ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

CHUYÊN ĐỀ

TỔ HỢP

NỘI DUNG CHUYÊN ĐỀ

Chương 1. Xếp đặt và Hoán vị

1.1. Hàm và các cách xếp đặt

1.2. Hoán vị của tập hợp

1.3. Một số thuật toán sinh các hoán vị

Bài tập Chương 1

Chương 2. Các tập con của tập hợp

2.1. Các tập con của tập hợp và tập bội

2.2. Các tập con *k*-phần tử

2.3. Sinh các tập con *k*-phần tử

Chương 3. Phân hoạch của tập hợp
3.1. Dàn các phân hoạch của một tập hợp
3.2. Các số Stirling và các số Bell
3.3. Thuật toán sinh phân hoạch
3.4. Sinh các *k*-phân hoạch
3.5. Bài toán phân tích số
Bài tập Chương 3

Chương 4. Một số kỹ thuật giải tích và ứng dụng
4.1. Kỹ thuật hàm sinh
4.2. Nguyên lý bù - trừ
Bài tập Chương 4

TÀI LIỆU THAM KHẢO

1) Hoàng Chí Thành, ***Giáo trình Tổ hợp***, NXB ĐHQG Hà Nội, 1999, 2000, 2001

2) A. Nijenhus and H.S. Wilf, ***Combinatorical Algorithms***, Academic Press - New York, 1975

3) T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, ***Introduction to Algorithms***, The MIT Press, 2003 (có bản dịch tiếng Việt)

Chương 1. XẾP ĐẶT VÀ HOÁN VỊ

Giả sử có *m* "đối tượng" và *n* cái "hộp".

Ta xếp đặt các "đối tượng" này vào các "hộp" theo một số qui tắc khác nhau. Công việc này được gọi là *xếp đặt*.

1.1. Hàm và các cách xếp đặt

Bài toán 1.1: Với *m* "đối tượng" và *n* "cái hộp" trên, hỏi có bao nhiêu cách xếp đặt các "đối tượng" vào các "hộp"?

Một số dạng khác của bài toán

Bài toán tô màu: Có *m* đồ gỗ và *n* màu sơn. Hỏi có bao nhiêu cách sơn màu các đồ gỗ, nếu mỗi đồ gỗ chỉ sơn bằng một màu?

Bài toán du lịch: Một đoàn du lịch gồm m khách đến ở một khách sạn. Quản lý khách sạn dành cho đoàn n phòng. Hỏi có bao nhiêu cách xếp các vị khách vào ở trong các phòng của khách sạn?

Ký hiệu: X - tập các "đối tượng", Y - tập các "hộp".

Mỗi cách xếp đặt của Bài toán 1.1 là một ánh xạ f từ tập X sang tập Y:

$$f \ : \ X_{(m)} \rightarrow Y_{(n)}$$

Bài toán này đưa về việc tìm tất cả các ánh xạ từ tập X sang tập Y. Mỗi ánh xạ f như thế có thể biểu diễn bằng một dãy m thành phần:

$$f \equiv < f(x_1), f(x_2), \ldots, f(x_m) > \text{ với } f(x_i) \in Y, i = 1, 2, \ldots, m. \quad (1.1)$$

Mỗi thành phần của dãy có n cách chọn.

Định lý 1.1: Số tất cả các cách xếp đặt m "đối tượng" vào n "hộp" là n^m.

Mỗi cách xếp đặt có thể viết gọn lại là $< y_1, y_2, \ldots, y_m >$ trong đó y_i là "hộp" chứa "đối tượng" x_i.

Thuật toán 1.2

Đầu vào: Hai số nguyên dương m và n

Đầu ra: Dãy các dãy số $< y_1, y_2, \ldots y_m >$ thỏa mãn (1.1) được sắp tăng dần theo thứ tự từ điển

```
Begin
   input m, n ;
   T[1..m] ← 1 ;
   repeat
      print T[1..m] ;
      i ← m ;
```

```
      while  T[i] = n  do
           { T[i] ← 1 ; i ← i - 1 } ;
      if  i > 0  then  T[i] ← T[i] + 1 ;
   until  i = 0 ;
End.
```

Độ phức tạp của thuật toán: Dễ thấy rằng Thuật toán 1.2 có độ phức tạp là $O(m.n^m)$.

Xét trường hợp các vị khách du lịch có nhiều tiền và yêu cầu không ở chung phòng. Liệu khách sạn có thỏa mãn yêu cầu của họ hay không và có bao nhiêu cách xếp đặt.

Bài toán 1.2: Hỏi có bao nhiêu cách xếp đặt m “đối tượng” vào n “hộp” sao cho mỗi “hộp” chứa không quá một “đối tượng”?

Mỗi cách xếp đặt của Bài toán 1.2 được gọi là *xếp đặt đơn lẻ*. Dãy ảnh $< f(x_1), f(x_2), \ldots, f(x_m) >$ cần thỏa mãn thêm điều kiện:

$$f(x_i) \neq f(x_j) \text{ nếu } 1 \leq i < j \leq m \qquad (1.2)$$

Ánh xạ f trở thành một đơn ánh. Bài toán 1.2 đưa về việc tìm tất cả các đơn ánh $f : X_{(m)} \rightarrow Y_{(n)}$

Dễ thấy rằng có $[n]_m = n(n-1)(n-2)...(n-m+1)$ cách xếp đặt, với quy ước: $[n]_0 = 1$

Số nghiệm của Bài toán 1.2 là tích của m số nguyên liên tiếp bắt đầu từ n và giảm dần.

$$[n]_m = \frac{n!}{(n-m)!}$$

Định lý 1.3: Số tất cả các cách xếp đặt đơn lẻ m "đối tượng" vào n "hộp" là $[n]_m$.

Khi xếp khách du lịch vào ở trong các phòng của khách sạn, trưởng đoàn đã có danh sách các vị khách. Ông ta chỉ việc lần lượt chọn phòng và đọc số phòng theo thứ tự tương ứng với thứ tự các vị khách. Từ đó ta có khái niệm chọn có thứ tự như sau.

Định nghĩa 1.1: *Chọn có thứ tự* được hiểu theo nghĩa sau đây: hai cách chọn là khác nhau nếu hai tập được chọn là khác nhau hoặc hai tập được chọn giống nhau nhưng thứ tự chọn của các phần tử là khác nhau.

Chẳng hạn, khi chọn 3 phần tử trong tập $X = \{a, b, c, d, e\}$ thì các cách chọn có thứ tự sau đây là khác nhau: $(a, b, c) \neq (a, c, e) \neq (a, e, c)$ ...

Bài toán 1.2’: Hỏi có bao nhiêu cách chọn có thứ tự m phần tử trong tập n phần tử?

Bài toán 1.2’ tương đương với Bài toán 1.2. Mỗi cách chọn có thứ tự m phần tử trong tập n phần tử được gọi là một *chỉnh hợp* chập m của n. Số tất cả chỉnh hợp là:

$$A_m^n = [n]_m = \frac{n!}{(n-m)!}$$

Đồng nhất tập $Y \equiv \{1, 2, ..., n\}$.

Khi đó, dãy $< y_1, y_2, \ldots, y_m >$ trở thành một dãy bị chặn giữa hai biên là $\langle \underbrace{1, 1, \ldots, 1}_{m} \rangle$ và $\langle \underbrace{n, n, \ldots, n}_{m} \rangle$ và thỏa mãn:

$$y_i \neq y_j \text{ nếu } 1 \leq i < j \leq m \qquad (1.3)$$

Để tìm các cách xếp đặt đơn lẻ (các cách chọn có thứ tự) ta có thể áp dụng Thuật toán 1.2 để sinh ra các dãy bị chặn, sau đó kiểm tra tính chất (1.3) để loại bỏ những dãy không thỏa mãn.

Việc kiểm tra tính chất (1.3) có thể thực hiện bằng cách kiểm tra xem tập hợp tạo bởi dãy $< y_1, y_2, \ldots, y_m >$ có đúng m phần tử hay không:

$$|\{y_1, y_2, \ldots, y_m\}| = m \qquad (1.4)$$

Áp dụng Thuật toán 1.2 ở trên, ta có thuật toán sinh ra tất cả các cách xếp đặt đơn lẻ.

Thuật toán 1.4

Đầu vào: Hai số nguyên dương m và n ($m \leq n$)

Đầu ra: Dãy các cách xếp đặt đơn lẻ $< y_1, y_2, \ldots\ y_m >$ được sắp tăng theo thứ tự từ điển

```
Begin
    input  m, n  ;
    T[1..m] ← 1 ;
    repeat
        if |{T[1], T[2], ..., T[m]}| = m  then  print T[1..m] ;
        i ← m ;
        while  T[i] = n  do
            { T[i] ← 1 ; i ← i - 1 } ;
        if  i > 0  then  T[i] ← T[i] + 1 ;
    until  i = 0 ;
End.
```

Độ phức tạp của thuật toán: Như Thuật toán 1.2, Thuật toán 1.4 có độ phức tạp là $O(m.n^m)$.

Chú ý: Khi $n = m$ thì ánh xạ f vừa là đơn ánh vừa là toàn ánh. Vậy nó trở thành một song ánh từ X lên Y.

Mỗi song ánh f được gọi là một *hoán vị* trên tập Y. Số các hoán vị này là: $[n]_n = n(n-1)(n-2)... .2.1 = n!$

Hệ quả 1.5: Số các hoán vị trên tập n phần tử là $n!$.

Ta quan tâm đến thứ tự các "đối tượng" nằm trong mỗi "hộp". Thứ tự này tạo nên nhiều cách xếp đặt khác nhau mà ta gọi là xếp đặt có thứ tự.

Định nghĩa 1.2: *Xếp đặt có thứ tự* được hiểu theo nghĩa sau đây: hai cách xếp đặt là khác nhau nếu trong một "hộp" nào đó hai tập phần tử là khác nhau hoặc hai tập phần tử giống nhau nhưng thứ tự của các phần tử lại khác nhau.

Ví dụ 1.3: Lấy các phần tử của tập $X = \{a, b, c, d, e, f\}$ xếp đặt vào 3 hộp. Các cách xếp đặt có thứ tự sau đây là khác nhau:

Hộp 1	Hộp 2	Hộp 3
$a\ b$	$c\ d$	$e\ f$
$a\ b$	$c\ d\ e$	f
$a\ b$	$e\ d\ c$	f

Bài toán 1.3: Hỏi có bao nhiêu cách xếp đặt có thứ tự *m* "đối tượng" vào *n* "hộp"?

Ký hiệu $[n]^m$ là số các cách xếp đặt có thứ tự *m* "đối tượng" vào *n* "hộp".

Không thể dùng ánh xạ để biểu diễn cách xếp đặt có thứ tự vì ánh xạ không diễn đạt được thứ tự của các "đối tượng" trong "hộp".

Ta xếp dần các "đối tượng" vào các "hộp".

- "Đối tượng" thứ nhất có *n* cách xếp đặt.

- "Đối tượng" thứ hai có $n+1$ cách xếp đặt: xếp vào n-1 "hộp" còn trống hoặc xếp nó vào "hộp" đang chứa "đối tượng" thứ nhất nhưng đặt trước hoặc đặt sau.

- Trước khi xếp đặt “đối tượng” thứ ba thì có hai tình huống xảy ra: hai “đối tượng” thứ nhất và thứ hai cùng nằm trong một “hộp” hoặc nằm ở hai “hộp” khác nhau.

* Tình huống thứ nhất cho: n-1 + 3 = n+2 cách xếp đặt “đối tượng” thứ ba.

* Tình huống thứ hai cũng cho: n-2 + 2 + 2 = n+2 cách xếp đặt “đối tượng” thứ ba.
Vậy có n+2 cách xếp đặt cho “đối tượng” thứ ba.

- Giả sử ta đã xếp đặt được k-1 “đối tượng” vào các “hộp” và “hộp” thứ i đang chứa r_i “đối tượng”, với i = 1, 2, ..., n.

Hiển nhiên, $r_1 + r_2 + ... + r_n = k$-1. Khi đó sẽ có r_i+1 cách xếp đặt "đối tượng" thứ k vào "hộp" thứ i. Số các cách xếp đặt "đối tượng" thứ k vào các "hộp" là: $(r_1+1) + (r_2+1) + ... + (r_n+1) = r_1 + r_2 + ... + r_n + n = n+k-1$.

- Tương tự, số các cách xếp đặt "đối tượng" thứ m vào các "hộp" trên là $n+m-1$.

Định lý 1.6: Số các cách xếp đặt có thứ tự m "đối tượng" vào n "hộp" là:

$$[n]^m = n(n+1)(n+2)...(n+m-1), \text{ với quy ước: } [n]^0 = 1.$$

Tổng kết lại, ta có các công thức đơn giản sau:

1) $[n]_m = A_m^n = \dfrac{n!}{(n-m)!}$

2) $[n]_m = (n - m + 1).[n]_{m-1}$

3) $[n]^m = [n+m-1]_m$

Chúng ta đã trình bày ba bài toán xếp đặt với độ "khó" tăng dần như sau:

	Bài toán 3.1	Bài toán 3.2	Bài toán 3.3
Số lượng nghiệm	n^m	$[n]_m$	$[n]^m$
Biểu diễn ánh xạ	$f : X_{(m)} \longrightarrow Y_{(n)}$	$f : X_{(m)} \xrightarrow{1-1} Y_{(n)}$	?

1.2. Hoán vị của tập hợp

Nếu bài toán sắp xếp biến một dãy các phần tử chưa được sắp thành một dãy được sắp theo một thứ tự nào đó thì bài toán hoán vị làm công việc ngược lại. Từ một dãy được sắp, bài toán hoán vị cho ta tất cả các cách đảo lộn các phần tử. Bài toán hoán vị cho ta tất cả các cách liệt kê các phần tử của một tập hợp.

1.2.1. Các định nghĩa

Giả sử X là một tập hợp n phần tử.

Định nghĩa 1.4: *Hoán vị* trên tập X là một cách liệt kê tất cả các phần tử của tập này.

Bài toán 1.2 về xếp đặt đơn lẻ đã chỉ ra rằng mỗi hoán vị trên tập X chính là một song ánh từ tập X vào chính nó.

Bài toán 1.3 về xếp đặt có thứ tự cũng chỉ ra rằng mỗi một cách xếp đặt có thứ tự n "đối tượng" của tập X vào một "hộp" chính là một hoán vị trên tập X.

Ký hiệu S_n là tập tất cả các hoán vị trên tập n phần tử.

Ví dụ 1.5: Các hoán vị của tập $X = \{1, 2, 3\}$ bao gồm:

1 2 3	2 3 1
1 3 2	3 1 2
2 1 3	3 2 1

1.2.2. Các cách biểu diễn hoán vị

Ta đồng nhất tập $X \equiv \{1, 2, \ldots, n\}$.

Mỗi hoán vị $f \in S_n$ có thể được biểu diễn bằng dãy số nguyên độ dài n như sau:

$f = \langle f(1)\, f(2) \ldots f(n) \rangle$, với $f(i) \in X$ và $f(i) \neq f(j)$ nếu $1 \leq i \neq j \leq n$

Để đơn giản ký hiệu người ta thường viết:

$f = \langle a_1\, a_2 \ldots a_n \rangle$, với $a_i \in X$ và $a_i \neq a_j$ nếu $1 \leq i \neq j \leq n$

- *Biểu diễn ma trận*: Ta biểu diễn hoán vị *f* bằng một ma trận 2 hàng *n* cột.

$$f = \begin{pmatrix} 1 & 2 \ldots\ldots\ldots & n-1 & n \\ f(1) & f(2)\ \ldots & f(n-1) & f(n) \end{pmatrix}$$

- *Biểu diễn dãy số*: Hoán vị *f* được biểu diễn bằng một dãy các ảnh của các phần tử.

$$f = < f(1)\, f(2)\, \ldots\, f(n) >$$

Ví dụ 1.6: Một hoán vị trên tập 4 phần tử.

$$g = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix} \quad \text{hoặc} \quad g = < 4\ 3\ 1\ 2 >$$

1.2.3. Tích các hoán vị

Mỗi hoán vị trên tập X là một song ánh từ X vào chính nó, nên ta có thể áp dụng phép toán tích ánh xạ trên các hoán vị này.

Giả sử f và g là hai hoán vị nào đó trên tập X. Khi đó, ánh xạ tích $f.g$ được định nghĩa như sau:

$$\forall x \in X : \quad f.g\,(x) = f\,(g\,(x))$$

Vì tích của hai song ánh là một song ánh nên $f.g$ trở thành một hoán vị trên tập X và ta gọi nó là *tích của hai hoán vị f và g.*

Tập S_n các hoán vị trên tập X đã được trang bị phép toán tích.

- Phép toán này là kết hợp do tính kết hợp của tích các ánh xạ.

- Ánh xạ đồng nhất: $e = < 1\ 2\ 3\ \ldots\ n\text{-}1\ n >$ trở thành đơn vị của phép toán trên.

- Mỗi hoán vị $f \in S_n$ được biểu diễn dưới dạng ma trận thì nghịch đảo của nó là hoán vị ứng với ma trận nhận được từ ma trận biểu diễn f và đổi hai hàng cho nhau.
  Chẳng hạn,

$$g = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix} \quad \text{thì} \quad g^{-1} = \begin{pmatrix} 4 & 3 & 1 & 2 \\ 1 & 2 & 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 \end{pmatrix}$$

Vậy tập S_n trở thành một nhóm và được gọi là *nhóm các hoán vị bậc n*.

1.2.4. Biểu diễn đồ thị và kiểu của hoán vị

Giả sử f là một hoán vị nào đó trên tập X.

Ta xây dựng đồ thị có hướng $G = (V, E)$ như sau:

- tập đỉnh $V = X$
- tập cạnh $E = \{ x \to f(x) \mid x \in V \}$.

Đồ thị G được gọi là *biểu diễn đồ thị* của hoán vị f.

Với mỗi đỉnh x của đồ thị G chỉ có duy nhất một cung đi ra là $x \to f(x)$ và một cung đi vào là $f^{-1}(x) \to x$.

Vì đồ thị G hữu hạn nên xuất phát từ một đỉnh nào đó x_1 ta đi qua các đỉnh $x_2 = f(x_1)$, $x_3 = f(x_2)$, …và sau một số hữu hạn bước ta lại quay về đỉnh xuất phát $f(x_k) = x_1$.

Đường đi như thế tạo thành chu trình $[x_1\ x_2\ \dots\ x_{k\text{-}1}\ x_k]$ gồm k đỉnh và k cung nên được gọi là một chu trình độ dài k. Từ chu trình này ta xây dựng hoán vị c như sau:

$$c(x_1) = x_2\ ,\ c(x_2) = x_3\ ,\ \dots\ ,\ c(x_{k\text{-}1}) = x_k\ ,\ c(x_k) = x_1 \text{ và}$$

$$c(x) = x\ ,\ \text{với } x \in X \setminus \{x_1, x_2, \dots, x_k\}$$

Hoán vị c được gọi là hoán vị chu trình. Đồ thị G của hoán vị f bao gồm một số chu trình rời nhau nên hoán vị f có thể biểu diễn thành tích của các hoán vị chu trình rời nhau:

$$f = c_{1,1}.c_{1,2}\dots c_{1,\lambda 1}.\ c_{2,1}.c_{2,2}\dots c_{2,\lambda 2}\dots\ c_{n,1}.c_{n,2}\dots c_{n,\lambda n}$$

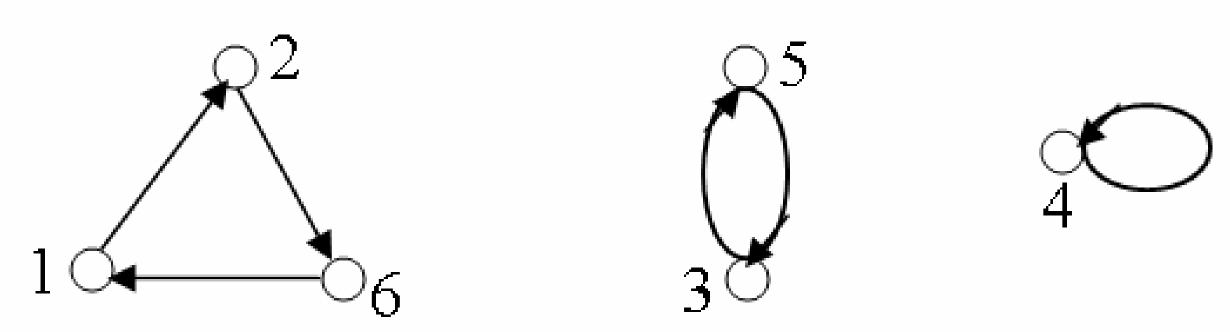
Số các chu trình có cùng độ dài cho ta biết kiểu của hoán vị đó.

Định nghĩa 1.7: Hoán vị f có kiểu là $< \lambda_1, \lambda_2, ..., \lambda_n >$ nếu trong biểu diễn chu trình của nó có λ_i chu trình độ dài *i*, với $i = 1, 2, 3,..., n$.

Vì $\lambda_1.1 + \lambda_2.2 + ...+ \lambda_n.n = n$ nên sẽ có nhiều thành phần λ_i bằng 0. Do vậy, kiểu của hoán vị f còn được viết ngắn gọn dưới dạng $1^{\lambda 1}2^{\lambda 2}...n^{\lambda n}$ và bỏ qua không viết $i^{\lambda i}$ nếu $\lambda_i = 0$.

Ví dụ 1.8: Cho một hoán vị trên tập 6 phần tử $h = < 2\ \ 6\ \ 5\ \ 4\ \ 3\ \ 1 >$.

Đồ thị có hướng biểu diễn hoán vị *h* là:



Hoán vị *h* có thể biểu diễn thành tích của các hoán vị chu trình:

$$h = [4] \,.\, [3\ 5] \,.\, [1\ 2\ 6]$$

Kiểu của hoán vị *h* là $1^1 2^1 3^1$

1.2.5. Dấu của hoán vị

Định nghĩa 1.9: Cặp (a_i, a_j), với $i < j$ được gọi là một *nghịch thế* của hoán vị $< a_1\, a_2 \ldots a_n >$ nếu $a_i > a_j$.

Số các nghịch thế của hoán vị *f* được ký hiệu là $I(f)$.

Định nghĩa 1.10: Dấu của hoán vị *f*, ký hiệu là sign(*f*), được định nghĩa như sau:

$$\mathrm{sign}(f) = (-1)^{I(f)}$$

Nếu hoán vị f có số nghịch thế chẵn thì dấu của nó là 1 và ta gọi nó là *hoán vị chẵn*. Ngược lại, hoán vị f có số nghịch thế lẻ thì dấu của nó là -1 và ta gọi nó là *hoán vị lẻ*.

Ví dụ 1.11: Cho hoán vị $h = \langle 2 \quad 6 \quad 5 \quad 4 \quad 3 \quad 1 \rangle$. Hoán vị này có các nghịch thế sau đây:

(6,5), (6,4), (5,4), (6,3), (5,3), (4,3), (2,1), (6,1), (5,1), (4,1) và (3,1).

Vậy $I(h) = 11$ và h là hoán vị lẻ, mang dấu trừ.

Khái niệm dấu của hoán vị chia tập S_n các hoán vị của một tập hợp thành hai tập con: tập S_n^+ chứa các hoán vị chẵn và tập S_n^- chứa các hoán vị lẻ.

Việc tính dấu của một hoán vị dựa vào Định nghĩa 1.10 đòi hỏi một thuật toán có độ phức tạp là $O(n^2)$. Ta có thể xây dựng một thuật toán tuyến tính để tìm được cả kiểu và dấu của một hoán vị.

Một số kết quả bổ trợ

Trong số các hoán vị của một tập hợp ta đặc biệt quan tâm tới hoán vị tạo bởi việc đổi chỗ hai phần tử nào đó của tập hợp. Hoán vị này có dạng:

$$t = < 1\ 2\ \ldots\ i\text{-}1\ \boldsymbol{j}\ i\text{+}1\ \ldots\ j\text{-}1\ \boldsymbol{i}\ j\text{+}1\ \ldots\ n\text{-}1\ n >$$

Hoán vị t được gọi là *phép đổi chỗ*. Trong biểu diễn đồ thị của t chỉ có duy nhất một chu trình độ dài 2. Do vậy, có thể viết gọn lại $t = [i\ \ j]$.

Nếu $j = i+1$ thì hai phần tử đứng cạnh nhau đổi chỗ cho nhau, nên ta gọi $t = [i\ \ i+1]$ là *phép đổi chỗ kế tiếp*.

Bổ đề sau đây cho ta một biểu diễn chi tiết của mỗi hoán vị.

Bổ đề 1.7: Mọi hoán vị f đều có thể biểu diễn qua tích của $I(f)$ phép đổi chỗ kế tiếp.

Chứng minh: Giả sử $f = < a_1\ a_2\ \ldots\ a_n >$ là một hoán vị nào đó trên tập X và $t = [i\ \ i{+}1]$ là một phép đổi chỗ kế tiếp. Thế thì:

$$f.t = < a_1\ a_2\ \ldots a_{i+1}\ a_i\ \ldots\ a_n >$$

Ký hiệu l_i - số nghịch thế sinh bởi phần tử $i \in X$. Nghĩa là:

$$l_i = |\{\ j \mid i = a_k\ ,\ j < k\ \&\ a_j > a_k\ \}|$$

Hiển nhiên, $I(f) = \sum_{i=1}^{n} l_i$

Ta dẫn hoán vị f về hoán vị đơn vị e nhờ các phép đổi chỗ kế tiếp.

- Đưa phần tử 1 về vị trí đầu tiên nhờ l_1 phép đổi chỗ kế tiếp: $< \underline{a_1 a_2} .... \underline{a_k} 1 ... a_n >$ ta nhận được hoán vị $< 1 \underline{a_2 a_3} ... \underline{a_l} 2 .. a_n >$.

- Đưa phần tử 2 về vị trí thứ hai nhờ l_2 phép đổi chỗ kế tiếp ta nhận được hoán vị $< 1\ 2\ a_3 \ldots a_n >$.

- Cứ tiếp tục như thế thì sau: $l_1 + l_2 + ... + l_n = I(f)$ phép đổi chỗ kế tiếp ta dẫn được hoán vị f về hoán vị đơn vị e. Nghĩa là: $f.t_1.t_2. \ldots t_{I(f)} = e$. Vậy thì:

$$f = (t_1.t_2. \ldots t_{I(f)})^{-1} = t_{I(f)}^{-1} \ldots t_2^{-1}.t_1^{-1}$$

$$f = t_{I(f)} \ldots t_2.t_1$$

Định lý 1.8: Với mọi cặp hoán vị $f, g \in S_n$ thì:

$$\text{sign}(f.g) = \text{sign}(f) \cdot \text{sign}(g)$$

Chứng minh:

Trước hết, giả sử $f = < a_1\ a_2\ \ldots\ a_i\ a_{i+1}\ \ldots\ a_n >$ và g là một phép đổi chỗ kế tiếp $t = [i\ \ i+1]$. Khi đó thì: $f.t = < a_1\ a_2\ \ldots\ a_{i+1}\ a_i\ \ldots\ a_n >$. Hiển nhiên:

$$I(f.t) = \begin{cases} I(f)+1 & \text{, nếu } a_i < a_{i+1} \\ I(f)-1 & \text{, ngược lại} \end{cases}$$

Tính chẵn lẻ của $I(f.t)$ và $I(f)$ đổi nhau. Vậy $\text{sign}(f.t) = -\ \text{sign}(f)$.

Trong trường hợp tổng quát, theo Bổ đề 1.7 ta có thể viết:

$$g = t_1.t_2.\ \ldots\ t_{I(g)}$$

Vậy thì:

$\mathrm{sign}(f.g) = \mathrm{sign}((f.t_1.t_2.\ \ldots\ ).t_{I(g)}) = -\ \mathrm{sign}(f.t_1.t_2.\ \ldots\ t_{I(g)-1}) = \ldots$

$= (-1)^{I(g)}\mathrm{sign}(f) = \mathrm{sign}(f)\ .\ \mathrm{sign}(g)$

Như vậy, phép tích các hoán vị bảo toàn dấu của hoán vị.

Tục ngữ Việt Nam có câu: "*Con hơn cha là nhà có phúc*" nói lên cái hay của khái niệm nghịch thế.

Định lý 1.9:

1) Phép đổi chỗ luôn là hoán vị lẻ.

2) Tổng quát, hoán vị chu trình độ dài k có dấu là $(-1)^{k-1}$.

Chứng minh:

1) Phép đổi chỗ $t = < 1\ 2\ \ldots\ i\text{-}1\ \boldsymbol{j}\ i\text{+}1\ \ldots\ j\text{-}1\ \boldsymbol{i}\ j\text{+}1\ \ldots\ n\text{-}1\ n >$

Các nghịch thế tạo nên bởi sự đổi chỗ của hai phần tử i với j và là:

$(j\ i+1), (j\ i+2), \ldots, (j\ j-1), (j\ i)$ và $(i+1\ i), (i+2\ i), \ldots, (j-1\ i)$

Số nghịch thế của phép đổi chỗ t là: $I(t) = (j - i) + (j - i - 1) = 2(j - i) - 1$.
Vậy phép đổi chỗ t là hoán vị lẻ.

2) Một hoán vị chu trình có k đỉnh:

$c_k = [a_{i1}, a_{i2}, \ldots, a_{ik}]$, $f(a_{i1}) = a_{i2}, f(a_{i2}) = a_{i3}, \ldots , f(a_{ik\text{-}1}) = a_{ik}\,, f(a_{ik}) = a_{i1}$.

Hoán vị chu trình trên có thể biểu diễn qua tích của k-1 phép đổi chỗ:

$$c_k = [a_{i1}\ a_{i2}].[a_{i2}\ a_{i3}]\ \dots\ [a_{ik-1}\ a_{ik}]$$

Theo Định lý 1.8, $\mathrm{sign}(c_k) = \underbrace{(-1)(-1)\dots(-1)}_{k-1} = (-1)^{k-1}$.

Định lý 1.10: Hoán vị f có kiểu < $\lambda_1, \lambda_2, \dots, \lambda_n$ > thì có dấu là:

$$\mathrm{sign}(f) = (-1)^{\lambda_2+\lambda_4+\dots+\lambda_{2[n/2]}}$$

Chứng minh:

Hoán vị f được biểu diễn thành tích của các hoán vị chu trình:

$$f = c_{1,1}.c_{1,2}\dots c_{1,\lambda 1}.\ c_{2,1}.c_{2,2}\dots c_{2,\lambda 2}\dots\ c_{n,1}.c_{n,2}\dots c_{n,\lambda n}$$

Vậy:

$$\begin{aligned}\operatorname{sign}(f) &= \operatorname{sign}(c_{1,1}.c_{1,2}\ldots c_{1,\lambda 1}.\ c_{2,1}.c_{2,2}\ldots c_{2,\lambda 2}\ldots\ c_{n,1}.c_{n,2}\ldots c_{n,\lambda n})\\ &= \operatorname{sign}(c_{1,1}).\operatorname{sign}(c_{1,2})\ldots\operatorname{sign}(c_{1,\lambda 1}).\\ &\quad \operatorname{sign}(c_{2,1}).\operatorname{sign}(c_{2,2})\ldots\operatorname{sign}(c_{2,\lambda 2})\ldots\\ &\quad \operatorname{sign}(c_{n,1}).\operatorname{sign}(c_{n,2})\ldots\operatorname{sign}(c_{n,\lambda n})\end{aligned}$$

$$= \underbrace{(-1)\ldots(-1)}_{\lambda_2}\underbrace{(-1)\ldots(-1)}_{\lambda_4}\ldots\underbrace{(-1)\ldots(-1)}_{\lambda_{2[n/2]}}$$

$$= (-1)^{\lambda_2+\lambda_4+\ldots+\lambda_{2[n/2]}}$$

1.2.6. Thuật toán tuyến tính tìm kiểu và dấu của hoán vị

Trước tiên, các đỉnh được đánh dấu là ‘mới” nhờ mảng logic *MOI*[1..*n*].

Thuật toán sẽ “đi qua” các cung để tới từng đỉnh của mỗi chu trình trong hoán vị.

Khi “đi qua” cung nào thì nó đổi dấu của biến *d* (trừ cung đầu tiên của mỗi chu trình) và tới đỉnh nào thì nó đổi dấu đỉnh này thành “cũ” để không quay lại đỉnh này nữa. Kỹ thuật này được gọi là cờ hiệu (*flag*).

Đồng thời, nó đếm số các cạnh để xác định độ dài của mỗi chu trình, từ đó tìm được kiểu của hoán vị và lưu vào mảng *LAMDA*[1..*n*].

Thuật toán 1.11

Đầu vào: Hoán vị $f \in S_n$

Đầu ra: Dấu $d = \text{sign}(f)$ và kiểu $< \lambda_1, \lambda_2, ..., \lambda_n >$ của hoán vị f lưu trong mảng $LAMDA[1..n]$

```
Begin
   input  n ;
   for  i ← 1  to  n  do { P[i] ← f(i) ; MOI[i] ← true ;
                                        LAMDA[i] ← 0 } ;
   d ← 1 ;
   for  i ← 1 to  n  do
      if  MOI[i]  then
         { j ← P[i]  ;  k ← 1 ;
            while  j ≠ i  do
```

9 $\{\ MOI[j] \leftarrow \text{false}\ ;\ j \leftarrow P[j]\ ;\ d \leftarrow -d\ ;\ k \leftarrow k+1\ \}$
;

10 $LAMDA[i] \leftarrow LAMDA[i] + 1\ \}$;

11 print d và $LAMDA[1..n]$;

12 **End.**

Độ phức tạp của Thuật toán 1.11:

Chu trình 3 có độ phức tạp là O(n).

Các câu lệnh 5-10 là hai chu trình lồng nhau. Song nhờ kỹ thuật cờ hiệu và câu lệnh 6 thuật toán chỉ tới "thăm" mỗi đỉnh một lần. Tại mỗi đỉnh chỉ thực hiện 4 thao tác (câu lệnh 9). Do vậy, độ phức tạp của chu trình 5-10 là O(n).

Vậy độ phức tạp tổng thể của Thuật toán 1.11 là O(n).

1.3. Một số thuật toán sinh các hoán vị

Bài toán hoán vị:

Cho tập X gồm n phần tử. Hãy tìm tất cả các hoán vị của tập X.

Số hoán vị của tập n phần tử là $n!$. Số n càng lớn thì thời gian tìm hết các hoán vị càng lớn.

Để đơn giản ký hiệu, dãy số nguyên trên được viết gọn lại là:

$$f = < a_1 a_2 \ldots a_n >, \text{ với } a_i = f(i), \; i = 1, 2, \ldots, n.$$

1.3.1. Thuật toán từ điển sinh các hoán vị

Giả sử X là một tập hợp n phần tử. Đồng nhất tập $X \equiv \{1, 2, ..., n\}$.

Khi đó, mỗi hoán vị trên tập X là một dãy số độ dài n bao gồm tất cả các số trong tập X. Mỗi dãy số trên có thể xem như là một từ trên bảng chữ cái X.

Vì các từ sinh bởi hoán vị có độ dài bằng nhau nên thứ tự từ điển trên các từ này được định nghĩa ngắn gọn như sau:

Định nghĩa 1.12: Giả sử $x_1 x_2 \ldots x_n$ và $y_1 y_2 \ldots y_n$ là hai từ có cùng độ dài n trên bảng chữ cái X. Ta nói rằng,

$x_1 x_2 \ldots x_n < y_1 y_2 \ldots y_n$ theo thứ tự từ điển $\Leftrightarrow$

$$\exists k \le n: \;\; \forall t < k,\, x_t = y_t \;\; \wedge \;\; x_k < y_k$$

Ví dụ 1.13: Các hoán vị của tập 2, 3 phần tử được sắp xếp tăng dần theo thứ tự từ điển như sau:

$n = 2$	$n = 3$
1 2	1 2 3
2 1	1 3 2
	2 1 3
	2 3 1
	3 1 2
	3 2 1

Dãy các hoán vị của tập n phần tử được sắp xếp tăng theo thứ tự từ điển có các tính chất lý thú sau đây:

1) Hoán vị đầu tiên (nhỏ nhất) là hoán vị đồng nhất: 1 2 3 … n-1 n và hoán vị cuối cùng (lớn nhất) là: n n-1 ... 2 1 và là đảo ngược của hoán vị đầu tiên.

2) Dãy n! hoán vị này có thể chia thành n dãy con có độ dài (n-1)! tương ứng với giá trị tăng dần của thành phần đầu tiên.

Khi bỏ thành phần đầu tiên q trong dãy con thứ q ta được dãy các hoán vị của tập {1, 2, …, q-1, q+1, …, n-1, n} cũng vẫn được sắp tăng theo thứ tự từ điển.

Xuất phát từ hoán vị đầu tiên, thuật toán vào vòng lặp để sinh ra các hoán vị còn lại.

Giả sử $t = a_1a_2 \dots a_{n-1}a_n$ là từ vừa tìm được.

Ta cần tìm từ $t' = a'_1a'_2 \dots a'_{n-1}a'_n$ kế tiếp từ t trong dãy được sắp tăng.

				(j)			
t	a_1	...	a_{j-1}	a_j	a_{j+1}	...	a_n
t'	a_1	...	a_{j-1}	a'_j	a'_{j+1}	...	a'_n

Theo thứ tự từ điển, vị trí thay đổi j là chỉ số lớn nhất mà $a_j < a_{j+1}$. Thế thì:

$$j = \max \{ i \mid 1 \le i \le n\text{-}1 \ \& \ a_i < a_{i+1} \}$$

Từ cần tìm t' kế thừa phần bên trái của từ t từ a_1 đến a_{j-1}.

Dễ thấy rằng, dãy con $< a_{j+1}, a_{j+2}, ..., a_n >$ là một dãy giảm.

Phần thay đổi trong t' từ vị trí thứ j đến vị trí n được xác định như sau:

1) a'_j sẽ là phần tử nhỏ nhất trong dãy con $< a_{j+1}, a_{j+2}, ..., a_n >$ nhưng lớn hơn a_j.

Ta đổi chỗ a_j với phần tử nhỏ nhất này:

$$a_j \leftrightarrow \min \{ a_i \mid j+1 \le i \le n \ \& \ a_i > a_j \}$$

Lưu ý rằng, sau khi tráo đổi dãy con $< a_{j+1}, a_{j+2}, ..., a_m >$ vẫn là dãy giảm.

2) Đảo ngược dãy con $< a_{j+1}, a_{j+2}, ..., a_n >$, biến nó thành dãy tăng và là dãy bé nhất trong số các hoán vị của tập $\{a_{j+1}, a_{j+2}, ..., a_n\}$.

Đó chính là dãy con $< a'_{j+1}, a'_{j+2}, ..., a'_n >$ trong t' cần tìm.

Vòng lặp kết thúc khi hoán vị lặp cuối cùng n n-1 ... 2 1 được sinh ra. Với hoán vị này thì vị trí thay đổi: $j = 0$

Nghĩa là đã sinh hết các hoán vị. Đây chính là điều kiện kết thúc của thuật toán.

Dùng mảng nguyên $P[1..n]$ chứa các hoán vị.

Thuật toán 1.12 (*Sinh các hoán vị theo thứ tự từ điển*)
Đầu vào: Số nguyên n
Đầu ra: Dãy tất cả các hoán vị của tập $\{1, 2, ..., n\}$ được sắp tăng theo thứ tự từ điển

```
Begin
   input  n ;
   for  i ← 1  to  n  do  P[i] ← i  ;
   repeat
       print P[1..n] ;
       p ← n - 1 ;
    while  P[p] >= P[p+1]  do  p ← p - 1;
       if  p > 0 then
          {  i ← p +1 ;
             while  P[i] > P[p] do  i ← i +1 ;
             Đổi chỗ P[p] với P[i-1]  ;
```

```
12              Đảo ngược dãy con P[p+1..n] ; }
13      until  p = 0 ;
14  End .
```

Độ phức tạp của thuật toán:

- Các câu lệnh 2-3 nhập dữ liệu đầu vào và hoán vị đầu tiên có độ phức tạp là O(n).
- Vòng lặp 5-12 tính toán và in ra một hoán vị, trong đó:

Câu lệnh 5 in ra một hoán vị với độ phức tạp là O(n).

Vòng lặp 7 tìm vị trí thay đổi j với độ phức tạp là O(n).

Các câu lệnh 10-11 tìm phần tử a'_j và đổi chỗ với a_j với độ phức tạp là O(n).

Câu lệnh 12 đảo ngược dãy con $< a_{j+1}, a_{j+2}, ..., a_n >$ với độ phức tạp là O(n). Vậy việc sinh và in ra một hoán vị có độ phức tạp là O(n).

Độ phức tạp tổng thể của Thuật toán 1.12 là O($n.n!$).

Ví dụ 1.14 (Sắp đặt công việc tối ưu)

Giả sử có n công việc cần thực hiện với thời gian thực hiện như nhau, chẳng hạn là một ngày. Các công việc được thực hiện lần lượt, không ngắt quãng công việc đang thực hiện và tại một thời điểm chỉ có một công việc được thực hiện.

Với công việc e_i ($i = 1, 2,..., n$) thì:

- d_i là thời hạn phải thực hiện xong,
- x_i là số tiền thưởng cho mỗi ngày do thực hiện xong công việc e_i trước hạn
- y_i là số tiền phạt tính theo ngày do không thực hiện công việc này đúng hạn.

Hãy xác định thứ tự thực hiện các công việc để tổng số tiền nhận được là lớn nhất.

Tập các công việc trên có thể biểu diễn như một tập hợp n phần tử X.

Mỗi cách sắp đặt các công việc chính là một hoán vị $< a_1\ a_2\ ...\ a_n >$ của tập này.

Xây dựng hàm số xác định số tiền thưởng/phạt khi thực hiện công việc e_i là:

$$tp(e_i) = \begin{cases} (d_i - j)x_i & \text{, nếu } e_i = a_j \text{ và } j < d_i \\ (d_i - j)y_i & \text{, nếu } e_i = a_j \text{ và } j > d_i \end{cases}$$

Với cách sắp đặt các công việc như trên thì tổng số tiền nhận được là:

$$tg = \sum_{i=1}^{n} tp(e_i)$$

Bài toán sắp đặt công việc tối ưu ở trên đưa về bài toán tìm hoán vị có tổng số tiền nhận được *tg* là lớn nhất.

Kết hợp Thuật toán 1.12 sinh các hoán vị và thuật toán tìm số lớn nhất ta có thuật toán tối ưu để sắp đặt các công việc.

Chẳng hạn, xét tập các công việc được cho như trong bảng dưới đây.

Công việc e_i	Thời hạn d_i	Tiền thưởng x_i	Tiền phạt y_i
1	2	10	5
2	3	12	8
3	3	6	9
4	4	20	2
5	2	14	7

Tính toán trên máy tính theo thuật toán trên ta nhận được phương án sắp đặt tối ưu các công việc là: 4 2 3 5 1 với tổng số tiền thưởng cao nhất là 43.

1.3.2. Thuật toán từ điển ngược sinh các hoán vị

Thứ tự từ điển ngược trên các từ được định nghĩa như sau.

Định nghĩa 4.12: Giả sử $x_1 x_2 \ldots x_n$ và $y_1 y_2 \ldots y_n$ là hai từ có cùng độ dài n trên bảng chữ cái X. Ta nói rằng,

$$x_1 x_2 \ldots x_n <^{N} y_1 y_2 \ldots y_n \text{ theo thứ tự từ điển ngược } \Leftrightarrow$$
$$\exists k \geq 1 : \quad \forall p > k \,,\, x_p = y_p \quad \& \quad x_k > y_k$$

Thứ tự từ điển ngược so sánh các cặp chữ cái của hai từ từ phải sang trái cho đến khi gặp cặp chữ cái khác nhau, rồi kết luận: từ trên nhỏ hơn từ dưới nếu chữ cái tương ứng của từ trên lớn hơn chữ cái tương ứng của từ dưới và ngược lại.

Ví dụ 1.15: Các hoán vị của tập 2, 3 phần tử được sắp xếp tăng dần theo thứ tự từ điển ngược như sau:

$n = 2$	$n = 3$
1 2	1 2 3
2 1	2 1 3
	1 3 2
	3 1 2
	2 3 1
	3 2 1

Dãy các hoán vị của tập n phần tử được sắp xếp tăng dần theo thứ tự từ điển ngược cũng có các tính chất lý thú như với thứ tự từ điển:

1) Hoán vị đầu tiên là hoán vị đồng nhất 1 2 3 … n-1 n và hoán vị cuối cùng là đảo ngược của hoán vị đầu tiên.

2) Dãy n! hoán vị này có thể chia thành n dãy con có độ dài (n-1)! tương ứng với giá trị giảm dần của thành phần cuối cùng.

Khi bỏ thành phần cuối cùng q trong dãy con thứ n-q+1 ta được dãy các hoán vị của tập {1, 2, …, q-1, q+1, …, n-1, n} cũng vẫn được sắp tăng theo thứ tự từ điển ngược.

Hai tính chất trên của thứ tự từ điển và thứ tự từ điển ngược gợi ý cho ta xây dựng một thuật toán đệ quy để sinh các hoán vị. Nhưng ta chọn thứ tự nào để xây dựng thuật toán dễ dàng.

Nếu dùng thứ tự từ điển thì cần hai tham số đệ quy:

- Tham số *n* là số phần tử của tập hợp tương ứng với cột bên phải, tham số này giảm dần về phía trái.
- Tham số *t* chỉ thành phần đầu tiên tương ứng với cột bên trái, tham số này tăng dần về phía phải.

Nếu ta dùng thứ tự từ điển ngược thì cũng cần hai tham số đệ quy:

- Tham số *n* là số phần tử của tập hợp tương ứng với cột bên phải, tham số này giảm dần về phía trái.
- Tham số *p* chỉ thành phần cuối cùng tương ứng với cột bên phải, tham số này cũng giảm dần về phía trái.

Hai tham số *n* và *p* trên là trùng nhau. Vậy với thứ tự từ điển ngược ta chỉ cần một tham số đệ quy.

Thuật toán 1.13

Đầu vào: Số nguyên dương n

Đầu ra: Dãy các hoán vị của tập n phần tử được sắp tăng theo thứ tự từ điển ngược

```
procedure DAO(m) ;
  begin
    tr ← 1 ; ph ← m ;
    while  tr < ph  do
      { P[tr] ↔ P[ph] ;
        tr ← tr + 1 ;
        ph ← ph - 1 }
  end ;

procedure TD_NGUOC(m) ;
  begin
```

```
    if  m = 1  then  print  P[1..n]
        else  for  i ← 1  to  m  do
                  {  TD_NGUOC(m-1) ;
                     if  i < m  then
                             { P[i] ↔ P[m] ;
                               DAO(m-1)  }  }
    end ;

Begin
  input  n ;
  for  i ← 1  to  n  do  P[i] ← i  ;
  TD_NGUOC(n) ;
End .
```

Tính đúng đắn của thuật toán:

Để chứng minh tính đúng đắn của Thuật toán 1.13 ta cần chỉ ra bằng phương pháp quy nạp theo n rằng:

Nếu $P[1] < P[2] < \ldots < P[m]$ thì lời gọi thủ tục *TD_NGUOC*(m) sinh đúng đắn tất cả các hoán vị của tập $\{P[1], P[2], \ldots, P[m]\}$ và không làm thay đổi giá trị của $P[m+1], P[m+2], \ldots, P[n]$.

a) Với $n = 2, 3$ thì Ví dụ 1.15 đã chỉ ra điều đó.

b) Đặt $a_i = P[i]$ với $i = 1,2,\ldots,n$. Thế thì: $a_1 < a_2 < \ldots < a_n$.

Xét chu trình 12-16. Kết quả thực hiện vòng lặp đầu tiên của chu trình này như sau:

Sau khi thực hiện:	$P[1]$	$P[2]$	…	$P[m\text{-}2]$	$P[m\text{-}1]$	$P[m]$
TD_NGUOC(m-1)	a_1	a_2	…	a_{m-2}	a_{m-1}	a_m
	…	…	…	…	…	…
	a_{m-1}	a_{m-2}	…	a_2	a_1	a_m
$P[i] \leftrightarrow P[m]$	a_m	a_{m-2}	…	a_2	a_1	a_{m-1}
DAO(m-1)	a_1	a_2	…	a_{m-2}	a_m	a_{m-1}

Theo giả thiết quy nạp, lời gọi thủ tục *TD_NGUOC*(m-1) sinh đúng đắn tất cả các hoán vị của tập { $P[1]$, $P[2]$, …, $P[m-1]$ } theo thứ tự từ điển ngược. Vậy vòng lặp đầu tiên sinh đúng đắn $(m-1)!$ hoán vị của tập $\{a_1, a_2, \dots, a_m\}$.

Bắt đầu vòng lặp thứ hai ta lại có: $P[1] < P[2] < \dots < P[m-1]$ (vì $a_1 < a_2 < \dots < a_{m-2} < a_m$). Tương tự như trên, ta chỉ ra rằng vòng lặp này sinh đúng đắn (m-1)! hoán vị của tập $\{a_1, a_2, ..., a_m\}$ theo thứ tự từ điển ngược.

Tiếp tục như thế, ta chứng minh được rằng vòng lặp thứ i sinh đúng đắn (m-1)! hoán vị của tập $\{a_1, \ldots, a_{m-i}, a_{m-i+2}, ..., a_m\}$ theo thứ tự từ điển ngược và $P[m] = a_{m-i+1}$.

Điều đó chỉ ra tính đúng đắn của giả thiết quy nạp.

1.3.3. Thuật toán đổi chỗ kế tiếp sinh các hoán vị

Thuật toán này do S.M. Johnson xây dựng vào năm 1963 với ý tưởng:

Hoán vị sau được tạo bởi hoán vị trước đó nhờ một phép đổi chỗ kế tiếp.

Ví dụ 1.16: Sinh các hoán vị của tập 3 phần tử $X_{(3)} = \{1, 2, 3\}$.

1) Trước hết sinh các hoán vị của tập $X_{(2)} = \{2, 3\}$ là:

$$< 2\ 3 > \text{ và } < 3\ 2 >$$

Sau đó chèn phần tử 1 vào hai hoán vị này và nhận được đủ 6 hoán vị của tập 3 phần tử là:

1 2 3	1 3 2
2 1 3	3 1 2
2 3 1	3 2 1

2) Xuất phát từ hoán vị đồng nhất < 1 2 3 > ta có thể nhận được dãy hoán vị trên bằng cách cho phần tử 1 di chuyển sang phải một vị trí đến khi chạm biên thì quay hướng di chuyển của 1 sang trái, cho 2 di chuyển một vị trí, sau đó cho 1 tiếp tục di chuyển cho đến khi nhận được hoán vị cuối cùng (cột 2).

3) Để tạo nên sự di chuyển của phần tử ta dùng phép đổi chỗ kế tiếp và nhận được dãy hoán vị giống như hai kết quả trên (cột 3).

Chèn một phần tử	Di chuyển một phần tử	Đổi chỗ kế tiếp
1 2 3	1 2 3	1↔2 3
2 1 3	2 1 3	2 1↔3
2 3 1	2 3 1	2 ↔3 1
3 2 1	3 2 1	3 2↔1
3 1 2	3 1 2	3↔1 2
1 3 2	1 3 2	1 3 2

Cùng một kết quả có thể thực hiện bằng ba cách khác nhau.

Giả sử $X_{(n)} = \{1, 2, ..., n\text{-}1, n\}$ là tập n phần tử. Để sinh ra dãy tất cả các hoán vị của tập này ta thực hiện thuật toán đệ quy:

1) Sinh dãy các hoán vị của tập $X_{(n-1)} = \{2, 3, ..., n\text{-}1, n\}$ mà hoán vị sau tạo bởi hoán vị trước đó nhờ một phép đổi chỗ kế tiếp.

2) Chèn phần tử 1 vào các hoán vị trong dãy vừa tìm được bằng mọi cách có thể.

Thuật toán đệ quy đòi hỏi bộ nhớ rất lớn.

Sử dụng cả ba cách thực hiện như trong Ví dụ 1.16 ta xây dựng phương án lặp cho thuật toán này như sau:

1) Nhập số n
2) Chọn hoán vị đầu tiên là hoán vị đồng nhất $< 1\ 2\ \ldots\ n\text{-}1\ n >$
3) Vòng lặp:
   - Tìm vị trí k để đổi chỗ kế tiếp
   - Đổi chỗ: $P[k] \leftrightarrow P[k+1]$
4) Lặp lại 3) cho đến khi không còn vị trí đổi chỗ.

Các biến mảng được dùng:

- mảng nguyên $P[1..n]$ để lưu hoán vị vừa tìm được,
- mảng logic $PHAI[1..n]$ để chỉ hướng di chuyển của các phần tử. Phần tử n không di chuyển. Phần tử i di chuyển sang phải khi $PHAI[i]$ = true và sang trái khi $PHAI[i]$ = false.

- Mảng nguyên $CHO[1..n]$ để lưu vị trí tương đối của các phần tử. Biến nguyên $CHO[i]$ cho biết vị trí tương đối của phần tử i trong tập hợp $\{i, i+1, ..., n-1, n\}$. Vị trí tương đối của phần tử i được đánh số từ 1 và tăng dần tới giá trị lớn nhất là $n-i+1$.

Phần tử di chuyển là phần tử nhỏ nhất chưa chạm biên.

Phần tử i chạm biên khi và chỉ khi vị trí tương đối của nó là lớn nhất:

$$CHO[i] = n - i + 1$$

Khi tìm được phần tử di chuyển i ta phải xác định vị trí k của nó trong mảng P để tiến hành đổi chỗ kế tiếp.

- Tất cả các phần tử nhỏ hơn i đang chạm biên.

- Ta phải quay hướng di chuyển và đánh số lại vị trí tương đối để tạo cơ hội cho các phần tử này di chuyển.
- Trong số các phần tử này, các phần tử chạm biên bên trái đã đẩy các phần tử trong tập $\{i, i+1, ..., n-1, n\}$ sang phải.
- Vậy vị trí của phần tử di chuyển *i* trong mảng *P* được tính bằng vị trí tương đối của nó trong tập $\{i, i+1, ..., n-1, n\}$ cộng với số các phần tử nhỏ hơn *i* đang chạm biên bên trái.

Xác định được vị trí *k* của phần tử di chuyển *i*, ta cho phần tử đó di chuyển nhờ phép đổi chỗ kế tiếp: $P[k] \leftrightarrow P[k+1]$ và nhận được hoán vị mới.

Thuật toán dừng khi sinh được hoán vị mà tất cả các phần tử của nó đều chạm biên.

Thuật toán 4.14

Đầu vào: Số nguyên dương n

Đầu ra: Dãy các hoán vị của tập hợp $\{1, 2, \ldots, n\}$ mà hoán vị sau tạo bởi hoán vị trước đó nhờ một phép đổi chỗ kế tiếp

```
Begin
    input n ;
    for i ← 1 to n do
            { P[i] ← i ; CHO[i] ← 1 ; PHAI[i] ← true } ;
    CHO[n] ← 0 ;
    print P[1..n] ;
    i ← 1 ;
    while i < n do
      { i ← 1 ; x ← 0 ;
        while CHO[i] = n - i + 1 do
```

```
11              { PHAI[i] ← not PHAI[i] ;   CHO[i] ← 1 ;
12                if  PHAI[i]  then  x ← x + 1 ;
13                i ← i + 1 } ;
14        if i < n  then
15              if PHAI[i] then k ← x + CHO[i]
                         else k ← x + n - i+1 - CHO[i] ;
16        P[k] ↔ P[k+1] ;
17        print P[1..n] ;
18        CHO[i] ← CHO[i] + 1 } ;
19   End .
```

Độ phức tạp của thuật toán:

Chu trình 10-13 có độ phức tạp là O(n).

Độ phức tạp tổng thể của Thuật toán 1.14 là O($n!.n$).

Ví dụ 1.17: Tính toán của Thuật toán 1.14 với $n = 4$.

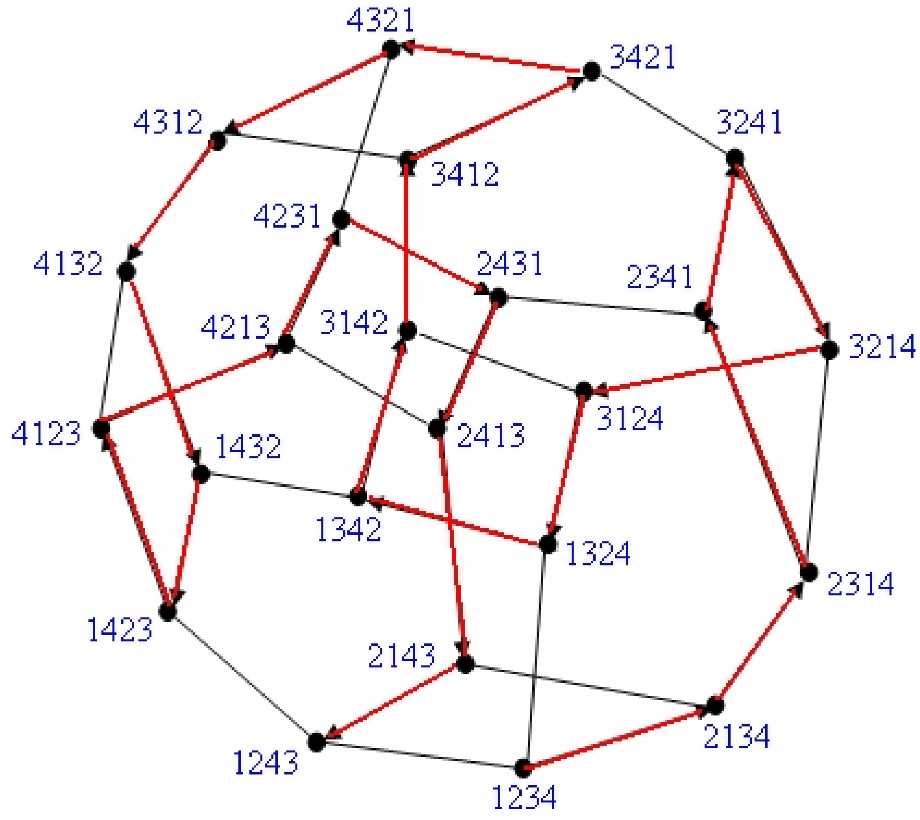
***P*[1]**	***P*[2]**	***P*[3]**	***P*[4]**	PH_1	PH_2	PH_3	PH_4	CHO_1	CHO_2	CHO_3	CHO_4	i	x	k
1	2	3	4	true	true	true	true	1	1	1	0			
2	1	3	4	true	true	true	true	2	1	1	0	1	0	1
2	3	1	4	true	true	true	true	3	1	1	0	1	0	2
2	3	4	1	true	true	true	true	4	1	1	0	1	0	3
3	2	4	1	false	true	true	true	1	2	1	0	2	0	1
3	2	1	4	false	true	true	true	2	2	1	0	1	0	3
3	1	2	4	false	true	true	true	3	2	1	0	1	0	2
1	3	2	4	false	true	true	true	4	2	1	0	1	0	1
1	3	4	2	true	true	true	true	1	3	1	0	2	1	3
3	1	4	2	true	true	true	true	2	3	1	0	1	0	1
3	4	1	2	true	true	true	true	3	3	1	0	1	0	2
3	4	2	1	true	true	true	true	4	3	1	0	1	0	3
4	3	2	1	false	false	true	true	1	1	2	0	3	0	1
4	3	1	2	false	false	true	true	2	1	2	0	1	0	3
4	1	3	2	false	false	true	true	3	1	2	0	1	0	2

1	4	3	2	false	false	true	true	4	1	2	0	1	0	1
1	4	2	3	true	false	true	true	1	2	2	0	2	1	3
4	1	2	3	true	false	true	true	2	2	2	0	1	0	1
4	2	1	3	true	false	true	true	3	2	2	0	1	0	2
4	2	3	1	true	false	true	true	4	2	2	0	1	0	3
2	4	3	1	false	false	true	true	1	3	2	0	2	0	1
2	4	1	3	false	false	true	true	2	3	2	0	1	0	3
2	1	4	3	false	false	true	true	3	3	2	0	1	0	2
1	2	4	3	false	false	true	true	4	3	2	0	1	0	1
				true	true	false	true	1	1	2	0	**4**	2	

Ý nghĩa đồ thị của thuật toán:

Xây dựng đồ thị vô hướng $G_n = (V, E)$ với tập đỉnh V là tập các hoán vị của tập n phần tử. Hai đỉnh tương ứng với hai hoán vị f và g, được nối với nhau bằng một cạnh $(f, g) \in E$ khi và chỉ khi g được tạo bởi f nhờ một phép đổi chỗ kế tiếp.

Dãy các hoán vị được sinh ra bởi Thuật toán 1.14 tạo nên một đường đi Hamilton trên đồ thị G_n.



BÀI TẬP CHƯƠNG 1

1.1. Có 10 vận động viên tham gia một cuộc thi thể thao. Hỏi có bao nhiêu cách trao tặng huy chương vàng, bạc và đồng?

1.2. Có bao nhiêu số nguyên nằm giữa 1000 và 10000 tạo bởi các chữ số lẻ, có bao nhiêu số tạo bởi các chữ số khác nhau?

1.3. Chứng minh rằng, số các cách xếp cho n trong số m vị khách ngồi cạnh một cái bàn tròn là $\frac{[m]_n}{n}$. Chú ý rằng, hai cách xếp chỉ khác nhau bởi sự dịch chuyển vòng tròn mọi người xung quanh bàn được xem là như nhau.

1.4. Có 5 người đứng thành hàng ngang để chụp ảnh. Hỏi có bao nhiêu cách đứng khác nhau để chụp?

1.5. Chứng minh rằng tập các hoán vị chẵn của một tập hợp tạo thành một nhóm còn tập các hoán vị lẻ thì không.

1.6. Chứng minh rằng, số các hoán vị $f \in S_n$ có cùng kiểu $1^{\lambda_1} 2^{\lambda_2} \ldots n^{\lambda_n}$ là:

$$\frac{n!}{1^{\lambda_1} 2^{\lambda_2} \ldots n^{\lambda_n} \lambda_1! \lambda_2! \ldots \lambda_n!}$$

Gợi ý: Vẽ một đồ thị n đỉnh, với i = 1, 2, ..., *n thì có* λ_i *chu trình độ dài i. Các chu trình này rời nhau. Xếp đặt có thứ tự các số nguyên* 1, 2, ..., *n trên các đỉnh này. Mỗi cách xếp đặt cho một hoán vị. Tính xem một hoán vị trùng với bao nhiêu hoán vị khác bằng cách xoay tròn từng chu trình và hoán vị các chu trình cùng độ dài.*

1.7. Chứng minh rằng, hai hoán vị $f, g \in S_n$ có cùng một kiểu khi và chỉ khi tồn tại một hoán vị $h \in S_n$ sao cho $g = h.f.h^{-1}$.

Gợi ý: Chứng minh rằng t.f.t bảo toàn chu trình của f với t = [*i i*+1] *là phép đổi chỗ kế tiếp. Xét trường hợp i và i*+1 *cùng nằm trên một chu trình và trường hợp chúng nằm trên hai chu trình*.

1.8. Hoán vị $f \in S_n$ được gọi là xoắn ốc nếu $f.f = e$. Chứng minh rằng hoán vị $f \in S_n$ là xoắn ốc khi và chỉ khi nó có kiểu là $1^{\lambda_1} 2^{\lambda_2}$.

1.9. Chứng minh rằng, mọi số nguyên m mà $0 \le m \le n! - 1$ đều có thể biểu diễn duy nhất dưới dạng:

$$m = \sum_{i=1}^{n-1} k_i i!, \text{ với } 0 \le k_i \le i.$$

Đồng thời các dãy số < $k_{n-1}\ k_{n-2}\ \ldots\ k_2\ k_1$ > tương ứng với các số liên tiếp xuất hiện theo thứ tự từ điển. Xây dựng thuật toán sinh các dãy số này.

1.10. Có hai đội cờ vua *A* và *B* thi đấu với nhau. Mỗi đội có *n* đấu thủ. Mỗi đối thủ của đội này chỉ đấu một trận với một đối thủ của đội kia và ngược lại. Biết rằng, trình độ của các đối thủ ở hai đội lần lượt là a_i và b_i, $i = 1, 2, \ldots, n$ và trong thi đấu đấu thủ nào có trình độ cao hơn thì thắng còn trình độ ngang nhau thì hòa. Đội thắng một trận được 2 điểm, hòa được 1 điểm và thua được 0 điểm. Cho đội *B* được quyền chọn cặp thi đấu. Hãy xây dựng thuật toán để đội *B* đạt số điểm cao nhất.