Tạp chí Tin học và Điều khiển học, T. 18, S. 4 (2002), 295–308

THUẬT TOÁN PHÂN RÃ LỚP THÔNG TIN CÓ CẤU TRÚC DẠNG CÂY NHỊ PHÂN VỚI THÔNG TIN CHỨA Ở ĐỈNH TRONG TRÊN TẬP KHÓA VÔ HẠN

ĐỖ ĐỨC GIÁO, PHẠM NGỌC HÙNG VÀ ĐỖ VIỆT HÙNG

Abstract. The notion of a binary search tree plays an important role in computer science, especially in the theory of data structures. For that reason we can find many papers concerned with theory of search tree in the literature. After having read these papers we noticed that, above all, questions of the optimal construction and inductive generation of search trees are studied, where equivalent transformations of search trees are often used. In this paper we will proper the theorem, which says that every binary search tree can be uniquely transformed into an optimal binary search tree by using a composition algorithm for the infinite set of keys of binary search trees with information in node.

Tóm tắt. Khái niệm về cây nhị phân đóng vai trò rất quan trọng trong khoa học máy tính, đặc biệt là trong lý thuyết cấu trúc dữ liệu. Nhiều kết quả nghiên cứu dựa trên các phép biến đổi tương đương tìm thấy trong các tài liệu đã công bố. Trong bài báo này chúng tôi sử dụng thuật toán phân dã để tìm cây tối ưu nhị phân trên tập khóa vô hạn với các thông tin chứa ở đỉnh trong của cây.

1. ĐẶT VẤN ĐỀ

Trong quá trình lưu trữ các thông tin dưới dạng cây tìm kiếm nhị phân, người ta thường đưa ra một tiêu chuẩn nào đó để đánh giá và tổ chức lại các thông tin dưới dạng cây tìm kiếm tối ưu và xây dựng bảng mã cho các thông tin được lưu trữ trong cây tối ưu đó.

Một trong những phương pháp hay dùng để tổ chức lại các thông tin và lưu trữ nó dưới dạng cây nhị phân là phương pháp tiên đề do H.Thiele nêu ra trong [5]. Từ kết quả đó của H.Thiele, người ta đã xây dựng các thuật toán tương đương và thuật toán tìm cây tối ưu dựa trên việc xét duyệt các khóa của cây đã cho ([1,..., 4]). Chính vì vậy khi tập khóa của cây quá lớn thì thuật toán tương đương và thuật toán tối ưu sẽ bị kéo dài quá trình tính toán nhất là khi tập khóa là tập vô hạn. Để khắc phục nhược điểm trên, trong bài báo này, chúng ta áp dụng thuật toán phân rã để tìm cây tối ưu và lưu trữ các thông tin đã được phân loại trong cây tối ưu đó. Thuật toán phân rã dựa trên việc chia nhỏ tập khóa ban đầu thành n tập con ($n-1$ tập hữu hạn và 1 tập vô hạn một phía). Sau đó áp dụng phương pháp tiên đề của H.Thiele để tìm cây chuẩn tắc trên từng tập con đó. Công việc này có thể tiến hành đồng thời trên tất cả các tập khóa con. Tiếp theo ghép nối các cây chuẩn tắc của từng tập con theo quy luật các khóa có giá trị tăng dần ta được cây chuẩn tắc trên toàn tập khóa ban đầu. Từ đó dùng thuật toán bẻ đôi cây chuẩn ta được cây tối ưu trên tập khóa ban đầu.

2. CÂY TỐI ƯU TRÊN TẬP KHÓA VÔ HẠN K

2.1. Cây nhị phân một chiều với thông tin chứa ở đỉnh trong trên tập khóa vô hạn

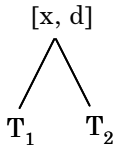
Giả sử D là tập không rỗng các documents nào đó và mỗi phần tử của nó ta gọi là một thông tin, K là một tập bất kì các phần tử mà trên đó thỏa mãn quan hệ so sánh, tức là với mọi phần tử $x, y \in K$ bao giờ cũng thoả mãn $x \leq y$ hoặc $x > y$. Không mất tính tổng quát và để cho đơn giản ta lấy K là tập số nguyên dương $K = \{1, 2, 3, ...\}$. K được gọi là tập khóa của các cây nhị

phân. Ta ký hiệu τ là cây rỗng và đặt $D_+ = D \cup \{\tau\}$.

Định nghĩa 1. (Định nghĩa đệ qui cây nhị phân với thông tin chứa ở đỉnh trong)

a) τ là một cây.

b) Nếu T_1, T_2 là hai cây và $x \in K,\ d \in D$ thì dãy ký hiệu $[x, d]\langle T_1, T_2\rangle$ cũng là một cây. Dạng đồ thị của cây $[x, d]\langle T_1, T_2\rangle$ như sau:



$[x, d]$ là đỉnh trong của cây, chứa khóa x và thông tin d, T_1 là cây con bên trái, T_2 là cây con bên phải. Đỉnh ngoài hay còn gọi là lá chứa ký hiệu rỗng. Tập tất cả các cây định nghĩa như trên ký hiệu qua TREE và gọi là tập các cây nhị phân với thông tin chứa ở đỉnh trong trên tập khóa K (gọi tắt là cây nhị phân).

Định nghĩa 2. (Định nghĩa hàm đánh giá hay hàm kết quả)

Ta ký hiệu "$\equiv$" ("$\not\equiv$") để chỉ sự đồng nhất bằng nhau (khác nhau) giữa các cây.

Giả sử T là một cây nhị phân và $l \in K$. Ta định nghĩa hàm f : TREE$\times K \to D_+$, như sau:

a) Nếu $T \equiv \tau$ thì $f(T, l) = f(\tau, l) = \tau$ với mọi $l \in K$

b) Nếu $T \equiv [x, d]\langle T_1, T_2\rangle$ thì:

$$f(T,l) = \begin{cases} f(T_1,l) & \text{nếu } l < x \\ d & \text{nếu } l = x \\ f(T_2,l) & \text{nếu } l > x \end{cases}$$

Định nghĩa 3. (Định nghĩa sự tương đương)

Giả sử $T_1, T_2 \in$ TREE. Ta nói T_1 là tương đương với T_2 trên K (ký hiệu $T_1 \approx (K)T_2$) khi và chỉ khi $f(T_1, l) = f(T_2, l)$ với mọi $l \in K$.

Từ định nghĩa trên, ta nói T_1 không tương đương với T_2 trên K (Ký hiệu $T_1 \not\approx (K)T_2$) nếu tồn tại một khóa $l_0 \in K$ sao cho $f(T_1, l_0) \neq f(T_2, l_0)$.

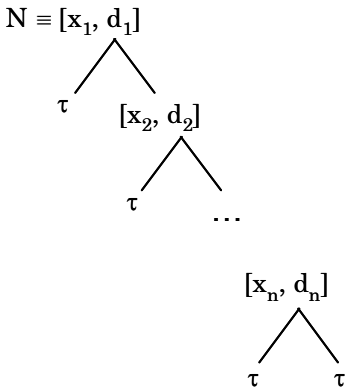
2.2. Cây chuẩn tắc trên tập khóa vô hạn K

Định nghĩa 4. (Định nghĩa cây chuẩn tắc). Cây $N \in$ TREE gọi là cây chuẩn tắc nếu N là một trong hai dạng sau:

a) $N \equiv \tau$

b) Nếu $N \not\equiv \tau$ thì $N \equiv [x_1, d_1] < \tau,\ [x_2, d_2] < \tau, ..., [x_n, d_n] < \tau,\ \ \tau > ... >$ Với $x_i \in K,\ d_i \in D$ và $x_1 < x_2 < ... < x_n$.

Dạng đồ thị của cây chuẩn tắc N:



Nếu $N, T \in$ TREE và $N \approx (K)T$ thì ta nói N là cây chuẩn tắc của T trên K.

2.3. Bảng mã của cây trên tập K

Giả sử $T \in$ TREE. Bảng mã của T ta ký hiệu là $m(T)$ và định nghĩa $m(T) = \{x/x \in K$ sao cho $f(T, x) \neq \tau\}$. Nếu $T \equiv \tau$ thì $m(T) = \varnothing$.

Nếu $N \equiv [x_1, d_1] < \tau,\ [x_2, d_2] < \tau, ..., [x_n, d_n] < \tau,\ \tau > ... >$ thì $m(N) = \{x_1, x_2, ..., x_n\}$, ở đây x_i là mã của d_i $(i = 1, 2, ..., n)$.

2.4. Cây tối ưu trên tập khóa vô hạn K

Trước hết ta đưa vào một số ký hiệu sau:

a. $\gamma(T) =$ số các đỉnh của cây T (kể cả lá)

b. Deep $(\tau) =$ số các cung của đường đi từ gốc của cây T đến lá τ trong T.

c. $h(T)$ là chiều cao của cây T và $h(T) = \max$ Deep$(\tau)/\tau$ là lá của $T\}$.

Định nghĩa 5. (Định nghĩa cây tối ưu trên K)

Cây $T_0 \in$ TREE gọi là cây tối ưu trên tập K, nếu T_0 thoả mãn các điều kiện sau đây:

a. $\gamma(T_0) = \min\{\gamma(T)\}/T \in$ TREE và $T \approx (K)T_0$.

b. $h(T_0) = \min\{h(T)\}/T \in$ TREE và $T \approx (K)T$.

c. $|\text{Deep}\tau_i) - \text{Deep}(\tau_j)| \leq 1$ với mọi $i \neq j$

d. Khóa ở đỉnh cha bất kỳ trong cây T_0 nhỏ hơn khóa ở đỉnh cây con phải và lớn hơn khóa ở đỉnh cây con trái.

3. THUẬT TOÁN PHÂN RÃ TÌM CÂY TỐI ƯU TRÊN TẬP KHÓA VÔ HẠN K

Input: Cây $T \in$ TREE và tập khóa vô hạn K

Output: Cây tối ưu $T_0 \approx (K)T$.

Trong [2], người ta đã chỉ ra thủ tục thực hiện tìm cây tối ưu T_0 của T trên tập vô hạn K. Ở đây ta dùng thuật toán phân rã để giải quyết bài toán trên với mục đích rút ngắn thời gian tính toán trong việc tìm kiếm cây tối ưu trên tập khóa vô hạn K.

3.1. Phân hoạch tương đương trên tập khóa vô hạn K

Chia K thành n tập con $K_1, K_2, ..., K_n$ sao cho chúng thỏa mãn đồng thời ba điều kiện sau:

1. $\bigcup_{i=1}^{n} K_i = K$
2. $K_i \cap K_j = \varnothing$ $(i \# j)$.
3. Mọi phần tử $x \in K_i$ và $y \in K_{i+1}$ đều thỏa mãn $x < y$ với $i = 1, 2, ..., n-1$.

Các tập $K_1, K_2, ..., K_n$ thỏa mãn ba điều kiện trên gọi là phân hoạch tương đương trên tập khóa K.

3.2. Quy tắc dẫn xuất

Trước hết ta đưa vào khái niệm phương trình cây. Giả sử $T_1, T_2 \in$ TREE và "=" là một ký hiệu không thuộc tập $K \cup D_+ \cup \{[,], <, >\}$. Khi đó dãy ký hiệu $T_1 = T_2$ gọi là một phương trình cây. Đặt EQU $=\{T_1 = T_2 / T_1, T_2 \in \text{TREE}\}$. EQU được gọi là tập tất cả những phương trình cây trên TREE.

Giả sử $X \subseteq$ EQU và $T_1 = T_2 \in$EQU.

Định nghĩa 6. (Định nghĩa dẫn được) Phương trình cây $T_1 = T_2$ là dẫn được từ X (ký hiệu $X \vdash T_1 = T_2$) khi và chỉ khi $T_1 = T_2 \in X$ hoặc $T_1 = T_2$ dẫn được từ các phần tử trong X bằng cách áp dụng các qui tắc dẫn xuất sau:

Quy tắc 1(R1): $X \vdash T = T$ với mọi $T \in$TREE

Quy tắc 2(R2): Nếu $X \vdash T_1 = T_2$ thì $X \vdash T_2 = T_1$

Quy tắc 3(R3): Nếu $X \vdash T_1 = T_2$ và $X \vdash T_2 = T_3$ thì $X \vdash T_1 = T_3$

Quy tắc 4(R4): Nếu $X \vdash T_1 = T_1'$ thì $X \vdash [x, d] < T_1, T_2 >= [x, d] < T_1', T_2 >$

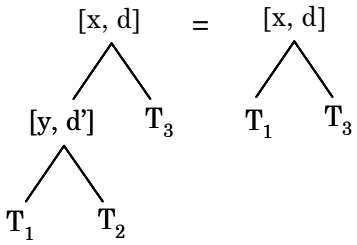
Quy tắc 5(R5): Nếu $X \vdash T_2 = T_2'$ thì $X \vdash [x, d] < T_1, T_2 >= [x, d] < T_1, T_2' >$

3.3. Các tiên đề của TREE trên tập khóa vô hạn K

Do K vô hạn một phía bên phải nên trong K tồn tại phần tử bé nhất mà ta ký hiệu là x_{min}.

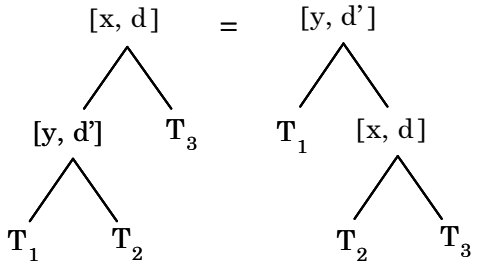
Tiên đề ax_1. $[x, d] < [y, d'] < T_1, T_2 >, T_3 >= [x, d] < T_1, T_3 >$ *là một tiên đề nếu* $x \leq y$

Dạng đồ thị:



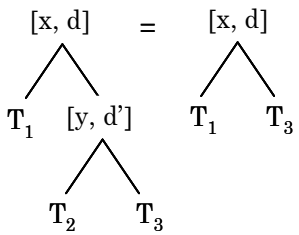
Tiên đề ax_2. $[x, d] < [y, d'] < T_1, T_2 >, T_3 >= [y, d'] < T_1, [x, d] < T_2, T_3 >>$ *là một tiên đề nếu* $y < x$

Dạng đồ thị:



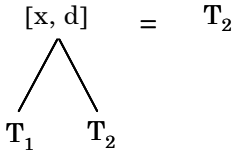
Tiên đề ax_3. $[x, d] < T_1, [y, d'] < T_2, T_3 >>= [x, d] < T_1, T_3 >$ *là một tiên đề nếu* $y \leq x$

Dạng đồ thị:



Tiên đề ax_4. $[x,d] < T_1, T_2 >= T_2$ *là một tiên đề nếu* $x < x_{min}$

Dạng đồ thị:



Đặt $AX = \{ax_1, ax_2, ax_3, ax_4\}$, và gọi AX là hệ tiên đề trên tập khóa vô hạn K.

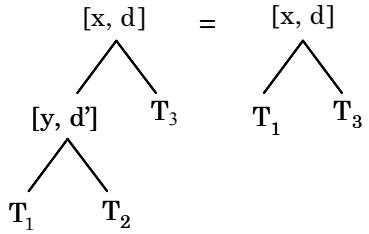
Trong [2] người ta đã chỉ ra được thuật toán tìm cây tối ưu T_0 của T trên tập khóa vô hạn K. Nhưng để tìm được T_0 cần phải duyệt toàn bộ khóa trong K, nên độ phức tạp lớn, quá trình tính toán chậm lại. Để khắc phục nhược điểm này ta xây dựng hệ tiên đề trên mỗi tập khóa con K_i.

3.4. Các tiên đề của TREE trên tập khóa hữu hạn K_i $(i = 1, 2, ..., n-1)$

Do K_i hữu hạn nên trong K_i tồn tại phần tử bé nhất và phần tử lớn nhất mà ta ký hiệu tương ứng là $x_{i\min}, x_{i\max}$.

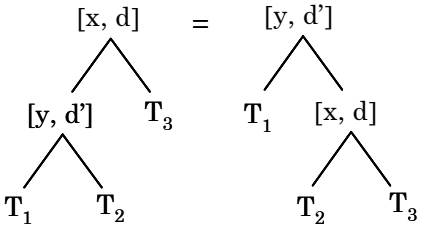
Tiên đề ax_{i1}. $[x,d] < [y,d'] < T_1, T_2 >, T_3 >= [x,d] < T_1, T_3 >$ *là một tiên đề nếu* $x_{i\min} \le x < y \le x_{i\max}$

Dạng đồ thị:



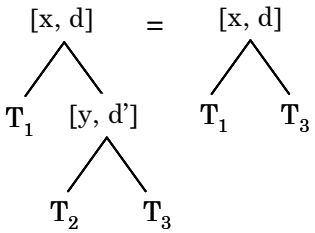
Tiên đề ax_{i2}. $[x,d] < [y,d'] < T_1, T_2 >, T_3 >= [y,d'] < T_1,\ [x,d] < T_2, T_3 >>$ *là một tiên đề nếu* $x_{i\min} \le y < x \le x_{i\max}$

Dạng đồ thị:



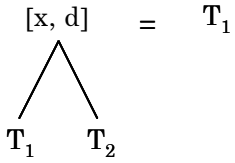
Tiên đề ax_{i3}. $[x,d] < T_1, [y,d'] < T_2, T_3 >>= [x,d] < T_1, T_3 >$ *là một tiên đề nếu* $x_{i\min} \le y \le x \le x_{i\max}$

Dạng đồ thị:



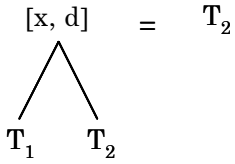
Tiên đề ax_{i4}. $[x,d] < T_1, T_2 >= T_1$ *là một tiên đề nếu* $x > x_{i\max}$

Dạng đồ thị:



Tiên đề ax_{i5}. $[x,d] < T_1, T_2 >= T_2$ *là một tiên đề nếu* $x < x_{i\min}$

Dạng đồ thị:



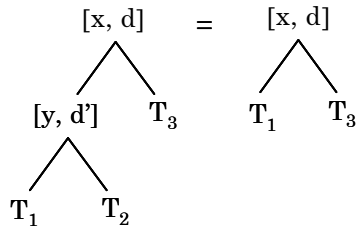
Đặt $AX_i = \{ax_{i1}, ax_{i2}, ax_{i3}, ax_{i4}, ax_{i5}\}$ $(i = 1, 2, ..., n-1)$ và gọi AX_i là hệ tiên đề trên tập khóa hữu hạn K_i.

3.5. Các tiên đề của TREE trên tập khóa vô hạn K_n

Do K_n vô hạn một phía bên phải nên trong K_n tồn tại phần tử bé nhất mà ta ký hiệu là $x_{n\min}$.

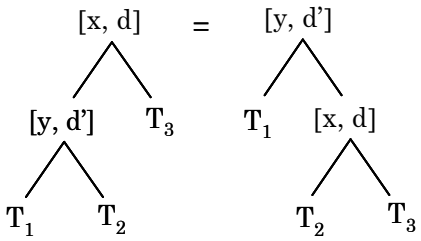
Tiên đề ax_{n1}. $[x,d] < [y,d'] < T_1, T_2 >, T_3 >= [x,d] < T_1, T_3 >$ *là một tiên đề nếu* $x < y$

Dạng đồ thị:



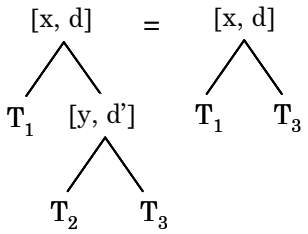
Tiên đề ax_{n2}. $[x,d] < [y,d'] < T_1, T_2 >, T_3 >= [y,d'] < T_1, [x,d] < T_2, T_3 >>$ *là một tiên đề nếu* $y < x$

Dạng đồ thị:



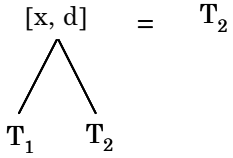
Tiên đề ax_{n3}. $[x,d] < T_1, [y,d'] < T_2, T_3 >>= [x,d] < T_1, T_3 >$ *là một tiên đề nếu* $y \leq x$

Dạng đồ thị:



Tiên đề ax_{n4}. $[x,d] < T_1, T_2 >= T_2$ *là một tiên đề nếu* $x < x_{n\min}$

Dạng đồ thị:



Đặt $AX_n = \{ax_{n1}, ax_{n2}, ax_{n3}, ax_{n4}\}$ và gọi AX_n là hệ tiên đề trên tập khóa hữu hạn K_n.

3.6. Cây chuẩn tắc trên K_i $(i = 1, 2..., n)$. Cây chuẩn tắc trên K_i định nghĩa như cây chuẩn tắc trên K (Định nghĩa 4).

Định lý 1. (Tính duy nhất của cây chuẩn tắc)

Giả sử N_1, N_2 là hai cây chuẩn tắc trên K_i. Khi đó nếu $N_1 \approx (K_i)N_2$ thì $N_1 \equiv N_2$.

Chứng minh. Dùng phương pháp phản chứng dựa vào định nghĩa của cây chuẩn tắc.

Định lý 2. (Sự tồn tại dạng chuẩn tắc)

Với mọi $T \in TREE$ tồn tại duy nhất cây chuẩn tắc N_i trên K_i sao cho:

a. $AX_i \vdash T = N_i$

b. $T \approx (K_i)N_i$

Chứng minh. Tính duy nhất của cây chuẩn tắc suy từ Định lý 1. Phần b. suy ra từ phần a. Do mỗi tiên đề trên các K_i là một phương trình cây gồm hai cây tương đương nhau và các quy tắc dẫn xuất bảo tồn tính tương đương. Phần a được chứng minh dựa vào định nghĩa đệ quy của cây T.

3.7. Cây tối ưu trên tập $K_i (i = 1, 2, ..., n)$. Cây tối ưu T_{i0} trên tập K_i cũng định nghĩa tương tự như cây tối ưu trên tập K (Định nghĩa 5).

Định lý 3. (Sự tồn tại của cây tối ưu trên K_i)

Với mỗi cây $T \in TREE$ bao giờ cũng tồn tại cây $T_{i0} \in TREE$ thỏa mãn đồng thời các điều kiện sau:

a. $AX_i \vdash T = T_{i0}$.

b. $T_{i0} \approx (K_i)T$

c. $\gamma(T_{i0}) = \min\{\gamma(T)/T \in TREE$ *và* $T \approx (K_i)T_{i0}\}$

d. $h(T_{i0}) = \min\{(h(T)/T \in TREE$ *và* $T \approx (K_i)T_{i0}\}$

e. $|Deep(\tau_i) - Deep(\tau_j)| \leq 1$ *với mọi* $i \neq j$

f. Khóa ở đỉnh cha bất kỳ của cây T_{i0} nhỏ hơn khóa ở đỉnh cây con bên phải và lớn hơn khóa ở đỉnh cây con bên trái.

3.8. Thuật toán chuyển về cây chuẩn tắc trên tập khóa K_i $(i = 1, 2, ..., n-1)$

Mục đích của thuật toán tối ưu cây nhị phân là đưa một cây nhị phân bất kỳ về cây nhị phân tối ưu tương đương với nó trên tập khóa K_i. Quá trình này qua hai bước sau:

1. Đưa cây bất kỳ về dạng chuẩn tắc.
2. Từ dạng chuẩn tắc chuyển về dạng hoàn chỉnh.

Thuật toán 1.
Input: Cây $T \in$ TREE, tập khóa K_i $(i = 1, 2, ..., n-1)$.
Output: $N_i \in$ TREE là cây chuẩn tắc trên K_i, $N_i \approx (K_i)T$.

Bước 1:
+ Áp dụng tiên đề ax_{i4} với cây T (loại các đỉnh có khóa không thuộc tập khóa K_i)
+Áp dụng tiên đề ax_{i5} với cây T (loại các đỉnh có khóa không thuộc tập khóa K_i)

Bước 2: Kiểm tra xem N_i có phải là dạng chuẩn tắc không ?
+ Nếu đúng thì chuyển sang bước 4.
+ Nếu sai thì chuyển sang bước 3.

Bước 3: Thực hiện lần lượt các tiên đề sau:
+ Thực hiện tiên đề ax_{i1}.
+ Thực hiện tiên đề ax_{i2}.
+ Thực hiện tiên đề ax_{i3}
Quay lại bước 2.

Bước 4: Cây $N_i \approx (K_i)T$ là dạng chuẩn tắc.

Thuật toán 1 thực hiện đồng thời trên các tập khóa $K_1, K_2, ..., K_{n-1}$ và cho ta các dạng chuẩn tắc $N_1, N_2, ..., N_{n-1}$, ở đây $N_i \approx (K_i)T$ với $i = 1, 2, ..., n-1$.

3.9. Thuật toán tìm cây chuẩn tắc của T trên tập khóa vô hạn K_n

Input: Cây $T \in$ TREE, tập khóa vô hạn K_n.
Output: $N_n \in$ TREE là cây chuẩn tắc trên $K_n (N_n \approx (K_n)T)$

Thuật toán 2.

Bước 1: Áp dụng tiên đề ax_{n4} với cây T (loại các đỉnh không thuộc tập khóa vô hạn Kn)

Bước 2: Kiểm tra xem N_n có phải là dạng chuẩn tắc không ?
+ Nếu đúng thì chuyển sang bước 4.
+ Nếu sai thì chuyển sang bước 3.

Bước 3: Thực hiện lần lượt các tiên đề sau:
Thực hiện tiên đề ax_{n1}.
Thực hiện tiên đề ax_{n2}.
Thực hiện tiên đề ax_{n3}.
Quay lại bước 2.

Bước 4: Cây $N_n \approx (K_n)T$ là dạng chuẩn tắc.

3.10. Thuật toán nối các cây chuẩn tắc $N_i \approx (K_i)T$ $(i = 1, 2, ..., n)$ thành cây chuẩn tắc $N \approx (K)T$.

Thuật toán 3.

Input: Các cây chuẩn tắc $N_i \approx (K_i)T$ $(i = 1, 2, ..., n)$
Out put: Cây chuẩn tắc $N \approx (K)T$.

Bước 1: Khởi tạo N là cây rỗng. Khởi tạo biến đếm $i = 1$.

Bước 2: Kiểm tra $i > n$?
+ Nếu đúng thì sang bước 4.
+ Nếu sai thì sang bước 3.

Bước 3: Kiểm tra cây N_i có là cây rỗng không ?
+ Nếu là cây rỗng thì tăng $i = i + 1$ (loại bỏ cây rỗng ra khỏi quá trình ghép).

+ Nếu là cây không rỗng thì nối cây N_i vào cuối cây N theo nguyên tắc đồng nhất lá bên phải nhất trong N với đích trên cùng của N_i và sau đó tăng $i = i + 1$.

Quay lại bước 2.

Bước 4: Cây chuẩn tắc $N \approx (K)T$.

3.11. Thuật toán chuyển cây chuẩn tắc của T về cây tối ưu trên tập khóa vô hạn K

Sau khi thực hiện thủ tục nối các cây chuẩn tắc N_i trong thuật toán 3 (mục 3.10) ta được cây chuẩn tắc $N \approx (K)T$. Sau đó chuyển N về cây tối ưu $T_0 \approx (K)N \approx (K)T$.

Thuật toán 4.

Input: Cây chuẩn tắc $N \approx (K)T$.

Output: Cây tối ưu $T_0 \approx (K)N \approx (K)T$.

Bước 1: Dùng hàm Numr(p) để đo chiều cao của cây chuẩn tắc N (p là con trỏ trỏ đến đỉnh của N). Sau đó áp dụng (Numr(p) div 2) lần tiên đề $ax_{2\_}$ ngược.

Bước 2: Dùng hàm Numl(p) và hàm Numr(p) để đo chiều cao của cây con trái và cây con phải của N (giả sử ta đo được m, n).

+ Áp dụng (m div 2) lần tiên đề $ax_{2\_}$ thuận cho cây con trái của N.

+ Áp dụng (n div 2) lần tiên đề $ax_{2\_}$ nghịch cho cây con phải của N.

Bước 3: Chuyển sang xét cây con trái và cây con phải của N.

Quay lại bước 2.

Bước 4: Cây hiện thời T_0 là cây tối ưu.

3.12. Thuật toán phân rã xây dựng cây tối ưu trên tập khóa vô hạn K

Input: $T \in$ TREE, tập khóa vô hạn K.

Output: Cây tối ưu $T_0 \approx (K)T$.

Bước 1: Chia tập khóa vô hạn K thành n tập con $K_1, K_2, ..., K_n$ sao cho chúng tạo nên một phân hoạch tương trên K.

Bước 2:

+ Áp dụng Thuật toán 1 để xây dựng các cây chuẩn tắc N_i của T trên tập khóa hữu hạn K_i $(i = 1, 2, ..., n-1)$.

+ Áp dụng Thuật toán 2 để xây dựng các cây chuẩn tắc N_n của T trên tập khóa vô hạn K_n.

Bước 3: Áp dụng Thuật toán 3 để ghép các cây N_i $(i = 1, 2, ..., n)$ thành cây chuẩn tắc $N \approx (K)T$.

Bước 4: Áp dụng Thuật toán 4 để chuyển cây chuẩn N (xây dựng ở bước 3) ta được cây tối ưu $T_0 \approx (K)N \approx (K)T$.

4. GIẢI THUẬT CHO CÁC TIÊN ĐỀ

4.1. Lưu trữ cây nhị phân

Vì T là cây nhị phân nên ta dùng cách lưu trữ móc nối để lưu trữ cây. Mỗi nút của cây sẽ là một con trỏ gồm các trường được khai báo như sau:

```
Type TREE = ^Node;
    Node = Record
        Key: integer; { khóa của nút}
        Info: String; { thông tin được lưu trữ tại nút}
        Left: TREE; {Trỏ đến cây con trái}
        Right: TREE; {Trỏ đến cây con phải}
            x, y: Integer; {Tọa độ của nút trên màn hình đồ họa}
    End;
```

4.2. Mô tả thuật toán của tiên đề ax_1, ax_{i1} $(i = 1, 2, ..., n-1)$

Bước 1: Trỏ p vào đỉnh của cây T. Giả sử p_1 là nút con trái của p.

Nếu $\begin{cases} + \text{ Nút } p \text{ khác rỗng} \\ + \text{ Nút con trái } p_1 \text{ khác rỗng} \\ + \text{ Nút con trái của } p_1 \text{ hoặc nút con phải của } p_1 \text{ khác rỗng} \end{cases}$

thì chuyển sang bước 2, ngược lại chuyển sang bước 3.

Bước 2: Nếu giá trị khóa của p không lớn hơn giá trị khóa của p_1 thì:

+ Nút con trái của p sẽ là nút con trái của p_1.
+ Xóa nút p_1 ra khỏi cây.
+ Chuyển sang bước 3.

Ngược lại chuyển sang bước 3.

Bước 3: Chuyển sang xét nút con trái (hoặc nút con phải) của p. Sau đó quay lại bước 1.

4.3. Mô tả thuật toán của tiên đề $ax_{2\_}$ thuận, $ax_{i2\_}$ thuận $(i = 1, 2, ..., n-1)$

Bước 1: Trỏ p vào đỉnh của cây T. Giả sử p_1 là nút con trái của p.

Nếu $\begin{cases} + \text{ Nút } p \text{ khác rỗng} \\ + \text{ Nút con trái } p_1 \text{ khác rỗng} \\ + \text{ Nút con trái của } p_1 \text{ hoặc nút con phải của } p_1 \text{ khác rỗng} \end{cases}$

thì chuyển sang bước 2, ngược lại chuyển sang bước 3.

Bước 2: Nếu giá trị khóa của p lớn hơn giá trị khóa của p_1 thì:

+ p_1 sẽ là đỉnh.
+ Nút p sẽ là nút con phải.
+ Nút con phải của p_1 sẽ là nút con trái của p_r.
+ Chuyển sang bước 3.

Ngược lại, nếu khóa của p nhỏ hơn khóa của p_1 thì chuyển sang bước 3.

Bước 3: Chuyển sang xét nút con trái (hoặc nút con phải) của p, sau đó quay lại bước 1.

4.4. Mô tả thuật toán của tiên đề $ax_{2\_}$ nghịch, $ax_{i2\_}$ nghịch $(i = 1, 2, ..., n-1)$

Bước 1: Trỏ p vào đỉnh của cây T. Giả sử p_r là nút con phải của p.

Nếu $\begin{cases} + \text{ Nút } p \text{ khác rỗng} \\ + \text{ Nút con phải } p_r \text{ khác rỗng} \\ + \text{ Nút con trái của } p_r \text{ hoặc nút con phải của } p_r \text{ khác rỗng} \end{cases}$

thì chuyển sang bước 2, ngược lại chuyển sang bước 3.

Bước 2: Nếu giá trị khóa của p nhỏ hơn giá trị khóa của p_r thì:

+ p_r sẽ là đỉnh.
+ Nút p sẽ là nút con trái.
+ Nút con phải của p_1 sẽ là nút con trái của p_r.
+ Chuyển sang bước 3.

Ngược lại, nếu khóa của p lớn hơn khóa của p_r thì chuyển sang bước 3.

Bước 3: Chuyển sang xét nút con trái (hoặc nút con phải) của p. Sau đó quay lại bước 1.

4.5. Mô tả thuật toán của tiên đề ax_3 nghịch, ax_{i3} $(i = 1, 2, ..., n-1)$

Bước 1: Trỏ p vào đỉnh của cây T. Giả sử p_r là nút con phải của p.

Nếu $\begin{cases} + \text{ Nút } p \text{ khác rỗng} \\ + \text{ Nút con phải } p_r \text{ khác rỗng} \\ + \text{ Nút con trái của } p_1 \text{ hoặc nút con phải của } p_r \text{ khác rỗng} \end{cases}$

thì chuyển sang bước 2, ngược lại chuyển sang bước 3.

Bước 2: Nếu giá trị khóa của p không nhỏ hơn giá trị khóa của p_r thì:

+ Nút con phải của p sẽ là nút con phải của p_r.
+ Xóa nút p_r ra khỏi cây.
+ Chuyển sang bước 3.

Ngược lại, nếu khóa của p nhỏ hơn khóa của p_r thì chuyển sang bước 3.

Bước 3: Chuyển sang xét nút con trái (hoặc nút con phải) của p, sau đó quay lại bước 1.

4.6. Mô tả thuật toán của tiên đề ax_{i4} $(i = 1, 2, ..., n-1)$

Bước 1: Trỏ p vào đỉnh của cây T.

Nếu $\begin{cases} + \text{ Nút } p \text{ khác rỗng} \\ + \text{ Nút con phải } p_r \text{ khác rỗng} \\ + \text{ Nút con trái của } p_1 \text{ khác rỗng} \end{cases}$

thì chuyển sang bước 2, ngược lại chuyển sang bước 3.

Bước 2: Nếu giá trị khóa của p lớn hơn $x_{i\max}$ thì:

+ Nút con trái p_1 sẽ là nút đỉnh.
+ Xóa nút con phải p_r và nút p ra khỏi cây.
+ Chuyển sang bước 3.

Ngược lại, chuyển sang bước 3.

Bước 3: Chuyển sang xét nút con trái (hoặc nút con phải) của p. Sau đó quay lại bước 1.

4.7. Mô tả thuật toán của tiên đề ax_4, ax_{i5} $(i = 1, 2, ..., n-1)$

Bước 1: Trỏ p vào đỉnh của cây T.

Nếu $\begin{cases} + \text{ Nút } p \text{ khác rỗng} \\ + \text{ Nút con phải } p_r \text{ khác rỗng} \\ + \text{ Nút con trái của } p_1 \text{ khác rỗng} \end{cases}$

thì chuyển sang bước 2, ngược lại chuyển sang bước 3.

Bước 2: Nếu giá trị khóa của p nhỏ hơn $x_{i\min}(x_{n\min}$ với tiên đề $ax_{n4})$ thì:

+ Nút con phải p_r sẽ là đỉnh.
+ Xóa nút con trái p_1 và nút p ra khỏi cây.
+ Chuyển sang bước 3.

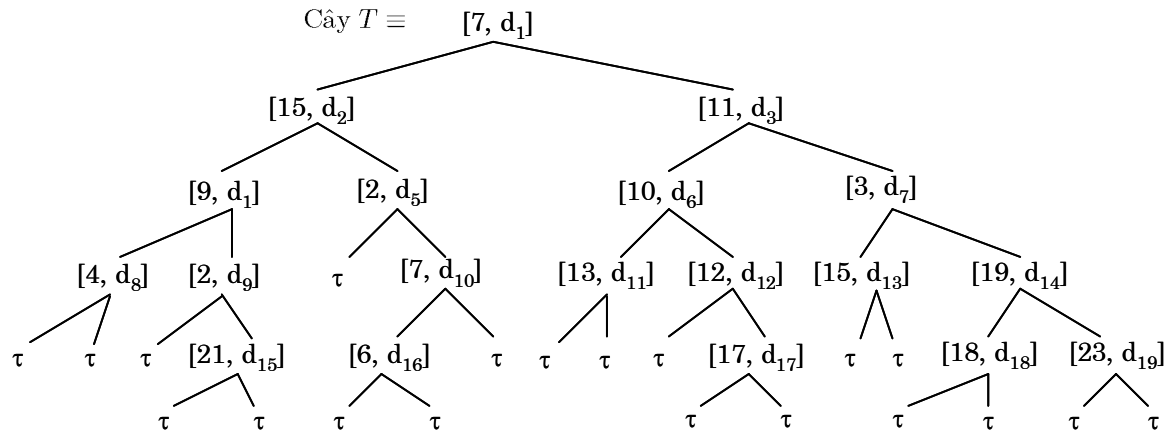
Ngược lại, chuyển sang bước 3.

Bước 3: Chuyển sang xét nút con trái (hoặc nút con phải) của p. Sau đó quay lại bước 1.

5. VÍ DỤ MINH HỌA

Bài báo này đã giải quyết đầy đủ bài toán: cho $T \in$ TREE tìm cây tối ưu của T tập khóa vô hạn bằng thuật toán phân rã. Nó có ý nghĩa trong việc nghiên cứu lớp thông tin có cấu trúc dạng cây nhị phân. Dưới đây ta xét một ví dụ của thuật toán phân rã trên tập khóa hữu hạn với mục đích mô phỏng hoạt động của thuật một cách trực quan.

Để cho đơn giản, ta chọn tập khóa $K = \{1, 2, ..., 25\}$



Bước 1: Chia tập K thành 3 tập con

$K_1 = \{1, 2, 3, 4, 5, 6, 8, 9, 10\}$,

$K_2 = \{11, 12, 13, 14, 15, 16, 17\}$

$K_3 = \{18, 19, 20, 21, 22, 23, 24, 25\}$.

Dễ thấy K_1, K_2 và K_3 là một phân hoạch tương đương trên K với:

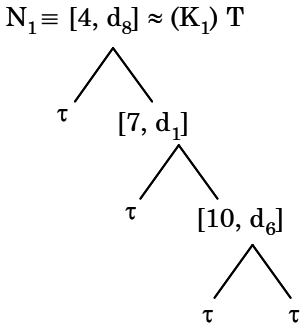
$x_{1\,\min} = 1, \quad x_{1\,\max} = 10,$

$x_{2\,\min} = 11, \quad x_{2\,\max} = 17,$

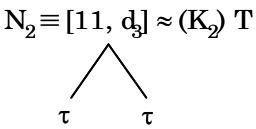
$x_{3\,\min} = 18, \quad x_{3\,\max} = 25,$

Bước 2: Áp dụng Thuật toán 1, trước hết loại bỏ các đỉnh có khóa không nằm trong K_i $(i = 1, 2, 3)$ bằng cách áp dụng ax_{i4}, ax_{i5}. Sau đó dùng các tiên đề $ax_{i1}, ax_{i2}, ax_{i3}$ để tìm cây chuẩn tắc N_i trên K_i $(i = 1, 2, 3)$.

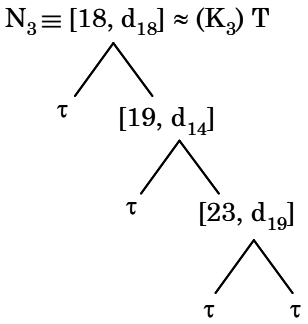
Dễ dàng kiểm tra N_1 là cây chuẩn tắc của T trên K_1 với N_1:



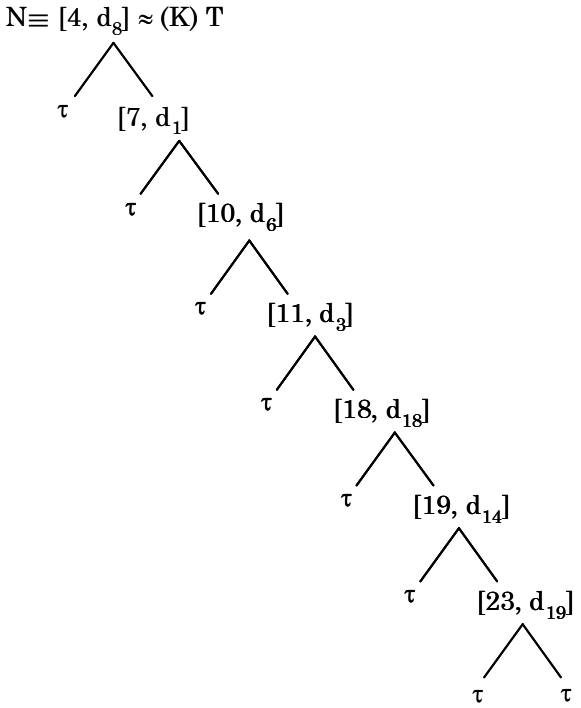
N_2 là cây chuẩn tắc của T trên K_2 với N_2 :



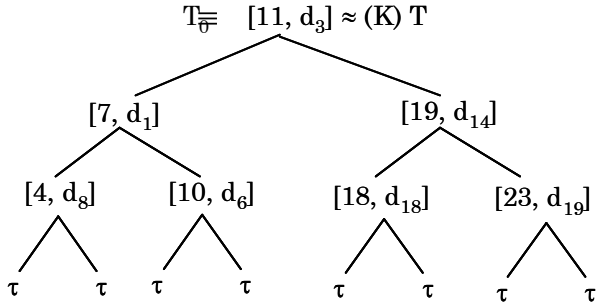
N_3 là cây chuẩn tắc của T trên với K_3 với N_3:



Bước 3: Áp dụng Thuật toán 3, ghép N_1, N_2, N_3 ta được cây chuẩn tắc N của T trên K với:



Bước 4: Sử dụng Thuật toán 4 ta được cây tối ưu cần tìm N_0 của T trên K là



6. KẾT LUẬN

Bài báo này là sự mở rộng về phương pháp luận của thuật toán phân rã trên tập khóa hữu hạn

với mục đích tăng tốc độ tính toán trong việc phân loại, tìm kiếm thông tin dưới dạng cây tìm kiếm nhị phân có kiểu dữ liệu động.

Phương pháp này có thể mở rộng cho lớp cây nhị phân với thông tin chứa ở các lá. Về mặt logic ta có thể mở rộng lớp cây một chiều trong bài báo này thành lớp cây n chiều và áp dụng thuật toán phân rã trên lớp cây n chiều. Vấn đề này chúng tôi sẽ đề cập trong bài báo tiếp theo.

TÀI LIỆU THAM KHẢO

[1] Do Duc Giao and A.M.Tjoa, Optimization for one dimensional binary search trees with information in leaf HNU. *Journal of . Science, Nat .Sci.* **2** (1995) 49 – 61.

[2] Đỗ Đức Giáo và Lê Anh Cường, Tối ưu hoá cây nhị phân một chiều với thông tin chứa ở các ở đỉnh trong trên tập khóa vô hạn *Tạp chí Tin học và Điều khiển học* **15** (4) (1999).

[3] Đỗ Đức Giáo, Đặng Thị Nga và Vũ Lê Tú , Tối ưu hoá cây nhị phân một chiều với thông tin chứa ở các ở đỉnh trong trên tập khóa hữu hạn *Tạp chí Tin học và Điều khiển học* **16** (3) (2000).

[4] Đỗ Đức Giáo, Nguyễn Thành Chung, Nguyễn Văn Hội, Tối ưu hoá lớp các thông tin có cấu trúc dạng cây nhị phân một chiều với thông tin chứa ở lá trên tập khóa hữu hạn *Tạp chí viễn thông, chuyên san các công trình nghiên cứu - triển khai viễn thông và công nghệ thông tin* (5) (2001) 35–44.

[5] H.Thiele , equivalent transformation of one demensional binary search trees with information in nodes *Pr.IPI.PAN.* (411) (1980) 87–88.

Nhận bài ngày 15 - 6 - 2001

Khoa Công nghệ - Đại học Quốc gia Hà nội