

Chương 4. MỘT SỐ KỸ THUẬT GIẢI TÍCH VÀ ỨNG DỤNG

Khi giải quyết một bài toán học búa nào đó ta thường hay nghĩ đến việc áp dụng các **công cụ khoa học hiện đại, phức tạp** mà ít chú ý tới các **công cụ khoa học cổ điển, đơn giản**.

Nếu biết vận dụng đúng chỗ các công cụ khoa học đơn giản thì nhiều bài toán học búa có thể được giải quyết một cách nhanh chóng.

Chương này trình **bày hai kỹ thuật giải tích đơn giản** hay được dùng khi giải các bài toán tổ hợp.

Đó là:

- Kỹ thuật hàm sinh
- Nguyên lý bù - trừ

4.1. Kỹ thuật hàm sinh

4.1.1. Hàm sinh

Giả sử có một dãy số thực vô hạn: $a_0, a_1, a_2, a_3, \dots$. Ta lập chuỗi lũy thừa:

$$\sum_{n=0}^{\infty} a_n x^n \quad (4.1)$$

Định nghĩa 4.1: Nếu chuỗi lũy thừa (4.1) hội tụ đến một hàm số $A(x)$ nào đó thì ta gọi hàm $A(x)$ là *hàm sinh* của dãy số $a_0, a_1, a_2, a_3, \dots$

Chẳng hạn,

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$$

$$\sum_{n=0}^{\infty} \frac{x^n}{n!} = e^x$$

Như vậy, hàm sinh của dãy số $1, 1, 1, \dots$ là hàm số $\frac{1}{1-x}$, còn hàm sinh của dãy số $\frac{1}{0!}, \frac{1}{1!}, \frac{1}{2!}, \dots$ là hàm mũ e^x .

Giả sử $A(x) = \sum_{n=0}^{\infty} a_n x^n$ và $B(x) = \sum_{n=0}^{\infty} b_n x^n$ là hai chuỗi lũy thừa nào đó.

a) **Phép cộng**

$$A(x) + B(x) = \sum_{n=0}^{\infty} (a_n + b_n) x^n$$

b) **Phép nhân vô hướng**

$$p.A(x) = \sum_{n=0}^{\infty} (p.a_n) x^n$$

c) Tích Cauchy

$$A(x).B(x) = \sum_{n=0}^{\infty} c_n x^n, \text{ với } c_n = a_0 b_n + a_1 b_{n-1} + a_2 b_{n-2} + \dots + a_n b_0 = \sum_{i=0}^n a_i b_{n-i}$$

Trong nhiều trường hợp ta chưa biết dãy số nhưng bằng những lý luận hợp lý ta lại biết hàm sinh của nó. Liệu từ hàm sinh có tìm được dãy số sinh ra nó hay không?

Nhìn vào đẳng thức: $A(x) = \sum_{n=0}^{\infty} a_n x^n$ ta dễ dàng thấy rằng đây là khai triển Maclaurine của hàm $A(x)$. Vậy:

$$a_n = \frac{1}{n!} \frac{d^n}{dx^n} A(x) \Big|_{x=0}, \quad n=0, 1, 2, 3, \dots \quad (4.2)$$

Công thức (4.2) cho ta một cách tìm dãy số từ hàm sinh của nó.

Hàm sinh cho một số dãy số đơn giản:

1) Dãy các đại lượng tổ hợp: $\binom{n}{0}, \binom{n}{1}, \binom{n}{2}, \dots$

$$A(x) = \sum_{k=0}^{\infty} \binom{n}{k} x^k = \sum_{k=0}^n \binom{n}{k} x^k = (1+x)^n$$

Vậy hàm sinh của dãy số: $\binom{n}{0}, \binom{n}{1}, \binom{n}{2}, \dots$ là hàm $(1+x)^n$.

2) **Dãy các lũy thừa của 2** là: $1, 2, 4, \dots, 2^k, \dots$

Ta có:

$$A(x) = \sum_{k=0}^{\infty} 2^k x^k = \sum_{k=0}^{\infty} (2x)^k = \frac{1}{1-2x}$$

Vậy hàm sinh của dãy số: $1, 2, 4, \dots, 2^k, \dots$ là hàm $\frac{1}{1-2x}$

3) **Dãy các số nguyên không âm**: $0, 1, 2, 3, \dots$

$$\begin{aligned} A(x) &= \sum_{k=0}^{\infty} k \cdot x^k = x \cdot \sum_{k=0}^{\infty} k \cdot x^{k-1} = x \cdot \frac{d}{dx} \sum_{k=0}^{\infty} x^k \\ &= x \cdot \frac{d}{dx} \frac{1}{1-x} = \frac{x}{(1-x)^2} \end{aligned}$$

Vậy hàm sinh của dãy số: $0, 1, 2, 3, \dots$ là hàm $\frac{x}{(1-x)^2}$

4.1.2. Một số ứng dụng của hàm sinh

1. Bài toán tập con bội k phần tử

Cho tập bội $X = (k_1 * x_1, k_2 * x_2, \dots, k_n * x_n)$ và k là một số nguyên không âm. Bài toán đặt ra là: **Hỏi có bao nhiêu tập con bội k phần tử của tập bội X ?**

Ký hiệu: m_k là số các tập con bội k phần tử của tập bội X , với $k = 0, 1, 2, 3, \dots$

Ta xây dựng hàm sinh $M(x)$ cho dãy số này:

$$M(x) = \sum_{k=0}^{\infty} m_k x^k$$

Như ta đã biết, nếu X là tập hợp n phần tử thông thường thì $m_k = \binom{n}{k}$ và hàm sinh của dãy số này là: $M(x) = (1 + x)^n$.

Hàm này có thể viết lại thành tích của n thừa số $(1 + x)$ là:

$$M(x) = \underbrace{(1 + x)(1 + x) \dots (1 + x)}_n = \underbrace{(x^0 + x^1)(x^0 + x^1) \dots (x^0 + x^1)}_n$$

Mỗi thừa số $x^0 + x^1$ tương ứng với một phần tử của tập X và **các số mũ lũy thừa của x chỉ sự xuất hiện của phần tử này trong các tập con** (0 lần hoặc 1 lần).

Vậy đơn thức $1 + x$ biểu diễn sự xuất hiện của mỗi phần tử trong các tập con.

Trở lại với tập bội X ở trên.

Nếu phần tử x_1 có bội là k_1 thì nó có thể xuất hiện trong các tập con bội là 0 lần, 1 lần, 2 lần, ... hoặc k_1 lần.

Vậy đa thức biểu diễn sự xuất hiện của phần tử x_1 trong các tập con bội là: $1 + x + x^2 + \dots + x^{k_1}$.

Tương tự như thế cho các phần tử x_2, x_3, \dots, x_n . Vậy hàm sinh của dãy $\{m_k\}_{k=0}^\infty$ là:

$$M(x) = (1 + x + x^2 + \dots + x^{k_1}).(1 + x + x^2 + \dots + x^{k_2}) \dots (1 + x + x^2 + \dots + x^{k_n})$$

Nhân các đa thức trên ta tìm được các hệ số m_k , đó chính là số các tập con bội k phần tử của tập bội X .

Ví dụ 4.2: Cho tập bội $X = (3 * x_1, 1 * x_2, 4 * x_3, 2 * x_4)$.

Hàm sinh cho dãy số $\{m_k\}_{k=0}^{\infty}$ là:

$$\begin{aligned}
 M(x) &= \sum_{k=0}^{\infty} m_k x^k = (1 + x + x^2 + x^3) \cdot (1 + x) \cdot (1 + x + x^2 + x^3 + x^4) \cdot (1 + x + x^2) \\
 &= 1 + 4x + 9x^2 + 15x^3 + 20x^4 + 22x^5 + 20x^6 + 15x^7 + 9x^8 + 4x^9 + x^{10}
 \end{aligned}$$

Vậy tập bội trên có:

1	tập con bội 0 phần tử
4	tập con bội 1 phần tử
9	tập con bội 2 phần tử
15	tập con bội 3 phần tử
20	tập con bội 4 phần tử
22	tập con bội 5 phần tử
20	tập con bội 6 phần tử
15	tập con bội 7 phần tử
9	tập con bội 8 phần tử
4	tập con bội 9 phần tử
1	tập con bội 10 phần tử

2. Chứng minh một số đẳng thức tổ hợp

Dùng kỹ thuật hàm sinh để chứng minh một số đẳng thức tổ hợp.

a) Đẳng thức Cauchy:
$$\binom{m+n}{k} = \sum_{s=0}^k \binom{m}{s} \binom{n}{k-s}$$

Chứng minh:

$$\begin{aligned} \sum_{k=0}^{m+n} \binom{m+n}{k} x^k &= (1+x)^{m+n} = (1+x)^m (1+x)^n \\ &= \sum_{i=0}^m \binom{m}{i} x^i \cdot \sum_{j=0}^n \binom{n}{j} x^j = \sum_{k=0}^{m+n} \left(\sum_{s=0}^k \binom{m}{s} \binom{n}{k-s} \right) x^k \end{aligned}$$

b) Số tập con bội k phần tử của tập bội $(k_1 * x_1, k_2 * x_2, \dots, k_n * x_n)$ với $k_i \geq k$ ($i = 1, 2, \dots, n$) là bằng $\binom{n+k-1}{k}$

Chứng minh:

Bội của các phần tử trong tập X đều lớn hơn k , nghĩa là sự xuất hiện của các phần tử trong các tập con bội là không hạn chế. Vậy thì hàm sinh:

$$\begin{aligned} M(x) &= \sum_{k=0}^{\infty} m_k x^k = (1 + x + x^2 + \dots)(1 + x + x^2 + \dots) \dots (1 + x + x^2 + \dots) \\ &= \underbrace{(1-x)^{-1} (1-x)^{-1} \dots (1-x)^{-1}}_n = (1-x)^{-n} \end{aligned}$$

Tính đạo hàm bậc k của hàm $(1 - x)^{-1}$ ta nhận được:

$$\frac{d^k}{dx^k} (1 - x)^{-n} = (-n)(-n-1)\dots(-n-k+1)(-1)^k (1 - x)^{-n-k} = [n]^k (1 - x)^{-n-k}$$

Khai triển Maclaurine hàm sinh $M(x)$ ta nhận được:

$$M(x) = (1 - x)^{-n} = \sum_{k=0}^{\infty} \frac{[n]^k}{k!} x^k = \sum_{k=0}^{\infty} \binom{n+k-1}{k} x^k$$

Nghĩa là: $m_k = \binom{n+k-1}{k}$, đó là điều phải chứng minh.

c) Với mọi số nguyên dương n thì số phân tích lẻ của n bằng số các phân tích khác nhau từng đôi của số này.

Chứng minh:

Hàm sinh của dãy số các phân tích khác nhau từng đôi:

$$K(x) = (1 + x)(1 + x^2)(1 + x^3) \dots (1 + x^i) \dots$$

Hàm sinh của dãy số các phân tích lẻ:

$$L(x) = (1 - x)^{-1}(1 - x^3)^{-1} \dots (1 - x^{2i-1})^{-1} \dots$$

Sử dụng đẳng thức $1 + x^i = \frac{1 - x^{2i}}{1 - x^i}$ ta có:

$$K(x) = \frac{1 - x^2}{1 - x} \cdot \frac{1 - x^4}{1 - x^2} \cdot \frac{1 - x^6}{1 - x^3} \dots = \frac{1}{1 - x} \cdot \frac{1}{1 - x^3} \cdot \frac{1}{1 - x^5} \dots = L(x)$$

Hai hàm sinh trùng nhau thì các hệ số tương ứng phải bằng nhau.

3. Giải hệ phương trình đệ quy tuyến tính

Hệ phương trình đệ quy tuyến tính bao gồm một số phương trình đệ quy tuyến tính với các hệ số cố định.

Để đơn giản trình bày ta áp dụng kỹ thuật hàm sinh cho dãy số Fibonacci.

Dãy số Fibonacci $\{F_n\}_{n=0}^{\infty}$ được định nghĩa đệ quy như sau:

$$F_0 = F_1 = 1, F_n = F_{n-1} + F_{n-2} \text{ với } n \geq 2 \quad (4.3)$$

Liệu có thể tìm được biểu diễn cho dãy số Fibonacci dưới dạng hàm số:

$$F_n = f(n) \text{ với } n = 0, 1, 2, 3, \dots$$

Xây dựng hàm sinh:

$$F(x) = \sum_{n=0}^{\infty} F_n x^n$$

Theo định nghĩa (8.3) ta có:

$$\begin{aligned} F(x) &= 1 + x + \sum_{n=2}^{\infty} (F_{n-1} + F_{n-2}) x^n \\ &= 1 + x + x \cdot \left(\sum_{n=0}^{\infty} F_n x^n - 1 \right) + x^2 \cdot \sum_{n=0}^{\infty} F_n x^n \\ &= 1 + x \cdot F(x) + x^2 \cdot F(x) \end{aligned}$$

Hàm sinh của dãy số Fibonacci:

$$F(x) = \frac{1}{1 - x - x^2}$$

Khai triển Maclaurine hàm số trên ta sẽ tìm được các số Fibonacci. Mẫu số của hàm là một tam thức bậc hai có hai nghiệm là:

$$a = \frac{1 + \sqrt{5}}{2}, \quad b = \frac{1 - \sqrt{5}}{2}$$

Sử dụng phương pháp hệ số bất định để tìm biểu diễn:

$$F(x) = \frac{1}{1 - x - x^2} = \frac{A}{1 - a.x} + \frac{B}{1 - b.x}$$

Ta nhận được:

$$F(x) = \sum_{n=0}^{\infty} \frac{a^{n+1} - b^{n+1}}{a - b} x^n$$

Vậy thì:

$$F_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{n+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{n+1} \right], \quad n = 0, 1, 2, 3...$$

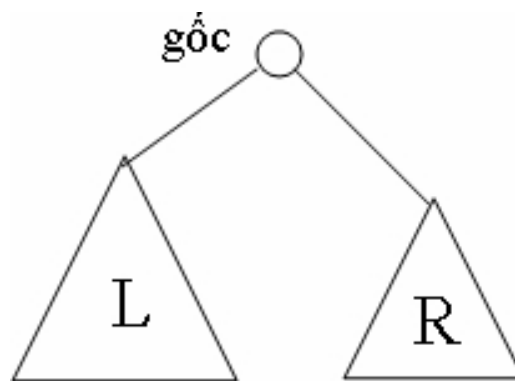
Ta đã biểu diễn được các số Fibonacci dưới dạng hàm số. Hay nói một cách khác là ta đã giải được hệ phương trình đệ quy tuyến tính (8.3).

Một điều lý thú là: các số Fibonacci là các số nguyên nhưng lại có thể biểu diễn qua các số vô tỷ. Đây là cơ sở để [A. Mitiasevich giải được trọn vẹn Bài toán thứ 10 của Hilbert vào năm 1972](#) khi anh ta mới tròn 27 tuổi và đang làm nghiên cứu sinh tại Đại học Leningrad (Thành phố Sant Peterbourg, Nga).

Áp dụng kỹ thuật trên ta có thể giải được nhiều hệ phương trình đệ quy tuyến tính khác.

4. Bài toán cây nhị phân

Bài toán: Cho trước n đỉnh trên mặt phẳng. Hỏi có thể xây dựng được bao nhiêu cây nhị phân từ các đỉnh này.


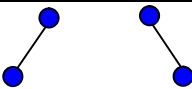
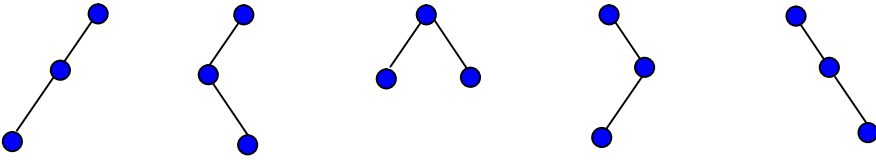


Trong mọi cây nhị phân n đỉnh, ta luôn có đẳng thức:

$$n = 1 + l + r$$

trong đó: l là số đỉnh của cây con trái và r là số đỉnh của cây con phải.

Ký hiệu: c_n là số cây nhị phân n đỉnh ($n = 0, 1, 2, 3, \dots$).

n	c_n	Các cây nhị phân
0	1	
1	1	
2	2	
3	5	
...

Tập các cây nhị phân n đỉnh có thể phân thành các lớp theo số đỉnh của cây con trái $l = 0, 1, 2, \dots, n-1$. Vậy thì:

$$c_n = c_0 c_{n-1} + c_1 c_{n-2} + c_2 c_{n-3} + \dots + c_{n-1} c_0, \quad n \geq 1 \quad (4.4)$$

Xây dựng hàm sinh: $C(x) = \sum_{n=0}^{\infty} c_n x^n$

Theo tính chất (4.4) ta có:

$$\begin{aligned} C(x) &= 1 + \sum_{n=1}^{\infty} (c_0 c_{n-1} + c_1 c_{n-2} + \dots + c_{n-1} c_0) x^n \\ &= 1 + x \cdot \sum_{n=0}^{\infty} (c_0 c_n + c_1 c_n + \dots + c_n c_0) x^n = 1 + x \cdot C^2(x) \end{aligned}$$

Ta nhận được phương trình bậc hai đối với $C(x)$ là:

$$x \cdot C^2(x) + C(x) + 1 = 0$$

Giải phương trình này ta nhận được **hàm sinh**:

$$C(x) = \frac{1 \pm \sqrt{1-4x}}{2x} \quad (4.5)$$

Để khai triển Maclaurine hàm $C(x)$ ta chỉ cần khai triển hàm $\sqrt{1-4x}$.

Với $n > 0$ thì đạo hàm cấp n của hàm số trên là:

$$\begin{aligned} \frac{d^n}{dx^n} (1-4x)^{\frac{1}{2}} &= \frac{1}{2} \left(\frac{1}{2} - 1 \right) \dots \left(\frac{1}{2} - n + 1 \right) (1-4x)^{\frac{1}{2}-n} (-1)^n \\ &= -2^n . 1.3.5 \dots (2n-3) (1-4x)^{\frac{1}{2}-n} \end{aligned}$$

Vậy thì:

$$\begin{aligned}\sqrt{1-4x} &= 1 - \sum_{n=1}^{\infty} \frac{2^n \cdot 1 \cdot 3 \cdot 5 \dots (2n-3)}{n!} x^n \\ &= 1 - \sum_{n=1}^{\infty} \frac{2^n (2n-2)!}{n! \cdot 2 \cdot 4 \cdot 6 \dots (2n-2)} x^n \\ &= 1 - 2 \sum_{n=1}^{\infty} \frac{(2n-2)!}{n! (n-1)!} x^n \\ &= 1 - 2 \sum_{n=1}^{\infty} \frac{1}{n} \binom{2n-2}{n-1} x^n\end{aligned}$$

Hàm sinh $C(x)$ là hàm dương nên trong công thức nghiệm (4.5) ta phải lấy dấu trừ và nhận được:

$$C(x) = \frac{1 - \sqrt{1-4x}}{2x} = \sum_{n=1}^{\infty} \frac{1}{n} \binom{2n-1}{n-1} x^{n-1} = \sum_{n=0}^{\infty} \frac{1}{n+1} \binom{2n}{n} x^n$$

Vậy số các cây nhị phân n đỉnh là:

$$c_n = \frac{1}{n+1} \binom{2n}{n}, \quad n = 0, 1, 2, 3, \dots$$

Dãy số c_n do E.C. Catalan tìm ra nên được gọi là *dãy số Catalan*.

n	c_n
0	1
1	1
2	2
3	5
4	14
5	42
6	132
7	429

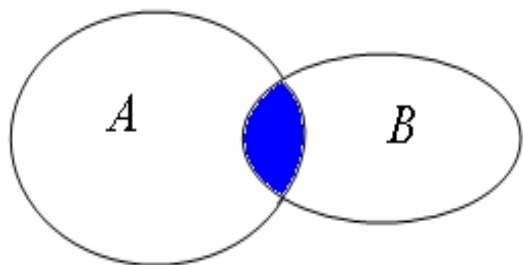
8	1 430
9	4 862
10	16 796
11	58 786
12	208 012
13	742 900
14	2 674 440
15	9 694 845
16	35 357 670
17	129 644 790
18	477 638 700
19	1 767 263 190
20	6 564 120 420
...	...

Dãy số Catalan tăng nhanh và xấp xỉ hàm số sau $c_n \approx \frac{4^n}{n^{3/2}\sqrt{\pi}}$

4.2. Nguyên lý bù - trừ

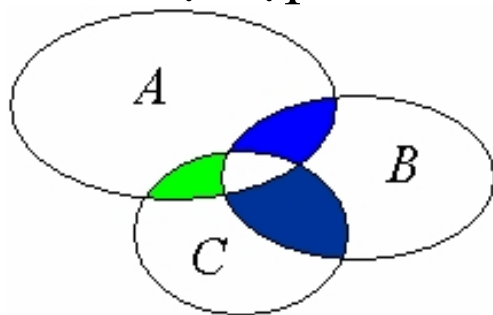
4.2.1. Nguyên lý bù - trừ

Giả sử ta có hai tập hợp A và B . Hiển nhiên:



$$|A \cup B| = |A| + |B| - |A \cap B|$$

Thêm một tập C nữa.



$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

Vậy với n tập hợp A_1, A_2, \dots, A_n thì lực lượng của hợp của n tập hợp này là $\bigcup_{i=1}^n A_i$ sẽ như thế nào. Liệu có thể đưa ra một công thức chung để tính được không?

- Nếu lấy "xấp xỉ" đầu tiên là $\sum_{i=1}^n |A_i|$ thì sẽ quá lớn vì nếu $A_i \cap A_j \neq \emptyset$ thì các phần tử thuộc $A_i \cap A_j$ được đếm hai lần.
- Trừ đi khỏi xấp xỉ đầu tiên tổng: $\sum_{1 \leq i < j \leq n} |A_i \cap A_j|$. Ta nhận được một đại lượng nói chung là bé so với lực lượng cần tìm vì nếu $A_i \cap A_j \cap A_k \neq \emptyset$ thì các phần tử thuộc tập giao này trước đây được tính ba lần nay lại trừ đi ba lần nên chưa được tính.
- Thêm vào tổng sau đây: $\sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k|$ thì lại nhận được một đại lượng nói chung là quá lớn.
- Song cứ tiếp tục thêm và bớt sau n bước ta luôn nhận được kết quả đúng.

Định lý 4.1 (Nguyên lý bù - trừ): Với mỗi bộ n tập hợp A_1, A_2, \dots, A_n tùy ý thì:

$$\left| \bigcup_{i=1}^n A_i \right| = \sum_{i=1}^n |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| + \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| - \dots + (-1)^{n-1} |A_1 \cap A_2 \cap \dots \cap A_n|$$

Chứng minh:

Lấy một phần tử tùy ý $x \in \bigcup_{i=1}^n A_i$. Phần tử này xuất hiện trong tập hợp ở vế trái 1 lần.

Giả sử x thuộc vào q tập hợp trong số n tập hợp trên với $1 \leq q \leq n$. Số lần xuất hiện của x trong các tập hợp ở vế phải là:

$$q - \binom{q}{2} + \binom{q}{3} - \dots + (-1)^{q-1} \binom{q}{q} = 1 - \left(\binom{q}{0} - \binom{q}{1} + \binom{q}{2} - \dots + (-1)^q \binom{q}{q} \right) = 1 - (1-1)^q = 1$$

4.2.2. Một số ứng dụng của nguyên lý bù - trừ

1. Bài toán tập con bội k -phần tử

Giả sử ta có tập bội $X = (k_1 * x_1, k_2 * x_2, \dots, k_n * x_n)$ và k là một số nguyên không âm.

Bài toán đặt ra là: **Hỏi có bao nhiêu tập con bội k phần tử của tập bội X ?**

Ký hiệu: m_k là số các tập con bội k phần tử của tập bội X . Trong Chương 2 ta đã xét các trường hợp sau đây:

- Nếu $k_1 = k_2 = \dots = k_n = 1$, tập bội X là tập thông thường và $m_k = \binom{n}{k}$
- Nếu $k_1, k_2, \dots, k_n \geq k$ thì $m_k = \binom{n+k-1}{k}$

- Trong trường hợp **tổng quát** nếu có bội nào đó $k_i \geq k$, ta có thể thay k_i bằng k nhưng m_k không thay đổi. Vậy, không mất tính tổng quát ta có thể giả thiết rằng: **$k_i \leq k$ với $i = 1, 2, \dots, n$.**

Dễ thấy rằng, bài toán tập con bội k phần tử tương đương với bài toán tìm các dãy nghiệm nguyên của hệ bất phương trình sau đây:

$$\begin{cases} t_1 + t_2 + \dots + t_n = k \\ 0 \leq t_i \leq k_i, \quad i = 1, 2, \dots, n \end{cases} \quad (4.6)$$

Song việc giải hệ bất phương trình (4.6) cũng không hề đơn giản. Do vậy, ta cần tìm một giải pháp khác đơn giản hơn.

Ký hiệu Z là tập tất cả các tập con bội k phần tử của tập bội $\overline{X} = (k * x_1, k * x_2, \dots, k * x_n)$

Tập Z bao gồm $\binom{n+k-1}{k}$ tập con. Mọi tập con bội k phần tử của tập bội X đều nằm trong Z .

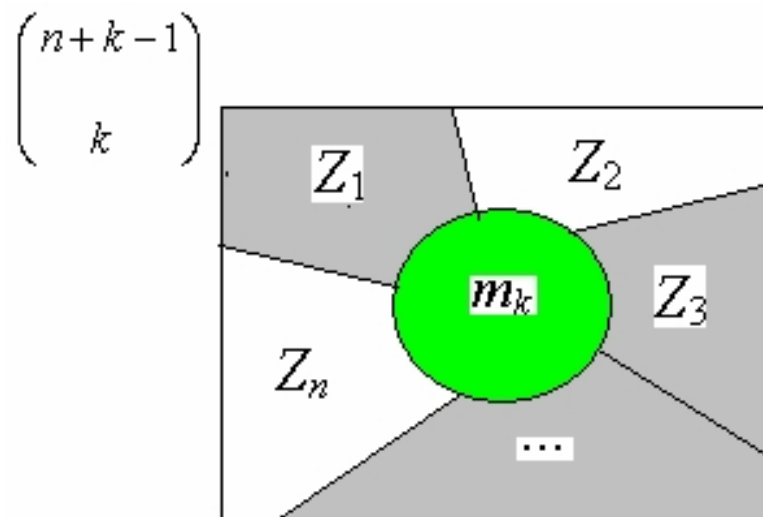
Vậy để tính m_k ta tìm cách loại bỏ các tập con bội thuộc Z nhưng không phải là tập con bội k phần tử của X .

Mỗi tập con bội k phần tử của X phải thoả mãn tất cả n bất đẳng thức kẹp trong hệ (4.6). Nếu không thoả mãn chỉ một bất đẳng thức thì tập con bội tương ứng sẽ không phải là tập con bội k phần tử của X .

Ta ký hiệu Z_i là tập tất cả các tập con bội thuộc Z mà bội của phần tử x_i là $t_i > k_i$, với $i = 1, 2, 3, \dots, n$. Khi đó thì:

$$m_k = |Z - \bigcup_{i=1}^n Z_i| = |Z| - |\bigcup_{i=1}^n Z_i|$$

Công thức trên được minh họa như sau:



Tính toán lực lượng của Z_i và các cặp giao của chúng, rồi áp dụng nguyên lý bù - trừ ta sẽ tìm được số các tập con bội k phần tử của tập bội X .

$$\begin{aligned}
m_k = & \binom{n+k-1}{k} - \sum_{i=1}^n \binom{n+k-k_i-2}{k-k_i-1} + \\
& + \sum_{1 \leq i < j \leq n} \binom{n+k-k_i-k_j-3}{k-k_i-k_j-2} - \dots + (-1)^n \binom{n+k-k_1-k_2-\dots-k_n-n-1}{k-k_1-k_2-\dots-k_n-n}
\end{aligned}$$

Chú ý: Trong các dấu tổ hợp ở trên nếu có tham số nào bé hơn 0 thì tổ hợp tương ứng sẽ bằng 0.

Ví dụ 4.3: Cho tập bội $X = (4 * x_1, 3 * x_2, 7 * x_3)$. Hãy tìm số các tập con bội 11 phần tử của tập X .

Ta xây dựng các tập hợp sau đây:

- Z là tập tất cả các tập con bội 11 phần tử của tập bội $(11 * x_1, 11 * x_2, 11 * x_3)$
- Z_1 là tập tất cả các tập con thuộc Z mà bội của x_1 lớn hơn 4 (≥ 5)
- Z_2 là tập tất cả các tập con thuộc Z mà bội của x_2 lớn hơn 3 (≥ 4)
- Z_3 là tập tất cả các tập con thuộc Z mà bội của x_3 lớn hơn 7 (≥ 8)

Khi đó ta có:

$$|Z| = \binom{3+11-1}{11} = \binom{13}{11} = 78$$

Mỗi tập con bội thuộc Z_1 chứa ít nhất 5 phần tử x_1 . Vậy lực lượng của Z_1 bằng số các tập con bội $11 - 5 = 6$ phần tử của tập bội ($11 * x_1, 11 * x_2, 11 * x_3$).

$$\text{Thế thì: } |Z_1| = \binom{3+6-1}{6} = \binom{8}{6} = 28$$

$$\text{Tương tự ta có: } |Z_2| = \binom{3+7-1}{7} = \binom{9}{7} = 36$$

$$\text{và } |Z_3| = \binom{3+3-1}{3} = \binom{5}{3} = 10$$

Lực lượng của $Z_1 \cap Z_2$ là bằng số các tập con bội 11-5-4 = 2 phần tử của tập bội $(11 * x_1, 11 * x_2, 11 * x_3)$. Vậy thì:

$$|Z_1 \cap Z_2| = \binom{3+2-1}{2} = \binom{4}{2} = 6$$

còn $Z_1 \cap Z_3 = Z_2 \cap Z_3 = \emptyset$

Theo nguyên lý bù - trừ, số các tập con bội 11 phần tử của tập bội X là:

$$\begin{aligned} & |Z| - |Z_1 \cup Z_2 \cup Z_3| \\ &= |Z| - |Z_1| - |Z_2| - |Z_3| + |Z_1 \cap Z_2| + |Z_1 \cap Z_3| + |Z_2 \cap Z_3| - |Z_1 \cap Z_2 \cap Z_3| \\ &= 78 - 28 - 36 - 10 + 6 + 0 + 0 - 0 = 10 \end{aligned}$$

2. Bài toán k -phân hoạch

Giả sử X là tập n phần tử. Một phân hoạch $\{A_1, A_2, \dots, A_k\}$ của tập X bao gồm k khối được gọi là k -phân hoạch.

Bài toán k -phân hoạch: Cho trước số n và k . Hãy tìm số các k -phân hoạch của tập n phần tử.

$S(n, k)$ - số các k -phân hoạch của tập n phần tử.

Đã có một số công thức đệ quy để tính số này. Song ta muốn có được một công thức hiện để tính số các k -phân hoạch của tập n phần tử một cách nhanh chóng.

Mỗi k -phân hoạch của tập n phần tử tạo nên $k!$ toàn ánh có chung một nhân tử tập n phần tử lên tập k phần tử và ngược lại. Vậy, thay vì việc tìm số các k -phân hoạch ta đi tìm số các toàn ánh này.

Ký hiệu Y là tập hợp có k phần tử.

Xét tất cả các **toàn ánh** $f : X \rightarrow Y$. Nghĩa là các ánh xạ f từ X sang Y mà $f(X) = Y$.

Ký hiệu số toàn ánh này là $s_{n,k}$. Ta có mối quan hệ sau đây:

$$S(n, k) = \frac{1}{k!} s_{n, k} \quad (4.7)$$

Giả sử $Y = \{y_1, y_2, \dots, y_k\}$.

Ký hiệu $F_i = \{f : X \rightarrow Y \mid y_i \notin f(X)\}$, $i = 1, 2, \dots, k$

Hiển nhiên:

$$f(X) \neq Y \quad \Leftrightarrow \quad f \in \bigcup_{i=1}^k F_i$$

Số tất cả các ánh xạ $f: X \rightarrow Y$ là k^n .

Vậy ta chỉ cần tìm lực lượng của tập hợp $\bigcup_{i=1}^k F_i$.

Áp dụng nguyên lý bù - trừ để làm việc này.

Giao của i tập hợp $F_{p1} \cap F_{p2} \cap \dots \cap F_{pi}$ chính là tất cả các ánh xạ $f: X \rightarrow Y$ mà $y_{p1}, y_{p2}, \dots, y_{pi} \notin f(X)$ và bằng tập các ánh xạ:

$$f: X \rightarrow Y \setminus \{y_{p1}, y_{p2}, \dots, y_{pi}\}$$

từ tập n phần tử sang tập $k-i$ phần tử. Vậy số các ánh xạ này là $(k-i)^n$.

Theo nguyên lý bù - trừ thì:

$$s_{n,k} = k^n - \left| \bigcup_{i=1}^k F_i \right| = k^n - \sum_{i=1}^{k-1} (-1)^i \binom{k}{i} (k-i)^n = \sum_{i=0}^{k-1} (-1)^i \binom{k}{i} (k-i)^n$$

Theo (4.7) ta có công thức tính số các k -phân hoạch như sau:

$$S(n, k) = \frac{1}{k!} \sum_{i=0}^{k-1} (-1)^i \binom{k}{i} (k-i)^n, \quad k = 1, 2, 3, \dots, n.$$

Ta có công thức tính trực tiếp số Bell B_n là số các phân hoạch của tập n phần tử.

$$B_n = \sum_{k=1}^n \frac{1}{k!} \sum_{i=0}^{k-1} (-1)^i \binom{k}{i} (k-i)^n$$

3. Bài toán hỗn độn

Giả sử X là một tập n phần tử. Ta đồng nhất $X \equiv \{1, 2, 3, \dots, n\}$.

Định nghĩa 4.3: Một *hỗn độn* trên tập X là một hoán vị f trên tập này mà:

$$f(i) \neq i \text{ với mọi } i = 1, 2, \dots, n$$

Bài toán hỗn độn: Cho một tập n phần tử X . Hãy tìm số tất cả các hỗn độn trên tập X .

Ký hiệu D_n là tập tất cả các hỗn độn trên tập X và:

$$F_i = \{ f \in S_n \mid f(i) = i \} \text{ với } i = 1, 2, \dots, n$$

Hoán vị f là một hỗn độn khi và chỉ khi nó không thuộc vào một tập F_i nào. Nghĩa là, nó không thuộc vào tập $\bigcup_{i=1}^n F_i$.

Theo nguyên lý bù - trừ ta có:

$$\begin{aligned} |D_n| &= |S_n| - \left| \bigcup_{i=1}^n F_i \right| \\ &= |S_n| - \sum_{i=1}^n |F_i| + \sum_{1 \leq i < j \leq n} |F_i \cap F_j| - \sum_{1 \leq i < j < k \leq n} |F_i \cap F_j \cap F_k| + \dots + (-1)^{n-1} |F_1 \cap F_2 \cap \dots \cap F_n| \end{aligned}$$

Mà $|F_{p1} \cap F_{p2} \cap \dots \cap F_{pi}| = (n - i)!$. Vậy thì:

$$|D_n| = \sum_{i=0}^n (-1)^i \binom{n}{i} (n-i)! = \sum_{i=0}^n (-1)^i \frac{n!}{i!}$$

$$= n! \left(\frac{1}{2!} - \frac{1}{3!} + \frac{1}{4!} - \dots + (-1)^n \frac{1}{n!} \right)$$

Số hỗn độn trên tập n phần tử xấp xỉ bằng $0,368.n!$

Một hình ảnh khác của bài toán hỗn độn là bài toán vui sau đây:

Bài toán viết thư: Một cô gái ngồi viết thư cho n bạn trai. Cô viết địa chỉ trên các phong bì trước, sau đó viết thư rồi mới bỏ thư vào phong bì.

Hỏi xác suất mà cô bỏ nhầm tất cả các thư là bao nhiêu?

Câu trả lời lấy từ bài toán hỗn độn ở trên, nó là: $0,368$.

BÀI TẬP CHƯƠNG 4

4.1. Hỏi có bao nhiêu dãy nhị phân độ dài 10 bắt đầu hoặc kết thúc bởi 01?

4.2. Hỏi có bao nhiêu số nguyên dương không vượt quá 10000 và không chia hết cho 3 hoặc 4 hoặc 7?

4.3. Chứng minh rằng, số các dãy nhị phân độ dài n không chứa chữ số 1 tại hai vị trí liên tiếp nhau nào là bằng số Fibonacci F_{n+1} .

4.4. Chứng minh rằng:

$$F_{n+1} = \sum_{k=0}^n \binom{n-k+1}{k}$$

4.5. Ký hiệu q_n là số các cách có thể đặt cặp dấu ngoặc vào tích $a_0 a_1 a_2 \dots a_n$. Chứng minh rằng, q_n bằng số Catalan c_n .

4.6. Chứng minh rằng, số các cách chia một đa giác lồi phẳng có $n+2$ cạnh thành các tam giác nhờ $n-1$ đường thẳng nối các đỉnh là bằng số Catalan c_n .

4.7. Chứng minh rằng, số các dãy nhị phân độ dài $2n$ gồm n chữ số 1 và n chữ số 0 mà trong mọi tiền tố của nó số chữ số 0 không nhiều hơn số chữ số 1 (từ Dyck) là bằng số Catalan c_n . Chẳng hạn, các từ Dyck có độ dài 6 bao gồm: 111000, 101100, 101010, 110010, 110100.

4.8 Ký hiệu $T(n)$ là số các số tự nhiên không vượt quá n và nguyên tố cùng nhau với n . Chứng minh rằng:

$$T(n) = n \cdot \left(1 - \sum_{i=1}^m \frac{1}{p_i} + \sum_{1 \leq i < j \leq m} \frac{1}{p_i p_j} - \dots + (-1)^m \frac{1}{p_1 p_2 \dots p_m} \right) = n \cdot \prod_{i=1}^m \left(1 - \frac{1}{p_i} \right)$$

với p_1, p_2, \dots, p_m là tất cả các ước số nguyên tố khác nhau của n .

4.9. Ký hiệu d_n là số tất cả các hỗn độ trên tập n phần tử, nghĩa là $d_n = |D_n|$. Chứng minh rằng:

$$d_n = (n - 1)(d_{n-1} + d_{n-2})$$

Từ đó xây dựng công thức tính số các hỗn độ của tập n phần tử.

PHỤ LỤC

MỘT SỐ CHƯƠNG TRÌNH C GIẢI CÁC BÀI TOÁN TỔ HỢP

1. CHƯƠNG TRÌNH SINH CÁC DÃY CHỌN CÓ THỨ TỰ

Dãy chọn có thứ tự m phần tử từ tập n phần tử là một dãy m phần tử tùy ý lấy ra từ tập n phần tử trên sao cho các phần tử trong dãy đều khác nhau từng đôi. Dãy chọn có thứ tự còn được gọi là *chỉnh hợp* chập m của n .

```
// Chương trình sinh các dãy chọn có thứ tự
#include <stdio.h>
#define Nmax 30
short i , tg , m , n ;
int B[Nmax] , T[Nmax] ;
```



```

void main()
{
    clrscr() ;
    printf("Nhap m = ") ; scanf("%d",&m) ;
    printf("Nhap n = ") ; scanf("%d",&n) ;
    printf("\n") ;
    for (i = 1 ; i <= m ; i++) T[i] = 1 ;
    printf("CAC DAY CHON CO THU TU LA \n") ;
    do
    {
        for (i = 1 ; i <= n ; i++) B[i] = 0 ;
        for (i = 1 ; i <= m ; i++) B[T[i]] = 1 ;
        tg = 0 ;
        for (i = 1 ; i <= n ; i++) tg += B[i] ;
        if (tg == m)
            { for (i = 1 ; i <= m ; i++) printf("%3d", T[i]) ;
              printf("\n") ; }
    }
}

```

```

i = m ;
while (T[i] == n) { T[i] = 1 ; i-- ; }
if (i > 0) T[i]++ ;
} while (i > 0) ;
getch() ;
}

```

2. CHƯƠNG TRÌNH SINH CÁC CÁCH XẾP ĐẶT CÓ THỨ TỰ

Xếp đặt có thứ tự m “đối tượng” vào n “hộp” là việc xếp các đối tượng vào các hộp có quan tâm tới thứ tự của các đối tượng trong mỗi hộp. Thực chất, mỗi cách xếp đặt có thứ tự m đối tượng vào n hộp chính là một cách chọn có thứ tự m phần tử trong tập $m+n-1$ phần tử (số 0 trong dãy kết quả để phân biệt giữa các hộp).

```

//Chương trình sinh các cách xếp đặt có thứ tự
#include <stdio.h>

```

```

#define Nmax 30
short i , tg , dem , m , n , l ;
int B[Nmax], T[Nmax] ;

void main()
{
    clrscr() ;
    printf("Nhap m = ") ; scanf("%d",&m) ;
    printf("Nhap n = ") ; scanf("%d",&n) ;
    printf("\n") ;
    l = m + n - 1 ;
    for (i = 1 ; i <= m ; i++) T[i] = 1 ;
    dem = 1 ;
    printf("CAC CACH XEP DAT CO THU TU m DT VAO n HOP \n") ;
    do
    {
        for (i = 1 ; i <= l ; i++) B[i] = 0 ;
        for (i = 1 ; i <= m ; i++) B[T[i]] = 1 ;
    }
}

```

```

tg = 0 ;
for (i = 1 ; i <= 1 ; i++) tg += B[i] ;
if (tg == m)
{
    for (i = 1 ; i <= 1 ; i++) B[i] = 0 ;
    for (i = 1 ; i <= m ; i++) B[T[i]] = i ;
    for (i = 1 ; i <= 1 ; i++) printf("%3d", B[i]) ;
    dem++ ; printf("\n") ;
}
i = m ;
while (T[i] == 1) { T[i] = 1 ; i-- ; }
if (i > 0) T[i]++ ;
if (dem % 22 == 0) { getch() ; clrscr() ; }
} while (i > 0) ;
printf("So cac cach xep dat co thu tu %d", dem-1) ;
getch() ;
}

```

3. CHƯƠNG TRÌNH SINH CÁC HOÁN VỊ

// Chương trình sinh các hoán vị theo thứ tự tự nhiên

```
#include <stdio.h>
```

```
#define max 20
```

```
int i, j, n, Pt, P[max] ;
```

```
void main()
```

```
{
```

```
clrscr() ;
```

```
printf("Nhập số phần tử n : ") ;
```

```
scanf("%d",&n) ;
```

```
for (i=1 ; i<=n ; i++) P[i] = i ;
```

```
printf("\n");
```

```
do
```

```
{
```

```
for (i=1 ; i<=n ; i++) printf("%2d ",P[i]) ;
```

```

printf("\n") ;
j = n-1 ;
while (P[j] >= P[j+1]) j-- ;
if (j>0)
{
    i = j+1 ;
    while (P[i] > P[j]) i++ ;
    Pt = P[j] ; P[j] = P[i-1] ; P[i-1] = Pt ;
    j++ ; i = n ;
    while (j < i)
        { Pt = P[j] ; P[j] = P[i] ; P[i] = Pt ; j++ ; i-- ; }
    }
} while (j > 0) ;
getch() ;
}

```

Chương trình sắp đặt công việc tối ưu

Có n công việc cần thực hiện với thời gian thực hiện như nhau, chẳng hạn là một ngày. Với công việc e_i ($i = 1, 2, \dots, n$) 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 trước hạn và 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.

```
// Chương trình sắp đặt công việc tối ưu
```

```
#include <stdio.h>
```

```
#define max 20
```

```
int i, j, p, n, At, Tg, Tmax, d[max], X[max], Y[max],  
A[max], Amax[max] ;
```

```
void main()
```

```

{
    clrscr() ;
    printf("Nhap so cong viec n : ") ;
    scanf("%d",&n) ;
    for (i = 1 ; i <= n ; i++)
    {
        A[i] = i ;
        printf("Nhap thoi han cong viec thu %d : ", i) ;
        scanf("%d", &d[i]) ;
        printf("Nhap tien thuong cong viec thu %d : ", i) ;
        scanf("%d", &X[i]) ;
        printf("Nhap tien phat cong viec thu %d : ", i) ;
        scanf("%d", &Y[i]) ;
        printf("\n") ;
    }
    Tmax = -1000 ;
    do

```



```

{
Tg = 0 ;
for (j = 1 ; j <= n ; j++)
    Tg += j < d[A[j]] ? (d[A[j]]-j)*X[A[j]] : (d[A[j]]-j)*Y[A[j]] ;
if (Tmax < Tg)
    {
        Tmax = Tg ;
        for (i = 1 ; i <= n ; i++) Amax[i] = A[i] ;
    }
p = n-1 ;
while (A[p] >= A[p+1]) p-- ;
if (p > 0)
    {
        j = p+1 ;
        while (A[j] > A[p]) j++ ;
        At = A[p] ; A[p] = A[j-1] ; A[j-1] = At ;
        p++ ; j = n ;
    }

```

```

        while (p < j)
            { At = A[p] ; A[p] = A[j] ; A[j] = At ; p++ ; j-- ; }
        }
    } while (p > 0) ;
    printf("Xep dat cong viec toi uu voi tong tien la %d : ", Tmax) ;
    printf("\n") ;
    for (i = 1 ; i <= n ; i++) printf("%2d ",Amax[i]) ;
    printf("\n") ;
    getch() ;
}

```

Sử dụng thứ tự từ điển ngược và tính đệ quy của tập các hoán vị trên một tập hợp, ta xây dựng được chương trình ngắn gọn sinh các hoán vị sau đây.

```

// Chương trình đệ quy sinh các hoán vị theo thứ tự từ điển ngược
#include <stdio.h>

```

```

#define Nmax 20
int i , n , P[Nmax] ;

void DAO(int m)
{
    int i = 1 , j = m , tg ;
    while (i < j)
    {
        tg = P[i] ; P[i] = P[j] ;
        P[j] = tg ; i++ ; j-- ;
    }
}

void TD_NGUOC(int m)
{
    static dem ;
    int i , tg ;

```

```

if (m == 0)
{
    for (i = 1; i <= n ; i++) printf("%4d", P[i]) ;
    printf("\n") ; dem++ ;
    if (dem % 22 == 0) { getch() ; clrscr() ; }
}
else for (i = 1; i <= m ; i++)
{
    TD_NGUOC(m-1) ;
    if (i < m) {
        tg = P[m] ;
        P[m] = P[i] ;
        P[i] = tg ;
        DAO(m-1) ;
    }
}
}

```

```

void main()
{
    clrscr() ;
    printf("Luc luong cua tap X : ") ; scanf("%d", &n) ;
    printf("\n") ;
    for (i = 1; i <= n ; P[i] = i++) ;
    printf("CAC HOAN VI LA \n") ;
    TD_NGUOC(n) ;
    getch() ;
}

```

4. CHƯƠNG TRÌNH SINH CÁC TẬP CON

Với tập n phần tử X cho trước, chương trình có nhiệm vụ sinh ra tất cả 2^n tập con của tập này. Việc tìm các tập con đưa về việc tìm các dãy nhị phân độ dài n .

Thuật toán dãy nhị phân

Chương trình lần lượt sinh ra các dãy nhị phân theo thứ tự từ điển. Dãy đầu tiên gồm n chữ số 0 và dãy cuối cùng gồm n chữ số 1.

Để tìm được dãy sau ta chỉ việc đọc dãy trước đó từ phải sang trái. Nếu $b_i = 1$ thì thay b_i bằng 0 và b_i đầu tiên bằng 0 thì được thay bằng 1. Phần còn lại giữ nguyên.

// Chương trình sinh tất cả các tập con của tập hợp

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
    long  so, dem = 1 ;
```

```
    int  i, n, A[20] ;
```

```
    clrscr() ;
```

```
    printf("Luc luong cua tap  X : ") ; scanf("%d" , &n) ;
```

```
    puts("\a CAC TAP CON CUA TAP HOP LA ") ;
```

```

printf("\n") ;
so = 1 << n ;
for (i = 0 ; i < n ; i++)
{
    A[i] = 0 ;
    printf("%4d", A[i]) ;
}
printf("\n") ;
do
{
    i = n-1 ;
    while (A[i] == 1) { A[i] = 0 ; i-- ; }
    A[i] = 1 ;
    for (i = 0; i < n; i++)
{
    printf("%4d", A[i]) ;
    sound(i * 80) ;
}

```

```

    delay(100) ;
}
    printf("\n") ;
    dem ++ ;
    if (dem % 24 == 0) { getch() ; clrscr() ; }
}
while (dem < so) ;
nosound() ;
printf("\n So cac tap con la : %d \n", so) ;
getch() ;
}

```

Thuật toán thêm/bớt một phần tử

Ta quan sát thấy rằng: từ một tập con đã biết nào đó của tập hợp, thêm vào một phần tử ngoài tập này hay bớt đi một phần tử trong tập này thì ta sẽ nhận được tập con khác. Hơn nữa, các dãy nhị phân biểu

diễn các tập con này khác nhau chỉ một vị trí. Do vậy, việc thay đổi chúng trở nên rất đơn giản.

// Chương trình để quy sinh tất cả các tập con bằng cách thêm/bớt

```
#include<stdio.h>
```

```
long so ;
```

```
int i , n , A[20] ;
```

```
void Tap_Con (int nt , int Sao)
```

```
{
```

```
int i ;
```

```
if (nt == 0)
```

```
{
```

```
for (i = 1; i <= n; i++) printf("%4d", A[i]) ;
```

```
printf("\n") ;
```

```
so++ ;
```

```
if (so % 20 == 0) { getch() ; clrscr() ; }
```

```

    }
else
    if (Sao == 0)
    {
        A[nt] = 0 ; Tap_Con (nt-1, 0) ;
        A[nt] = 1 ; Tap_Con (nt-1, 1) ;
    }
else
    {
        A[nt] = 1 ; Tap_Con (nt-1, 1) ;
        A[nt] = 0 ; Tap_Con (nt-1, 0) ;
    }
}

void main()
{
    clrscr() ;

```

```

so = 0 ;
printf("Luc luong cua tap X : ") ; scanf("%d" , &n) ;
printf("\n") ;
for (i = 1; i <= n; A[i] = 0 , i++) ;
printf("CAC TAP CON CUA TAP HOP LA \n") ;
Tap_Con (n, 0) ;
printf("\n") ;
printf("So tap con la : %d\n", so) ;
getch() ;
}

```

Thuật toán sinh các tập con k -phần tử

Để sinh các tập con k -phần tử của một tập n phần tử, ta xuất phát từ tập hợp gồm k phần tử đầu tiên sau đó thay một phần tử trong tập con tìm được bằng một phần tử khác ta sẽ có một danh sách $G(n,k)$ bao gồm

tất cả các tập con k phần tử. Danh sách $G(n,k)$ được xác định bằng công thức đệ quy sau đây:

$$G(n,k) = G(n-1,k), G(n-1,k-1)^* \cup \{n\}$$

// Chương trình đệ quy tìm các tập con k -phần tử của tập n phần tử

```
#include<stdio.h>
```

```
int j , so , n , k , A[20] ;
```

```
void G(int nt, int kt, int Gsao)
```

```
{
```

```
    int i ;
```

```
    if (kt == 0 || kt == nt)
```

```
    {
```

```
        for (i = 1 ; i <= kt ; i ++)
```

```
            printf("%04d", i) ;
```

```
        for (i = 1 ; i <= nL ; i ++)
```

```

        if (A[i] != 0) printf("%4d" , A[i]) ;
    printf("\n") ; so ++ ;
}
if (kt > 0 && kt < nt)
    if (Gsao == 0)
    {
        G(nt-1, kt, 0) ; A[nt] = nt ;
        G(nt-1, kt-1, 1) ; A[nt] = 0 ;
    }
else
    {
        A[nt] = nt ; G(nt-1, kt-1, 0) ;
        A[nt] = 0 ; G(nt-1, kt, 1) ;
    }
}

```

```

void main()

```

```

{
    clrscr() ;
    printf("Luc luong cua tap X : ") ; scanf("%d", &n) ;
    printf("Luc luong cua tap con : ") ; scanf("%d", &k) ;
    printf("\n") ;
    so = 0 ;
    for (j = 1; j <= n; A[j] = 0 , j ++ ) ;
    G(n, k, 0) ;
    printf("\n") ;
    printf("So cac tap con la : %d\n", so) ;
    getch() ;
}

```

5. CHƯƠNG TRÌNH SINH CÁC TẬP CON BỘI

Mở rộng khái niệm tập hợp thông thường thành *tập bội*. Tập bội là tập hợp mà mỗi phần tử của nó có thể xuất hiện nhiều hơn một lần. Các

số nguyên không âm đứng trước mỗi phần tử được gọi là bội của phần tử ấy.

Thuật toán dựa trên dãy bị chặn

Việc sinh các tập con bội của một tập bội tương đương với bài toán sinh các dãy bị chặn với hai biên là $\langle 0 \ 0 \ \dots \ 0 \ 0 \rangle$ và $\langle k_1 \ k_2 \ \dots \ k_{n-1} \ k_n \rangle$.

// Chương trình sinh các tập con bội của tập bội

```
#include <stdio.h>
```

```
#define Nmax 50
```

```
short i , n ;
```

```
int T[Nmax] , K[Nmax] ;
```

```
void main()
```

```
{
```

```
clrscr() ;
```

```

printf("Nhap n = ") ; scanf("%d",&n) ;
printf("\n") ;
for (i = 1 ; i <= n ; i++)
{
    printf("Nhap boi cua phan tu thu %d : ", i) ;
    scanf("%d", &K[i]) ; T[i] = 0 ;
}
printf("\n") ;
printf("CAC TAP CON BOI LA \n") ;
do
{
    for (i = 1 ; i <= n ; i++) printf("%3d", T[i]) ;
    printf("\n") ;
    i = n ;
    while (T[i] == K[i]) { T[i] = 0 ; i-- ; }
    if (i > 0) T[i]++ ;
} while (i > 0) ;

```



```
getch() ;  
}
```

Thuật toán thêm/bớt một phần tử

Từ một tập con bội vừa tìm được, ta thêm vào hay bớt đi một phần tử thì nhận được tập con bội khác. Chương trình dưới đây sinh ra tất cả các tập con bội của một tập bội mà tập con sau sai khác tập con trước chỉ một phần tử.

```
// Chương trình tính sinh các tập con bội của tập bội bằng cách thêm/bớt  
#include<stdio.h>  
#define Nmax 10  
char B[Nmax] , K[Nmax] ;  
long j , i = 0 ;  
short p , n , t , q , s ;  
void main()
```

```

{
    clrscr() ;
    printf("Nhap n = ") ; scanf("%d", &n) ;
    for (t = 1 ; t <= n ; B[t] = 0, t++)
    {
        printf("Boi thu %d ", t) ;
        scanf("%d", &K[t]) ;
    }
    clrscr() ;
    do
    {
        for (t = 1; t <= n; t++) printf("%3d", B[t]) ;
        printf("\n") ;
        i++ ;
        p = 1 ;
        j = i ;
        while (j % (K[p] + 1) == 0)

```

```

{
    j = j / (K[p] + 1) ;
    p++ ;
}
if (i % 24 == 0) { getch() ; clrscr() ; }
if (p <= n)
{
    q = j / (K[p]+1) ;
    s = j % (K[p]+1) ;
    if (q % 2 == 0) B[p] = s ;
    else B[p] = K[p] - s ;
}
} while (p <= n) ;
printf("\n") ;
printf("So cac tap con boi la: %d \n", i) ;
getch() ;
}

```

6. CHƯƠNG TRÌNH SINH CÁC PHÂN HOẠCH

Mỗi phân hoạch của tập n phần tử có thể biểu diễn bằng dãy các chỉ số và xem như là một từ có độ dài n . Ta sắp xếp các từ đó tăng dần theo thứ tự từ điển. Chương trình dưới đây sinh ra các từ (phân hoạch) này.

// Chương trình sinh các phân hoạch của tập hợp

```
#include <stdio.h>
```

```
#define Nmax 50
```

```
short i , n , CS[Nmax] , Max[Nmax] ;
```

```
void main()
```

```
{
```

```
clrscr() ;
```

```
printf("Nhap n = ") ; scanf("%d", &n) ;
```

```
printf("\n") ;
```

```
for (i = 1 ; i <= n ; i++) CS[i] = 1 ;
```

```

printf("CAC PHAN HOACH CUA TAP HOP LA \n") ;
for (i = 1 ; i <= n ; i++) printf("%3d", CS[i]) ;
printf("\n") ;
while (CS[n] != n)
{
    Max[1] = 0 ;
    for (i = 2 ; i <= n ; i++)
        if (Max[i-1] < CS[i-1]) Max[i] = CS[i-1] ;
        else Max[i] = Max[i-1] ;
    i = n ;
    while (CS[i] == Max[i]+1) { CS[i] = 1 ; i-- ; }
    if (i > 0) CS[i]++ ;
    for (i = 1 ; i <= n ; i++) printf("%3d", CS[i]) ;
    printf("\n") ;
} ;
getch() ;
}

```

7. CHƯƠNG TRÌNH PHÂN TÍCH SỐ

Với số nguyên dương n đã cho, hãy tính xem có bao nhiêu phân tích của số n và tìm ra tất cả các phân tích đó.

Chương trình tính số các phân tích

// Chương trình tính số tất cả các phân tích của số nguyên n

```
#include<stdio.h>
```

```
long n , k , t , ts ;
```

```
long PT_SO(long n, long k)
```

```
{
```

```
    if (n < k) return 0 ;
```

```
    else
```

```
        if ((k == n) || (k == 1)) return 1 ;
```

```
        else return(PT_SO(n-1, k-1) + PT_SO(n-k, k)) ;
```

```
}
```

```
void main()
```

```
{
```

```
clrscr() ;
```

```
printf("Cho n : ") ; scanf("%d" , &n) ;
```

```
ts = 0 ;
```

```
for (k = 1; k <= n; k++)
```

```
{
```

```
    t = PT_SO(n, k) ;
```

```
    printf("So cac phan tich %ld thanh phan la: %ld \n", k, t) ;
```

```
    ts += t ;
```

```
}
```

```
printf("\n So tat ca cac phan tich: %ld \n", ts) ;
```

```
getch();
```

```
}
```

Chương trình sinh các phân tích

// Chương trình sinh các phân tích số

```
#include <stdio.h>
```

```
#define Nmax 20
```

```
int i , j , n , d , l , sum , dem = 1 ;
```

```
int S[Nmax] , R[Nmax] ;
```

```
void main()
```

```
{
```

```
clrscr() ;
```

```
printf("Cho biet so n : ") ; scanf("%d", &n) ;
```

```
printf("\n") ;
```

```
S[1] = n ; R[1] = d = 1 ;
```

```
printf("%4d \n", n) ;
```

```
while (S[1] > 1)
```

```
{
```



```

sum = 0 ;
if (S[d] == 1) { sum += R[d] ; d-- ; }
sum += S[d] ;
R[d]-- ;
l = S[d] - 1 ;
if (R[d] > 0) d += 1 ;
S[d] = 1 ;
R[d] = sum / l ;
l = sum % l ;
if (l > 0) { S[++d] = 1 ; R[d] = 1 ; }
dem ++ ;
for (i = 1 ; i <= d ; i++)
    for (j = 1 ; j <= R[i] ; j++) printf("%4d", S[i]) ;
printf("\n") ;
if (dem % 22 == 0) { getch() ; clrscr() ; }
}
printf("\nSo tat ca cac phan tich : %d \n" , dem) ;

```

```
    getch() ;  
}
```

8. CHƯƠNG TRÌNH TÍNH SỐ CÁC HỖN ĐỘN

Hỗn độn trên một tập hợp là một hoán vị (song ánh) của tập này mà ảnh và nghịch ảnh của mỗi phần tử đều khác nhau. Chương trình sau đây giúp ta tính số các hỗn độn trên tập n phần tử và so sánh tỷ lệ với số các hoán vị của tập này.

```
// Chương trình tính số các hỗn độn  
#include <stdio.h>  
#include <math.h>  
short i , n ;  
float a , t ;  
long gt , hd ;
```

```

void main()
{
    clrscr() ;
    printf("Cho biet luc luong cua tap hop : ") ; scanf("%d", &n) ;
    gt = 1 ;
    for (i = 2 ; i <= n ; i++) gt *= i ;
    printf("\n So cac hoan vi la: %ld \n", gt) ;
    t = a = 1 ;
    for (i = 1 ; i <= n ; i++) { a = -a / i ; t += a ; }
    hd = ceil(gt * t) ;
    printf("So cac hon don la: %ld \n", hd) ;
    i = ceil(100.0*hd/gt) ;
    printf("Ty le: %d %\n", i) ;
    getch() ;
}

```