Tạp chí Tin học và Điều khiển học, T.29, S.1 (2013), 79–91

NGUYÊN LÝ KẾ THỪA VÀ MỘT SỐ BÀI TOÁN DÃY BỊ CHẶN*

HOÀNG CHÍ THÀNH

Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội

Tóm tắt. Bài báo đề xuất nguyên lý kế thừa trong việc thiết kế các thuật toán tổ hợp. Dựa trên nguyên lý kế thừa phát triển bài toán dãy bị chặn được đề xuất trong [5] thành một số bài toán dãy bị chặn dạng đặc biệt và giải quyết chúng bằng các thuật toán ngắn gọn. Áp dụng các thuật toán này cho các bài toán tổ hợp có nhiều ứng dụng như: bài toán tập con, bài toán tập con bội, bài toán tập con k-phần tử và bài toán phân hoạch tập hợp.

Từ khóa. Nguyên lý kế thừa, bài toán dãy bị chặn, bài toán tổ hợp.

Abstract. In this paper, we propose an inheritance principle for combinatorial algorithm design. Based on the principle we extend the bounded sequence problem presented in [4] to some special bounded sequence problems and solve them by shorter algorithms. Then we apply these algorithms to solve some well-known combinatorial problems, such as subset problem, multi-subset problem, k-element subset problem and partition problem.

Key words. Inheritance principle, bounded sequence problems, combinatorial problems.

1. MỞ ĐẦU

Các khái niệm tập con, tập con bội, phân hoạch... được ứng dụng nhiều trong tính toán khoa học và xử lý thông tin, đặc biệt là trong các bài toán mang tính "vét cạn". Do vậy, việc xây dựng các thuật toán hữu hiệu để sinh ra các tập con của một tập hợp, các tập con bội của một tập bội, các phân hoạch của một tập hợp... vẫn được nhiều người quan tâm.

Đã có một số thuật toán giải bài toán tập con của tập hợp, bài toán tập con bội của tập bội, bài toán tập con k-phần tử, bài toán phân hoạch [1, 2, 4]. Những bài toán này có nhiều ứng dụng trong thực tế. D. Singh et al. đã tổng quan một số ứng dụng của tập bội trong toán học và công nghệ thông tin [3]...

Trong [5] tác giả đã đề xuất và giải quyết bài toán dãy bị chặn bằng một thuật toán ngắn gọn. Các nghiệm của bài toán được sắp xếp mang tính "kế thừa" nên độ phức tạp tính toán của thuật toán là khá nhỏ. Dựa trên bài toán này chúng tôi phát triển việc thiết kế các thuật toán tổ hợp thành nguyên lý kế thừa và áp dụng nó cho một số dạng đặc biệt của bài toán dãy bị chặn và các bài toán tổ hợp cụ thể. Những thuật toán mới được xây dựng góp phần phát triển tính toán tổ hợp và có thể ứng dụng tốt trong thực tế. Hơn nữa, những kết quả này còn chỉ ra rằng, nhiều bài toán tổ hợp khác nhau có chung một ngôn ngữ thể hiện, đó là bài toán dãy bị chặn.

*Bài báo được thực hiện với sự hỗ trợ từ Trung tâm Hỗ trợ Nghiên cứu Châu Á & Quỹ Giáo dục Cao học Hàn Quốc

Bố cục bài báo gồm: Mục 2 trình bày nguyên lý kế thừa khi giải quyết một bài toán, đặc biệt là bài toán tổ hợp. Mục 3 nhắc lại bài toán dãy bị chặn và trình bày thuật toán giải nó dựa trên nguyên lý kế thừa và một số ứng dụng. Hai dạng đặc biệt của bài toán dãy bị chặn được đề xuất trong các phần tiếp theo. Mục 4 dành cho bài toán dãy đơn điệu bị chặn. Mục 5 trình bày bài toán dãy bị chặn với ràng buộc. Cuối mỗi phần đều có các ứng dụng cụ thể. Cuối cùng là kết luận nêu một số hướng phát triển về lý thuyết và ứng dụng của bài toán dãy bị chặn.

2. NGUYÊN LÝ KẾ THỪA

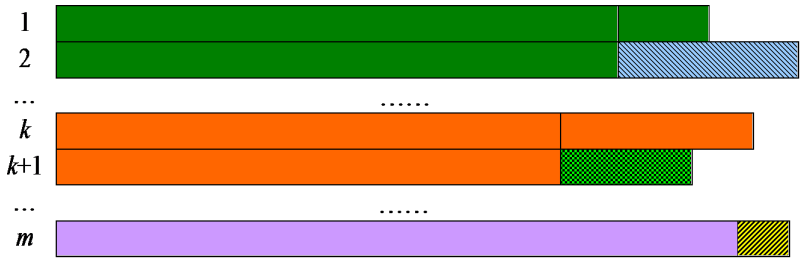
Bài toán tổ hợp là bài toán mà các ràng buộc của nó là sự kết nối của nhiều điều kiện. Bài toán liệt kê, bài toán đếm, bài toán chọn,... là những ví dụ điển hình cho bài toán tổ hợp.

Một bài toán tổ hợp thường có nhiều nghiệm. Để tìm các nghiệm này ta sắp xếp chúng thành một dãy theo một thứ tự nào đó, sao cho:

(1) Nghiệm đầu tiên rất dễ tìm.

(2) Nghiệm sau là "tổ hợp" một phần lớn nhất có thể của nghiệm trước đó và một phần mới.

Các tính chất (1)-(2) ở trên chính là "*nguyên lý kế thừa*", một nguyên lý chỉ đạo quan trọng trong việc thiết kế các thuật toán tổ hợp.



Hình 2.1. Sự kế thừa của các nghiệm

Trong mỗi bước lặp để tìm nghiệm, trước hết ta tìm vị trí thay đổi. Phần bên trái của vị trí thay đổi sẽ là phần kế thừa. Ta chỉ việc sao chép phần này sang cho nghiệm mới. Phần bên phải từ vị trí thay đổi trở đi là phần mới cần tìm. Thông thường, phần kế thừa càng lớn thì phần mới cần tìm càng nhỏ. Do vậy, khối lượng tính toán để tìm nghiệm của bài toán cũng nhỏ theo. So với các thuật toán khác, thuật toán tổ hợp dựa trên nguyên lý kế thừa thường là tối ưu.

Thứ tự hay được dùng để sắp xếp dãy các nghiệm là thứ tự từ điển hoặc thứ tự từ điển ngược.

Cấu trúc chung của một thuật toán tổ hợp được xây dựng dựa trên nguyên lý kế thừa như sau:

1) Nhập dữ liệu đầu vào.
2) Chọn nghiệm đầu tiên.
3) Vòng lặp:
   - Tìm vị trí thay đổi
   - Thay đổi
4) Lặp lại 3) cho đến khi hết nghiệm.

Trong bài báo này, nguyên lý kế thừa được ứng dụng để xây dựng thuật toán cho một số bài toán tổ hợp quen thuộc và có nhiều ứng dụng.

3. BÀI TOÁN DÃY BỊ CHẶN

Ký hiệu Z là tập các số nguyên. Cho n là một số nguyên dương nào đó. Giả sử p và q là hai dãy số nguyên độ dài n, với:

$$p = \langle p_1 p_2 ... p_n \rangle \text{ và } q = \langle q_1 q_2 ... q_n \rangle, \ \forall i \in \{1, 2, ..., n\} : p_i, q_i \in Z.$$

Định nghĩa 3.1. Ta nói rằng,

1) $p \leq q$ khi và chỉ khi $\forall i \in \{1, 2, ..., n\} : p_i \leq q_i$.

2) $p < q$ khi và chỉ khi $\forall i \in \{1, 2, ..., n\} : p_i \leq q_i$ và $\exists j \in \{1, 2, ..., n\} : p_j < q_j$.

3.1. Bài toán dãy bị chặn

Ta nhắc lại bài toán dãy bị chặn đã được đưa ra trong [5] như sau:

Cho hai dãy số nguyên s và g độ dài n, mà $s < g$. Hãy tìm tất cả các dãy số nguyên t độ dài n sao cho $s \leq t \leq g$.

Giả sử $s = \langle s_1 s_2 ... s_n \rangle$ và $g = \langle g_1 g_2 ... g_n \rangle$. Hai dãy số này được gọi là các dãy biên. Các dãy cần tìm $t = \langle t_1 t_2 ... t_n \rangle$ phải thỏa mãn

$$\forall i = 1, 2, ..., n : t_i \in Z \wedge s_i \leq t_i \leq g_i. \tag{3.1}$$

Ví dụ 3.2. Cho hai dãy biên $s = \langle 2\ 1\ 0\ 3 \rangle$ và $g = \langle 3\ 1\ 1\ 5 \rangle$. Tất cả các dãy số nguyên t thỏa mãn $s \leq t \leq g$ được sắp tăng dần theo thứ tự từ điển trong bảng dưới đây

No	Các dãy bị chặn		
1	2 1 0 3	7	3 1 0 3
2	2 1 0 4	8	3 1 0 4
3	2 1 0 5	9	3 1 0 5
4	2 1 1 3	10	3 1 1 3
5	2 1 1 4	11	3 1 1 4
6	2 1 1 5	12	3 1 1 5

Thành phần t_1 của dãy cần tìm có thể nhận một trong các giá trị $s_1, s_1 + 1, s_1 + 2, ..., g_1$. Vậy nó có $(g_1 - s_1 + 1)$ cách chọn.

Thành phần t_2 có thể nhận một trong các giá trị $s_2, s_2 + 1, s_2 + 2, ..., g_2$. Vậy nó có $(g_2 - s_2 + 1)$ cách chọn,...

Tương tự, thành phần t_n có thể nhận một trong các giá trị $s_n, s_n + 1, s_n + 2, ..., g_n$. Vậy nó có $(g_n - s_n + 1)$ cách chọn.

Suy ra số các dãy bị chặn cần tìm là: $\prod_{i=1}^{n}(g_i - s_i + 1)$.

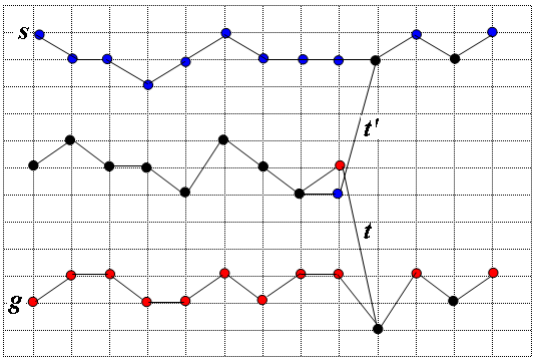
Bây giờ ta tìm các dãy số nguyên này.

3.2. Thuật toán sinh các dãy bị chặn

Mỗi dãy số nguyên $t = \langle t_1\ t_2\ ...\ t_n\rangle$ độ dài n có thể xem như một từ có độ dài n trên bảng chữ cái Z. Vậy ta có thể sắp xếp các từ này tăng dần theo thứ tự từ điển:

- Từ đầu tiên (nhỏ nhất) sẽ là $s_1\ s_2\ ...\ s_n$, tương ứng với dãy số s,
- Từ cuối cùng (lớn nhất) sẽ là $g_1\ g_2\ ...\ g_n$, tương ứng với dãy số g.

Xuất phát từ dãy đầu tiên $\langle s_1\ s_2\ ...\ s_n\rangle$ ta xây dựng vòng lặp để sinh ra các dãy còn lại. Giả sử $t = \langle t_1\ t_2\ ...\ t_n\rangle$ là một dãy số độ dài n thỏa mãn (3.1) vừa tìm được. Ta cần phải tìm dãy số $t' = \langle t'_1\ t'_2\ ...\ t'_n\rangle$ thỏa mãn (3.1) kế tiếp sau dãy số trên trong dãy được sắp tăng.



Hình 3.1. Sắp xếp tăng theo thứ tự từ điển của các dãy bị chặn

Theo thứ tự từ điển, dãy số t' được kế thừa phần bên trái nhiều nhất có thể của dãy số t từ thành phần đầu tiên đến thành phần thứ $i-1$, với

$$i = \max\{j | t_j < g_j\}.$$

Khi đó, các thành phần từ đầu tiên đến thứ $i-1$ sẽ được giữ nguyên

$$t'_j = t_j,\ j = 1, 2, ..., i-1.$$

Các thành phần từ vị trí thay đổi i đến thành phần cuối cùng được xác định như sau

$$t'_i = t_i + 1,\ \text{và}\ t'_j = s_j,\ j = i+1, i+2, ..., n.$$

Quá trình sinh các dãy số kết thúc khi đã sinh ra tất cả các dãy số thỏa mãn (3.1). Nghĩa là khi dãy số lớn nhất $\langle g_1\ g_2\ ...\ g_n\rangle$ được sinh ra. Khi đó thì $i = 0$. Đây cũng là điều kiện kết thúc của thuật toán.

Từ những lý luận trên ta xây dựng thuật toán chi tiết sinh các dãy bị chặn như sau.

Thuật toán 3.1. Sinh các dãy bị chặn

Đầu vào: Số nguyên dương n, hai dãy số biên $\langle s_1\ s_2\ ...\ s_n\rangle$ và $\langle g_1\ g_2\ ...\ g_n\rangle$
Đầu ra: Dãy các dãy số $\langle t_1, t_2, ..., t_n\rangle$ thỏa mãn (3.1) được sắp tăng theo thứ tự từ điển

```
1  Begin
2    input n; S[i] ⟵ s_i, G[i] ⟵ g_i, i = 1,2,...,n;
3    T[1..n] ⟵ S[1..n];
4    repeat
5      print dãy số T[1..n];
6      i ⟵ n;
```

7 **while** $T[i] = G[i]$ **do**
8 $\{T[i] \longleftarrow S[i];\ i \longleftarrow i - 1\}$;
9 **if** $i \geq 1$ **then** $T[i] \longleftarrow T[i] + 1$;
10 **until** $i = 0$;
11 **End.**

Độ phức tạp của Thuật toán 3.1:

Các câu lệnh 5–9 sinh ra một dãy số mới với độ phức tạp $O(n)$. Vậy độ phức tạp tổng thể của thuật toán là $O(n \times \prod_{i=1}^{n}(g_i - s_i + 1))$.

Độ phức tạp tổng thể của thuật toán xấp xỉ bằng $O(n.a^n)$, với

$$a = \max(g_1 - s_1 + 1, g_2 - s_2 + 1, ..., g_n - s_n + 1).$$

3.3. Một số ứng dụng của bài toán dãy bị chặn

Một số bài toán tổ hợp có thể đưa về bài toán dãy bị chặn. Do vậy, ta có thể áp dụng Thuật toán 3.1 để tìm nghiệm cho các bài toán này một cách nhanh chóng. Trong phần này, sẽ áp dụng thuật toán trên cho bài toán tập con của tập hợp và bài toán tập con bội của tập bội.

1) Bài toán tập con của tập hợp

Giả sử X là một tập hợp n-phần tử. Hãy tìm tất cả các tập con của tập X.

Ký hiệu $X = \{x_1, x_2, ..., x_n\}$. Mỗi tập con $A \subseteq X$ có thể biểu diễn bằng một dãy nhị phân độ dài n, $\langle b_1, b_2...b_n\rangle$, với:

$$b_i = \begin{cases} 1, \text{ nếu } x_i \in A \\ 0, \text{ ngược lại.} \end{cases}$$

Số tất cả các tập con của tập X là 2^n. Việc tìm tất cả các tập con của tập X đưa về việc tìm tất cả các dãy nhị phân độ dài n.

Các dãy $\langle b_1, b_2...b_n\rangle$ cần tìm phải thỏa mãn:

$$0 \leq b_i \leq 1,\ b_i \in \{0, 1\}. \tag{3.2}$$

Vậy bài toán tìm các dãy nhị phân độ dài n có thể đưa về bài toán dãy bị chặn với hai dãy biên là $s = \langle 0\ 0\ ...\ 0\rangle$ và $g = \langle 1\ 1\ ...\ 1\rangle$. Khi đó, ta có thuật toán đơn giản tìm các dãy nhị phân như sau:

Thuật toán 3.2. Sinh các tập con của tập hợp

Đầu vào: Số nguyên dương n
Đầu ra: Dãy các dãy nhị phân độ dài n biểu diễn các tập con của tập n phần tử được sắp tăng theo thứ tự từ điển

1 **Begin**
2 input n;
3 $B[1..n] \longleftarrow 0$;
4 **repeat**
5 print $B[1..n]$;
6 $i \longleftarrow n$;
7 **while** $B[i] = 1$ **do** $\{B[i] \longleftarrow 0;\ i \longleftarrow i - 1\}$;

8 **if** $i \geq 1$ **then** $B[i] \longleftarrow 1$;
9 **until** $i = 0$;
10 **End.**

Độ phức tạp của Thuật toán 3.2:

Dễ thấy rằng, độ phức tạp tổng thể của thuật toán là $O(n.2^n)$.

Thuật toán này đơn giản, ngắn gọn và là một trong các thuật toán sinh các tập con có độ phức tạp nhỏ nhất. Thuật toán sinh các tập con thường được sử dụng trong các bài toán chọn, mật mã và xử lý ảnh [1, 2]...

2) Bài toán tập con bội của tập bội

Tập bội (multi-set) là tập hợp, mà nó cho phép các phần tử xuất hiện nhiều hơn một lần. Chẳng hạn, tập giá trị của các biến trong một chương trình, bộ đánh dấu trong một hệ mạng...

Tập bội được viết dưới dạng tổng quát như sau:

$$X = (k_1 \times x_1, k_2 \times x_2, ..., k_n \times x_n), \quad \text{trong đó} \quad k_i \geq 0,\ i = 1, 2, ..., n.$$

Các phần tử $x_1, x_2, ..., x_n$ được gọi là cơ sở của tập bội, còn $k_1, k_2, ..., k_n$ là bội của các phần tử tương ứng.

Tập con bội của một tập bội là một tập bội mà bội của mỗi phần tử cơ sở trong tập con không vượt quá bội của phần tử này trong tập mẹ.

Giả sử, tập bội $A = (t_1 \times x_1, t_2 \times x_2, ..., t_n \times x_n)$.

$$A \subseteq X \Longleftrightarrow \forall i = 1, 2, ..., n,\ 0 \leq t_i \leq k_i. \tag{3.3}$$

Người ta thường biểu diễn tập bội A bằng dãy số nguyên $\langle t_1\ t_2\ ...\ t_n\rangle$ độ dài n bao gồm bội của các phần tử cơ sở trong tập này. Chẳng hạn, tập bội $A = (3 \times x_1, 4 \times x_2, 1 \times x_3, 5 \times x_4)$ có thể biểu diễn bởi dãy số $\langle 3\ 4\ 1\ 5\rangle$.

Bài toán. Cho tập bội $X = (k_1 \times x_1, k_2 \times x_2, ..., k_n \times x_n)$. Hãy tìm tất cả các tập con bội của tập này.

Với cách biểu diễn tập con bội như trên, bài toán tập con bội của tập bội có thể đưa về bài toán dãy bị chặn với hai dãy số biên là $s = \langle 0\ 0\ ...\ 0\rangle$ và $g = \langle k_1\ k_2\ ...\ k_n\rangle$.

Số tập con bội của tập bội X là: $\prod_{i=1}^{n}(k_1 + 1)$. Việc tìm tất cả các tập con bội A của tập bội X đưa về việc tìm tất cả các dãy số $\langle t_1\ t_2\ ...\ t_n\rangle$ thỏa mãn (3.3) biểu diễn các tập con bội cần tìm.

Áp dụng Thuật toán 3.1 ta có thuật toán tìm các tập con bội của tập bội như sau.

Thuật toán 3.3. Sinh các tập con bội

Đầu vào: Các số nguyên dương $n, k_1, k_2, ..., k_n$.
Đầu ra: Dãy các tập con bội mà các biểu diễn dãy số $\langle t_1 t_2 ... t_n\rangle$ của chúng được sắp tăng theo thứ tự từ điển.

1 **Begin**
2 input $n, K[i] \longleftarrow k_i,\ i = 1, 2, ..., n$;
3 $T[1..n] \longleftarrow 0$;
4 **repeat**
5 print $T[1..n]$;
6 $i \longleftarrow n$;

7 **while** $T[i] = K[i]$ **do** $\{T[i] \longleftarrow 0;\ i \longleftarrow i-1\}$;
9 **if** $i \geq 1$ **then** $T[i] \longleftarrow T[i]+1$;
10 **until** $i = 0$;
11 **End.**

Độ phức tạp của thuật toán:

Độ phức tạp tổng thể của thuật toán trên là $O((\prod_{i=1}^{n}(k_i+1)).n)$. Độ phức tạp của thuật toán xấp xỉ bằng $O(n.l^n)$, với $l = \max\{k_1, k_2, ..., k_n\}$.

Thuật toán này ngắn gọn và đơn giản hơn nhiều so với Thuật toán đệ quy 1.14 trình bày trong [2] để sinh các tập con bội của một tập bội. Ta xét ví dụ ứng dụng sau đây.

Ví dụ 3.3. Bài toán xếp ba lô

Giả sử ta có n loại đồ vật quý và một ba lô có dung tích c. Loại đồ vật thứ i có số lượng là k_i và mỗi đồ vật loại này có giá trị là p_i và dung lượng là s_i $(i = 1, 2, ..., n)$. Hãy chọn và xếp các đồ vật vào ba lô sao cho tổng giá trị các đồ vật xếp trong ba lô là lớn nhất có thể.

Ký hiệu đồ vật loại thứ nhất là x_1, loại thứ hai là x_2, ... và loại thứ n là x_n. Tập các đồ vật trên có thể biểu diễn bằng tập bội: $X = (k_1 \times x_1, k_2 \times x_2, ..., k_n \times x_n)$. Mỗi phương án chọn các đồ vật để xếp vào ba lô chính là một tập con bội A của tập bội trên, sao cho

$$A = (t_1 \times x_1, t_2 \times x_2, ..., t_n \times x_n),\ A \subseteq X \ \text{ và } \sum_{i=1}^{n} t_i.s_i \leq c.$$

Phương án tối ưu là phương án mà $\sum_{i=1}^{n} t_i.p_i$ đạt giá trị lớn nhất.

Kết hợp Thuật toán 3.3 sinh các tập con bội và thuật toán tìm số lớn nhất ta sẽ nhận được nghiệm tối ưu của bài toán một cách nhanh chóng.

Chẳng hạn, xét bài toán trên với dữ liệu được cho trong bảng dưới đây.

Loại đồ vật	a	b	c	d	e
Số lượng	3	5	2	4	6
Giá trị	25	28	20	12	15
Dung lượng	10	11	8	5	6

và ba lô có dung lượng là 100.

Tính toán theo các thuật toán trên, ta nhận được phương án xếp ba lô tối ưu là $A = (0 \times a, 5 \times b, 2 \times c, 1 \times d, 4 \times e)$ với tổng dung lượng là 100 (đầy ba lô) và tổng giá trị lớn nhất là 252.

Bài toán xếp ba lô được ứng dụng nhiều trong các hệ thống truyền tin và mạng máy tính.

4. BÀI TOÁN DÃY ĐƠN ĐIỆU BỊ CHẶN

Khi thêm một số ràng buộc trên các dãy bị chặn ta sẽ nhận được bài toán dãy bị chặn dạng đặc biệt.

4.1. Bài toán dãy đơn điệu bị chặn

Trong Bài toán 3.1 dãy bị chặn, nếu đòi hỏi hai dãy biên s, g và các dãy bị chặn t là đơn điệu tăng thì ta có bài toán dãy đơn điệu bị chặn.

Xuất phát từ dãy số đơn điệu tăng đầu tiên $\langle s_1\ s_2\ ...\ s_n\rangle$ ta xây dựng vòng lặp để sinh ra các dãy số đơn điệu tăng còn lại. Giả sử $t = \langle t_1\ t_2\ ...\ t_n\rangle$ là một dãy số đơn điệu tăng độ dài n thỏa mãn (3.1) vừa tìm được. Ta cần phải tìm dãy số $t' = \langle t'_1\ t'_2\ ...\ t'_n\rangle$ đơn điệu tăng thỏa mãn (3.1), kế tiếp sau dãy số trên trong dãy được sắp tăng.

Theo thứ tự từ điển, dãy số t' được kế thừa phần bên trái nhiều nhất có thể của dãy số t từ thành phần đầu tiên đến thành phần thứ $i-1$, với

$$i = \max\{j | t_j < g_j\}.$$

Khi đó, các thành phần từ đầu tiên đến thứ $i-1$ sẽ được giữ nguyên:

$$t'_j = t_j,\ j = 1, 2, ..., i-1.$$

Các thành phần từ vị trí thay đổi i đến thành phần cuối cùng được xác định như sau:

$$t'_i = t_i + 1,\ \text{và}\ t'_j = \max(t'_{j-1} + 1, s_j)\ \text{với}\ j = i+1, i+2, ..., n.$$

Từ đó, ta có thuật toán sinh các dãy đơn điệu bị chặn như sau.

Thuật toán 4.1. Sinh các dãy đơn điệu bị chặn
Đầu vào: Số nguyên dương n, hai dãy số biên đơn điệu tăng $\langle s_1\ s_2\ ...\ s_n\rangle$ và $\langle g_1\ g_2\ ...\ g_n\rangle$.
Đầu ra: Dãy các dãy số đơn điệu tăng $\langle t_1, t_2, ..., t_n\rangle$ thỏa mãn (3.1) được sắp tăng theo thứ tự từ điển.

```
1  Begin
2    input n, S[i] ⟵ s_i, G[i] ⟵ g_i, i = 1,2,...,n;
3    T[1..n] ⟵ S[1..n];
4    repeat
5      print T[1..n];
6      i ⟵ n;
7       while T[i] = G[i] do i ⟵ i − 1;
8       if i ≥ 1 then for j ⟵ n downto i do A[j] ⟵ max(A[i] + j − i + 1, S[j]);
10     until i = 0;
11 End.
```

Độ phức tạp của Thuật toán 4.1:

Tương tự như Thuật toán 3.1, Thuật toán 4.1 có độ phức tạp tổng thể là

$$O(n.\prod_{i=1}^{n}(g_i - s_i + 1)).$$

4.2. Ứng dụng cho bài toán tập con *k*-phần tử

Giả sử X là một tập hợp n phần tử và k là một số nguyên không âm.

Bài toán đặt ra là tìm tất cả các tập con k-phần tử của tập X.

Đồng nhất tập n-phần tử X với tập các số nguyên dương $\{1, 2, ..., n\}$. Khi đó, mỗi tập con k-phần tử của tập X tương ứng 1-1 với một dãy các số nguyên dương đơn điệu tăng độ dài k. Mỗi dãy như thế có thể xem như là một từ trên bảng chữ cái $\{1, 2, ..., n\}$.

Sắp xếp các từ trên theo thứ tự từ điển. Từ đầu tiên (nhỏ nhất) là $\langle 1\ 2\ ...\ k-1\ k\rangle$ và từ cuối cùng (lớn nhất) là $\langle n-k+1\ n-k+2\ ...\ n-1n\rangle$.

Vậy bài toán tìm các tập con k-phần tử của tập n phần tử có thể đưa về bài toán tìm các dãy đơn điệu bị chặn với hai dãy biên là $\langle 1\ 2\ ...\ k-1\ k\rangle$ và $\langle n-k+1\ n-k+2\ ...\ n-1\ n\rangle$. Áp dụng Thuật toán 4.1, ta có thuật toán ngắn gọn sau đây sinh các tập con k-phần tử của tập n phần tử.

Thuật toán 4.2. Sinh các tập con k-phần tử
Đầu vào: Hai số nguyên không âm n và k
Đầu ra: Dãy tất cả các tập con k phần tử của tập hợp n phần tử được sắp tăng theo thứ tự từ điển

```
1   Begin
2    input n, k;
3     for i ⟵ 1 to k do A[i] ⟵ i;
4    p ⟵ k;
5     while p ≥ 1 do
6      { print A[1..k];
7      if A[k] < n then p ⟵ k else p ⟵ p − 1;
8      if p ≥ 1 then
9        for i ⟵ k downto p do A[i] ⟵ A[p] + i − p + 1};
10  End.
```

Độ phức tạp của thuật toán:

Chu trình 5-9 sinh ra một tập con k phần tử với độ phức tạp $O(k)$. Vậy độ phức tạp tổng thể của Thuật toán 4.2 là $O(\binom{n}{k}.k)$.

5. BÀI TOÁN DÃY BỊ CHẶN VỚI RÀNG BUỘC

Trở lại với Bài toán 3.1, nếu ta đòi hỏi các dãy bị chặn t thỏa mãn ràng buộc (*) nào đó thì ta có bài toán dãy bị chặn với ràng buộc.

Ràng buộc (*) thường được cho dưới dạng một bất biến đối với các dãy bị chặn cần tìm. Khi đó, trong Thuật toán 3.1 thay câu lệnh 5 như sau:

5 **if** Dãy số $T[1..n]$ thỏa mãn ràng buộc (*) **then** print $T[1..n]$;

ta có một thuật toán đơn giản sinh các dãy bị chặn với ràng buộc bất kỳ.

Ứng dụng cho bài toán phân hoạch

Bài toán phân hoạch đặt ra như sau: Cho tập n phần tử X. Hãy tìm tất cả các phân hoạch của tập này.

Mỗi phân hoạch của tập X là một họ các tập con không rỗng, đôi một không giao nhau và hợp các tập con này lại phải cho ta đúng tập X.

Ký hiệu $\Pi(X)$ là tập tất cả các phân hoạch của tập X. Số lượng các phân hoạch của tập n phần tử là số Bell B_n được xác định theo công thức đệ quy sau đây [2, 4]:

$$B_n = \sum_{i=0}^{n-1} \binom{n-1}{i} B_i, \text{ trong đó } B_0 = 1.$$

Giả sử $\pi = \{A_1, A_2, ..., A_k\}$ là một phân hoạch của tập X. Mỗi tập con A_i được gọi là một khối của phân hoạch. Để đảm bảo tính duy nhất của biểu diễn, các khối trong một phân hoạch được sắp xếp theo thứ tự tăng dần của phần tử bé nhất nằm trong mỗi khối.

Phần tử $x \in X$ thuộc vào một khối A_i nào đó sẽ có chỉ số i, nghĩa là mỗi phần tử có thể biểu diễn bởi chỉ số của khối chứa nó. Vậy mỗi phân hoạch sẽ được biểu diễn bởi một dãy n chỉ số. Với biểu diễn dãy chỉ số thì số khối của phân hoạch bằng chính thành phần lớn nhất trong dãy chỉ số. Dãy này có thể xem như là một từ có độ dài n trên bảng chữ cái X. Ta sắp xếp các từ này tăng dần theo thứ tự từ điển. Khi đó thì,
- Từ nhỏ nhất (đầu tiên) sẽ là: 1 1 1 ... 1 - tương ứng với phân hoạch $\{\{1, 2, 3, ..., n\}\}$ chỉ gồm một khối là toàn bộ tập X.
- Từ lớn nhất (cuối cùng) sẽ là: 1 2 3 ... n - tương ứng với phân hoạch $\{\{1\}, \{2\}, \{3\}, ..., \{n\}\}$ có n khối, mỗi khối chỉ có một phần tử.

Dùng mảng nguyên $A[1..n]$ để chứa dãy chỉ số của một phân hoạch trong máy tính. Phần tử $A[i]$ lưu chỉ số của khối chứa phần tử i.
- Phần tử 1 luôn thuộc khối thứ nhất ($A[1] = 1$).
- Phần tử 2 chỉ có thể thuộc khối thứ nhất hoặc khối thứ hai.
- Nếu phần tử 2 thuộc khối thứ nhất thì phần tử 3 chỉ có thể thuộc khối thứ nhất hoặc khối thứ hai; còn nếu phần tử 2 thuộc khối thứ hai thì phần tử 3 có thể thuộc khối thứ nhất, thứ hai hoặc thứ ba.
- Do vậy, phần tử i chỉ có thể thuộc các khối: $1, 2, ..., \max(A[1], A[2], ..., A[i-1]) + 1$. Điều này có nghĩa là:

$$1 \le A[i] \le \max(A[1], A[2], ..., A[i-1]) + 1 \le i. \tag{5.1}$$

Ký hiệu Φ là tập tất cả các dãy số nguyên độ dài n với thành phần đầu tiên bằng 1, các thành phần còn lại thuộc tập $\{1, 2, ..., n\}$ và thỏa mãn (5.1). Lưu ý rằng, mỗi dãy trong tập Φ đều chứa tất cả các số nguyên từ 1 đến số lớn nhất nằm trong dãy.

Xây dựng ánh xạ $f : \Pi(X) \longrightarrow \Phi$ như sau:

$$\forall \pi \in \Pi(X),\ f(\pi) = A[1..n],$$

trong đó $A[1..n]$ là biểu diễn dãy chỉ số của phân hoạch p được xác định như ở trên.

Định lý 5.1. *Ánh xạ f là một song ánh từ tập* $\Pi(X)$ *các phân hoạch lên tập* Φ *các dãy số nguyên thỏa mãn (5.1).*

Chứng minh:

Trước hết, ta chứng minh rằng f là một đơn ánh. Giả sử β, γ là hai phân hoạch nào đó của tập X và $\beta \neq \gamma$. Phân hoạch $\beta = \{B_1, B_2, ..., B_k\}$ có biểu diễn dãy chỉ số là $B[1..n]$ và phân hoạch $\gamma = \{C_1, C_2, ..., C_l\}$ có biểu diễn dãy chỉ số là $C[1..n]$. Ta phải chỉ ra rằng $B[1..n] \neq C[1..n]$.

Xét hai trường hợp:
• $k \neq l$, hai phân hoạch trên có số khối khác nhau. Khi đó, các số lớn nhất trong mỗi dãy $B[1..n]$ và $C[1..n]$ là khác nhau. Vậy hai dãy $B[1..n]$ và $C[1..n]$ là khác nhau.
• $k = l$, hai phân hoạch trên có chung số khối. Vì hai phân hoạch β, γ là khác nhau nên phải tồn tại ít nhất một phần tử $i \in X$ sao cho $B[i] \neq C[i]$. Vậy thì $B[1..n] \neq C[1..n]$.

Bây giờ ta chỉ ra rằng ánh xạ f là một toàn ánh. Giả sử $A[1..n]$ là một dãy số nguyên nào đó thuộc tập Φ. Xây dựng họ η các tập con của tập X như sau

$$\eta = \{A_1, A_2, ..., A_k\}, \quad \text{trong đó } k = \max(A[i],\ i = 1, 2, ..., n),$$

và
$$A_j = \{i | i \in X,\ A[i] = j\}, \quad \text{với } j = 1, 2, ..., k. \tag{5.2}$$

Theo nhận xét trên, các số nguyên $1, 2, ..., k$ đều xuất hiện trong dãy $A[1..n]$, vậy các tập con A_j đều khác rỗng. Mỗi phần tử i chỉ có một chỉ số $A[i]$ nên nó chỉ thuộc vào một tập con, nghĩa là các tập con trên đôi một rời nhau. Dãy số nguyên $A[1..n]$ có độ dài n nên hợp các tập con A_j lại phải cho ta tập X. Điều này chứng tỏ họ các tập con $\eta = \{A_1, A_2, ..., A_k\}$ là một phân hoạch của tập X nhận $A[1..n]$ là biểu diễn dãy chỉ số của nó. Ánh xạ f là một toàn ánh.

Vậy thì, f là một song ánh từ tập $\Pi(X)$ lên tập Φ. ■

Công thức (5.2) cho ta một cách đơn giản để xác định phân hoạch từ dãy chỉ số.

Định lý 5.1 chỉ ra rằng tập Φ bao gồm tất cả các dãy chỉ số của các phân hoạch trong tập $\Pi(X)$. Vậy để tìm các phân hoạch của tập n phần tử X ta chỉ cần tìm tất cả các dãy số nguyên $A[1..n]$ thuộc tập Φ được sắp tăng theo thứ tự từ điển.

Dùng mảng nguyên $Maxi[1..n]$ để lưu dãy chỉ số lớn nhất có thể. Cụ thể là:

$$Maxi[1] = 0;\ Maxi[i] = \max(A[1], A[2], ..., A[i-1]),\ i = 2, 3, ..., n.$$

Hiển nhiên,

$$Maxi[i] = \max(Maxi[i-1], A[i-1]) < i,\ i = 2, 3, ..., n. \tag{5.3}$$

Xuất phát từ dãy chỉ số đầu tiên là: 1 1 1 ... 1 ta xây dựng vòng lặp để sinh ra các dãy chỉ số còn lại.

Giả sử $A[1..n]$ là một dãy chỉ số vừa tìm được. Ta cần phải tìm dãy chỉ số $A'[1..n]$ kế tiếp sau dãy chỉ số trên trong dãy được sắp tăng. Theo thứ tự từ điển, dãy chỉ số $A'[1..n]$ được kế thừa phần bên trái nhiều nhất có thể của dãy $A[1..n]$từ tọa độ 1 đến tọa độ thứ $i-1$, với

$$i = \max\{j | A[j] \le Maxi[j]\}.$$

Khi đó, các tọa độ từ tọa độ đầu tiên đến tọa độ thứ $i-1$ sẽ được giữ nguyên:

$$A'[j] = A[j],\ j = 1, 2, ..., i-1.$$

Các tọa độ từ vị trí thay đổi i đến tọa độ cuối cùng được xác định như sau:

$$A'[i] = A[i] + 1, \text{ và } A'[j] = 1,\ j = i+1, i+2, ..., n.$$

Việc sinh các dãy chỉ số kết thúc khi dãy chỉ số cuối cùng: 1 2 3 ... $n-1$ n được sinh ra. Khi đó thì

$$A[n] = n.$$

Đây cũng là điều kiện kết thúc của thuật toán.

Dựa vào các bất biến (5.1) và (5.3) của các dãy chỉ số và thứ tự từ điển, ta xây dựng thuật toán lặp sinh các phân hoạch như sau.

Thuật toán 5.2. Sinh các phân hoạch của một tập hợp

Đầu vào: Số nguyên dương n
Đầu ra: Dãy các phân hoạch của tập $\{1, 2, ..., n\}$ mà các biểu diễn chỉ số của chúng được sắp tăng theo thứ tự từ điển

```
1  Begin
2    input n;
3    A[1..n − 1] ⟵ 1;
4    A[n] ⟵ Maxi[1] ⟵ 0;
5     repeat
6      for i ⟵ 2 to n do
7        {Maxi[i] ⟵ Maxi[i − 1];
8        if Maxi[i] < A[i − 1] then Maxi[i] ⟵ A[i − 1]; }
9      i ⟵ n;
10      while A[i] = Maxi[i] + 1 do {A[i] ⟵ 1; i ⟵ i − 1};
11      A[i] ⟵ A[i] + 1;
12      print phân hoạch tương ứng với dãy chỉ số A[1..n];
13     until A[n] = n;
14   End.
```

Độ phức tạp của thuật toán:

Các câu lệnh 2-4 trong phần khởi tạo có độ phức tạp $O(n)$.

Mỗi vòng lặp 6-12 sinh và in ra một phân hoạch:
- Chu trình 6-8 xây dựng mảng $Maxi$ với độ phức tạp $O(n)$.
- Các câu lệnh 9-11 tìm vị trí thay đổi i trên mảng chỉ số và xây dựng mảng chỉ số mới có độ phức tạp $O(n)$.
- Các câu lệnh 12 in phân hoạch có độ phức tạp $O(n)$.

Vòng lặp 5-13 tìm hết B_n phân hoạch. Vậy độ phức tạp tổng thể của Thuật toán 5.2 là $O(n.B_n)$.

Thuật toán trên ngắn gọn và đơn giản hơn rất nhiều so với Thuật toán 1.19 trong [2] sinh các phân hoạch bằng cách di chuyển một phần tử từ khối này sang khối khác nhờ các con trỏ nguyên.

Thuật toán sinh các phân hoạch cùng các thuật toán trình bày ở trên có thể ứng dụng tốt vào các bài toán khai phá dữ liệu và đặc biệt là bài toán phân lớp dữ liệu. Những ứng dụng thuật toán sinh phân hoạch tập hợp vào các bài toán phân lớp dữ liệu, đặc biệt là các bài toán phân lớp dữ liệu chuỗi theo thời gian (time-series data) cùng các ứng dụng trong y tế, chứng khoán, môi trường,...

6. KẾT LUẬN

Bài báo đã đề xuất nguyên lý kế thừa để áp dụng trong việc thiết kế các thuật toán tối ưu, đặc biệt là các thuật toán tổ hợp. Từ đó phát triển bài toán dãy bị chặn cùng ứng dụng của nó và đề xuất hai dạng đặc biệt của bài toán trên là bài toán dãy đơn điệu bị chặn và bài toán dãy bị chặn với ràng buộc. Đồng thời cũng chỉ ra rằng bốn bài toán tổ hợp quen thuộc và có nhiều ứng dụng trong thực tế là: Bài toán tập con, bài toán tập con bội của tập bội, bài toán tập con k-phần tử và bài toán phân hoạch đều có thể đưa về bài toán dãy bị chặn. Áp dụng thuật toán của bài toán dãy bị chặn tổng quát để xây dựng các thuật toán cho bốn bài

toán tổ hợp trên. Các thuật toán này đều có thể song song hóa bằng kỹ thuật phân đoạn dãy nghiệm như trình bày trong [6, 7]. Những kết quả nhận được góp phần phát triển các thuật toán tổ hợp và có thể ứng dụng trong thực tế tính toán.

Việc nghiên cứu song song hóa các thuật toán đã xây dựng và phát triển bài toán dãy bị chặn trong trường hợp các dãy biên thay đổi và áp dụng vào một số bài toán tính toán khác là bài toán cần giải quyết.

TÀI LIỆU THAM KHẢO

[1] K.J. Batenburg and J. Sijbers, Generic iterative subset algorithms for discrete tomography, *Discrete Applied Mathematics* **157** (3) (2009) 438–451.

[2] W. Lipski, *Kombinatoryka dla programistów*, WNT, Warszawa, 1982.

[3] D. Singh, A. M. Ibrahim, T. Yohanna and J. N. Singh, An overview of the applications of multisets, *Novi Sad J. Math.* **37** (2) (2007) 73–92.

[4] H.C. Thành, *Giáo trình Tổ hợp*, NXB ĐHQG Hà Nội, 1999.

[5] H.C. Thanh, Bounded sequence problem and some its applications, *Proceedings of Japan - Vietnam Workshop on Software Engineering*, 2010 (74–83).

[6] H.C. Thanh and N.Q. Thanh, An efficient parallel algorithm for the set partition problem, *Studies in Computational Intelligence*, Springer, Vol. **351** (2011) 25–32.

[7] H.C. Thanh, Parallel generation of permutations by inversion vectors, *Proceedings of IEEE-RIVF International Conference on Computing and Communication Technologies, IEEE*, 2012 (129–132).

Ngày nhận bài 19 - 3 - 2012
Ngày lại sau sửa ngày 30 - 8 - 2012