COPYRIGHT NOTICE
THÔNG BÁO BẢN QUYỀN

COPYRIGHT (English):

This document is not up to date and may contain several errors or outdated information.

Last revision date: 2023-03-16

BẢN QUYỀN (Tiếng Việt):

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

Ngày sửa đổi cuối cùng: 2023-03-16

VNU-HUS MAT3500: Toán rời rạc

Thuật toán I

Giới thiệu, một số thuật toán tìm kiếm và sắp xếp, độ tăng của hàm

Hoàng Anh Đức

Bộ môn Tin học, Khoa Toán-Cơ-Tin học
Đại học KHTN, ĐHQG Hà Nội
hoanganhduc@hus.edu.vn

Nội dung

Thuật toán
- Định nghĩa và một số khái niệm
- Bất biến vòng lặp

Tìm kiếm và Sắp xếp
- Một số thuật toán tìm kiếm
- Một số thuật toán sắp xếp

Độ tăng của các hàm
- Giới thiệu
- Một số ký hiệu quan trọng

Thuật toán

Định nghĩa và một số khái niệm

- 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

Thuật toán

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

Định nghĩa và một số khái niệm

- Một thuật toán cũng có thể được mô tả bằng một ngôn ngữ máy tính. 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

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

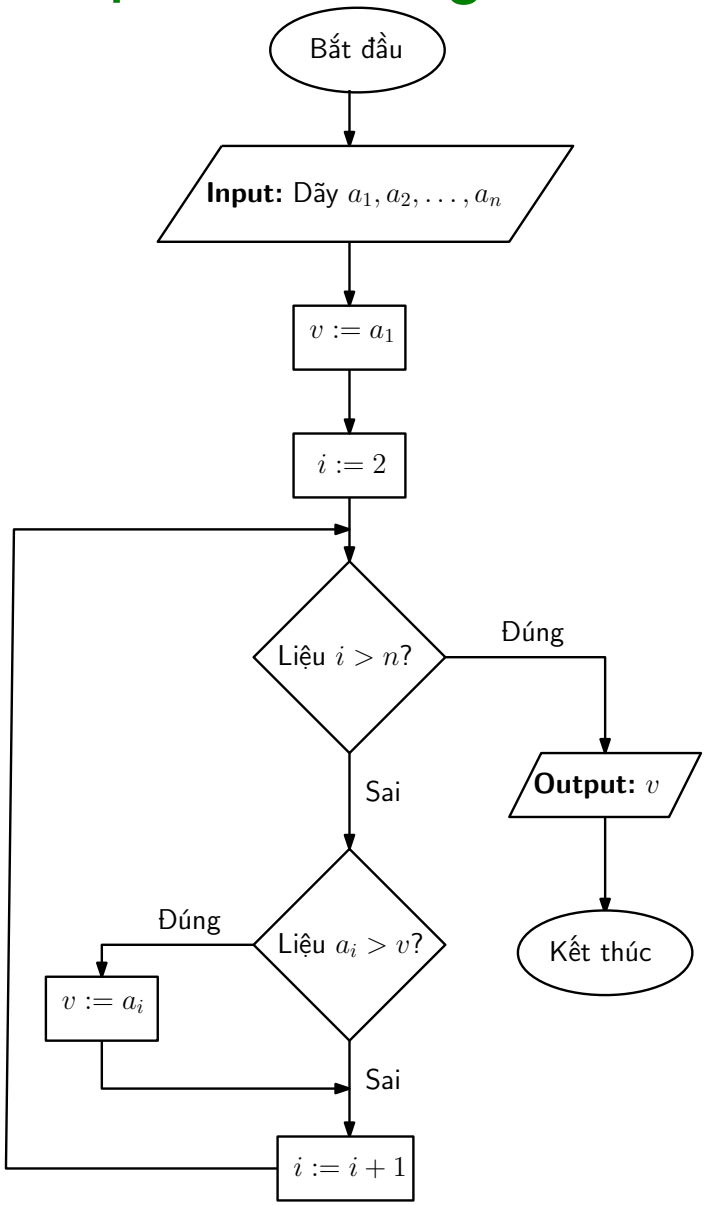
- **Bài toán:**
  - **Input:** $a_1, a_2, \ldots, a_n$: dãy số nguyên
  - **Output:** Giá trị của phần tử lớn nhất trong dãy
- **Tìm giá trị của phần tử lớn nhất:**
  (1) Gán giá trị của một biến tạm thời v (phần tử lớn nhất đến thời điểm hiện tại) bằng a_1
  (2) Xét phần tử ngay tiếp theo trong dãy
  (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

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
$\boxed{7}$	$\boxed{12}$	$\boxed{5}$	$\boxed{16}$	$\boxed{9}$
	$i = 2$	$i = 3$	$i = 4$	$i = 5$
$v = 7$	$v = 12$	$v = 12$	$v = 16$	$v = 16$

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



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

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

Input: $a_1, a_2, \ldots, a_n$: 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, ..., 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, ..., a_i
5 return v
```

Thuật toán
Bất biến vòng lặp

- 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)

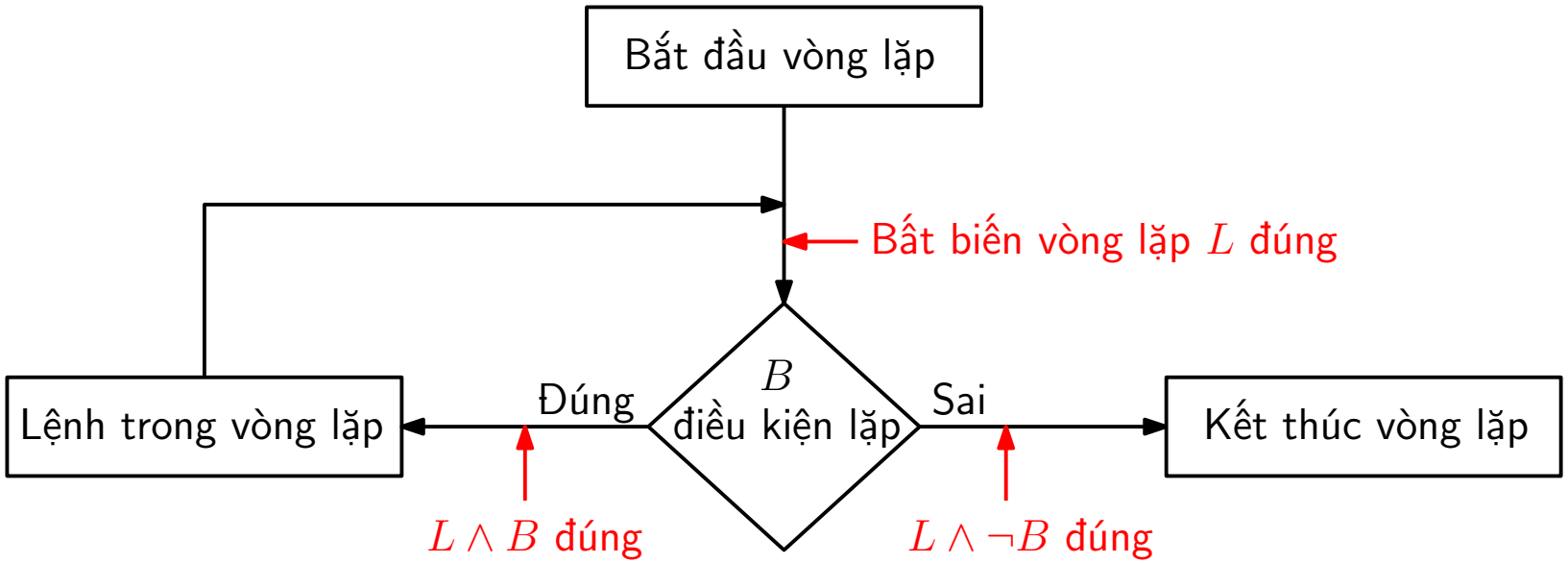
Thuật toán
Bất biến vòng lặp

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 (Maintainance) 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

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 $a_1, \ldots, a_n$

Ở trước lần lặp i, $v = \max\{a_1, a_2, \ldots, a_{i-1}\}$

Gọi v_i là giá trị của v trước lần lặp 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 bất biến vòng lặp đúng ở trước lần lặp $i = k$ nào đó, nghĩa là $v_k = \max\{a_1, \ldots, a_{k-1}\}$. Ta chứng minh bất biến vòng lặp đúng ở trước lần lặp $i = k + 1$, nghĩa là $v_{k+1} = \max\{a_1, \ldots, a_{k-1}, a_k\}$. 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 $\max\{a_1, \ldots, a_{k-1}, a_k\} = \max\{v_k, a_k\} = v_k$. Suy ra $v_{k+1} = \max\{a_1, \ldots, a_{k-1}, a_k\}$
  - 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 $\max\{a_1, \ldots, a_{k-1}, a_k\} = \max\{v_k, a_k\} = a_k$. Suy ra $v_{k+1} = \max\{a_1, \ldots, a_{k-1}, a_k\}$
- **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), $v = \max\{a_1, \ldots, a_n\}$ và do đó là giá trị lớn nhất của các phần tử trong dãy đầu vào

Bài toán tìm kiếm

Cho một dãy n phần tử $a_1, a_2, \ldots, a_n$ 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)

Thuật toán
Tìm kiếm tuyến tính

- **Bài toán:**
  - **Input:** $a_1, \ldots, a_n$: 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

Ví dụ 6 (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
$\boxed{2}$	$\boxed{5}$	$\boxed{6}$	$\boxed{8}$	12
$i = 1$	$i = 2$	$i = 3$	$i = 4$	
$\neq x$	$\neq x$	$\neq x$	$= x$	

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

Input: $a_1, \ldots, a_n$: 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 ≤ n và x ≠ a_i do // Chưa xong và chưa tìm thấy
3     i := i + 1    // Đi tới vị trí tiếp theo trong dãy
4 if i ≤ n then
5     location := i                               // Tìm thấy x trong dãy
6 else
7     location := 0                         // Không tìm thấy x trong dãy
8 return location
```

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

Ở trước lần lặp i, $x \notin \{a_1, \ldots, a_{i-1}\}$

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

Thuật toán
Tìm kiếm nhị phân

- **Bài toán:**
  - **Input:** $a_1, \ldots, a_n$: 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 $a_1, \ldots, a_n$ thành hai dãy con (a) $a_1, \ldots, a_m$ và (b) $a_{m+1}, \ldots, a_n$. 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

Ví dụ 7 (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 3: Tìm kiếm nhị phân (Binary Search)

Input: $a_1, \ldots, a_n$: 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 := ⌊(i + j)/2⌋                    // 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 3 (vòng **while** ở Dòng 3–8) tìm kiếm nhị phân số nguyên x trong dãy thực sự tăng $a_1, \ldots, a_n$

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

- **Khởi động ($i = 1, j = n$):** Bất biến vòng lặp hiển nhiên đúng
- **Duy trì:** Giả sử ở trước lần lặp với các biến k, ℓ, nếu $x \in \{a_1, \ldots, a_n\}$ thì $x \in \{a_k, a_{k+1}, \ldots, a_\ell\}$. 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 $x \in \{a_1, \ldots, a_n\}$ thì tương ứng (a') $x \in \{a_k, \ldots, a_{\lfloor (k+\ell)/2 \rfloor}\}$ hoặc (b') $x \in \{a_{\lfloor (k+\ell)/2 \rfloor + 1}, \ldots, a_\ell\}$. Với (a), điều kiện ở Dòng 7 cần được thỏa mãn, nghĩa là $x \leq a_{\lfloor (k+\ell)/2 \rfloor}$. Do đó, nếu $x \in \{a_k, a_{k+1}, \ldots, a_\ell\}$ 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 $x \in \{a_k, a_{k+1}, \ldots, a_\ell\}$ thì (b') đúng
- **Dừng:** Vòng lặp **while** dừng khi $i = j$, và từ bất biến vòng lặp, ta có nếu $x \in \{a_1, \ldots, a_n\}$ thì $x \in \{a_i\}$. Do đó Thuật toán 3 trả lại vị trí chính xác của x hoặc kết luận không tìm được x (Dòng 9–13)

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)

Thuật toán
Sắp xếp nổi bọt

- **Bài toán:**
  - **Input:** $a_1, a_2, \ldots, a_n$: 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), $\ldots$, (a_{n-1}, a_n). Lúc này, a_n là phần tử lớn nhất trong dãy
  (4) Lặp lại (1) – (3) với dãy $a_1, \ldots, a_{n-1}$, và sau đó với dãy $a_1, \ldots, a_{n-2}$, dãy $a_1, \ldots, a_{n-3}, \ldots$, cho đến dãy a_1, a_2

Ví dụ 8

- **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

Thuật toán

Sắp xếp nổi bọt

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

Input: $a_1, a_2, \ldots, a_n$: 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}, ..., a_n đã được sắp xếp
6 // a_1, ..., a_n đã được sắp xếp
```

Thuật toán
Sắp xếp nổi bọt

Thuật toán 4 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 $a_{n-i+1}, \ldots, a_n$ là dãy tăng chứa các phần tử lớn hơn hoặc bằng mọi phần tử trong $a_1, \ldots, a_{n-i}$

- Một bất biến vòng lặp cho vòng lặp trong là

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

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

Thuật toán
Sắp xếp chèn

- **Bài toán:**
  - **Input:** $a_1, a_2, \ldots, a_n$: 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 $a_4, a_5, \ldots, a_n$. Cuối cùng ta thu được dãy $a_1, \ldots, a_n$ đã được sắp xếp theo đúng thứ tự

Thuật toán
Sắp xếp chèn

- **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

Thuật toán 5: Sắp xếp chèn (Insertion Sort)

Input: $a_1, a_2, \ldots, a_n$: 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, ..., a_{i-1}
3     j := i - 1
4     while j ≥ 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, ..., a_i đã được sắp thứ tự
```

Thuật toán 5 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 $a_1, \ldots, a_{i-1}$ là dãy tăng

- Một bất biến vòng lặp cho vòng lặp trong là

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

Độ tăng của các hàm
Giới thiệu

- 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 $f : \mathbb{R} \to \mathbb{R}$ hoặc $f : \mathbb{N} \to \mathbb{R}$, 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 tăng của các hàm
Ký hiệu O-lớn

Ký hiệu O-lớn

Cho f và g là các hàm $\mathbb{R} \to \mathbb{R}$. 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 $x > k$

- $\exists C, k \forall x > k \, (|f(x)| \leq C|g(x)|)$
- 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 liệu f có là $O(g)$ hay không, chỉ cần một cặp bằng chứng là đủ

Ví dụ 9

Ta chứng minh hàm f 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
  $f(x) = x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 = 4x^2$
- Ta chọn $C = 4$ và $k = 1$

Cách 2:

- Chú ý rằng khi $x > 2$, ta có $2x \leq x^2$ và $1 \leq x^2$
- Do đó với mọi $x > 1$, ta có
  $f(x) = x^2 + 2x + 1 \leq x^2 + x^2 + x^2 = 3x^2$
- Ta chọn $C = 3$ và $k = 2$

Độ tăng của các hàm
Ký hiệu O-lớn

Bài tập 1

Chứng minh

(a) $7x$ *là* $O(x^3)$

(b) x^3 *không là* $O(x^2)$

(c) $1 + 2 + \cdots + n$ *là* $O(n^2)$

(d) $n! = 1 \times 2 \times \cdots \times n$ *là* $O(n^n)$

(e) $\log(n!)$ *là* $O(n \log n)$ *(Nếu không đề cập gì thêm thì* $\log n = \log_2 n$*)*

(f) n^2 *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 2

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

Độ tăng của các hàm
Ký hiệu O-lớn

Một số xấp xỉ hữu ích

- Nếu $d > c > 1$, thì n^c là $O(n^d)$, nhưng n^d không là $O(n^c)$
- Nếu $b > 1$ và c, d là các số dương, thì $(\log_b n)^c$ là $O(n^d)$ nhưng n^d không là $O((\log_b n)^c)$
- Nếu $b > 1$ và d là số dương, thì n^d là $O(b^n)$ nhưng b^n không là $O(n^d)$
- Nếu $c > b > 1$, thì b^n là $O(c^n)$ nhưng c^n không là $O(b^n)$
- Nếu $f_1(x)$ là $O(g_1(x))$ và $f_2(x)$ là $O(g_2(x))$ thì $(f_1 + f_2)(x)$ là $O(\max(|g_1(x)|, |g_2(x)|))$
- Nếu f_1 là $O(g_1)$ và f_2 là $O(g_2)$ thì $f_1 \circ f_2$ là $O(g_1 \circ g_2)$

Độ tăng của các hàm
Ký hiệu Ω-lớn

Ký hiệu Ω-lớn

Cho f và g là các hàm $\mathbb{R} \to \mathbb{R}$. 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 $x > k$

Bài tập 3

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

Ký hiệu Θ-lớn

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

Bài tập 4

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