đủ lớn. Ta nói rằng  $f$  là  $\Omega(g)$  nếu  $\lim_{x \rightarrow \infty} \frac{|f(x)|}{|g(x)|}$  khác 0

## Bài tập 8

Chứng minh rằng  $f$  là  $\Omega(g)$  khi và chỉ khi  $g$  là  $O(f)$

# Độ phức tạp tính toán

Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

27

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Chú ý

$f$  là  $O(g)$  theo nghĩa nào đó là “độ tăng của  $f \leq$  độ tăng của  $g$ ”. Tuy nhiên, bạn cần cẩn thận! Với hai số thực  $a, b \in \mathbb{R}$ , các bất đẳng thức  $a \leq b$  và  $b \leq a$  không thể cùng sai. Nhưng tồn tại hàm  $f$  sao cho  $f = O(g)$  và  $f = \Omega(g)$  cùng sai

## Bài tập 9 (★)

Tìm các ví dụ của các hàm  $f : \mathbb{R} \rightarrow \mathbb{R}$  thỏa mãn các điều kiện (a) – (d) tương ứng

	$f(n)$ là $O(n^3)$	$f(n)$ không là $O(n^3)$
$f(n)$ là $\Omega(n^3)$	(a)	(b)
$f(n)$ không là $\Omega(n^3)$	(c)	(d)

Cụ thể, ở (a), bạn cần tìm ví dụ về một hàm  $f(n)$  đồng thời là  $O(n^3)$  và  $\Omega(n^3)$  và chứng minh ví dụ bạn tìm ra là đúng. Tương tự cho các phần (b), (c), và (d)

# Độ phức tạp tính toán

Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

28

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Ký hiệu $\Theta$ -lớn

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Ta nói rằng  $f$  là  $\Theta(g)$  nếu  $f$  là  $O(g)$  và  $f$  là  $\Omega(g)$

## Ký hiệu $\Theta$ -lớn

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Giả sử  $g(x) \neq 0$  với  $x \in \mathbb{R}$  đủ lớn. Ta nói rằng  $f$  là  $\Theta(g)$  nếu  $\lim_{x \rightarrow \infty} \frac{|f(x)|}{|g(x)|}$  là hữu hạn và khác 0

## Bài tập 10

Chứng minh rằng  $1 + 2 + \dots + n$  là  $\Theta(n^2)$

## Bài tập 11

Chứng minh rằng với các hàm  $f, g$  từ  $\mathbb{R}$  đến  $\mathbb{R}$ ,  $f$  là  $\Theta(g)$  khi và chỉ khi tồn tại các hằng số dương  $C_1, C_2$ , và  $k$  sao cho  $C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|$  với mọi  $x > k$

# Độ phức tạp tính toán

## Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

29

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

### Ký hiệu $o$ -nhỏ

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Ta nói rằng  $f$  là  $o(g)$  nếu với mọi  $C > 0$  tồn tại  $k$  sao cho  $|f(x)| < C|g(x)|$  với mọi  $x > k$

### Ký hiệu $o$ -nhỏ

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Giả sử  $g(x) \neq 0$  với  $x \in \mathbb{R}$  đủ lớn. Ta nói rằng  $f$  là  $o(g)$  nếu  $\lim_{n \rightarrow \infty} \frac{|f(x)|}{|g(x)|} = 0$

### Ký hiệu $o$ -nhỏ

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Ta nói rằng  $f$  là  $o(g)$  nếu  $f$  là  $O(g)$  nhưng  $f$  không là  $\Omega(g)$

# Độ phức tạp tính toán

Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

## Ký hiệu $\omega$ -nhỏ

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Ta nói rằng  $f$  là  $\omega(g)$  nếu với mọi  $C > 0$  tồn tại  $k$  sao cho  $|f(x)| > C|g(x)|$  với mọi  $x > k$

## Ký hiệu $\omega$ -nhỏ

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Giả sử  $g(x) \neq 0$  với  $x \in \mathbb{R}$  đủ lớn. Ta nói rằng  $f$  là  $\omega(g)$  nếu  $\lim_{x \rightarrow \infty} \frac{|f(x)|}{|g(x)|} = \infty$

## Ký hiệu $\omega$ -nhỏ

Cho  $f$  và  $g$  là các hàm  $\mathbb{R} \rightarrow \mathbb{R}$ . Giả sử  $g(x) \neq 0$  với  $x \in \mathbb{R}$  đủ lớn. Ta nói rằng  $f$  là  $\omega(g)$  nếu  $f$  là  $\Omega(g)$  nhưng  $f$  không là  $O(g)$

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

30

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp



# Độ phức tạp tính toán

## Độ tăng của các hàm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

31

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

Tóm lại, với các hàm  $f$  và  $g$  từ  $\mathbb{R}$  đến  $\mathbb{R}$

■  $f$  là  $O(g)$  “ $f$  tăng không nhanh hơn  $g$ ” tương tự “ $\leq$ ”

$$\exists C, k \forall x > k \quad |f(x)| \leq C|g(x)|$$

■  $f$  là  $\Omega(g)$  “ $f$  tăng ít nhất nhanh như  $g$ ” tương tự “ $\geq$ ”

$$\exists C > 0, k \forall x > k \quad |f(x)| \geq C|g(x)|$$

■  $f$  là  $\Theta(g)$  “ $f$  tăng nhanh như  $g$ ” tương tự “ $=$ ”

$$\exists C_1 > 0, C_2 > 0, k \forall x > k \quad C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|$$

■  $f$  là  $o(g)$  “ $f$  tăng chậm hơn  $g$ ” tương tự “ $<$ ”

$$\forall C > 0 \exists k \forall x > k \quad |f(x)| < C|g(x)|$$

■  $f$  là  $\omega(g)$  “ $f$  tăng nhanh hơn  $g$ ” tương tự “ $>$ ”

$$\forall C > 0 \exists k \forall x > k \quad |f(x)| > C|g(x)|$$

# Độ phức tạp tính toán

## Định nghĩa và khái niệm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

32

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

- Một thuật toán cần *luôn xuất ra kết quả đúng* và tốt nhất là *có hiệu suất cao*
- *Độ phức tạp (complexity)* của một tính toán là một cách đo độ “khó” của việc thực hiện tính toán đó
  - *Độ phức tạp theo thời gian (time complexity)*: Số các toán tử hoặc số bước cần thiết
  - *Độ phức tạp theo bộ nhớ (space complexity)*: Số các bit trong bộ nhớ cần thiết
- Phần lớn các thuật toán có độ phức tạp khác nhau đối với các đầu vào có kích thước khác nhau
  - Tìm kiếm trong một dãy dài thường tốn nhiều thời gian hơn tìm kiếm trong một dãy ngắn
- Do đó, độ phức tạp tính toán thường được biểu diễn dưới dạng một *hàm (function) của kích thước đầu vào (input size)*

# Độ phức tạp tính toán

## Độ phức tạp tính toán theo thời gian



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

33

62

- Khi xét độ phức tạp của một thuật toán, ta không quan tâm đến số lượng chính xác các toán tử/bit cần thiết mà chỉ cần một *đánh giá tiệm cận (asymptotic estimate)*
- Một công cụ hữu ích cho việc đánh giá độ phức tạp tính toán là các ký hiệu  $O$ -lớn,  $\Omega$ -lớn, và  $\Theta$ -lớn
- Chúng ta sẽ tập trung vào *độ phức tạp tính toán theo thời gian (time complexity)* (chủ yếu là *trong trường hợp xấu nhất*)
  - *Độ phức tạp trong trường hợp xấu nhất (worst-case complexity)*: xấp xỉ thời gian nhiều nhất cần để giải quyết các trường hợp đầu vào với mỗi kích thước đầu vào
  - *Độ phức tạp trong trường hợp trung gian (average-case complexity)*: xấp xỉ thời gian trung bình cần để giải quyết các trường hợp đầu vào với mỗi kích thước đầu vào
  - *Độ phức tạp trong trường hợp tốt nhất (best-case complexity)*: xấp xỉ thời gian ít nhất cần để giải quyết các trường hợp đầu vào với mỗi kích thước đầu vào

# Độ phức tạp tính toán

## Độ phức tạp tính toán theo thời gian



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

- Để đơn giản, ta thường mô tả thời gian chạy của thuật toán theo  $O$ -lớn. Chú ý rằng với nhiều thuật toán, ta có thể thu được đánh giá tốt hơn với xấp xỉ theo  $\Theta$ -lớn. Tuy nhiên không phải lúc nào ta cũng làm được điều này
- Nhiều tác giả viết “ $f(x)$  là  $O(g(x))$ ” trong khi điều họ thực sự muốn thể hiện là “ $f(x)$  là  $\Theta(g(x))$ ”

34

62

# Độ phức tạp tính toán

## Độ phức tạp tính toán theo thời gian



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

Một số quy tắc tính độ phức tạp thuật toán theo thời gian

- Thời gian thực hiện các lệnh gán ( $:=$ ), trả lại (**return**) là  $O(1)$ , và giả sử thời gian thực hiện các phép toán cơ bản (cộng, trừ, nhân, chia, so sánh, v.v...) cũng là  $O(1)$ 
  - Chú ý rằng với phần cứng hữu hạn (64-bit CPU chỉ biểu diễn được tối đa các số nguyên có giá trị nhỏ hơn  $2^{64}$  dưới dạng số nhị phân), việc cộng hai số nguyên độ dài  $n$  bit biểu diễn dưới dạng số nhị phân có độ phức tạp  $O(n)$
- Độ phức tạp của một chuỗi tuần tự các bước là tổng của độ phức tạp của từng bước
- Thời gian thực hiện cấu trúc **if...then...else** là thời gian lớn nhất thực hiện các lệnh sau **then** hoặc sau **else** và thời gian kiểm tra điều kiện
- Thời gian thực hiện vòng lặp là tổng thời gian thực hiện các lần lặp và thời gian kiểm tra điều kiện lặp. Nếu thời gian thực hiện mỗi lần lặp là giống nhau, thì tổng thời gian thực hiện các lần lặp là tích của số lần lặp và thời gian thực hiện mỗi lần lặp

35

62

# Độ phức tạp tính toán

## Độ phức tạp tính toán theo thời gian



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

36

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

**Bảng:** Một số thuật ngữ thường dùng

Độ phức tạp	Thuật ngữ
$O(1)$	Độ phức tạp hằng số (constant complexity)
$O(\log n)$	Độ phức tạp lôgarit (logarithmic complexity)
$O(n)$	Độ phức tạp tuyến tính (linear complexity)
$O(n \log n)$	Độ phức tạp $n \log n$ (linearithmic complexity)
$O(n^b)$	Độ phức tạp đa thức (polynomial complexity)
$O(b^n)$ , với $b > 1$	Độ phức tạp hàm mũ (exponential complexity)
$O(n!)$	Độ phức tạp giai thừa (factorial complexity)

# Độ phức tạp tính toán

Ví dụ



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

**Thuật toán 1:** Tìm giá trị của phần tử lớn nhất

**Input:**  $a_1, a_2, \dots, a_n$ : dãy số nguyên

**Output:** Giá trị của phần tử lớn nhất trong dãy

```
1  $v := a_1$  _____  $t_1$ 
2 for  $i := 2$  to  $n$  do
3   if  $a_i > v$  then _____  $t_4^i$ 
4   |  $v := a_i$  _____  $t_5$ 
5 return  $v$  _____  $t_3$ 
```

$$T(n) = t_1 + t_2 + t_3$$

$$T(n) \text{ là } O(n)$$

(xấu nhất)

$$T(n) \text{ là } O(n)$$

(tốt nhất)

$$t_1, t_3, t_5 \text{ là } O(1)$$

$$t_2 = \sum_{i=2}^n \left[ t_4^i + (\text{t.g. tăng } i \text{ và kiểm tra } i \leq n) \right]$$

$$t_4^i = t_5 + (\text{t.g. kiểm tra } a_i > v) = O(1)$$

37

62

# Thuật toán

## Một số bài tập



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Bài tập 12

- (a) Thiết kế thuật toán để tính tổng của tất cả các số hạng trong một dãy số nguyên  $a_1, a_2, \dots, a_n$  cho trước
- (b) Chứng minh thuật toán bạn thiết kế là đúng
- (c) Đánh giá thời gian chạy của thuật toán bạn thiết kế

## Bài tập 13

Một chuỗi ký tự được gọi là chuỗi đối xứng (palindrome) khi viết từ trái qua phải và viết từ phải qua trái thì chuỗi không thay đổi. Một ví dụ là chuỗi *ma.dam*.

- (a) Thiết kế thuật toán để kiểm tra xem một chuỗi ký tự có phải là chuỗi đối xứng hay không
- (b) Đánh giá thời gian chạy của thuật toán bạn thiết kế

## Bài tập 14

Cho  $f : A \rightarrow B$  là một hàm với các tập  $A, B$  là các tập con hữu hạn của  $\mathbb{Z}$ . Hãy thiết kế một thuật toán để kiểm tra xem

- (a) liệu  $f$  có là đơn ánh không;
- (b) liệu  $f$  có là toàn ánh không.

Trong mỗi trường hợp, hãy đánh giá thời gian chạy của thuật toán bạn thiết kế

38

62



# Thuật toán

Một số thuật toán tìm kiếm



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

39

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Bài toán tìm kiếm

Cho một dãy  $n$  phần tử  $a_1, a_2, \dots, a_n$  và một phần tử  $x$ . Tìm  $x$  trong dãy đã cho hoặc kết luận rằng  $x$  không có trong dãy

- Tìm kiếm tuyến tính (Linear Search)
- Tìm kiếm nhị phân (Binary Search)



Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

40

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## ■ Bài toán:

■ **Input:**  $a_1, \dots, a_n$ : dãy số nguyên,  $x$ : số nguyên

■ **Output:** Chỉ số  $i$  thỏa mãn  $x = a_i$  hoặc 0 nếu  $x$  không có trong dãy

■ **Tìm kiếm tuyến tính:** Lần lượt xét các phần tử trong dãy cho đến khi tìm được  $x$  hoặc không còn phần tử nào để xét



Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

41

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Ví dụ 8 (Tìm kiếm tuyến tính)

- **Input:** Dãy  $a_1 = 2, a_2 = 5, a_3 = 6, a_4 = 8, a_5 = 12$  và  $x = 8$
- **Output:** Chỉ số  $i$  thỏa mãn  $x = a_i$  hoặc 0 nếu  $x$  không có trong dãy

$a_1$	$a_2$	$a_3$	$a_4$	$a_5$
<div>2</div>	<div>5</div>	<div>6</div>	<div>8</div>	12
$i = 1$	$i = 2$	$i = 3$	$i = 4$	
$\neq x$	$\neq x$	$\neq x$	$= x$	



Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

42

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Thuật toán 3: Tìm kiếm tuyến tính (Linear Search)

**Input:**  $a_1, \dots, a_n$ : dãy số nguyên,  $x$ : số nguyên

**Output:** Chỉ số  $i$  thỏa mãn  $x = a_i$  hoặc 0 nếu  $x$  không có trong dãy

```
1   $i := 1$                                 // Bắt đầu từ đầu dãy
2  while  $i \leq n$  và  $x \neq a_i$  do // Chưa xong và chưa tìm thấy
3       $i := i + 1$     // Đi tới vị trí tiếp theo trong dãy
4  if  $i \leq n$  then
5       $location := i$                 // Tìm thấy  $x$  trong dãy
6  else
7       $location := 0$                 // Không tìm thấy  $x$  trong dãy
8  return  $location$ 
```

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

43

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

Một bất biến vòng lặp trong Thuật toán 3 (vòng **while** ở Dòng 2–3) tìm kiếm tuyến tính số nguyên  $x$  trong dãy  $a_1, \dots, a_n$

$L :=$  *Ở trước lần lặp với biến  $i$ ,  $x \notin \{a_1, \dots, a_{i-1}\}$*

- **Khởi động** ( $i = 1$ ): Do  $i - 1 = 0$ , tập  $\{a_1, \dots, a_{i-1}\}$  là tập rỗng, và do đó  $x \notin \{a_1, \dots, a_{i-1}\}$ , nghĩa là  $L$  đúng
- **Duy trì**: Giả sử  $L$  đúng ở trước lần lặp với  $i = k$  nào đó, nghĩa là  $x \notin \{a_1, \dots, a_{k-1}\}$ . Ta chứng minh  $L$  đúng ở trước lần lặp với  $i = k + 1$ , nghĩa là  $x \notin \{a_1, \dots, a_k\}$ . Thật vậy, để thực hiện lần lặp  $i = k$ , điều kiện ở vòng **while** cần được thỏa mãn, nghĩa là  $k \leq n$  và  $x \neq a_k$ . Kết hợp với giả thiết, ta có điều cần chứng minh
- **Dừng**: Vòng lặp **while** kết thúc khi  $i = n + 1$  hoặc  $x = a_i$  với  $1 \leq i \leq n$ . Với trường hợp đầu tiên, bất biến vòng lặp  $L$  cho ta  $x \notin \{a_1, \dots, a_n\}$  và do đó kết luận không tìm được  $x$ . Với trường hợp thứ hai,  $x$  hiển nhiên thuộc dãy đã cho



Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

44

62

### ■ Bài toán:

- **Input:**  $a_1, \dots, a_n$ : dãy số nguyên *thực sự tăng*,  $x$ : số nguyên
- **Output:** Chỉ số  $i$  thỏa mãn  $x = a_i$  hoặc 0 nếu  $x$  không có trong dãy

### ■ Tìm kiếm nhị phân: (Một ví dụ về kỹ thuật *chia để trị* (*divide and conquer*) trong thiết kế thuật toán)

- (1) Tính  $m = \lfloor (1 + n)/2 \rfloor$ . Phần tử ở giữa của dãy là  $a_m$
- (2) Chia dãy  $a_1, \dots, a_n$  thành hai dãy con (a)  $a_1, \dots, a_m$  và (b)  $a_{m+1}, \dots, a_n$ . Nếu  $x > a_m$  thì ta chỉ tìm  $x$  trong dãy con (b), còn ngược lại thì ta chỉ tìm  $x$  trong dãy con (a)
- (3) Làm tương tự cho đến khi không gian tìm kiếm chỉ còn một phần tử  $a_i$ . Nếu  $x = a_i$  thì trả lại vị trí  $i$  của  $x$ , còn ngược lại thì trả lại 0

# Thuật toán

## Tìm kiếm nhị phân



### Thuật toán 2: Tìm kiếm tuyến tính (Linear Search)

**Input:**  $a_1, \dots, a_n$ : dãy số nguyên,  $x$ : số nguyên

**Output:** Chỉ số  $i$  thỏa mãn  $x = a_i$  hoặc 0 nếu  $x$  không có trong dãy

```
1  $i := 1$   $t_1$ 
2 while  $i \leq n$  và  $x \neq a_i$  do  $t_2$ 
3    $i := i + 1$   $t_5^i$ 
4 if  $i \leq n$  then  $t_3$ 
5    $location := i$   $t_6$ 
6 else
7    $location := 0$   $t_7$ 
8 return  $location$   $t_4$ 
```

$$T(n) = t_1 + t_2 + t_3 + t_4$$

$T(n)$  là  $O(n)$

$T(n)$  là  $O(1)$

$t_1, t_4, t_5^i, t_6, t_7$  là  $O(1)$

(xấu nhất)

(tốt nhất)

$$t_2 = \sum_{\{i | i \leq n \wedge x \neq a_i\}} \left[ t_5^i + (\text{t.g. kiểm tra } i \leq n \text{ và } x \neq a_i) \right]$$

$$t_3 = \max\{t_6, t_7\} + (\text{thời gian kiểm tra } i \leq n)$$

### Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

45

62

## Ví dụ 9 (Tìm kiếm nhị phân)

- **Input:** Dãy  $a_1 = 2, a_2 = 5, a_3 = 6, a_4 = 8, a_5 = 12$  và  $x = 8$
- **Output:** Chỉ số  $i$  thỏa mãn  $x = a_i$  hoặc 0 nếu  $x$  không có trong dãy

$a_1$	$a_2$	$a_3$	$a_4$	$a_5$
2	5	6	8	12
$i$		$m$		$j$

$< x$

$a_1$	$a_2$	$a_3$	$a_4$	$a_5$
2	5	6	8	12
		$i = m$	$j$	

$= x$

$a_1$	$a_2$	$a_3$	$a_4$	$a_5$
2	5	6	8	12
		$i = j$		

$= x$



# Thuật toán

## Tìm kiếm nhị phân



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

47

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

### Thuật toán 4: Tìm kiếm nhị phân (Binary Search)

**Input:**  $a_1, \dots, a_n$ : dãy số nguyên *thực sự tăng*,  $x$ : số nguyên

**Output:** Chỉ số  $i$  thỏa mãn  $x = a_i$  hoặc 0 nếu  $x$  không có trong dãy

```
1   $i := 1$                                 // Chỉ số bắt đầu khoảng tìm kiếm
2   $j := n$                                 // Chỉ số kết thúc khoảng tìm kiếm
3  while  $i < j$  do                        // Khi khoảng tìm kiếm có  $> 1$  phần tử
4       $m := \lfloor (i + j) / 2 \rfloor$            // Chỉ số của phần tử ở giữa
5      if  $x > a_m$  then
6           $i := m + 1$ 
7      else
8           $j := m$ 
9  if  $x = a_i$  then
10      $location := i$ 
11 else
12      $location := 0$ 
13 return  $location$ 
```

Một bất biến vòng lặp trong Thuật toán 4 (vòng **while** ở Dòng 3–8) tìm kiếm nhị phân số nguyên  $x$  trong dãy thực sự tăng  $a_1, \dots, a_n$

$L :=$  Ở trước mỗi lần lặp với các biến  $i, j$ , nếu  $x \in \{a_1, \dots, a_n\}$  thì  
 $x \in \{a_i, a_{i+1}, \dots, a_j\}$

- **Khởi động** ( $i = 1, j = n$ ):  $L$  hiển nhiên đúng
- **Duy trì**: Giả sử ở trước lần lặp với các biến  $k, \ell$ , nếu  $x \in \{a_1, \dots, a_n\}$  thì  $x \in \{a_k, a_{k+1}, \dots, a_\ell\}$ . Ta chứng minh rằng ở trước lần lặp kế tiếp với các biến (a)  $k, \lfloor (k + \ell)/2 \rfloor$  hoặc (b)  $\lfloor (k + \ell)/2 \rfloor + 1, \ell$ , nếu  $x \in \{a_1, \dots, a_n\}$  thì tương ứng (a')  $x \in \{a_k, \dots, a_{\lfloor (k+\ell)/2 \rfloor}\}$  hoặc (b')  $x \in \{a_{\lfloor (k+\ell)/2 \rfloor + 1}, \dots, a_\ell\}$ . Với (a), điều kiện ở Dòng 7 cần được thỏa mãn, nghĩa là  $x \leq a_{\lfloor (k+\ell)/2 \rfloor}$ . Do đó, nếu  $x \in \{a_k, a_{k+1}, \dots, a_\ell\}$  thì (a') đúng. Với (b), điều kiện ở Dòng 5 cần được thỏa mãn, nghĩa là  $x > a_{\lfloor (k+\ell)/2 \rfloor}$ . Do đó, nếu  $x \in \{a_k, a_{k+1}, \dots, a_\ell\}$  thì (b') đúng
- **Dừng**: Vòng lặp **while** dừng khi  $i = j$ , và từ  $L$ , ta có nếu  $x \in \{a_1, \dots, a_n\}$  thì  $x \in \{a_i\}$ . Do đó Thuật toán 4 trả lại vị trí chính xác của  $x$  hoặc kết luận không tìm được  $x$  (Dòng 9–13)

# Độ phức tạp tính toán

## Tìm kiếm nhị phân



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

49

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

**Thuật toán 3:** Tìm kiếm nhị phân (Binary Search)

**Input:**  $a_1, \dots, a_n$ : dãy số nguyên thực sự tăng,  $x$ : số nguyên

**Output:** Chỉ số  $i$  thỏa mãn  $x = a_i$  hoặc 0 nếu  $x$  không có trong dãy

```
1  $i := 1$ 
2  $j := n$ 
3 while  $i < j$  do
4    $m := \lfloor (i + j) / 2 \rfloor$ 
5   if  $x > a_m$  then
6      $i := m + 1$ 
7   else
8      $j := m$ 
9 if  $x = a_i$  then
10   $location := i$ 
11 else
12   $location := 0$ 
13 return  $location$ 
```

Diagram illustrating the execution flow and time complexity markers for the Binary Search algorithm:

- $t_1$  is associated with the initialization of  $i$  and  $j$  (lines 1-2).
- $t_2$  is associated with the **while** loop body (lines 4-8).
- $t_3$  is associated with the **if** statement and assignment of  $location$  (lines 9-12).
- $t_4$  is associated with the **return** statement (line 13).
- $t_5^{i,j}$  is associated with the calculation of  $m$  (line 4).
- $t_6^{i,j}$  is associated with the **if** statement (line 5).
- $t_7$  is associated with the assignment  $i := m + 1$  (line 6).
- $t_8$  is associated with the assignment  $j := m$  (line 8).
- $t_9$  is associated with the assignment  $location := i$  (line 10).
- $t_{10}$  is associated with the assignment  $location := 0$  (line 12).

# Độ phức tạp tính toán

Tìm kiếm nhị phân



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

50

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

$$T(n) = t_1 + t_2 + t_3 + t_4$$

$t_1, t_4, t_5^{i,j}, t_7, \dots, t_{10}$  là  $O(1)$

$$t_2 = \sum_{\text{lần lặp while với cặp } i, j} \left[ (t_5^{i,j} + t_6^{i,j}) + \right. \\ \left. + (\text{thời gian kiểm tra } i < j) \right]$$

$$t_6^{i,j} = \max\{t_7, t_8\} \\ + (\text{thời gian kiểm tra } x > a_m)$$

$$t_3 = \max\{t_9, t_{10}\} \\ + (\text{thời gian kiểm tra } x = a_i)$$

# cặp  $i, j$  trong vòng **while** là  $O(\log n)$

$T(n)$  là  $O(\log n)$

(xấu nhất)

$T(n)$  là  $O(\log n)$

(tốt nhất)

# Độ phức tạp tính toán

## Tìm kiếm nhị phân



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

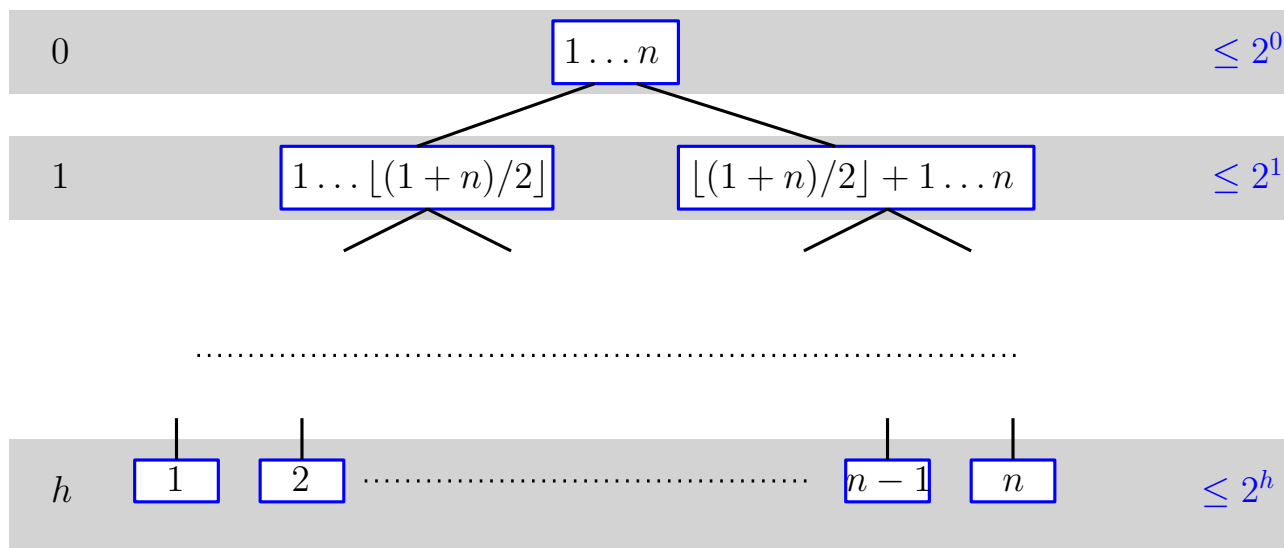
51

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

*Tại sao # cặp  $i, j$  trong vòng lặp **while** là  $O(\log n)$ ?*

- Đánh số các lần lặp **while** lần lượt bằng  $0, 1, 2, \dots, h$ . Mỗi hàng  $0, 1, 2, \dots, h$  liệt kê các khoảng  $i \dots j$  có khả năng xuất hiện ở lần lặp đánh số tương ứng  $\Rightarrow$  # cặp  $i, j \leq h + 1$
- Vòng lặp **while** dừng ở lần lặp  $h$  khi mọi khoảng  $i \dots j$  chỉ có một phần tử  $i = j$
- Ở mỗi hàng  $k \in \{1, \dots, h\}$ , tổng số phần tử của tất cả các khoảng là  $n$ , và có  $\leq 2^k$  khoảng
- Do đó,  $h$  là số nguyên nhỏ nhất thỏa mãn  $n \leq 2^h$ . Suy ra  $h = \lceil \log n \rceil = O(\log n)$



62

# Thuật toán

Một số thuật toán sắp xếp



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Bài toán sắp xếp

Cho một dãy  $n$  phần tử và một cách so sánh hai phần tử bất kỳ trong dãy. Hãy sắp xếp dãy theo thứ tự tăng dần

- Sắp xếp nổi bọt (Bubble Sort)
- Sắp xếp chèn (Insertion Sort)

52

62



Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

53

### ■ Bài toán:

- **Input:**  $a_1, a_2, \dots, a_n$ : dãy số thực ( $n \geq 2$ )
- **Output:** Dãy đã cho được sắp xếp theo thứ tự tăng dần

### ■ Sắp xếp nổi bọt:

- (1) So sánh các phần tử liên tiếp, bắt đầu với cặp  $(a_1, a_2)$
- (2) Nếu  $a_1 > a_2$ , hoán đổi giá trị của chúng
- (3) Lặp lại (1) và (2) với các cặp  $(a_2, a_3), (a_3, a_4), \dots, (a_{n-1}, a_n)$ . Lúc này,  $a_n$  là phần tử lớn nhất trong dãy
- (4) Lặp lại (1) – (3) với dãy  $a_1, \dots, a_{n-1}$ , và sau đó với dãy  $a_1, \dots, a_{n-2}$ , dãy  $a_1, \dots, a_{n-3}, \dots$ , cho đến dãy  $a_1, a_2$



### Ví dụ 10

- **Input:** Dãy  $a_1 = 34, a_2 = 13, a_3 = 21, a_4 = 3, a_5 = 89$
- **Output:** Dãy sắp xếp theo thứ tự tăng dần

	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$
	34	13	21	3	89
$i = 1$	13	21	3	34	89
$i = 2$	13	3	21	34	
$i = 3$	3	13	21		
$i = 4$	3	13			
	3	13	21	34	89

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp





Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

### Thuật toán 5: Sắp xếp nổi bọt (Bubble Sort)

**Input:**  $a_1, a_2, \dots, a_n$ : dãy số thực ( $n \geq 2$ )

**Output:** Dãy đã cho được sắp xếp theo thứ tự tăng dần

```
1 for  $i := 1$  to  $n - 1$  do // Lặp lại  $n - 1$  lần
2   for  $j := 1$  to  $n - i$  do
3     if  $a_j > a_{j+1}$  then
4       Hoán đổi giá trị của  $a_j$  và  $a_{j+1}$ 
5   //  $a_{n-i+1}, \dots, a_n$  đã được sắp xếp
6 //  $a_1, \dots, a_n$  đã được sắp xếp
```

# Thuật toán

## Sắp xếp nổi bọt



### Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

Thuật toán 5 có hai vòng lặp **for**: vòng lặp trong ở Dòng 2–4 và vòng lặp ngoài (chứa vòng lặp trong) ở Dòng 1–4

- Một bất biến vòng lặp cho vòng lặp ngoài là

*Ở trước lần lặp  $i$ , dãy  $a_{n-i+1}, \dots, a_n$  là dãy tăng chứa các phần tử lớn hơn hoặc bằng mọi phần tử trong  $a_1, \dots, a_{n-i}$*

- Một bất biến vòng lặp cho vòng lặp trong là

*Ở trước lần lặp  $j$ ,  $a_j = \max\{a_1, \dots, a_j\}$*

## Sơ đồ chứng minh:

- Chứng minh bước **Khởi động** cho bất biến vòng lặp ngoài
- Ở bước **Duy trì** cho vòng lặp ngoài
  - Chứng minh bất biến vòng lặp trong (**Khởi động**, **Duy trì**, **Dừng**)
  - Sử dụng bất biến vòng lặp trong để chứng minh cho vòng lặp ngoài
- Chứng minh bước **Dừng** cho bất biến vòng lặp ngoài

56

62

# Thuật toán

## Sắp xếp nổi bọt



### Thuật toán 4: Sắp xếp nổi bọt (Bubble Sort)

**Input:**  $a_1, a_2, \dots, a_n$ : dãy số thực ( $n \geq 2$ )

**Output:** Dãy đã cho được sắp xếp theo thứ tự tăng dần

```
1 for i := 1 to n - 1 do
2   for j := 1 to n - i do
3     if  $a_j > a_{j+1}$  then
4       Hoán đổi giá trị của  $a_j$  và  $a_{j+1}$ 
```

Diagram illustrating the nested loops and operations in the Bubble Sort algorithm. The outer loop (lines 1-2) is labeled  $t_1$ . The inner loop (lines 3-4) is labeled  $t_2^i$ . The comparison operation (line 3) is labeled  $t_3^j$ . The swap operation (line 4) is labeled  $t_4$ .

$$T(n) = t_1 = \sum_{i=1}^{n-1} \left[ t_2^i + (\text{t.g. tăng } i \text{ và kiểm tra } i \leq n - 1) \right]$$

$$t_2^i = \sum_{j=1}^{n-i} \left[ t_3^j + (\text{t.g. tăng } j \text{ và kiểm tra } j \leq n - i) \right]$$

$t_4$  là  $O(1)$  ( $v := a_j, a_j := a_{j+1}, a_{j+1} := v$ )

$t_3^j = t_4 + (\text{thời gian kiểm tra } a_j > a_{j+1})$

$T(n)$  là  $O(n^2)$

(xấu nhất)

$T(n)$  là  $O(n^2)$

(tốt nhất)

### Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

57

62



### ■ Bài toán:

- **Input:**  $a_1, a_2, \dots, a_n$ : dãy số thực ( $n \geq 2$ )
- **Output:** Dãy đã cho được sắp xếp theo thứ tự tăng dần

### ■ Sắp xếp chèn:

- (1) Xét  $a_2$ . Tìm vị trí trong dãy 1 phần tử  $a_1$  để chèn  $a_2$  bằng cách duyệt toàn bộ các phần tử trong dãy từ phải sang trái và đẩy mọi phần tử lớn hơn  $a_2$  sang phải một bước. Chèn  $a_2$  vào ngay sau phần tử đầu tiên nhỏ hơn hoặc bằng  $a_2$  khi duyệt dãy ở trên. Đến đây dãy  $a_1, a_2$  đã được sắp xếp theo đúng thứ tự
- (2) Xét  $a_3$ . Tìm vị trí trong dãy 2 phần tử  $a_1, a_2$  để chèn  $a_3$  bằng cách duyệt toàn bộ các phần tử trong dãy từ phải sang trái và đẩy mọi phần tử lớn hơn  $a_3$  sang phải một bước. Chèn  $a_3$  vào ngay sau phần tử đầu tiên nhỏ hơn hoặc bằng  $a_3$  khi duyệt dãy ở trên. Đến đây dãy  $a_1, a_2, a_3$  đã được sắp xếp theo đúng thứ tự
- (3) Tiếp tục với  $a_4, a_5, \dots, a_n$ . Cuối cùng ta thu được dãy  $a_1, \dots, a_n$  đã được sắp xếp theo đúng thứ tự

# Thuật toán

## Sắp xếp chèn



Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

59

62

- **Input:** Dãy  $a_1 = 34, a_2 = 13, a_3 = 21, a_4 = 3, a_5 = 89$
- **Output:** Dãy sắp xếp theo thứ tự tăng dần

	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$
	34	13	21	3	89
$i = 2$	13	34	21	3	89
$i = 3$	13	21	34	3	89
$i = 4$	3	13	21	34	89
$i = 5$	3	13	21	34	89

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

## Thuật toán 6: Sắp xếp chèn (Insertion Sort)

**Input:**  $a_1, a_2, \dots, a_n$ : dãy số thực ( $n \geq 2$ )

**Output:** Dãy đã cho được sắp xếp theo thứ tự tăng dần

```
1 for  $i = 2$  to  $n$  do
2      $m := a_i$  //  $m$  sắp được chèn vào dãy  $a_1, \dots, a_{i-1}$ 
3      $j := i - 1$ 
4     while  $j \geq 1$  và  $m < a_j$  do // Nếu  $m < a_j$ , đẩy  $a_j$ 
        sang phải để có chỗ chèn  $m$ 
5          $a_{j+1} := a_j$ 
6          $j := j - 1$ 
7      $a_{j+1} := m$  // Chèn  $m$ 
8     // Dãy  $a_1, \dots, a_i$  đã được sắp thứ tự
```



Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

Thuật toán 6 có hai vòng lặp: vòng lặp trong **while** ở Dòng 4–6 và vòng lặp ngoài **for** (chứa vòng lặp trong) ở Dòng 1–7

- Một bất biến vòng lặp cho vòng lặp ngoài là

*Ở trước lần lặp  $i$ , dãy  $a_1, \dots, a_{i-1}$  là dãy tăng*

- Một bất biến vòng lặp cho vòng lặp trong là

*Ở trước lần lặp  $j$ ,  $m \leq \min\{a_{j+1}, \dots, a_i\}$*

# Thuật toán

## Sắp xếp chèn



### Thuật toán 5: Sắp xếp chèn (Insertion Sort)

**Input:**  $a_1, a_2, \dots, a_n$ : dãy số thực ( $n \geq 2$ )

**Output:** Dãy đã cho được sắp xếp theo thứ tự tăng dần

```

1 for  $i = 2$  to  $n$  do
2    $m := a_i$   $t_2^i$ 
3    $j := i - 1$   $t_3^i$ 
4   while  $j \geq 1$  và  $m < a_j$  do
5      $a_{j+1} := a_j$   $t_4^i$ 
6      $j := j - 1$   $t_6^{i,j}$ 
7    $a_{j+1} := m$   $t_5^i$ 

```

$t_1$

$$T(n) = t_1 = \sum_{i=2}^n [(t_2^i + t_3^i + t_4^i + t_5^i) + (\text{t.g. tăng } i \text{ và kiểm tra } i \leq n)]$$

$t_2^i, t_3^i, t_5^i, t_6^{i,j}$  là  $O(1)$

$$t_4^i = \sum_{1 \leq j \leq i-1 \text{ và } m < a_j} [t_6^{i,j} + (\text{t.g. kiểm tra } j \geq 1 \text{ và } m < a_j)]$$

$T(n)$  là  $O(n^2)$

(xấu nhất)

$T(n)$  là  $O(n)$

(tốt nhất)

Thuật toán I

Hoàng Anh Đức

Định nghĩa và một số khái niệm

Mô tả thuật toán

Quy tắc viết giả mã

Ví dụ

Chứng minh thuật toán

Bất biến vòng lặp

Ví dụ

Độ phức tạp tính toán

Độ tăng của các hàm

Định nghĩa và khái niệm

Độ phức tạp tính toán theo thời gian

Ví dụ

Tìm kiếm và Sắp xếp

Một số thuật toán tìm kiếm

Một số thuật toán sắp xếp

62

62