B-Tree

Xóa một khóa khỏi một B-cây

Thủ tục B-TREE-DELETE(x, k) để xóa khóa k khỏi cây con có gốc tại x bảo đảm rằng khi B-TREE-DELETE được gọi đệ quy lên x thì

- số khóa trong x phải $\geq t$ (bậc tối thiểu của cây).

Do đó, khi thủ tục thực thi, đôi khi một khóa được di chuyển (từ một nút thích hợp khác) vào một nút trước khi đệ quy xuống nút đó.

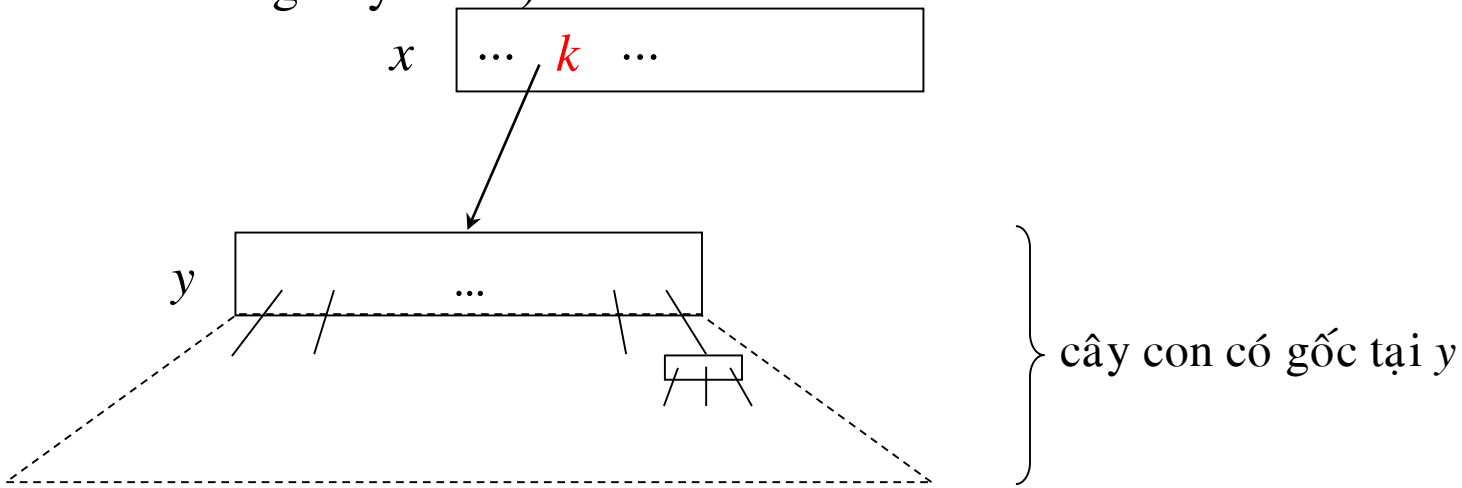
Xóa một khóa khỏi một B-cây

B-TREE-DELETE(x, k)

1. Nếu khóa k có trong nút x và x là một nút lá thì xóa k khỏi x, *rồi tái cân bằng (phần sau)*.

2. Nếu khóa k có trong nút x và x là một nút trong thì

a. Nếu nút con y ở trước k có ít nhất t khóa thì tìm *khóa trước* (predecessor) k' của k trong cây con có gốc tại y. Xóa k', trong nút x thay k bằng k'. (Tìm và xóa k' có thể thực thi trong một lượt đi xuống duy nhất.)



Xóa một khóa khỏi một B-cây

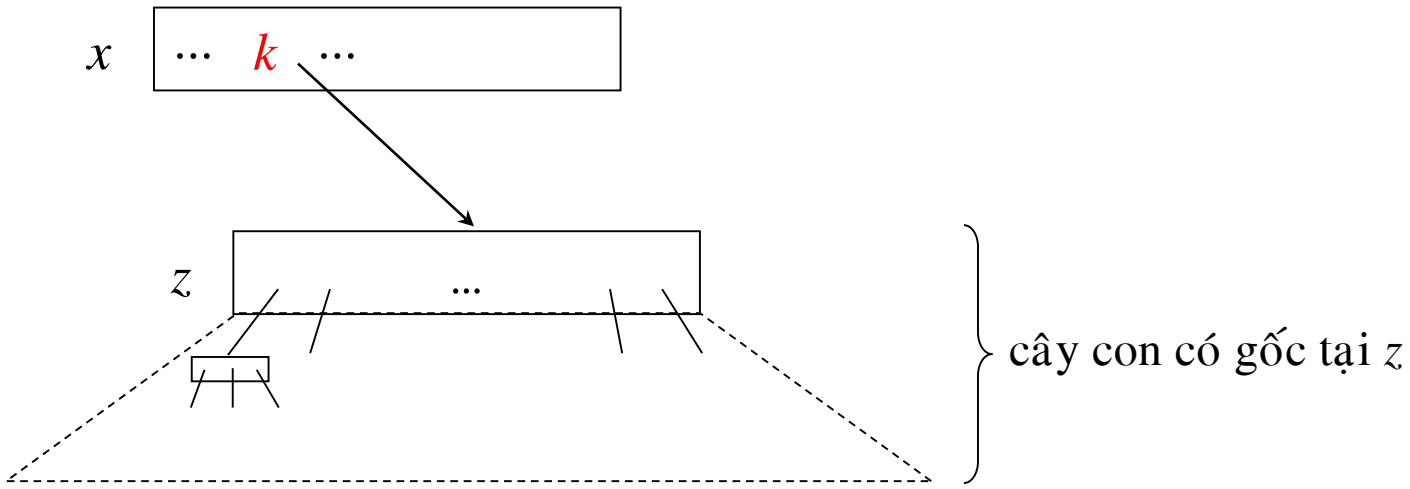
B-TREE-DELETE(x, k)

1. ...

2. Nếu khóa k có trong nút x và x là một nút trong thì

a. ...

b. Tương tự, nếu nút con z ở sau k có ít nhất t khóa thì tìm *khóa sau* (successor) k' của k trong cây con có gốc tại z. Xóa k', trong x thay k bằng k'. (Tìm và xóa k' có thể thực thi trong một lượt đi xuống duy nhất.)



Xóa một khóa khỏi một B-cây

B-TREE-DELETE(x, k)

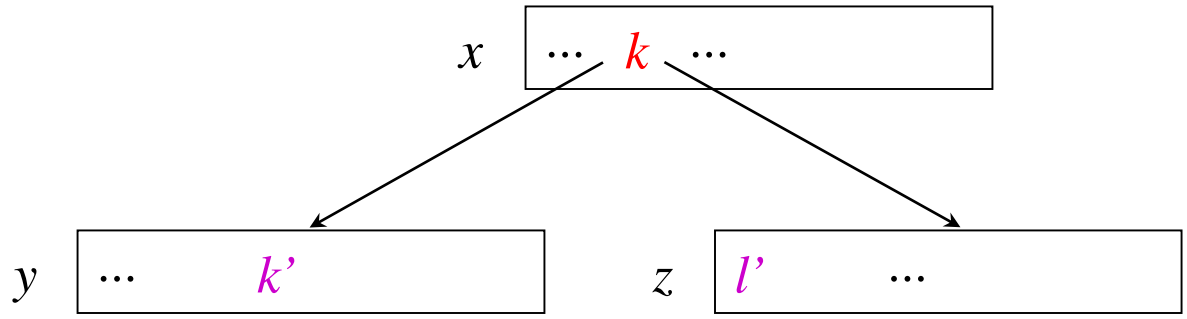
1. ...

2. Nếu khóa k có trong nút x và x là một nút trong thì

a. ...

b. ...

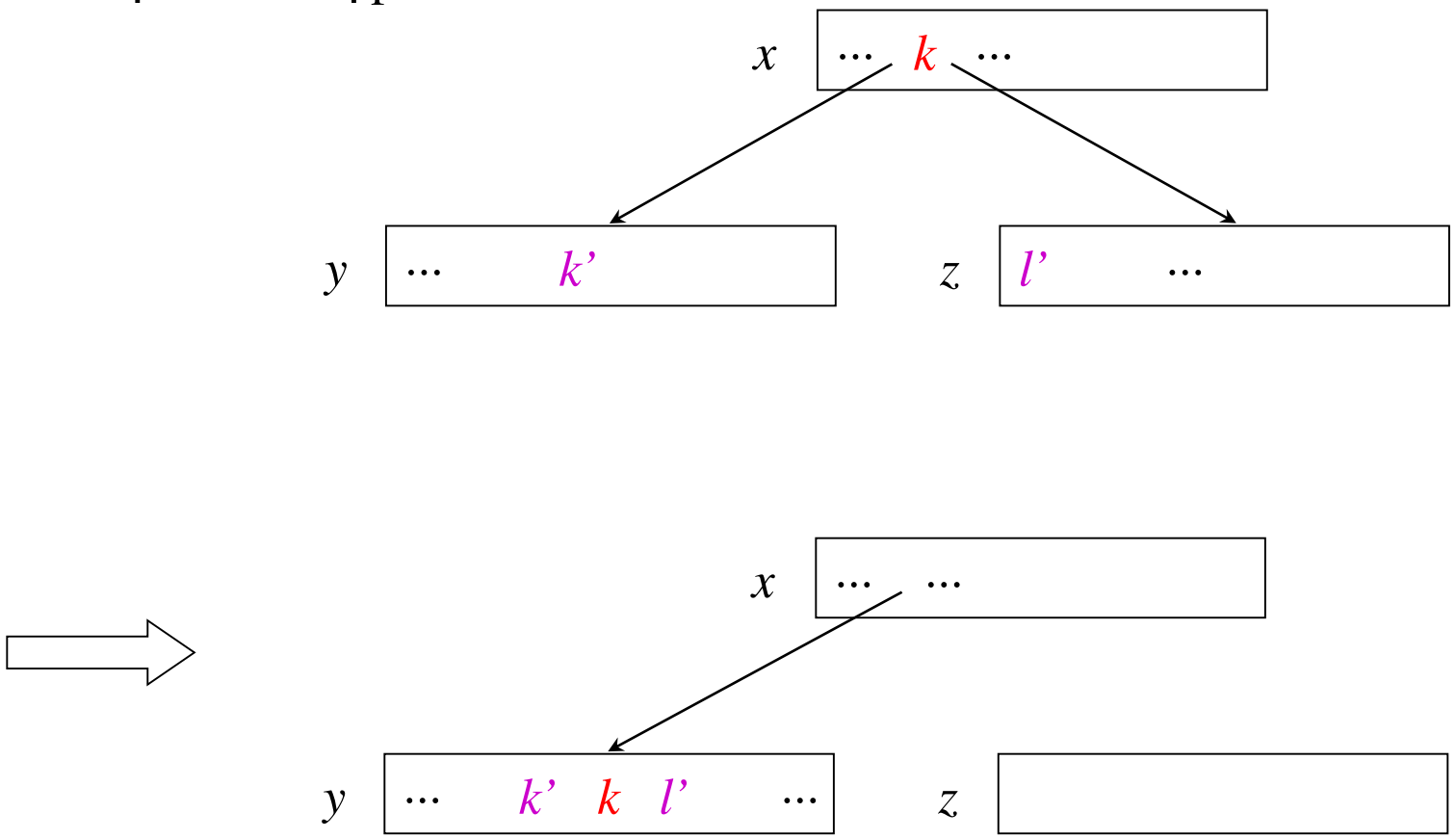
c. Nếu không, cả y lẫn z đều chỉ có $t - 1$ khóa, hợp nhất k và nguyên cả z vào y, thành ra x mất k và con trỏ đến z , và bây giờ y chứa $2t - 1$ khóa. Giải phóng (free) z và gọi đệ quy B-TREE-DELETE(y, k) để xóa k khỏi cây con có gốc y.



Xóa một khóa khỏi một B-cây

(tiếp)

Minh họa bước hợp nhất



Xóa một khóa khỏi một B-cây

B-TREE-DELETE(x, k)

1. ...

2. ...

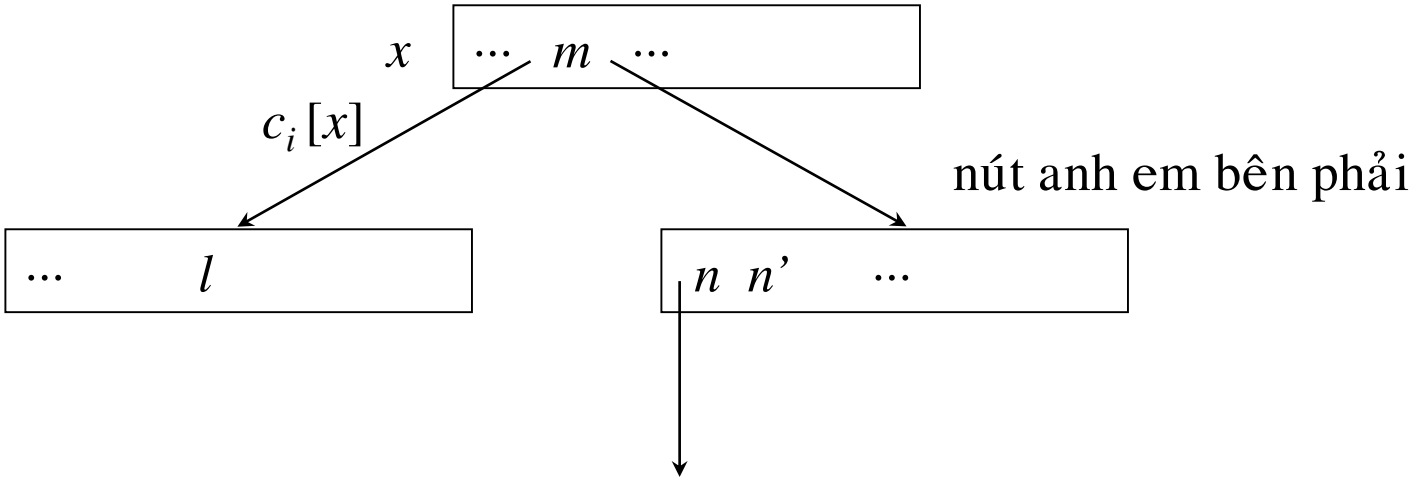
3. Nếu khóa k không có trong nút trong x thì xác định gốc $c_i[x]$ của cây con chứa k, nếu k có trong cây. Nếu $c_i[x]$ chỉ có $t - 1$ khóa, thực thi bước 3a hay 3b nếu cần (để đảm bảo rằng giải thuật, khi được gọi đệ quy, sẽ xuống một nút chứa ít nhất t khóa). Xong rồi gọi B-TREE-DELETE lên nút con thích hợp của x.

Xóa một khóa khỏi một B-cây

(tiếp)

3. ...

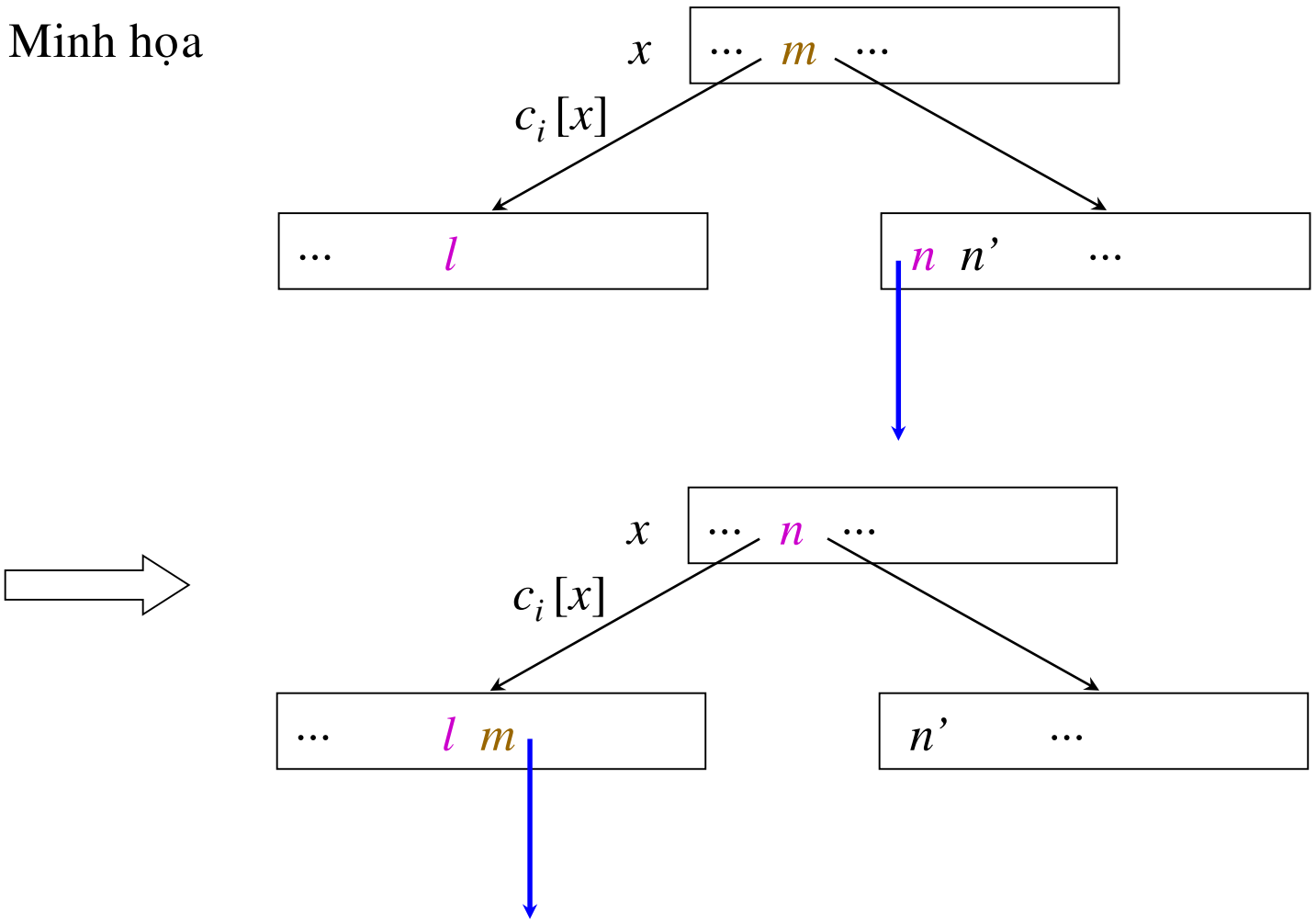
a. Nếu $c_i[x]$ chỉ có $t - 1$ khóa, nhưng lại có một nút anh em bên phải (hay bên trái) với ít nhất t khóa, thì cho nút $c_i[x]$ thêm một khóa bằng cách đem một khóa từ x xuống $c_i[x]$, đem một khóa từ nút anh em của $c_i[x]$ lên x, và đem con trỏ tương ứng từ nút anh em vào nút $c_i[x]$.



Xóa một khóa khỏi một B-cây

(tiếp)

Minh họa



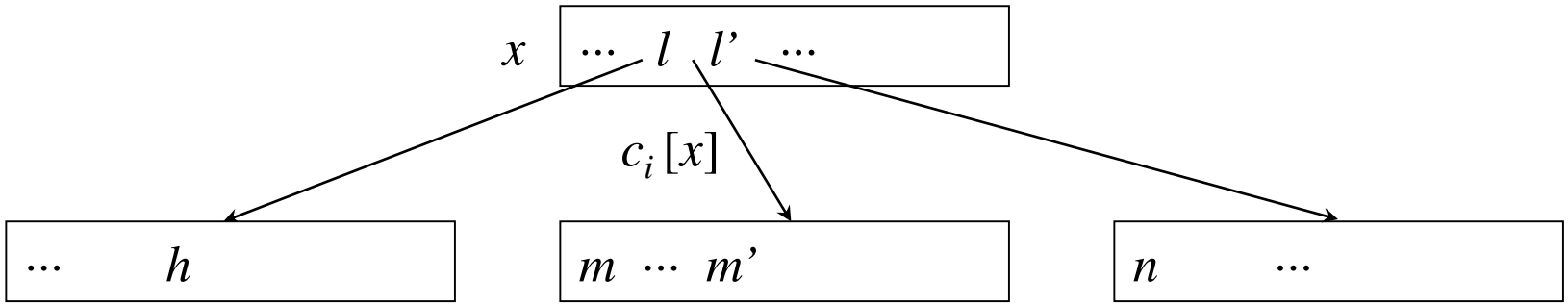
Xóa một khóa khỏi một B-cây

(tiếp)

3. ...

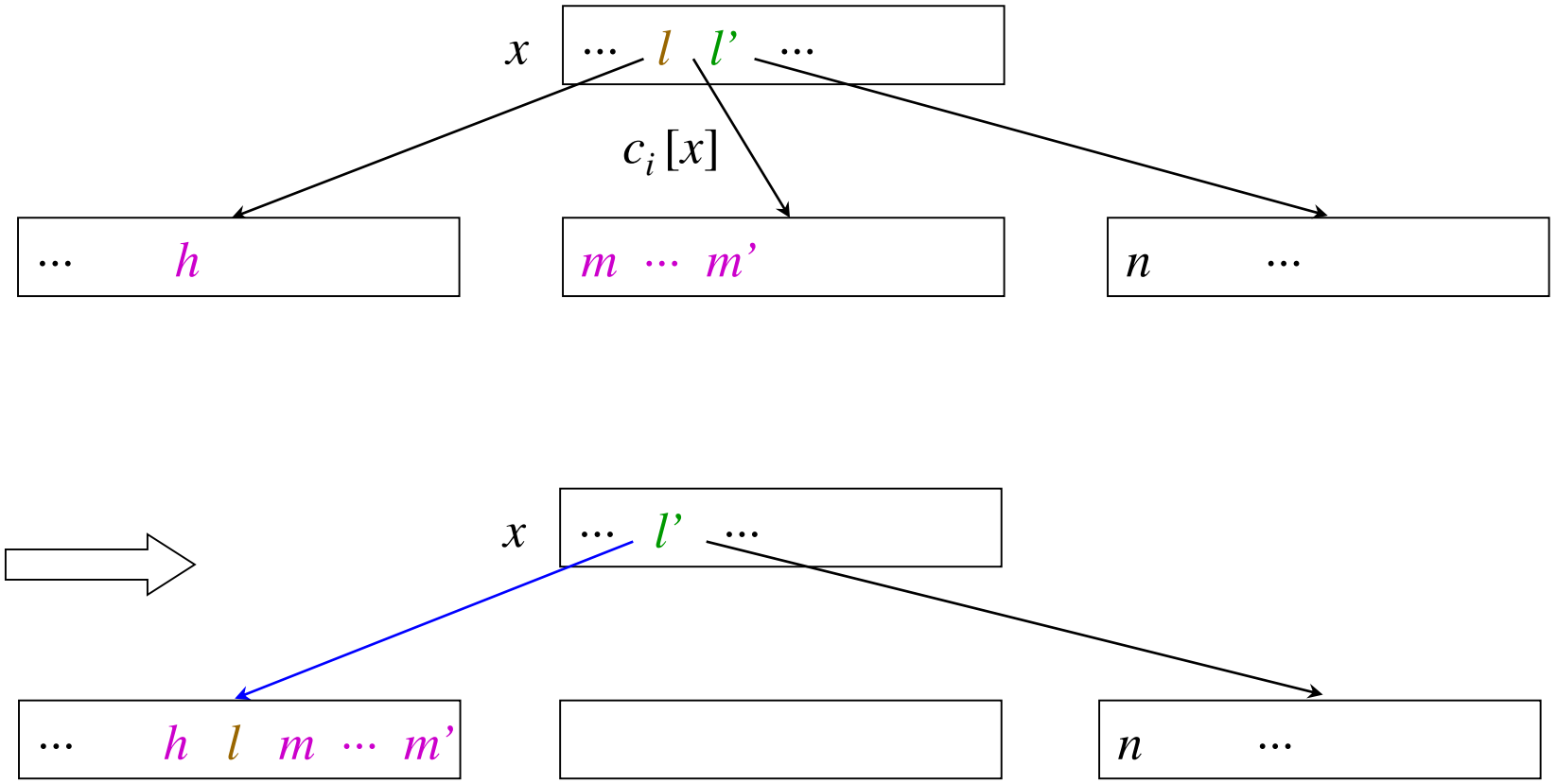
a. ...

b. Nếu $c_i[x]$ và mọi nút anh em của nó chỉ có $t - 1$ khóa, thì hợp nhất $c_i[x]$ và một nút anh em bằng cách đem một khóa từ x xuống nút mới tạo, khóa này sẽ là khóa giữa của nút.



Xóa một khóa khỏi một B-cây

(tiếp)
Minh họa



Xóa một khóa khỏi một B-cây

(tiếp)

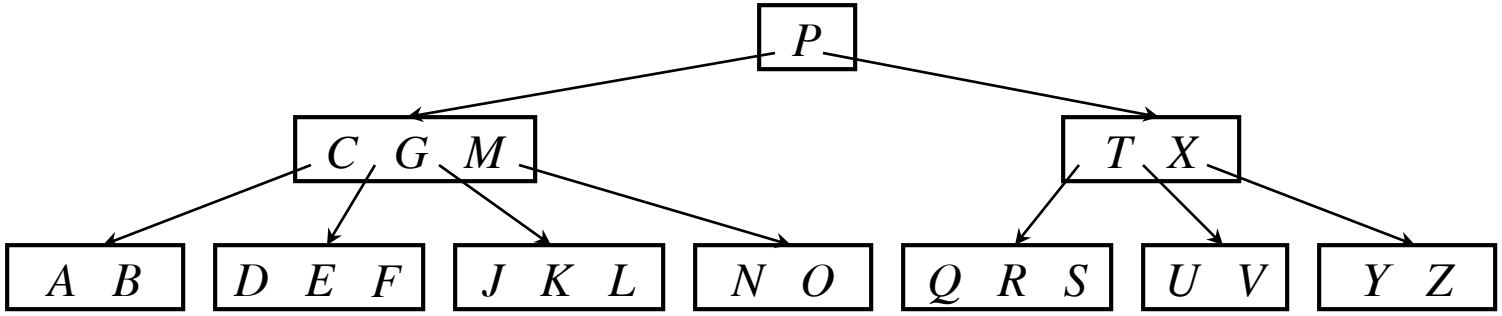
Thủ tục B-TREE-DELETE cần

- số truy cập lên đĩa là $O(h)$ vì có $O(1)$ lần gọi DISK-READ và DISK-WRITE giữa các gọi đệ quy của thủ tục.
- thời gian CPU của thủ tục là $O(t\,h) = O(t \log_t n)$.

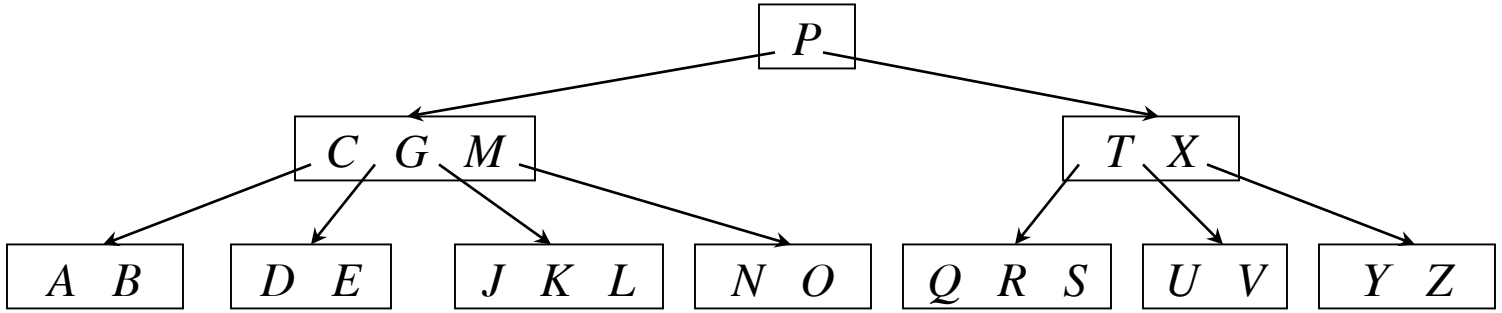
Ví dụ cho các trường hợp khi xóa một khóa khỏi một B-cây

Cho một B-cây có bậc tối thiểu $t = 3$

- Cây lúc đầu, xóa *F* khỏi cây

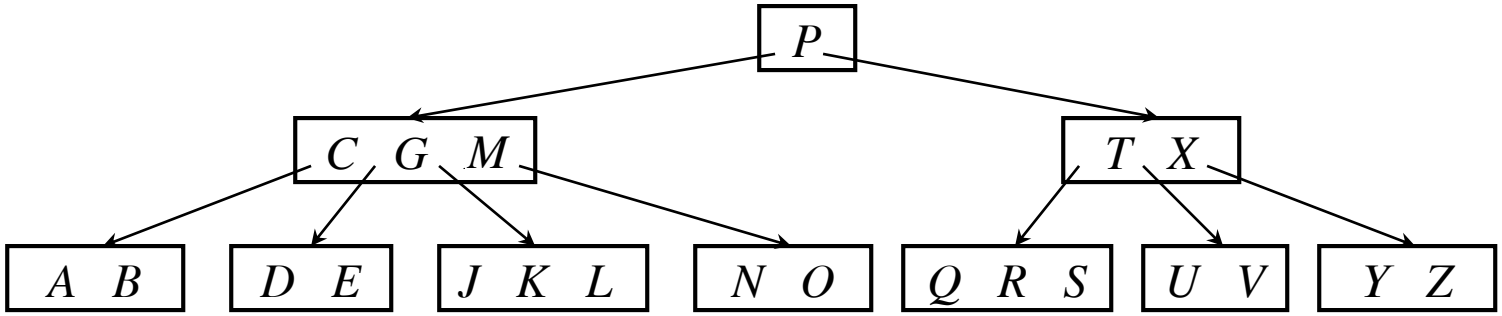


- *F* đã được xóa: trường hợp 1

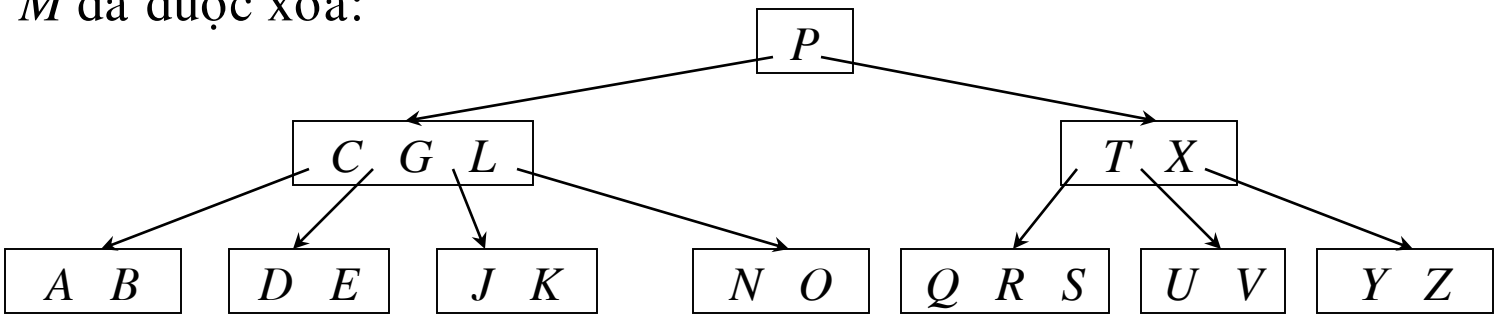


Ví dụ cho các trường hợp khi xóa một khóa khỏi một B-cây

- Xóa *M* khỏi cây: trường hợp 2a

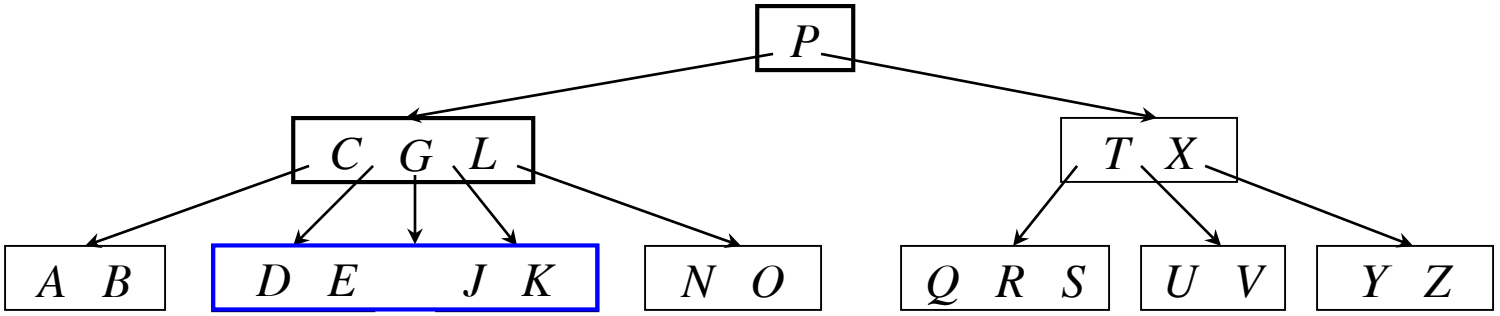


- *M* đã được xóa:

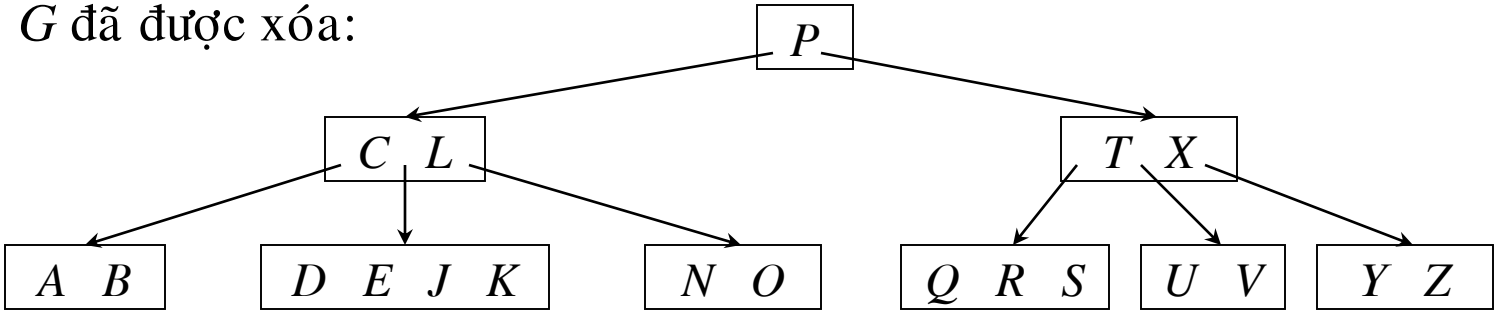


Ví dụ cho các trường hợp khi xóa một khóa khỏi một B-cây

- Xóa *G* khỏi cây: trường hợp 2c

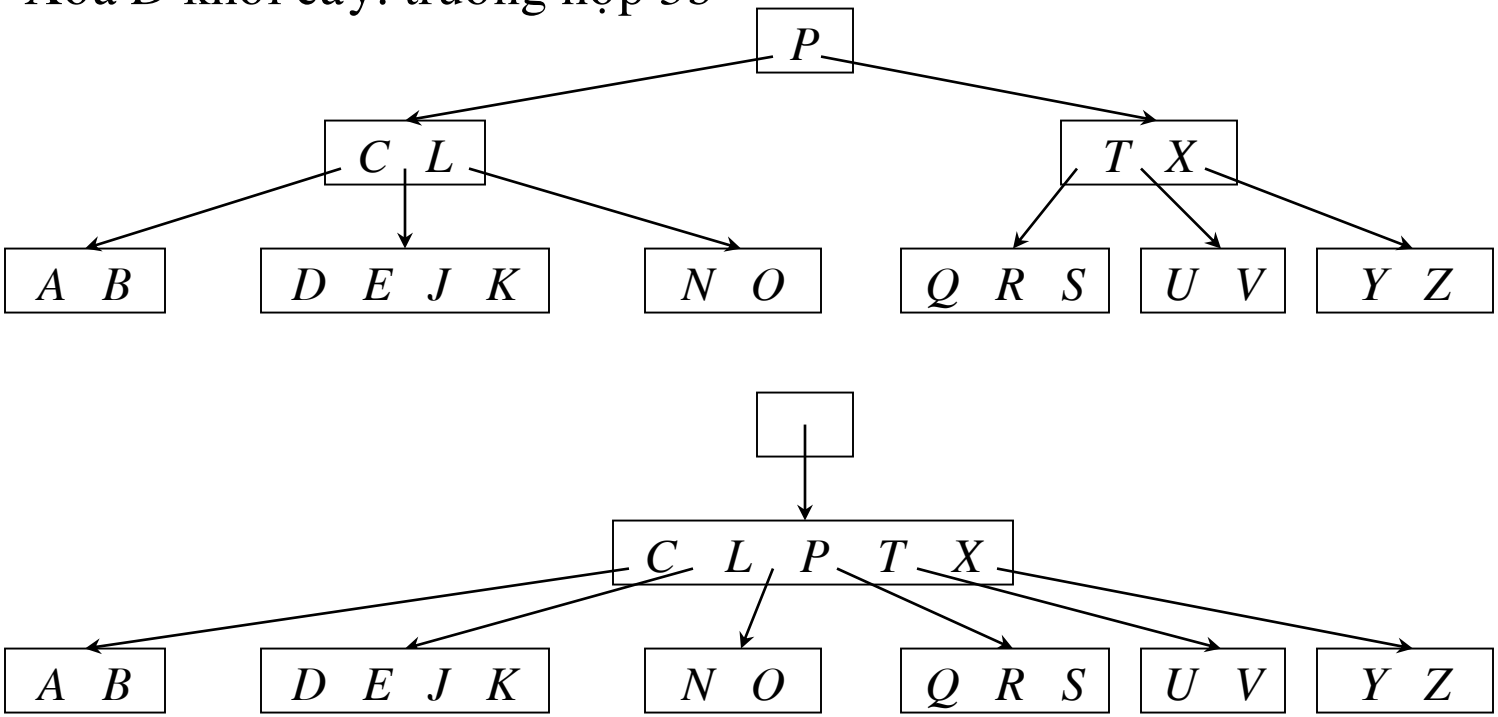


- *G* đã được xóa:

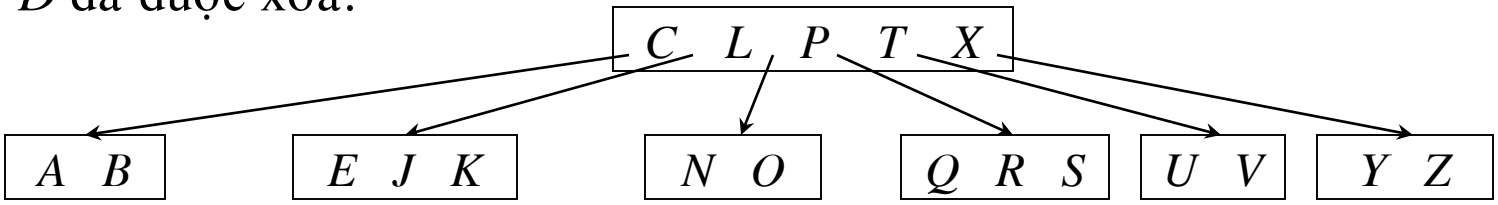


Ví dụ cho các trường hợp khi xóa một khóa khỏi một B-cây

- Xóa *D* khỏi cây: trường hợp 3b

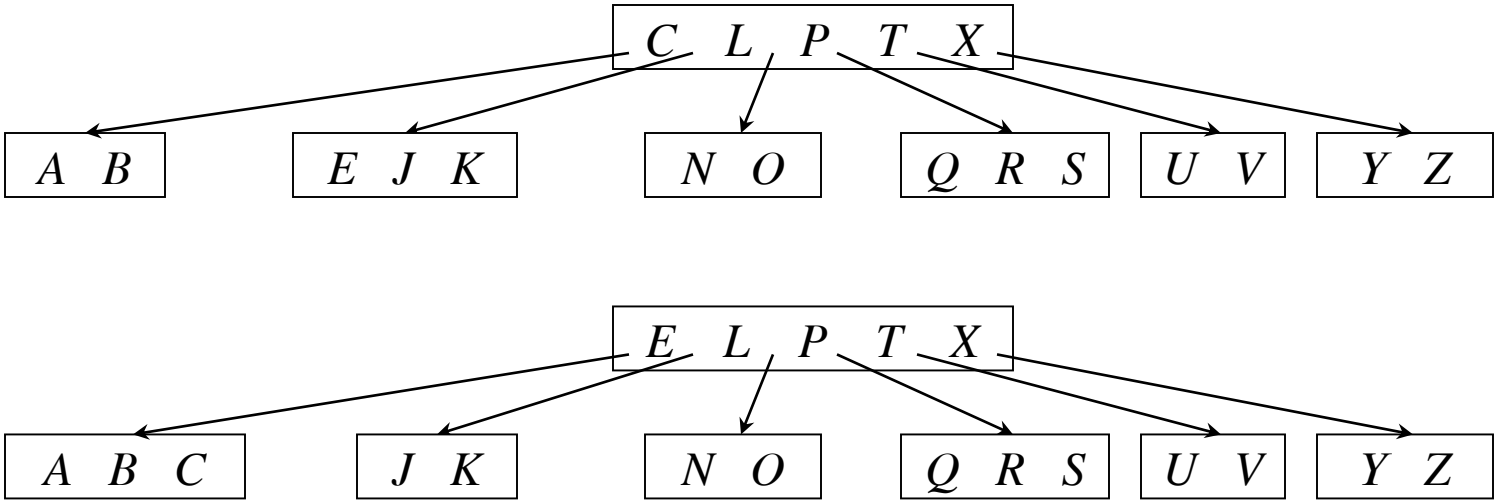


- *D* đã được xóa:



Ví dụ cho các trường hợp khi xóa một khóa khỏi một B-cây

- Xóa *B* khỏi cây: trường hợp 3a



- *B* đã được xóa khỏi cây: