XẾP ĐẶT VÀ HOÁN VỊ

Bài giảng chuyên đề "*Một số thuật toán tổ hợp*"

Lê Hồng Phương[1]

[1]Khoa Toán–Cơ–Tin học
Trường Đại học Khoa học Tự nhiên, ĐHQG Hà Nội
<*phuonglh@gmail.com*>

07/2012

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Bài toán 1

Bài toán

Có bao nhiêu cách xếp n đồ vật vào m cái hộp?

Một số cách phát biểu tương đương:

1. Cho hai tập hợp hữu hạn X và Y, trong đó $|X| = n \in \mathbb{N}$ và $|Y| = m \in \mathbb{N}$. Có bao nhiêu hàm số $f : X \to Y$?
2. Có n đồ vật và m màu. Có bao nhiêu cách tô màu các đồ vật nếu mỗi vật chỉ được tô một màu?

Bài toán 1

- Kí hiệu: $X = \{x_1, x_2, \ldots, x_n\}$ và $Y = \{y_1, y_2, \ldots, y_m\}$.
- Mỗi hàm $f : X \to Y$ ứng với một dãy

$$\langle y_1, y_2, \ldots, y_n \rangle = \langle f(x_1), f(x_2), \ldots, f(x_n) \rangle$$

- Mỗi y_i có m cách chọn $\forall i = 1, 2, \ldots, n$.
- Như vậy số các hàm f là $\underbrace{m \times m \times \cdots \times m}_{n} = m^n$.

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Bài toán 2

Bài toán

Có bao nhiêu cách xếp n đồ vật vào m cái hộp sao cho không có hộp nào chứa nhiều hơn một vật?

Cách xếp đặt này tương ứng với việc tìm các hàm f đơn ánh, tức là

$$\forall x_1, x_2 \in X, x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2).$$

Bài toán 2

Ta thấy:

- Có thể chọn y_1 bằng m cách từ tập Y;
- Sau khi chọn y_1 thì có thể chọn y_2 bằng $m-1$ cách từ tập $Y \setminus \{y_1\}$;
- Sau khi chọn y_1, y_2 thì có thể chọn y_3 bằng $m-2$ cách từ tập $Y \setminus \{y_1, y_2\}$;
- Tổng quát, có thể chọn y_i bằng $m-(i-1)$ cách từ tập $Y \setminus \{y_1, y_2, \ldots, y_{i-1}\}$.

Như vậy, ta có $m(m-1)\ldots(m-n+1)$ cách chọn dãy $y_1, y_2, \ldots, y_n$ có giá trị khác nhau.

Bài toán 2

- Chú ý rằng ta cần giả thiết $m \geq n$, tức số hộp phải không bé hơn số đồ vật.
- Số cách xếp đặt này chính là số cách chọn *có thứ tự*, hay số chỉnh hợp chập n của m phần từ:

$$A_m^n = \frac{m!}{(m-n)!}, \quad m \geq n.$$

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Bài toán 3 – Xếp đặt có thứ tự

Bài toán

Có bao nhiêu cách xếp n đồ vật vào m cái hộp sao cho mỗi hộp có thể chứa một dãy có thứ tự các vật?

Cách xếp đặt này được gọi là *xếp đặt có thứ tự*. Ví dụ, có 2 vật a, b và 3 hộp. Ta có 12 cách xếp như sau[1]:

$(a, b, \square)$	$(a, \square, b)$	$(b, a, \square)$	$(b, \square, a)$	$(\square, a, b)$	$(\square, b, a)$
$(ab, \square, \square)$	$(ba, \square, \square)$	$(\square, ab, \square)$	$(\square, ba, \square)$	$(\square, \square, ab)$	$(\square, \square, ba)$

[1] Kí hiệu $\square$ là hộp rỗng.

Bài toán 3 – Xếp đặt có thứ tự

- Vật thứ nhất có m cách xếp vào một trong m hộp rỗng;
- Vật thứ hai có $(m-1)+2 = m+1$ cách xếp: hoặc xếp nó vào $m-1$ hộp rỗng còn lại, hoặc xếp nó vào hộp đang chứa vật thứ nhất với hai cách hoặc xếp trước hoặc xếp sau vật đó;
- Giả sử ta đã xếp được $i-1$ vật và trong hộp thứ k đang chứa r_k vật, $\forall k = 1, 2, \ldots, m$. Dễ thấy $\sum_{k=1}^{m} r_k = i-1$. Ta có thể xếp vật thứ i vào một trong các hộp thứ k với $1+r_k$ cách xếp. Như vậy tổng số cách xếp vật thứ i là

$$\sum_{k=1}^{m}(1+r_k) = m + \sum_{k=1}^{m} r_k = m + (i-1).$$

- Do đó, số cách xếp đặt có thứ tự là $\boxed{m \times (m+1) \times (m+n-1)}$.

Bài toán 3 – Xếp đặt có thứ tự

- Vật thứ nhất có m cách xếp vào một trong m hộp rỗng;
- Vật thứ hai có $(m-1)+2 = m+1$ cách xếp: hoặc xếp nó vào $m-1$ hộp rỗng còn lại, hoặc xếp nó vào hộp đang chứa vật thứ nhất với hai cách hoặc xếp trước hoặc xếp sau vật đó;
- Giả sử ta đã xếp được $i-1$ vật và trong hộp thứ k đang chứa r_k vật, $\forall k = 1, 2, \ldots, m$. Dễ thấy $\sum_{k=1}^{m} r_k = i-1$. Ta có thể xếp vật thứ i vào một trong các hộp thứ k với $1+r_k$ cách xếp. Như vậy tổng số cách xếp vật thứ i là

$$\sum_{k=1}^{m}(1+r_k) = m + \sum_{k=1}^{m} r_k = m + (i-1).$$

- Do đó, số cách xếp đặt có thứ tự là $\boxed{m \times (m+1) \times (m+n-1)}$.

Bài toán 3 – Xếp đặt có thứ tự

- Vật thứ nhất có m cách xếp vào một trong m hộp rỗng;
- Vật thứ hai có $(m-1)+2 = m+1$ cách xếp: hoặc xếp nó vào $m-1$ hộp rỗng còn lại, hoặc xếp nó vào hộp đang chứa vật thứ nhất với hai cách hoặc xếp trước hoặc xếp sau vật đó;
- Giả sử ta đã xếp được $i-1$ vật và trong hộp thứ k đang chứa r_k vật, $\forall k = 1, 2, \ldots, m$. Dễ thấy $\sum_{k=1}^{m} r_k = i-1$. Ta có thể xếp vật thứ i vào một trong các hộp thứ k với $1+r_k$ cách xếp. Như vậy tổng số cách xếp vật thứ i là

$$\sum_{k=1}^{m}(1+r_k) = m + \sum_{k=1}^{m} r_k = m + (i-1).$$

- Do đó, số cách xếp đặt có thứ tự là $\boxed{m \times (m+1) \times (m+n-1)}$.

Bài toán 3 – Xếp đặt có thứ tự

- Vật thứ nhất có m cách xếp vào một trong m hộp rỗng;
- Vật thứ hai có $(m-1)+2 = m+1$ cách xếp: hoặc xếp nó vào $m-1$ hộp rỗng còn lại, hoặc xếp nó vào hộp đang chứa vật thứ nhất với hai cách hoặc xếp trước hoặc xếp sau vật đó;
- Giả sử ta đã xếp được $i-1$ vật và trong hộp thứ k đang chứa r_k vật, $\forall k = 1, 2, \ldots, m$. Dễ thấy $\sum_{k=1}^{m} r_k = i-1$. Ta có thể xếp vật thứ i vào một trong các hộp thứ k với $1+r_k$ cách xếp. Như vậy tổng số cách xếp vật thứ i là

$$\sum_{k=1}^{m}(1+r_k) = m + \sum_{k=1}^{m} r_k = m + (i-1).$$

- Do đó, số cách xếp đặt có thứ tự là $\boxed{m \times (m+1) \times (m+n-1)}$.

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Hoán vị

- Mỗi hoán vị của n phần tử là một cách xếp đặt n phần tử đó trên một hàng. Với ba phần tử a, b, c ta có 6 hoán vị sau:

$$abc, \quad acb, \quad bac, \quad bca, \quad cab, \quad cba.$$

- Có bao nhiêu hoán vị của n phần tử?

Hoán vị

- Có n cách chọn phần tử thứ nhất;
- Sau khi đã chọn phần tử thứ nhất thì có $n-1$ cách chọn phần tử thứ hai từ những phần tử còn lại. Như vậy có $n(n-1)$ cách chọn hai phần tử đầu tiên;
- Sau khi đã chọn hai phần tử đầu tiên thì có $n-2$ cách chọn phần tử thứ ba từ những phần tử còn lại. Như vậy có $n(n-1)(n-2)$ cách chọn ba phần tử đầu tiên;
- Nói chung, có $n(n-1)(n-2)\dots(n-k+1)$ cách chọn k phần tử từ n phần tử để xếp đặt chúng vào một hàng.
- Do đó, tổng số hoán vị là $n(n-1)\cdots(1)$. Số này được gọi là n giai thừa, kí hiệu là:

$$n! = n(n-1)\cdots 2\cdot 1 = \prod_{k=1}^{n} k.$$

Hoán vị

- Có n cách chọn phần tử thứ nhất;
- Sau khi đã chọn phần tử thứ nhất thì có $n-1$ cách chọn phần tử thứ hai từ những phần tử còn lại. Như vậy có $n(n-1)$ cách chọn hai phần tử đầu tiên;
- Sau khi đã chọn hai phần tử đầu tiên thì có $n-2$ cách chọn phần tử thứ ba từ những phần tử còn lại. Như vậy có $n(n-1)(n-2)$ cách chọn ba phần tử đầu tiên;
- Nói chung, có $n(n-1)(n-2)\dots(n-k+1)$ cách chọn k phần tử từ n phần tử để xếp đặt chúng vào một hàng.
- Do đó, tổng số hoán vị là $n(n-1)\cdots(1)$. Số này được gọi là n giai thừa, kí hiệu là:

$$n! = n(n-1)\cdots 2\cdot 1 = \prod_{k=1}^{n} k.$$

Hoán vị

- Có n cách chọn phần tử thứ nhất;
- Sau khi đã chọn phần tử thứ nhất thì có $n-1$ cách chọn phần tử thứ hai từ những phần tử còn lại. Như vậy có $n(n-1)$ cách chọn hai phần tử đầu tiên;
- Sau khi đã chọn hai phần tử đầu tiên thì có $n-2$ cách chọn phần tử thứ ba từ những phần tử còn lại. Như vậy có $n(n-1)(n-2)$ cách chọn ba phần tử đầu tiên;
- Nói chung, có $n(n-1)(n-2)\dots(n-k+1)$ cách chọn k phần tử từ n phần tử để xếp đặt chúng vào một hàng.
- Do đó, tổng số hoán vị là $n(n-1)\cdots(1)$. Số này được gọi là n giai thừa, kí hiệu là:

$$n! = n(n-1)\cdots 2\cdot 1 = \prod_{k=1}^{n} k.$$

Hoán vị

- Có n cách chọn phần tử thứ nhất;
- Sau khi đã chọn phần tử thứ nhất thì có $n-1$ cách chọn phần tử thứ hai từ những phần tử còn lại. Như vậy có $n(n-1)$ cách chọn hai phần tử đầu tiên;
- Sau khi đã chọn hai phần tử đầu tiên thì có $n-2$ cách chọn phần tử thứ ba từ những phần tử còn lại. Như vậy có $n(n-1)(n-2)$ cách chọn ba phần tử đầu tiên;
- Nói chung, có $n(n-1)(n-2)\dots(n-k+1)$ cách chọn k phần tử từ n phần tử để xếp đặt chúng vào một hàng.
- Do đó, tổng số hoán vị là $n(n-1)\cdots(1)$. Số này được gọi là n giai thừa, kí hiệu là:

$$n! = n(n-1)\cdots 2\cdot 1 = \prod_{k=1}^{n} k.$$

Hoán vị

- Có n cách chọn phần tử thứ nhất;
- Sau khi đã chọn phần tử thứ nhất thì có $n-1$ cách chọn phần tử thứ hai từ những phần tử còn lại. Như vậy có $n(n-1)$ cách chọn hai phần tử đầu tiên;
- Sau khi đã chọn hai phần tử đầu tiên thì có $n-2$ cách chọn phần tử thứ ba từ những phần tử còn lại. Như vậy có $n(n-1)(n-2)$ cách chọn ba phần tử đầu tiên;
- Nói chung, có $n(n-1)(n-2)\dots(n-k+1)$ cách chọn k phần tử từ n phần tử để xếp đặt chúng vào một hàng.
- Do đó, tổng số hoán vị là $n(n-1)\cdots(1)$. Số này được gọi là n giai thừa, kí hiệu là:

$$n! = n(n-1)\cdots 2\cdot 1 = \prod_{k=1}^{n} k.$$

Giai thừa

Ta quy ước $0! = 1$.[2] Ta có thể viết

$$n! = (n-1)!n, \quad \forall n \in \mathbb{Z}.$$

Giá trị giai thừa tăng rất nhanh:

$$0! = 1, \quad 1! = 1, \quad 2! = 2, \quad 3! = 6, \quad 4! = 24, \quad 5! = 120.$$

Số 1000! là số có hơn 2500 chữ số thập phân. James Stirling đưa ra công thức ước lượng độ lớn của một giai thừa như sau:

$$\boxed{n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n}.$$

Ví dụ, ta có:

$$40320 = 8! \approx 4\sqrt{\pi}\left(\frac{8}{e}\right)^8 \approx 39902.$$

[2] Nếu $n = 0$ thì ta thấy chỉ có một cách để chọn là chọn phần tử rỗng.

Hoán vị và hàm song ánh

- Ta cũng có thể biểu diễn mỗi hoán vị dưới dạng một hàm song ánh như sau. Cho X là tập gồm n phần tử. Một *hoán vị* của X là một hàm song ánh $\sigma : X \to X$. Ví dụ

$$\sigma = \begin{pmatrix} a & b & c & d & e \\ c & d & a & e & b \end{pmatrix}.$$

- Kí hiệu $X = \{x_1, x_2, \ldots, x_n\}$ và S_n là tập tất cả các hoán vị của X.
- Tập S_n chứa các hoán vị được biểu diễn dưới dạng các dãy

$$\sigma = \langle \sigma(x_1), \sigma(x_2), \ldots, \sigma(x_n) \rangle.$$

- Chú ý rằng $\forall i, j : i \neq j \Leftrightarrow x_i \neq x_j$. Như vậy

$$\boxed{|S_n| = n!}$$

Nghịch thế

- Với mỗi hoán vị σ, ta gọi cặp (x_i, x_j) là một *nghịch thế* của σ nếu $x_i < x_j$ nhưng $\sigma(x_i) > \sigma(x_j)$.
- Mỗi hoán vị đều nằm ở một trong hai lớp kích thước bằng nhau là lớp các *hoán vị chẵn* và lớp các *hoán vị lẻ*.
- Tính chẵn lẻ của một hoán vị σ của X là tính chẵn lẻ của số nghịch thế của σ:
  - Nếu số cặp (x_i, x_j) trong đó $x_i < x_j$ và $\sigma(x_i) > \sigma(x_j)$ là một số chẵn thì σ là hoán vị chẵn;
  - Ngược lại σ là hoán vị lẻ.

Nghịch thế

- Ví dụ, số nghịch thế của hoán vị $\sigma = (1, 2, 3)$ là 0 nên đây là một hoán vị chẵn.
- Số nghịch thế của hoán vị $\sigma = (3, 2, 1)$ là 3 nên đây là một hoán vị lẻ.
- Dấu của hoán vị được định nghĩa bởi

$$\boxed{\text{sign}(\sigma) = (-1)^{N(\sigma)}},$$

trong đó $N(\sigma)$ là số nghịch thế của σ.

Bài tập

Bài tập 1. Viết chương trình kiểm tra xem hai dãy số nguyên dương cho trước có phải là hoán vị của nhau hay không.

- Input: Hai mảng số nguyên, mỗi mảng có n phần tử.
- Output: true/false.

Bài tập 2. Viết chương trình tìm dấu của một hoán vị.

- Input: Một mảng số nguyên chứa các số tự nhiên từ 1 tới n biểu diễn một hoán vị.
- Output: Dấu của hoán vị ($+1$ hoặc -1).

Sinh các hoán vị

Bài toán

Hãy sinh tất cả n! hoán vị độ dài n.

- Bài toán sinh các hoán vị là một trong những bài toán quan trọng của tổ hợp, có nhiều ứng dụng trong thực tế;
- Các hoán vị là cơ sở cấu trúc của nhiều thuật toán tìm kiếm quay lui;
- Có nhiều thuật toán sinh các hoán vị, thuật toán cổ nhất xuất hiện từ những năm 1650.

Sinh các hoán vị

Một số lưu ý:

- n cần phải nằm trong khoảng 10 và 20. Nếu n quá lớn thì số hoán vị là rất lớn, các thuật toán có độ phức tạp thời gian cao;
- Việc xử lí một hoán vị thường tốn nhiều thời gian hơn là việc sinh một hoán vị.

Sinh các hoán vị

Thời gian sinh các hoán vị theo độ lớn của n và tốc độ tính toán

n	**Số hoán vị**	**triệu/giây**	**tỉ/giây**	**ngàn tỉ/giây**
10	3,628,800			
11	39,916,800	giây		
12	479,001,600	phút		
13	6,227,020,800	giờ	giây	
14	87,178,291,200	ngày	phút	
15	1,307,674,368,000	tuần	phút	
16	20,922,789,888,000	tháng	giờ	giây
17	355,687,428,096,000	năm	ngày	phút
18	6,402,373,705,728,000	∞	tháng	giờ
19	121,645,100,408,832,000	∞	năm	ngày
20	2,432,902,008,176,640,000	∞	∞	tháng

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Thuật toán quay lui

- Ý tưởng: Đưa bài toán sinh các dãy hoán vị độ dài n về n bài toán sinh các dãy hoán vị độ dài $n-1$.
- Giả sử cần sinh tất cả các hoán vị của 5 phần tử *abcde*. Ta thấy các hoán vị của 5 phần tử này là một trong những dãy sau:
  1. kết thúc bởi a và dãy trước đó là một trong 4! hoán vị của *bcde*;
  2. kết thúc bởi b và dãy trước đó là một trong 4! hoán vị của *acde*;
  3. kết thúc bởi c và dãy trước đó là một trong 4! hoán vị của *abde*;
  4. kết thúc bởi d và dãy trước đó là một trong 4! hoán vị của *abce*;
  5. kết thúc bởi e và dãy trước đó là một trong 4! hoán vị của *abcd*.

Thuật toán quay lui

- Để sinh các hoán vị độ dài $n-1$ ta làm tương tự (đệ quy): đưa việc sinh một hoán vị độ dài $n-1$ về việc sinh $n-1$ hoán vị độ dài $n-2$.
- Lặp lại quá trình này cho tới khi $n = 1$ thì dừng.

Thuật toán quay lui

Giả sử dãy hoán vị được lưu trong một mảng $a[1..n]$. Thuật toán được mô tả tổng quát như sau: với mọi $i = 1, 2, \ldots, n$:

- Hoán đổi $a[i]$ với $a[n]$;
- Gọi đệ quy để sinh mọi hoán vị của $a[1..n-1]$ nếu $n > 1$;
- Sau khi kết thúc đệ quy, hoán đổi lại $a[i]$ với $a[n]$ (*quay lui*).

Thuật toán quay lui

Chú ý rằng khi cài đặt thuật toán bằng C/Java, chỉ số của mảng nằm trong đoạn $[0..n-1]$ thay vì đoạn $[1..n]$.

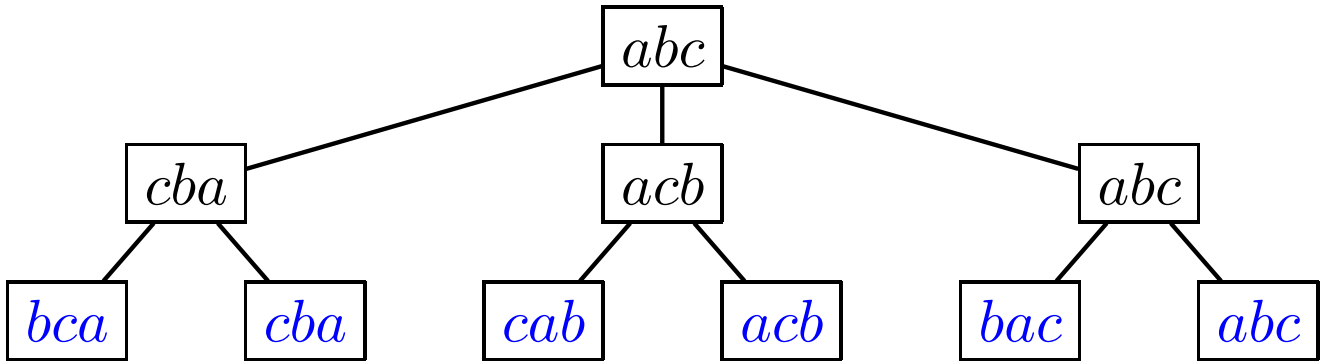
```
void enumerate(char a[], int n) {
  int i;
  if (n == 0) {
    printf("%s\n", a);
  } else {
    for (i = 0; i < n; i++) {
      swap(a, i, n - 1);
      enumerate(a, n - 1);
      swap(a, i, n - 1);
    }
  }
}
```

```
void swap(char a[], int i, int j)
{
  char c;
  c = a[i];
  a[i] = a[j];
  a[j] = c;
}

int main(int argc, char **argv) {
  char a[] = "abc";
  enumerate(a, 3);
  return 0;
}
```

Thuật toán quay lui

Với 3 phần tử abc ta có 6 hoán vị: $bca, cba, cab, acb, bac, abc$ như sau:



Hạn chế của thuật toán? Phải hoán đổi nhiều lần. Cụ thể là thủ tục `swap()` được gọi $\boxed{2n!}$ lần.

Bài tập

Bài tập 3. Vẽ sơ đồ sinh các hoán vị của 4 phần tử bằng phương pháp quay lui.

Bài tập 4. Viết chương trình sinh các hoán vị của n phần tử bằng thuật toán quay lui. Đo thời gian chạy của chương trình với các giá trị n khác nhau.

- Input: Một số tự nhiên $n < 15$.
- Output: Mọi hoán vị của dãy $1..n$ và thời gian chạy.

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Thuật toán đệ quy

Thuật toán này tốt hơn so với thuật toán quay lui ở chỗ ta chỉ cần hoán đổi một lần phần tử $a[i]$ với $a[n]$.

```
void enumerate(char a[], int n) {
  int i;
  if (n == 0) {
    printf("%s\n", a);
  } else {
    for (i = 0; i < n; i++) {
      enumerate(a, n - 1);
      swap(a, ?, n - 1);
    }
  }
}
```

Ta cần tìm các giá trị cụ thể của vị trí ? để hoán đổi với vị trí cuối.

Thuật toán đệ quy

- Để tìm giá trị của `?`, ta cần tính *bảng chỉ số* từ hoán vị cho $n-1$ phần tử (đã biết).
- Trước tiên, chỉ có 2 hoán vị của 2 phần tử là

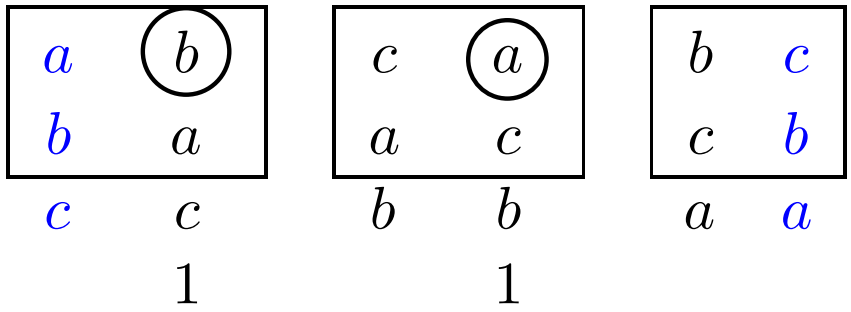
$$\boxed{\begin{matrix} a & b \\ b & a \end{matrix}}$$

- Với 3 phần tử, ta có 6 hoán vị như sau:

$$\begin{matrix} 1: \\ 2: \\ 3: \end{matrix} \quad \begin{bmatrix} a \\ b \\ c \end{bmatrix} \rightarrow \begin{bmatrix} b \\ a \\ c \end{bmatrix} \rightarrow \begin{bmatrix} c \\ a \\ b \end{bmatrix} \rightarrow \begin{bmatrix} a \\ c \\ b \end{bmatrix} \rightarrow \begin{bmatrix} b \\ c \\ a \end{bmatrix} \rightarrow \begin{bmatrix} c \\ b \\ a \end{bmatrix}$$

Thuật toán đệ quy

Các hoán vị này được sinh theo sơ đồ dưới đây, trong đó các phần đóng khung là các hoán vị của 2 phần tử đầu tiên của dãy.



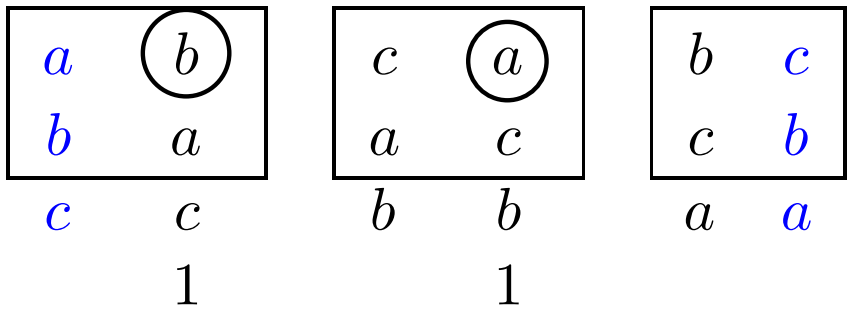
- Trước tiên, hoán vị của hai phần tử đầu tiên được sinh (ab, ba).
- Chỉ số 1 đầu tiên ứng với việc ta sẽ hoán đổi vị trí cuối (phần tử c) với phần tử đang ở vị trí số 1 (phần tử b).

Thuật toán đệ quy

- Chú ý rằng ta luôn hoán đổi phần tử cuối với phần tử có thứ tự ngay trước nó. Phần tử cuối đang là c thì cần tìm phần tử b để hoán đổi. Ở đây, b đang ở vị trí số 1.
- Sau khi đổi chỗ, ta sinh các hoán vị của 2 phần tử đầu tiên (ca, ac).
- Chỉ số 1 thứ hai ứng với việc ta sẽ hoán đổi vị trí cuối (phần tử b) với phần tử a đang ở vị trí số 1.
- Sau khi đổi chỗ, ta sinh các hoán vị của hai phần tử đầu tiên (bc, cb).

Như vậy các chỉ số cần dùng để sinh mọi hoán vị của 3 phần tử theo phương pháp này là $(1, 1)$.

Thuật toán đệ quy



Ta thấy nếu sinh hoán vị bằng cách đổi chỗ như trên thì sau khi sinh mọi hoán vị của abc ta sẽ có hoán vị cuối là cba:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \Longrightarrow \begin{bmatrix} c \\ b \\ a \end{bmatrix}$$

Thuật toán đệ quy

Tiếp theo, với 4 phần tử, các hoán vị đầu và cuối của 4 phần tử với phần tử cuối cố định bằng d tương ứng là $abcd$ và $cbad$:

$$\left[\begin{bmatrix} a \\ b \\ c \end{bmatrix} d\right] \Longrightarrow \left[\begin{bmatrix} c \\ b \\ a \end{bmatrix} d\right]$$

Sau khi sinh mọi hoán vị với phần tử cuối là d thì d sẽ được hoán đổi với c ở chỉ số 1 và có:

$$\begin{bmatrix} \textcircled{c} \\ b \\ a \\ d \end{bmatrix} \rightarrow \begin{bmatrix} d \\ b \\ a \\ c \end{bmatrix}$$

Tương tự như trên, vì sau khi sinh mọi hoán vị của 3 phần tử dba ta có hoán vị cuối là abd nên các hoán vị đầu và cuối của 4 phần tử với phần tử cuối cố định bằng c tương ứng là $dbac$ và $abdc$:

$$\begin{bmatrix} \begin{bmatrix} d \\ b \\ a \end{bmatrix} \\ c \end{bmatrix} \Longrightarrow \begin{bmatrix} \begin{bmatrix} a \\ b \\ d \end{bmatrix} \\ c \end{bmatrix}$$

Sau khi sinh mọi hoán vị với phần tử cuối là c thì c sẽ được hoán đổi với b ở chỉ số 2 và có:

$$\begin{bmatrix} a \\ \textcircled{b} \\ d \\ c \end{bmatrix} \rightarrow \begin{bmatrix} a \\ c \\ d \\ b \end{bmatrix}$$

Vì sau khi sinh mọi hoán vị của 3 phần tử acd ta có hoán vị cuối là dca nên các hoán vị đầu và cuối của 4 phần tử với phần tử cuối cố định bằng b tương ứng là $acdb$ và $dcab$:

$$\begin{bmatrix} \begin{bmatrix} a \\ c \\ d \end{bmatrix} \\ b \end{bmatrix} \Longrightarrow \begin{bmatrix} \begin{bmatrix} d \\ c \\ a \end{bmatrix} \\ b \end{bmatrix}$$

Sau khi sinh mọi hoán vị với phần tử cuối là b thì b sẽ được hoán đổi với a ở chỉ số 3 và có:

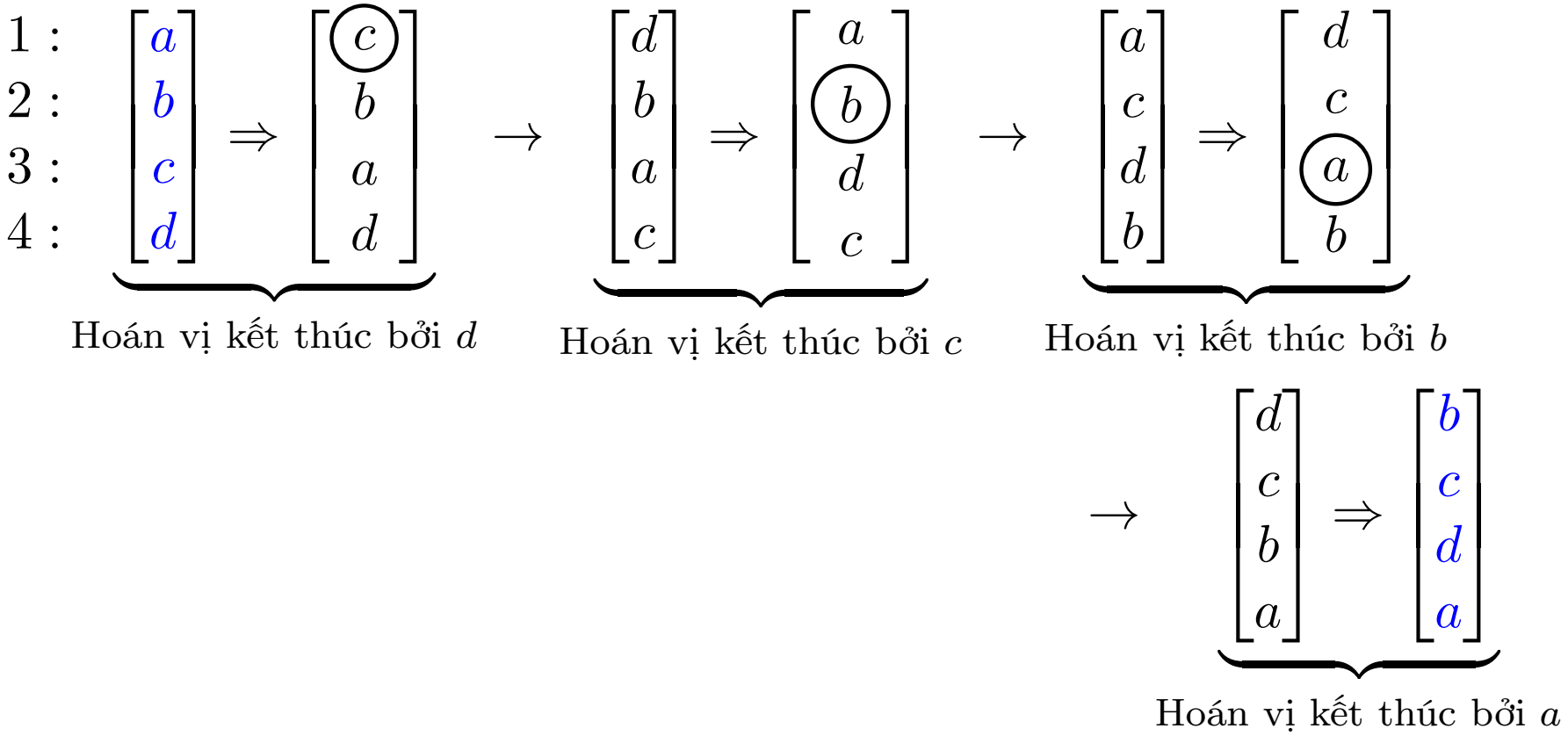
$$\begin{bmatrix} d \\ c \\ \textcircled{a} \\ b \end{bmatrix} \rightarrow \begin{bmatrix} d \\ c \\ b \\ a \end{bmatrix}$$

Cuối cùng, vì sau khi sinh mọi hoán vị của 3 phần tử dcb ta có hoán vị cuối là bcd nên các hoán vị đầu và cuối của 4 phần tử với phần tử cuối cố định bằng a tương ứng là $dcba$ và $bcda$:

$$\begin{bmatrix} \begin{bmatrix} d \\ c \\ b \end{bmatrix} \\ a \end{bmatrix} \Longrightarrow \begin{bmatrix} \begin{bmatrix} b \\ c \\ d \end{bmatrix} \\ a \end{bmatrix}$$

Thuật toán đệ quy

Tóm lại, ta có lược đồ sinh các hoán vị của 4 phần tử như sau:



Các chỉ số cần dùng để sinh mọi hoán vị của 4 phần tử theo phương pháp này là $(1, 2, 3)$.

Thuật toán đệ quy

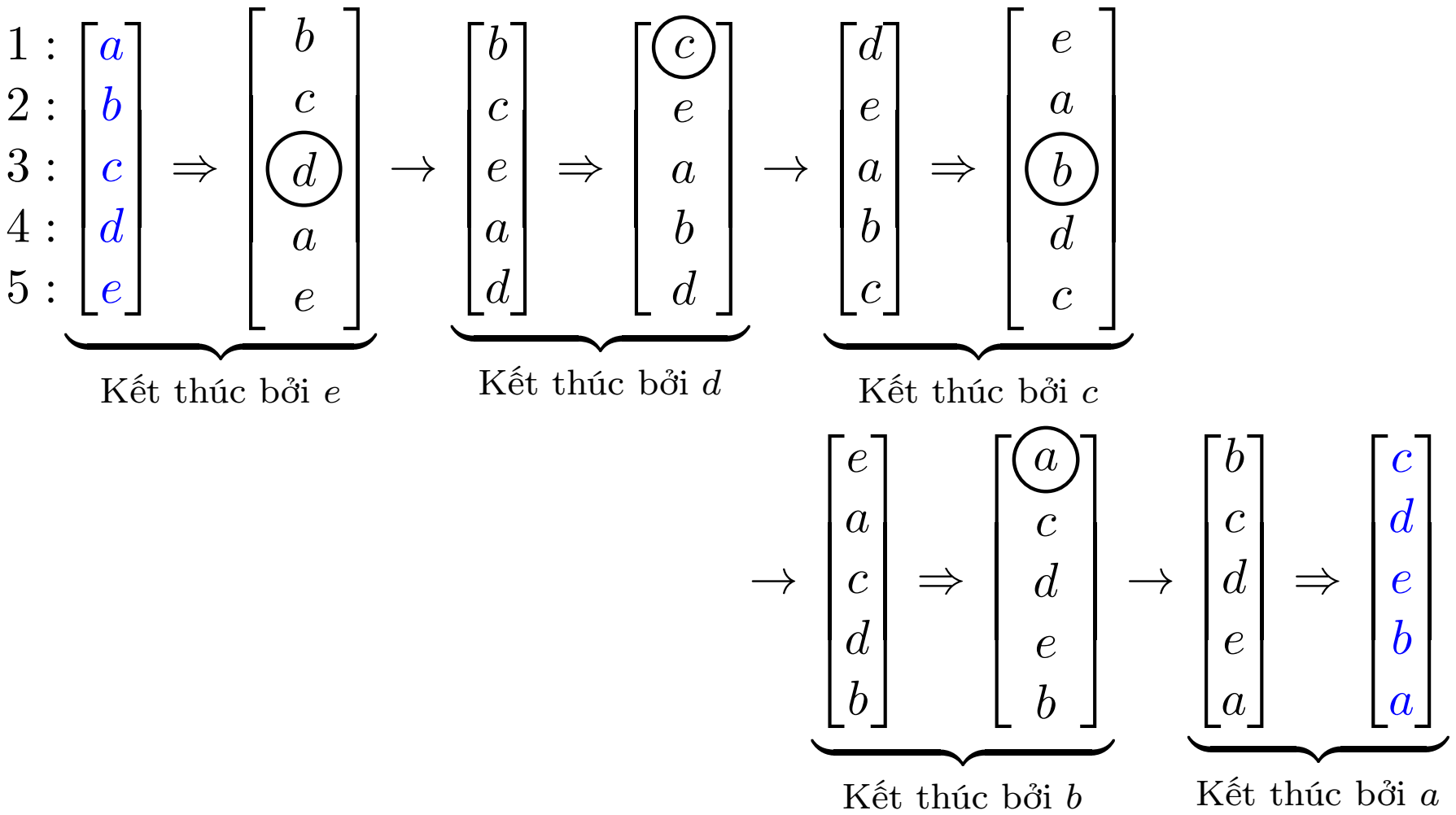
Ta thấy nếu sinh hoán vị bằng cách đổi chỗ kế tiếp như trên thì sau khi sinh mọi hoán vị của $abcd$ ta sẽ có hoán vị cuối là $bcda$:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \Longrightarrow \begin{bmatrix} b \\ c \\ d \\ a \end{bmatrix}$$

Đây là cơ sở để ta tiếp tục xây dựng các hoán vị của 5 phần tử.

Thuật toán đệ quy

Các chỉ số cần dùng để sinh mọi hoán vị của 5 phần tử là $(3, 1, 3, 1)$:



Thuật toán đệ quy

- Cứ tiếp tục như vậy, ta có thể sinh được mọi hoán vị của dãy n phần tử.
- Để sinh các hoán vị theo thuật toán này, ta cần tính được trước bảng chỉ số để tìm các vị trí cụ thể cho dấu ?.

Bảng chỉ số với 11 hàng đầu tiên như sau:

$$\texttt{index} = \begin{pmatrix}
1 & & & & & & & & & \\
1 & & & & & & & & & \\
1 & 1 & & & & & & & & \\
1 & 2 & 3 & & & & & & & \\
3 & 1 & 3 & 1 & & & & & & \\
3 & 4 & 3 & 2 & 3 & & & & & \\
5 & 3 & 1 & 5 & 3 & 1 & & & & \\
5 & 2 & 7 & 2 & 1 & 2 & 3 & & & \\
7 & 1 & 5 & 5 & 3 & 3 & 7 & 1 & & \\
7 & 8 & 1 & 6 & 5 & 4 & 9 & 2 & 3 & \\
9 & 7 & 5 & 3 & 1 & 9 & 7 & 5 & 3 & 1
\end{pmatrix}$$

Chú ý rằng hai hàng đầu của bảng tương ứng với các trường hợp $n = 1$ và $n = 2$.

- Với $n = 1$ chỉ có 1 hoán vị, ta không cần hoán đổi vị trí nhưng để thống nhất trong cách xử lí, có thể coi đây là trường hợp tự hoán đổi: hoán đổi vị trí 1 với vị trí 1.
- Với $n = 2$ ta có hai hoán vị, phần tử thứ 2 sẽ được hoán đổi với phần tử thứ 1 nên cũng có một chỉ số 1.

Ta cũng cần bổ sung thêm thông tin $\texttt{index}[n][n] = n, \forall n$ để sinh các hoán vị kết thúc bởi phần tử cuối cùng (hoán đổi phần tử cuối với chính nó).

Thuật toán đệ quy

Sau khi tính được bảng chỉ số **index** thì thuật toán sinh các hoán vị bằng phương pháp đệ quy trên được cụ thể hóa như sau:

```
void enumerate(char a[], int n) {
  int i;
  if (n == 0) {
    printf("%s\n", a);
  } else {
    for (i = 0; i < n; i++) {
      enumerate(a, n - 1);
      swap(a, index[n-1][i], n - 1);
    }
  }
}
```

Thuật toán đệ quy

Danh sách các hoán vị của bốn phần tử $abcd$ được sinh bằng thuật toán này là:

$abcd$	$bacd$	$cabd$	$acbd$	$bcad$	$cbad$
$dbac$	$bdac$	$adbc$	$dabc$	$badc$	$abdc$
$acdb$	$cadb$	$dacb$	$adcb$	$cdab$	$dcab$
$dcba$	$cdba$	$bdca$	$dbca$	$cbda$	$bcda$

Ta thấy thuật toán này chỉ sử dụng $n!$ phép hoán đổi vị trí các phần tử để sinh $n!$ hoán vị. Vì vậy thuật toán là tối ưu.

Bài tập

Bài tập 5. Kiểm tra xem dãy chỉ số cần dùng để sinh các hoán vị của 6 phần tử có đúng là $(3, 4, 3, 2, 3)$ hay không.

Bài tập 6. Hãy xây dựng thuật toán và viết chương trình tự động tính bảng chỉ số ở trên.

- Input: Một số tự nhiên $n < 20$.
- Output: Bảng `index`$[n][n]$.

Bài tập 7. Viết chương trình sinh các hoán vị của n phần tử bằng thuật toán đệ quy. Đo thời gian chạy của chương trình với các giá trị n khác nhau.

- Input: Một số tự nhiên $n < 15$.
- Output: Mọi hoán vị của dãy $1..n$ và thời gian chạy.

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Thuật toán Heap

- B. R. Heap đề xuất một thuật toán đệ quy sinh các hoán vị mà không cần tính bảng chỉ số `index` như trong thuật toán trên. [3]
- Để tìm phần tử tiếp theo đặt vào vị trí cuối, ta chỉ cần sử dụng quy tắc đơn giản trong khi gọi đệ quy: *vị trí đó là 1 nếu n là lẻ, là i nếu n là chẵn.*

[3] B. R. Heap, "Permutations by interchanges," *Computer Journal*, vol. 6, no. 293-294, 1963.

Thuật toán Heap

Thuật toán Heap được cài đặt như sau:

```
void enumerate(char a[], int n) {
  int i;
  if (n == 0) {
    printf("%s\n", a);
  } else {
    for (i = 0; i < n; i++) {
      enumerate(a, n - 1);
      swap(a, n % 2 ? 0 : i, n - 1);
    }
  }
}
```

Thuật toán Heap

Danh sách các hoán vị của bốn phần tử $abcd$ được sinh bằng thuật toán này là:

$abcd$	$bacd$	$cabd$	$acbd$	$bcad$	$cbad$
$dbca$	$bdca$	$cdba$	$dcba$	$bcda$	$cbda$
$dacb$	$adcb$	$cdab$	$dcab$	$acdb$	$cadb$
$dabc$	$adbc$	$bdac$	$dbac$	$abdc$	$badc$

Bài tập 8. Viết chương trình sinh các hoán vị của n phần tử bằng thuật toán Heap.

- Input: Một số tự nhiên $n < 15$.
- Output: Mọi hoán vị của dãy $1..n$.

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Thuật toán Steinhauss–Johnson–Trotter

Ý tưởng: sinh các hoán vị trong đó hai hoán vị liên tiếp chỉ khác nhau bởi một phép đổi chỗ kế tiếp.
Ví dụ, các hoán vị của 3 phần tử abc được sinh như sau:

$$\underline{ab}c \to b\underline{ac} \to bca$$
$$\underline{bc}a \to c\underline{ba} \to cab$$
$$\underline{ca}b \to a\underline{cb} \to abc$$

Ta thấy trong mỗi bước có một cặp phần tử kề nhau được đổi chỗ.

Thuật toán Steinhauss–Johnson–Trotter

Thuật toán Steinhauss–Johnson–Trotter được đề xuất bởi Hugo Steinhaus, Selmer M. Johnson và Hale F. Trotter trong các tài liệu

- S. M. Johnson, "Generation of permutations by adjacent transposition," *Mathematics of Computation*, vol. 17, pp. 282–285, 1963
- H. Steinhaus, *One hundred problems in elementary mathematics*. New York, 1964
- H. F. Trotter, "Algorithm 115: Perm," *Communications of the ACM*, vol. 8, no. 5, pp. 434–435, 1964

Thuật toán này còn được gọi là thuật toán Johnson–Trotter hoặc thuật toán đổi chỗ đơn giản ("plain changes").

Thuật toán Steinhauss–Johnson–Trotter

- Thực ra thuật toán này đã được phát minh từ thế kỉ thứ 17 bởi những người thợ kéo chuông nhà thờ ở Anh.
- Thuật toán này là thuật toán tốt và rất hiệu quả.
- Tương tự như thuật toán đệ quy, ưu điểm lớn nhất là ta có thể tăng tốc các tính toán kế tiếp nhau vì hai hoán vị liên tiếp chỉ khác nhau 2 vị trí, $n - 2$ vị trí còn lại là giống nhau.

Thuật toán Steinhauss–Johnson–Trotter

Cấu trúc đệ quy: Chuỗi các hoán vị của n phần tử có thể được sinh từ chuỗi các hoán vị của $n-1$ phần tử bằng cách đặt phần tử thứ n vào từng vị trí của các chuỗi hoán vị đó theo cách sau:

- Nếu hoán vị của $n-1$ phần tử là một *hoán vị chẵn* thì phần tử thứ n được đặt vào mọi vị trí có thể theo thứ tự giảm dần, từ n tới 1;
- Nếu khi hoán vị của $n-1$ phần tử là một *hoán vị lẻ* thì phần tử thứ n được đặt vào mọi vị trí có thể theo thứ tự tăng dần, từ 1 tới n.

Thuật toán Steinhauss–Johnson–Trotter

Ta giả sử các phần tử là các số nguyên dương. Từ hoán vị với một phần tử 1 ta có thể đặt 2 vào mọi vị trí có thể theo thứ tự giảm dần để lập các hoán vị của hai phần tử:

$$\begin{matrix} 1 & \mathbf{2} \\ \mathbf{2} & 1 \end{matrix}$$

Sau đó, ta có thể đặt phần tử 3 vào một trong các vị trí khác nhau của các hoán vị này, theo thứ tự giảm dần với hoán vị $(1, 2)$ và theo thứ tự tăng dần với hoán vị $(2, 1)$:

$$\begin{matrix} 1 & 2 & \mathbf{3} \\ 1 & \mathbf{3} & 2 \\ \mathbf{3} & 1 & 2 \\ \mathbf{3} & 2 & 1 \\ 2 & \mathbf{3} & 1 \\ 2 & 1 & \mathbf{3} \end{matrix}$$

Thuật toán Steinhauss–Johnson–Trotter

- Ở bước tiếp theo, phần tử 4 sẽ lần lượt được đặt vào hoán vị $(1, 2, 3)$ theo thứ tự giảm dần, đặt vào hoán vị $(1, 3, 2)$ theo thứ tự tăng dần, đặt vào hoán vị $(3, 1, 2)$ theo thứ tự giảm dần....
- Việc đặt phần tử cứ được tiếp tục theo cách như vậy, luân phiên giữa thứ tự tăng dần và giảm dần, áp dụng cho các giá trị n lớn bất kì.
- Như vậy, mỗi hoán vị đứng sau chỉ khác với hoán vị trước nó hoặc bởi sự di chuyển của phần tử thứ n, hoặc bởi sự thay đổi của hai phần tử nhỏ hơn n trong dãy hoán vị gồm $n-1$ phần tử trước đó.
- Trong cả hai trường hợp này, sự khác nhau giữa hai hoán vị trước và sau chỉ là sự hoán đổi vị trí của hai phần tử kề nhau.

Thuật toán Steinhauss–Johnson–Trotter

- Ở bước tiếp theo, phần tử 4 sẽ lần lượt được đặt vào hoán vị $(1, 2, 3)$ theo thứ tự giảm dần, đặt vào hoán vị $(1, 3, 2)$ theo thứ tự tăng dần, đặt vào hoán vị $(3, 1, 2)$ theo thứ tự giảm dần....
- Việc đặt phần tử cứ được tiếp tục theo cách như vậy, luân phiên giữa thứ tự tăng dần và giảm dần, áp dụng cho các giá trị n lớn bất kì.
- Như vậy, mỗi hoán vị đứng sau chỉ khác với hoán vị trước nó hoặc bởi sự di chuyển của phần tử thứ n, hoặc bởi sự thay đổi của hai phần tử nhỏ hơn n trong dãy hoán vị gồm $n-1$ phần tử trước đó.
- Trong cả hai trường hợp này, sự khác nhau giữa hai hoán vị trước và sau chỉ là sự hoán đổi vị trí của hai phần tử kề nhau.

Thuật toán Steinhauss–Johnson–Trotter

- Ở bước tiếp theo, phần tử 4 sẽ lần lượt được đặt vào hoán vị $(1, 2, 3)$ theo thứ tự giảm dần, đặt vào hoán vị $(1, 3, 2)$ theo thứ tự tăng dần, đặt vào hoán vị $(3, 1, 2)$ theo thứ tự giảm dần....
- Việc đặt phần tử cứ được tiếp tục theo cách như vậy, luân phiên giữa thứ tự tăng dần và giảm dần, áp dụng cho các giá trị n lớn bất kì.
- Như vậy, mỗi hoán vị đứng sau chỉ khác với hoán vị trước nó hoặc bởi sự di chuyển của phần tử thứ n, hoặc bởi sự thay đổi của hai phần tử nhỏ hơn n trong dãy hoán vị gồm $n - 1$ phần tử trước đó.
- Trong cả hai trường hợp này, sự khác nhau giữa hai hoán vị trước và sau chỉ là sự hoán đổi vị trí của hai phần tử kề nhau.

Thuật toán Steinhauss–Johnson–Trotter

- Ở bước tiếp theo, phần tử 4 sẽ lần lượt được đặt vào hoán vị $(1, 2, 3)$ theo thứ tự giảm dần, đặt vào hoán vị $(1, 3, 2)$ theo thứ tự tăng dần, đặt vào hoán vị $(3, 1, 2)$ theo thứ tự giảm dần....
- Việc đặt phần tử cứ được tiếp tục theo cách như vậy, luân phiên giữa thứ tự tăng dần và giảm dần, áp dụng cho các giá trị n lớn bất kì.
- Như vậy, mỗi hoán vị đứng sau chỉ khác với hoán vị trước nó hoặc bởi sự di chuyển của phần tử thứ n, hoặc bởi sự thay đổi của hai phần tử nhỏ hơn n trong dãy hoán vị gồm $n - 1$ phần tử trước đó.
- Trong cả hai trường hợp này, sự khác nhau giữa hai hoán vị trước và sau chỉ là sự hoán đổi vị trí của hai phần tử kề nhau.

Thuật toán Steinhauss–Johnson–Trotter

- Ta có thể viết một hàm đệ quy đơn giản để cài đặt thuật toán.
- Nhưng hàm đó là không hiệu quả vì nó có độ phức tạp thời gian và không gian rất lớn:
  - Các cấu trúc hoán vị trung gian bị lặp lại nhiều lần;
  - Để sinh được các hoán vị cấp n thì cần sinh trước các hoán vị cấp $n-1$.
- Vì vậy, ta sẽ cài đặt thuật toán bằng phương pháp lặp thay vì đệ quy.

Thuật toán Steinhauss–Johnson–Trotter

- Steinhauss, Johnson và Trotter độc lập nhau đề xuất thuật toán sinh các hoán vị của n phần tử bằng phương pháp hoán đổi kế tiếp.
- Mỗi phần tử (số nguyên) trong dãy hoán vị được gắn thêm thông tin về hướng di chuyển của nó.
- Hướng di chuyển có thể nhận một trong hai giá trị logic ứng với "trái" và "phải".
- Nếu hướng di chuyển của số k là sang trái thì ta viết $\langle k$; nếu hướng của nó là sang phải thì ta viết $k\rangle$.

Thuật toán Steinhauss–Johnson–Trotter

Một số nguyên có hướng được gọi là *di động* nếu nó lớn hơn số bên cạnh theo chiều di chuyển của nó.

Thuật toán Steinhauss–Johnson–Trotter:

1. Khởi tạo hoán vị đầu tiên là $\langle 1 \quad \langle 2 \quad \cdots \quad \langle n$;
2. Chừng nào còn tồn tại một số nguyên di động:
   - Tìm số nguyên di động k lớn nhất;
   - Hoán đổi k với số nguyên bên cạnh nó theo chiều di chuyển của k;
   - Đảo chiều di chuyển của mọi số nguyên lớn hơn k.

Các hoán vị của ba phần tử được sinh bởi thuật toán này:

$$\begin{bmatrix} \langle 1 & \langle 2 & \langle 3 \\ \langle 1 & \langle 3 & \langle 2 \\ \langle 3 & \langle 1 & \langle 2 \\ 3\rangle & \langle 2 & \langle 1 \\ \langle 2 & 3\rangle & \langle 1 \\ \langle 2 & \langle 1 & 3\rangle \end{bmatrix}$$

Thuật toán Steinhauss–Johnson–Trotter

Các hoán vị của 4 phần tử được sinh bởi thuật toán này như sau:

$$\begin{bmatrix}
\langle 1 & \langle 2 & \langle 3 & \langle 4 \\
\langle 1 & \langle 2 & \langle 4 & \langle 3 \\
\langle 1 & \langle 4 & \langle 2 & \langle 3 \\
\langle 4 & \langle 1 & \langle 2 & \langle 3 \\
4\rangle & \langle 1 & \langle 3 & \langle 2 \\
\langle 1 & 4\rangle & \langle 3 & \langle 2 \\
\langle 1 & \langle 3 & 4\rangle & \langle 2 \\
\langle 1 & \langle 3 & \langle 2 & 4\rangle
\end{bmatrix}
\quad
\begin{bmatrix}
\langle 3 & \langle 1 & \langle 2 & \langle 4 \\
\langle 3 & \langle 1 & \langle 4 & \langle 2 \\
\langle 3 & \langle 4 & \langle 1 & \langle 2 \\
\langle 4 & \langle 3 & \langle 1 & \langle 2 \\
4\rangle & 3\rangle & \langle 2 & \langle 1 \\
3\rangle & 4\rangle & \langle 2 & \langle 1 \\
3\rangle & \langle 2 & 4\rangle & \langle 1 \\
3\rangle & \langle 2 & \langle 1 & 4\rangle
\end{bmatrix}
\quad
\begin{bmatrix}
\langle 2 & 3\rangle & \langle 1 & \langle 4 \\
\langle 2 & 3\rangle & \langle 4 & \langle 1 \\
\langle 2 & \langle 4 & 3\rangle & \langle 1 \\
\langle 4 & \langle 2 & 3\rangle & \langle 1 \\
4\rangle & \langle 2 & \langle 1 & 3\rangle \\
\langle 2 & 4\rangle & \langle 1 & 3\rangle \\
\langle 2 & \langle 1 & 4\rangle & 3\rangle \\
\langle 2 & \langle 1 & 3\rangle & 4\rangle
\end{bmatrix}$$

Bài tập

Bài tập 9. Viết chương trình cài đặt thuật toán Steinhauss–Johnson–Trotter.

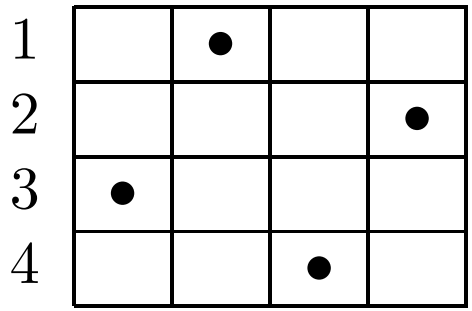
- Input: Một số tự nhiên $n < 20$.
- Output: Mọi hoán vị của dãy $1..n$.

Bài tập 10. Sử dụng chương trình tìm dấu của một hoán vị đã viết ở Bài tập 2 để kiểm tra xem các hoán vị thu được từ chương trình cài đặt thuật toán Steinhauss–Johnson–Trotter có đổi dấu luân phiên hay không.

- Input: Các hoán vị sinh bởi thuật toán S–J–T.
- Output: Các dấu ứng với các hoán vị.

Bài tập

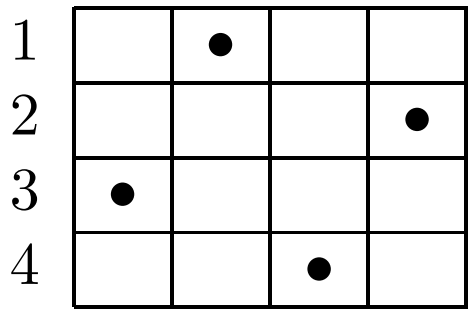
Bài tập 11. (*Bài toán xếp hậu*) Cho một bàn cờ vua kích thước $n \times n$. Hãy tìm mọi cách xếp n quân hậu trên bàn cờ sao cho không quân nào không chế được quân khác.
Ví dụ, với $n = 4$, ta có một cách xếp hậu như sau:



- Mỗi cột của bàn cờ chỉ có thể chứa một quân hậu nên mỗi cách xếp hậu tương ứng với một dãy số nguyên $a[1..n]$, trong đó $a[j] = i$ nếu hậu thứ j được đặt ở hàng thứ i.
- Cách xếp hậu như trên ứng với dãy $a[1..4] = (3, 1, 4, 2)$. Đây là một hoán vị của dãy $(1, 2, 3, 4)$.
- Quy việc tìm cách xếp hậu về việc tìm hoán vị của dãy $(1, 2, \ldots, n)$ thỏa mãn điều kiện của bài toán.

Bài tập

Bài tập 11. (*Bài toán xếp hậu*) Cho một bàn cờ vua kích thước $n \times n$. Hãy tìm mọi cách xếp n quân hậu trên bàn cờ sao cho không quân nào không chế được quân khác.
Ví dụ, với $n = 4$, ta có một cách xếp hậu như sau:



- Mỗi cột của bàn cờ chỉ có thể chứa một quân hậu nên mỗi cách xếp hậu tương ứng với một dãy số nguyên $a[1..n]$, trong đó $a[j] = i$ nếu hậu thứ j được đặt ở hàng thứ i.
- Cách xếp hậu như trên ứng với dãy $a[1..4] = (3, 1, 4, 2)$. Đây là một hoán vị của dãy $(1, 2, 3, 4)$.
- Quy việc tìm cách xếp hậu về việc tìm hoán vị của dãy $(1, 2, \ldots, n)$ thỏa mãn điều kiện của bài toán.

Nội dung

1. Xếp đặt
   - Bài toán 1
   - Bài toán 2
   - Bài toán 3
2. Hoán vị
   - Các khái niệm cơ bản
   - Thuật toán quay lui
   - Thuật toán đệ quy
   - Thuật toán Heap
   - Thuật toán Steinhauss–Johnson–Trotter
3. Tóm lược

Tóm lược

Các nội dung chính của bài giảng:

- Ba bài toán sắp xếp, tính số cách sắp xếp tương ứng
- Các khái niệm cơ bản: hoán vị, nghịch thế, tính chẵn lẻ của hoán vị
- Bốn thuật toán sinh mọi hoán vị của n phần tử:
  1. Thuật toán quay lui
  2. Thuật toán đệ quy
  3. Thuật toán Heap
  4. Thuật toán Steinhauss–Johnson–Trotter
- Các bài tập lập trình để hiểu rõ và củng cố kiến thức