B-TREE

DATA STRUCTURES AND ALGORITHMS

ThS Nguyễn Thị Ngọc Diễm
diemntn@uit.edu.vn

Nội dung

1. Định nghĩa using degree/order
2. Các phép toán cơ bản
   i. B-TREE-TRAVERSE
   ii. B-TREE-SEARCH
   iii. B-TREE-INSERTION
   iv. B-TREE DELETION

Định nghĩa B-tree using order

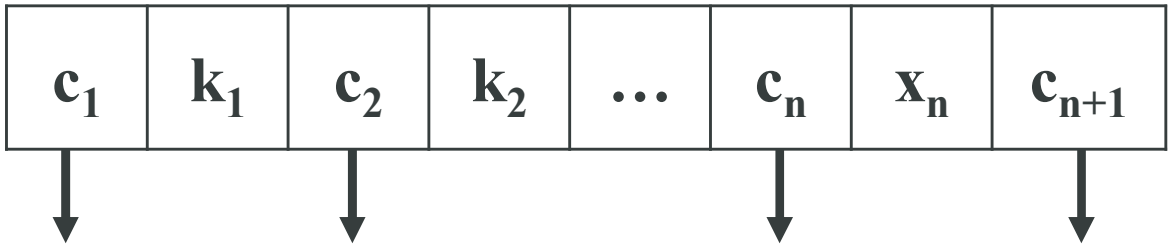
Cây B-tree T (với node root là T) có các tính chất sau:

1. Mỗi node x có các trường sau:
   - **x.n**: số lượng khóa (keys) hiện tại lưu trữ trên node x ,
   - x.n khóa được lưu theo thứ tự không giảm dần: $\mathbf{x.key_1} \leq \mathbf{x.key_2} \leq \ldots \leq \mathbf{x.key_{x.n}}$,
   - **x.leaf:** trả về giá trị boolean, TRUE nếu x là node lá, FALSE nếu x là node trong (internal node)

* Cây B-tree do R.Bayer và E.M.McCreight đưa ra năm 1972 trong bài báo: "*Bayer, R.; McCreight, E. (1972), "Organization and Maintenance of Large Ordered Indexes" (PDF), Acta Informatica, 1 (3): 173–189, doi:10.1007/bf00288683*"

Định nghĩa B-tree using order

2. Nếu **x là internal node** (node trong cây khác node root và node lá), thì nó chứa **x.n+1** con trỏ **$x.c_1, x.c_2, \ldots, x.c_{x.n+1}$** tới node con của nó. **Node lá không có node con**, vì vậy các trường c_i không được định nghĩa.



3. Các khóa **$x.key_i$** phân chia phạm vi các khóa được lưu trữ trong mỗi cây con: nếu k_i là bất kỳ khóa nào được lưu trữ trong cây con có gốc $c_i[x]$ thì:

$$k_1 \le x.key_1 \le k_2 \le x.key_2 \le \ldots \le x.key_{x.n} \le k_{x.n+1}$$

4. Các node lá có cùng độ sâu h (trong đó h là chiều cao của cây)

Định nghĩa B-tree using order

5. Số lượng cây con của một node có **chặn trên** và **chặn dưới**. Các giới hạn này được thể hiện dưới dạng một số nguyên cố định m ≥ 3 (m=2k+1, với k là số tự nhiên >=1), gọi là **bậc của cây B-tree**.

a. Mỗi node trong (khác node gốc và lá) chứa **ít nhất [m/2]** cây con và **nhiều nhất** m cây con. (Dấu [] lấy nguyên tròn lên trên, vd: 1.4 => lấy 2)

b. Tất cả các node, trừ node gốc , có từ **[m/2]-1** khóa cho đến m - 1 khóa (keys). Node gốc có từ 1 dến m-1 khóa.

TÓM LẠI, cho m ≥ 3 là bậc của cây B-tree, ta có:

[m/2]	≤	**số node con** của mỗi node (khác root và lá)	≤	m
2	≤	**số node con** của root (khác lá)	≤	m
[m/2]-1	≤	**số khóa** của mỗi node khác root	≤	m-1
1	≤	**số khóa** của node root	≤	m-1

Cây B-tree đơn giản nhất được tạo khi m=3. Mỗi internal node có thể có 2 hoặc 3 cây con. Đây là cây 2-3.

Nội dung

1. Định nghĩa using degree/order
2. Các phép toán cơ bản
   i. **B-TREE-TRAVERSE**
   ii. B-TREE-SEARCH
   iii. B-TREE-INSERTION
   iv. B-TREE DELETION

Nội dung

1. Định nghĩa using degree/order
2. Các phép toán cơ bản
   i. B-TREE-TRAVERSE
   ii. **B-TREE-SEARCH**
   iii. B-TREE-INSERTION
   iv. B-TREE DELETION

B-TREE-SEARCH

- Các trường hợp xảy ra khi tìm 1 node X. Nếu X không tìm thấy sẽ có 3 trường hợp sau xảy ra:
  - $K_i < X < K_{i+1}$: Tiếp tục tìm kiếm trên cây con C_{i+1}
  - $K_m < X$: Tiếp tục tìm kiếm trên C_{n+1}
  - $X < K_1$: Tiếp tục tìm kiếm trên C_1
- Quá trình này tiếp tục cho đến khi node được tìm thấy. Nếu đã đi đến node lá mà vẫn không tìm thấy khoá, việc tìm kiếm là thất bại.

B-TREE-SEARCH

- B-TREE-SEARCH is a generalization of the TREE-SEARCH procedure defined for binary search trees. It takes as input a pointer to the root node x of a subtree and a key k to be searched for in that subtree. The top-level call is thus of the form B-TREE-SEARCH(root[T], k)
- More precisely, at each internal node x, we make an (x.n+1)-way branching decision.
- If k is in the B-tree, B-TREE-SEARCH returns the ordered pair (y, i) consisting of a node y and an index i such that keyi[y] = k. Otherwise, the value NIL is returned.

B-TREE-INSERTION: (followed Bayer)

Quá trình thêm một khoá mới (newkey) vào B-tree có thể được mô tả như sau:

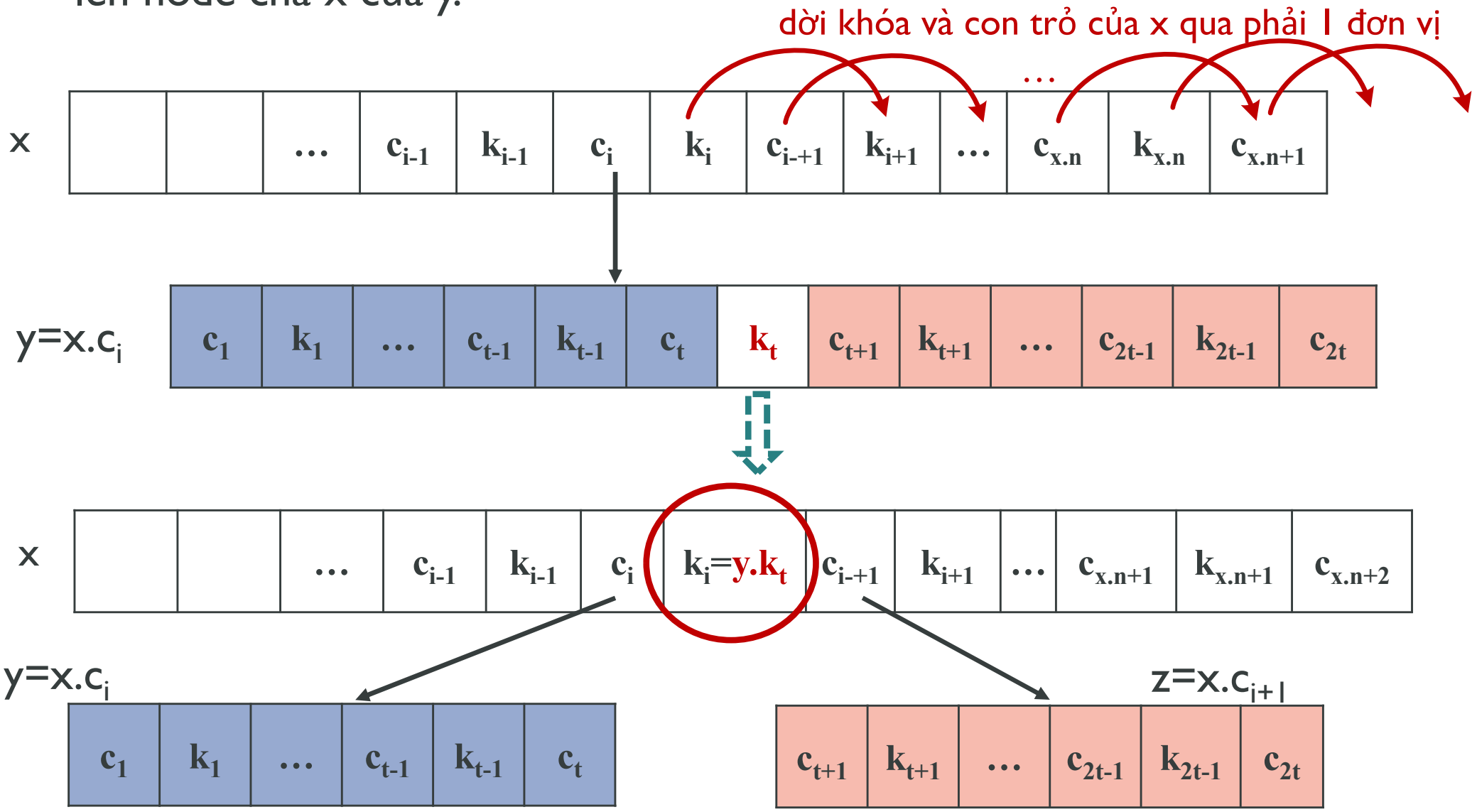
- Tìm node newkey, nếu tìm thấy thì kết thúc (không thêm vào nữa)
- Ngược lại, nếu không tìm thấy thì:
  - Thêm newkey vào node lá nếu node chưa đầy và kết thúc
  - Ngược lại: Khi node được thêm vào bị đầy, node này sẽ được tách thành 2 node cùng mức, khoá median sẽ được dời lên node cha, quá trình này có thể **lan truyền đến node gốc**
  - Trong trường hợp node gốc bị đầy, node gốc sẽ bị tách và dẫn đến việc tăng trưởng chiều cao của cây.

B-TREE-INSERTION: Tách node (followed Bayer)

- Cho node y có cha là x và hiện tại node y có **nhiều hơn** m-1 khóa
  - **Di chuyển** khóa ở giữa của y lên x
  - Tạo node y' mới, **di chuyển** phân nửa số node và phân nửa số cây con còn lại của y sang y'
  - Thêm y' vào danh sách cây con của x
- Thao tác splitting có thể lan truyền (propagate) đến nhiều mức phía trên.
- Split node gốc sẽ làm tăng chiều cao của cây.

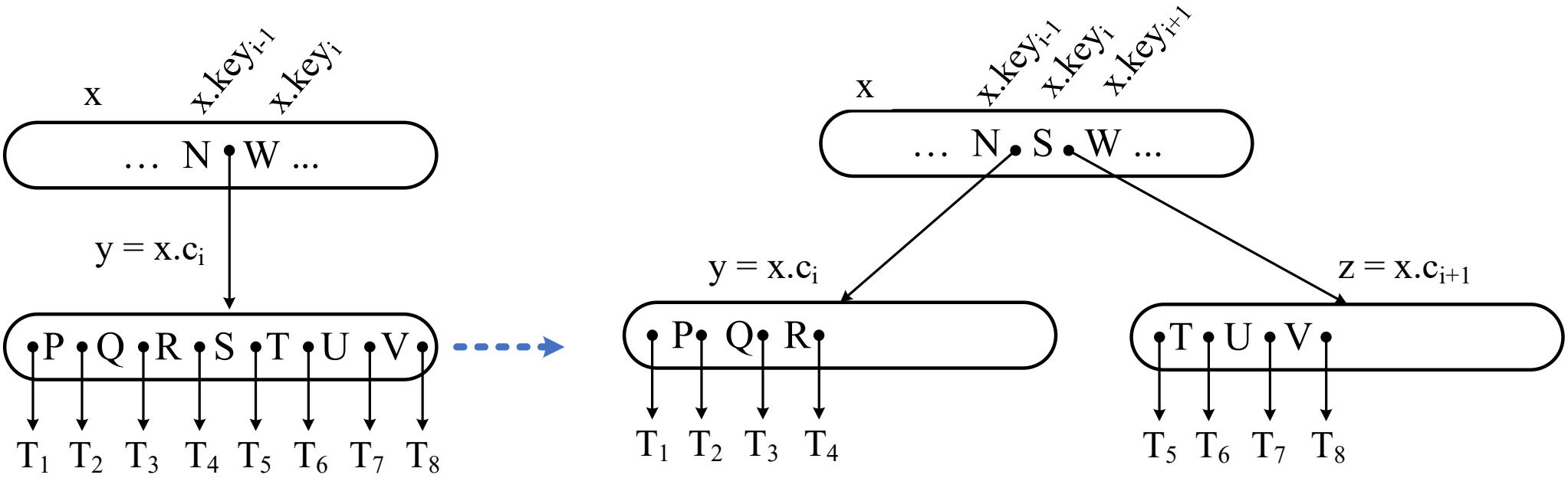
Splitting a node: Minh họa

- Minh họa: split node đầy y thành 2 node y và z, đưa khóa median từ y lên node cha x của y.



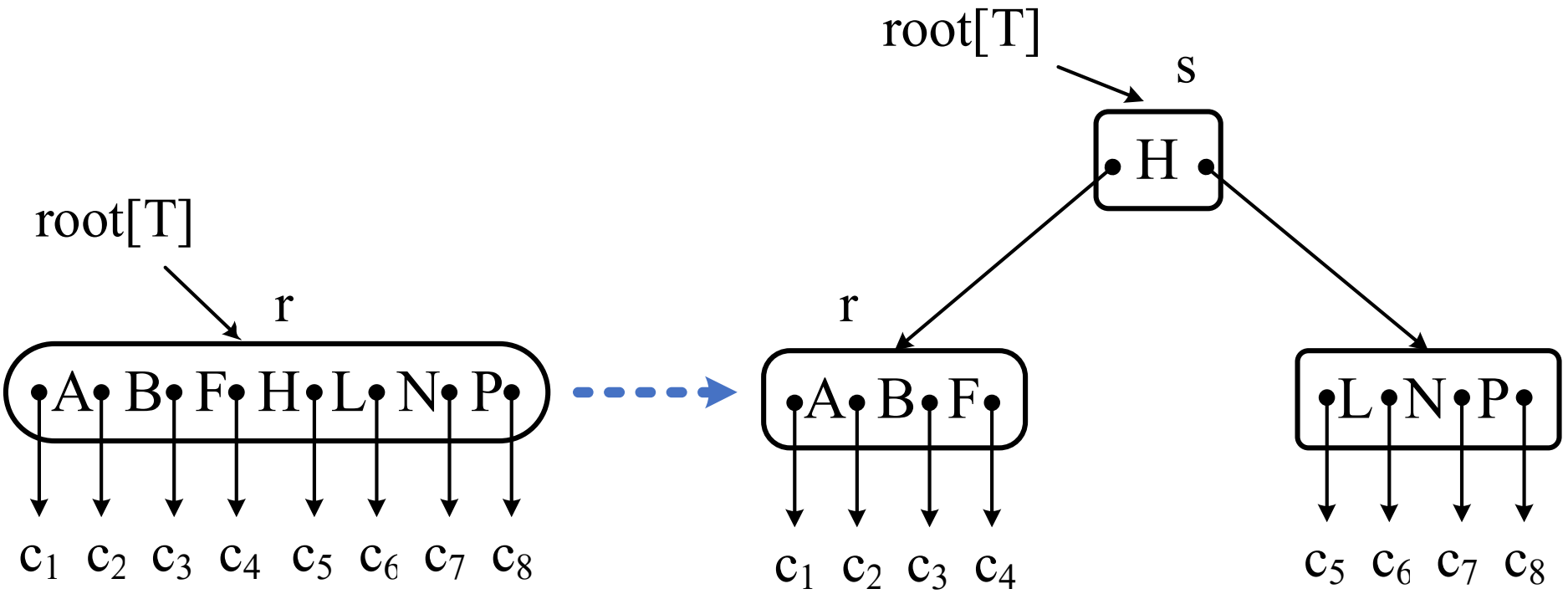
Splitting a node:Ví dụ:

- Ví dụ bên dưới minh họa quá trình split cho cây B-Tree bậc m=7.
  - Ta cần tách node đầy y=x.c_i với S là khóa median.
  - Khóa S sẽ di chuyển lên node cha của y là node x.
  - Tất cả khóa lớn hơn median sẽ di chuyển sang node mới là node z, node z trở thành con của node x.



Splitting a node: Split node Root

- Tách gốc với cây B-Tree bậc m=7. Root r được tách làm 2 node. Một root mới s được tạo ra chứa giá trị median của r, và nhận 2 node vừa được tách ra làm con. Cây B-Tree tăng trưởng lên 1 đơn vị chiều cao khi node root được tách.



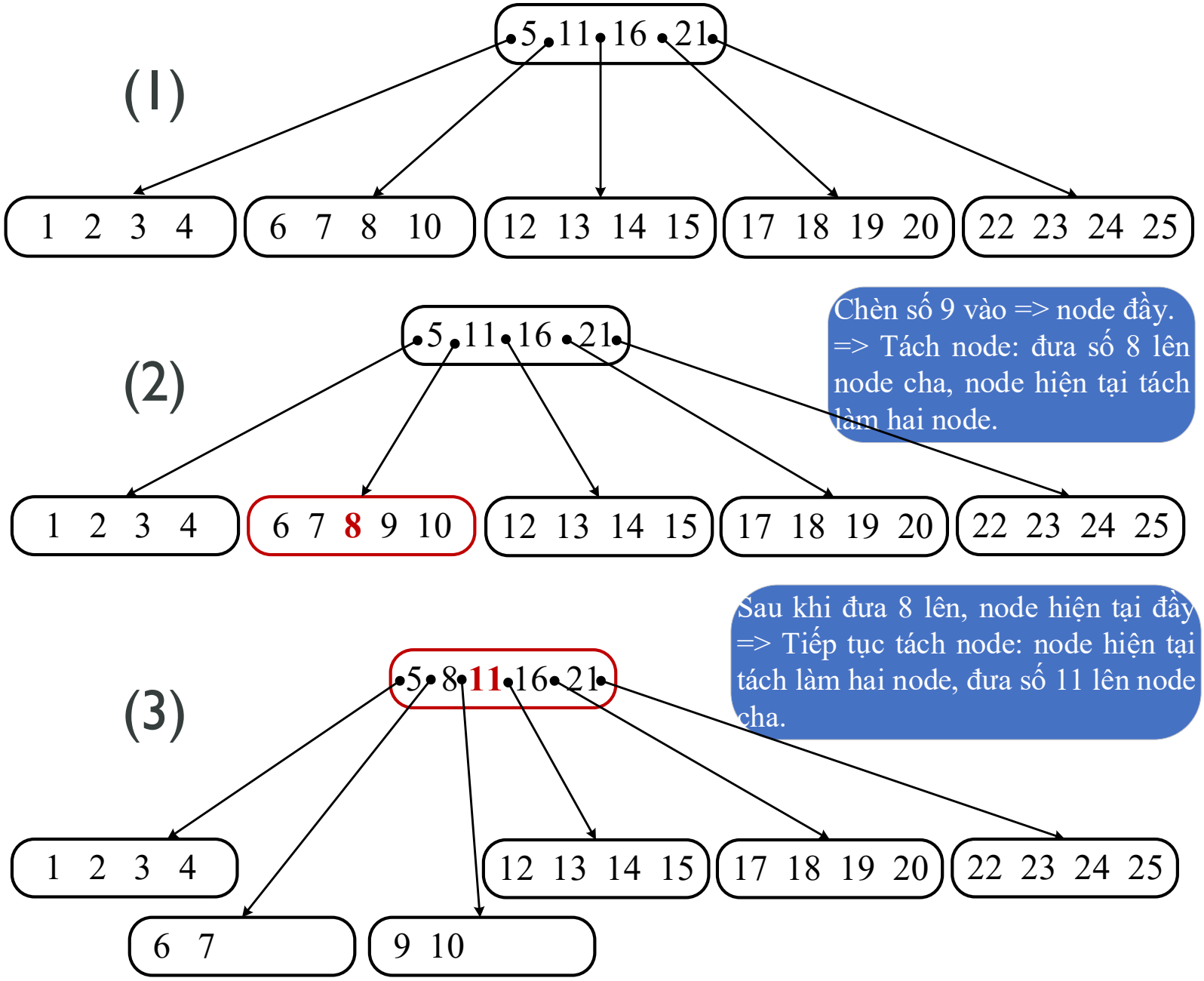
B-TREE-INSERTION:

- Một phương pháp tách khác gọi là: Proactive splitting - preemptive splitting (xem trong slide B-Tree using minimum degree)
  - Khi tìm node để thêm khóa mới, nếu gặp một node có vừa đủ m-1 khóa thì split luôn.
- Triệt tiêu lan truyền ngược
- Không tận dụng được hàm search
- Ưu nhược điểm khác ?

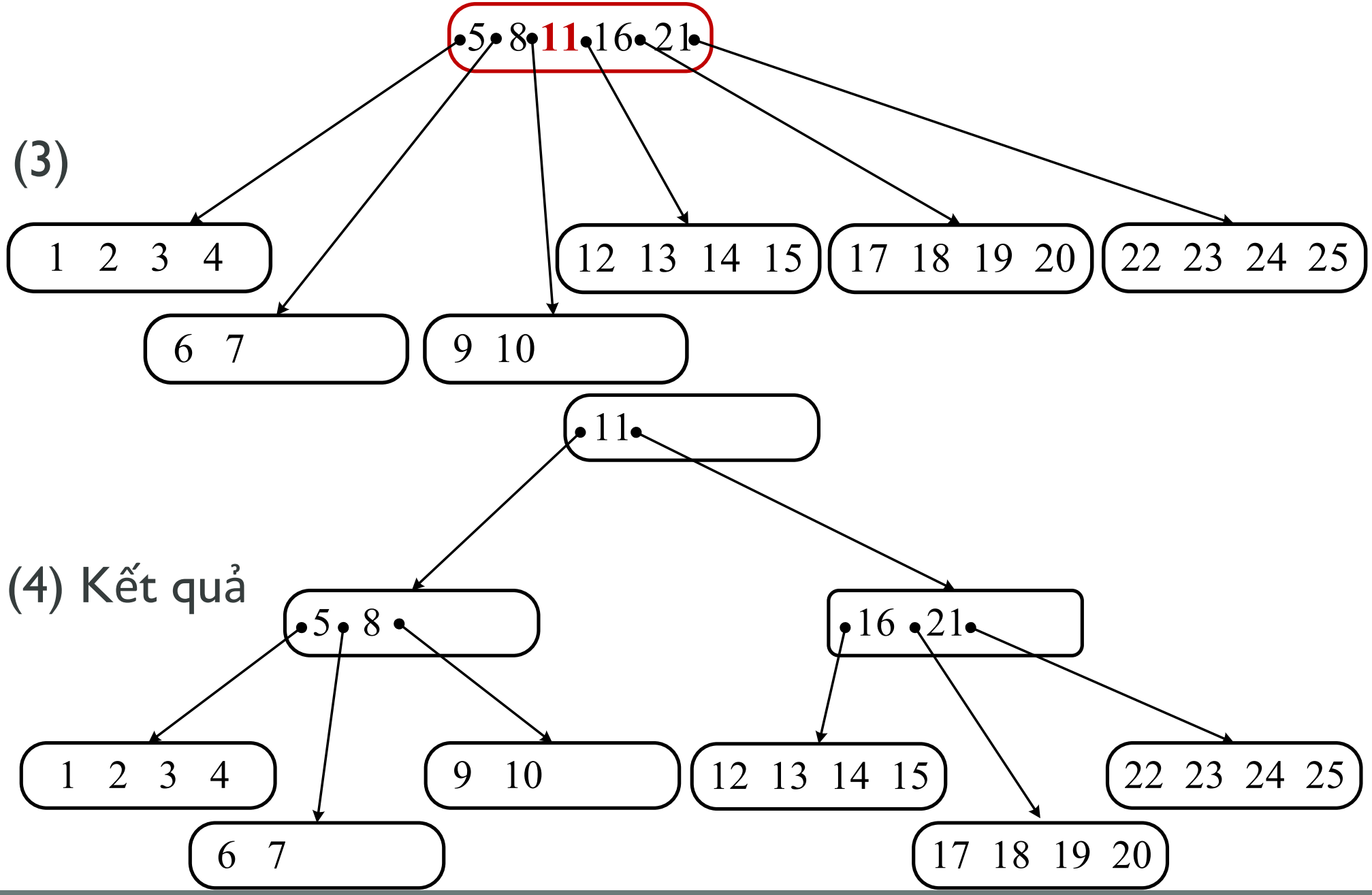
B-TREE-INSERTION: Example (followed Bayer)

Ví dụ:

Chèn số 9 vào cây B-tree bậc 5 cho sẵn như hình bên cạnh:



B-TREE-INSERTION: Example (followed Bayer)



B-TREE-INSERTION: Ví dụ 1

Ví dụ 1: Tạo cây B-Tree bậc m=3 từ dãy gồm 9 số nguyên từ 1 đến 9.

Bài làm:

Áp dụng vào bài, B-tree bậc 3 nên ta có:

1 ≤ **số khóa** của mỗi **node khác root** ≤ 2

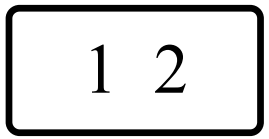
1 ≤ **số khóa** của node **root** ≤ 2

2 ≤ **số node con** của mỗi **node** (khác root và lá) ≤ 3

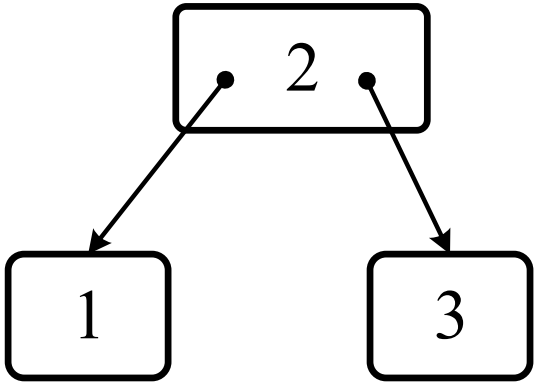
2 ≤ **số node con** của mỗi node **root** (khác lá) ≤ 3

B-TREE-INSERTION:Ví dụ 1

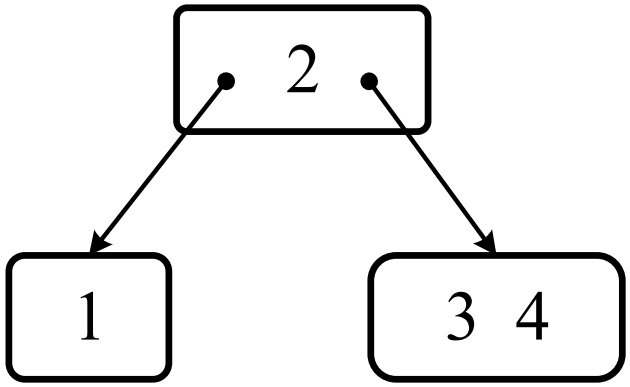
Insert 1, 2



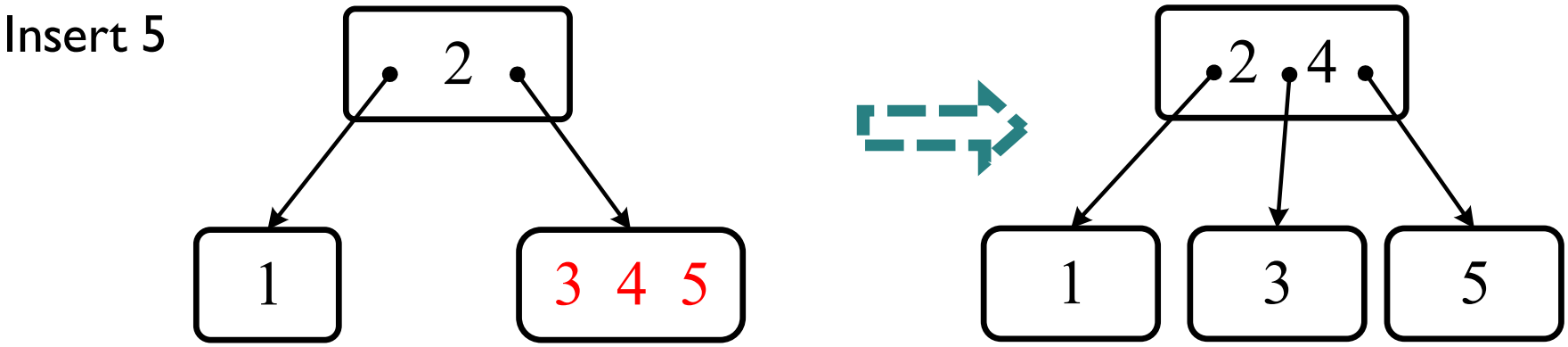
Insert 3

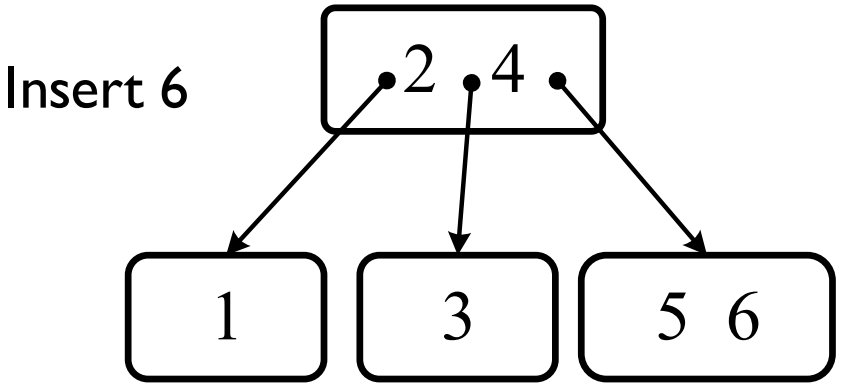


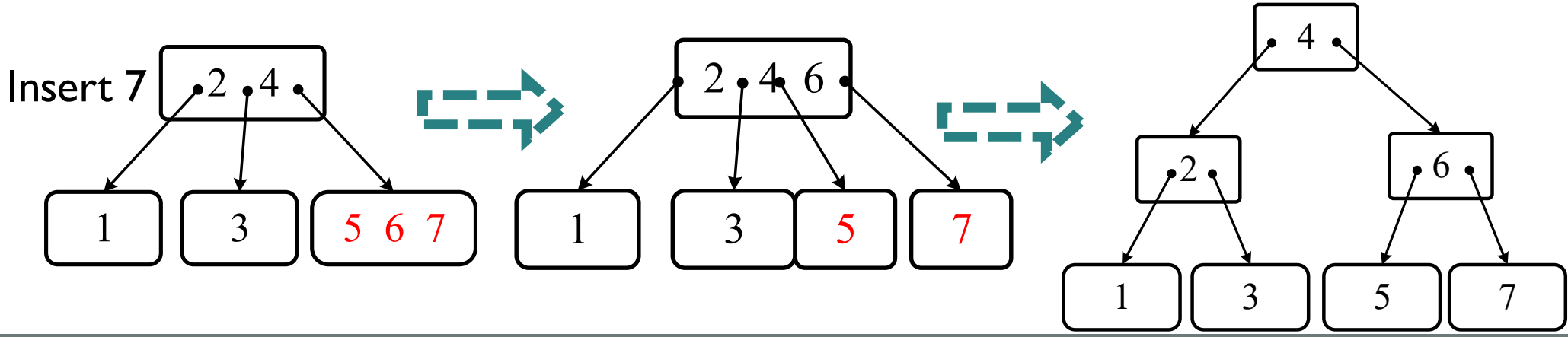
Insert 4



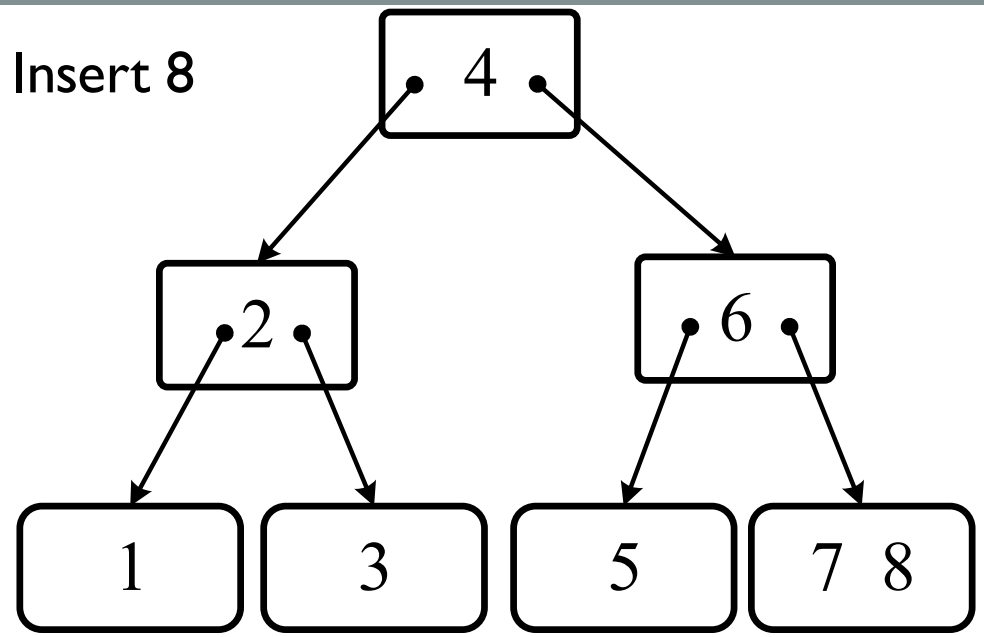
B-TREE-INSERTION:Ví dụ 1

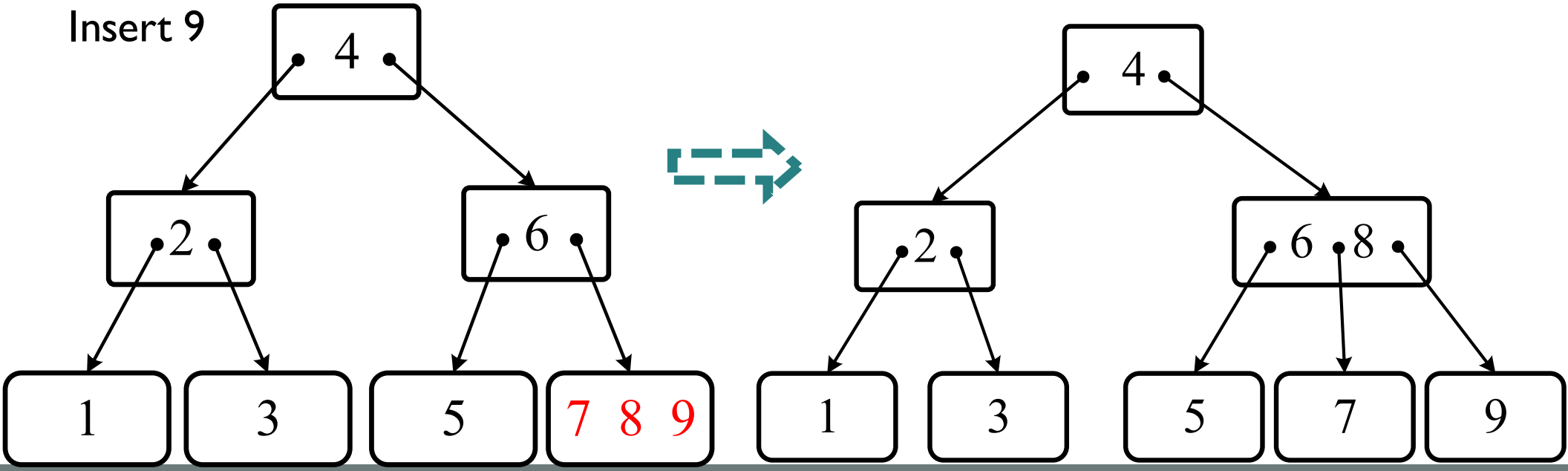






B-TREE-INSERTION:Ví dụ 1





B-TREE-INSERTION: Example (followed Bayer)

Ví dụ 2: Tạo cây B-Tree bậc m=3 từ dãy gồm 16 số nguyên: 27, 19, 23, 9, 1, 3, 11, 21, 5, 13, 17, 15, 29, 25, 20 và 22.

Bài làm:

Áp dụng vào bài, B-tree bậc 3 nên ta có:

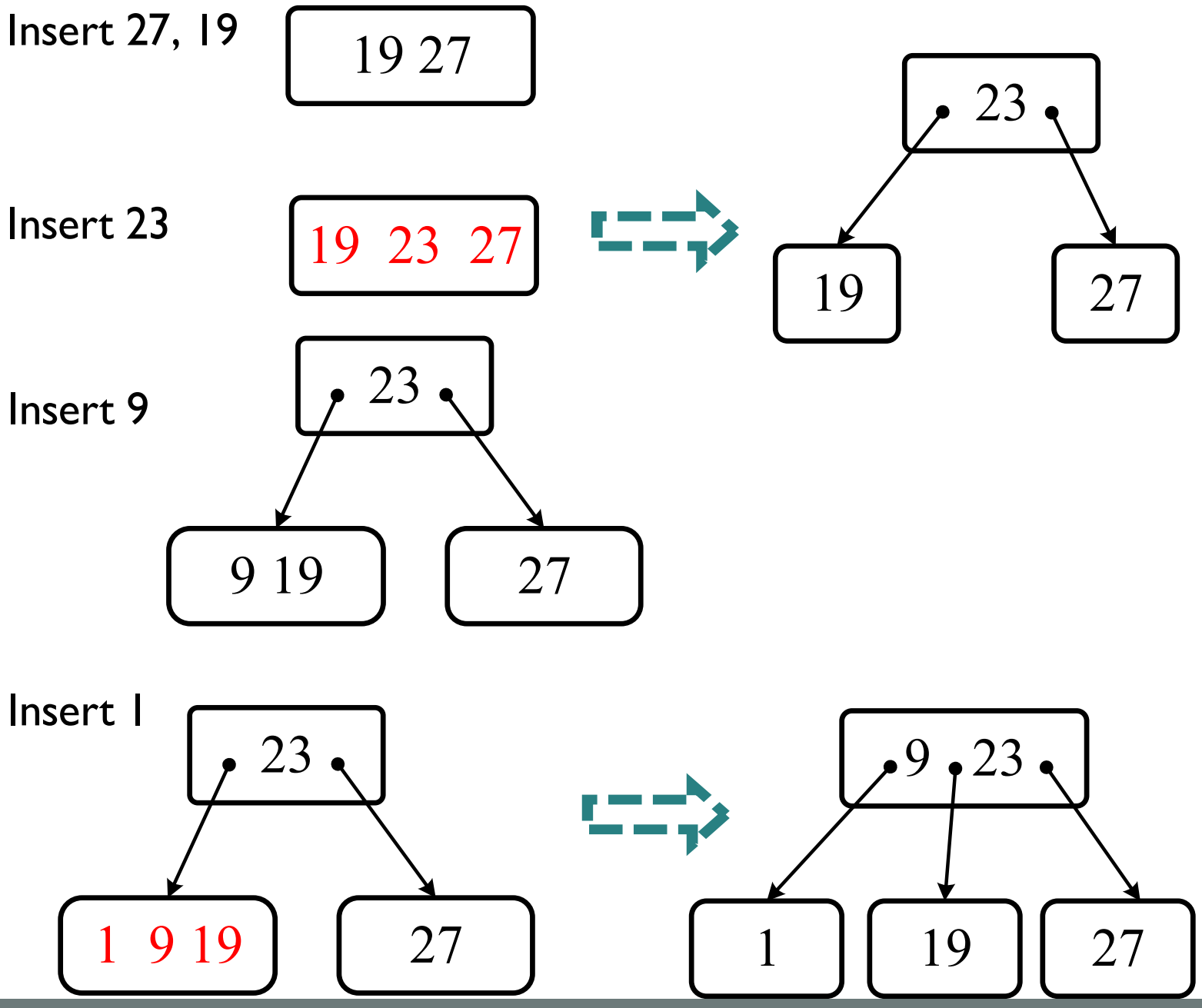
1 ≤ **số khóa** của mỗi **node khác root** ≤ 2

1 ≤ **số khóa** của node **root** ≤ 2

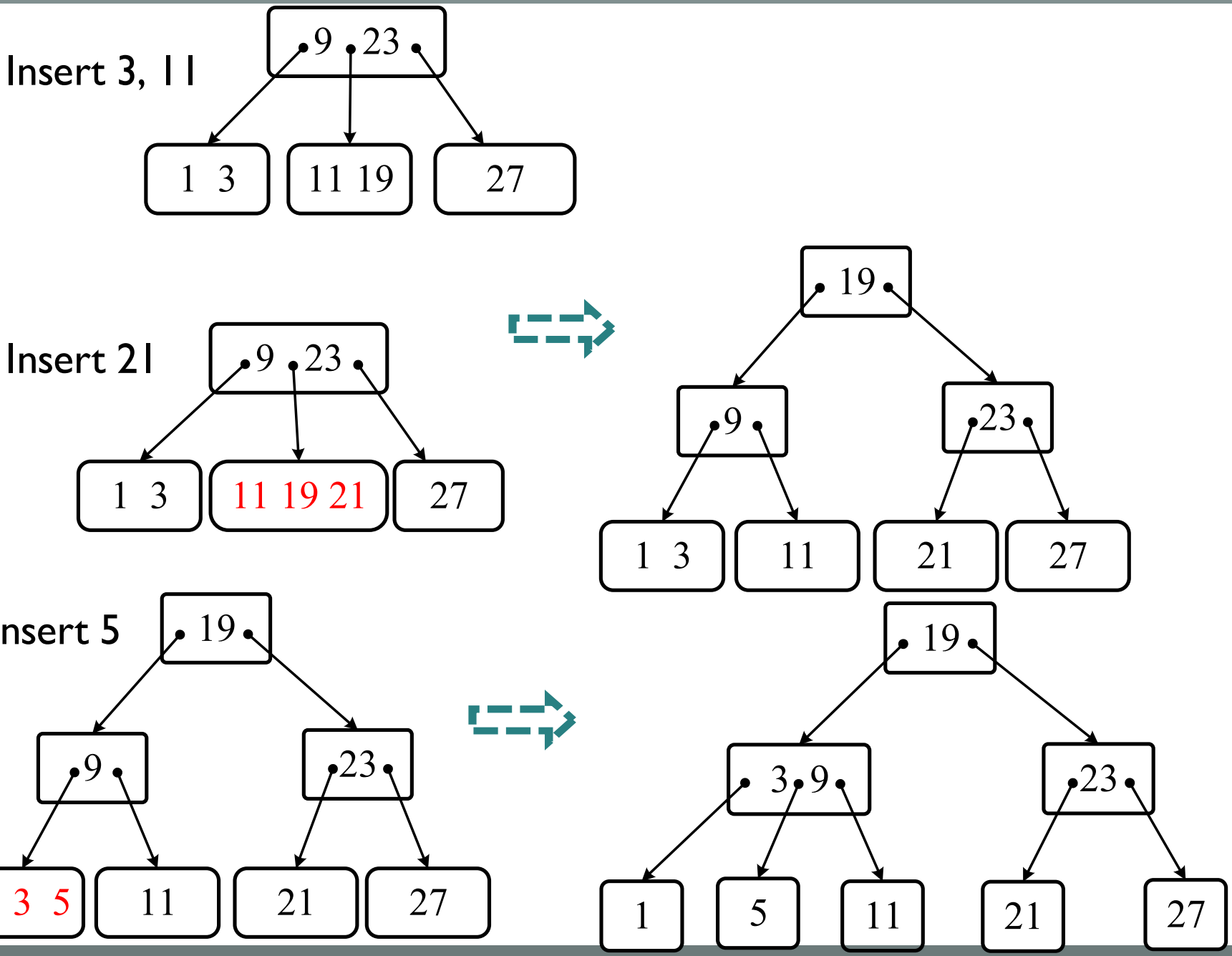
2 ≤ **số node con** của mỗi **node** (khác root và lá) ≤ 3

2 ≤ **số node con** của mỗi node **root** (khác lá) ≤ 3

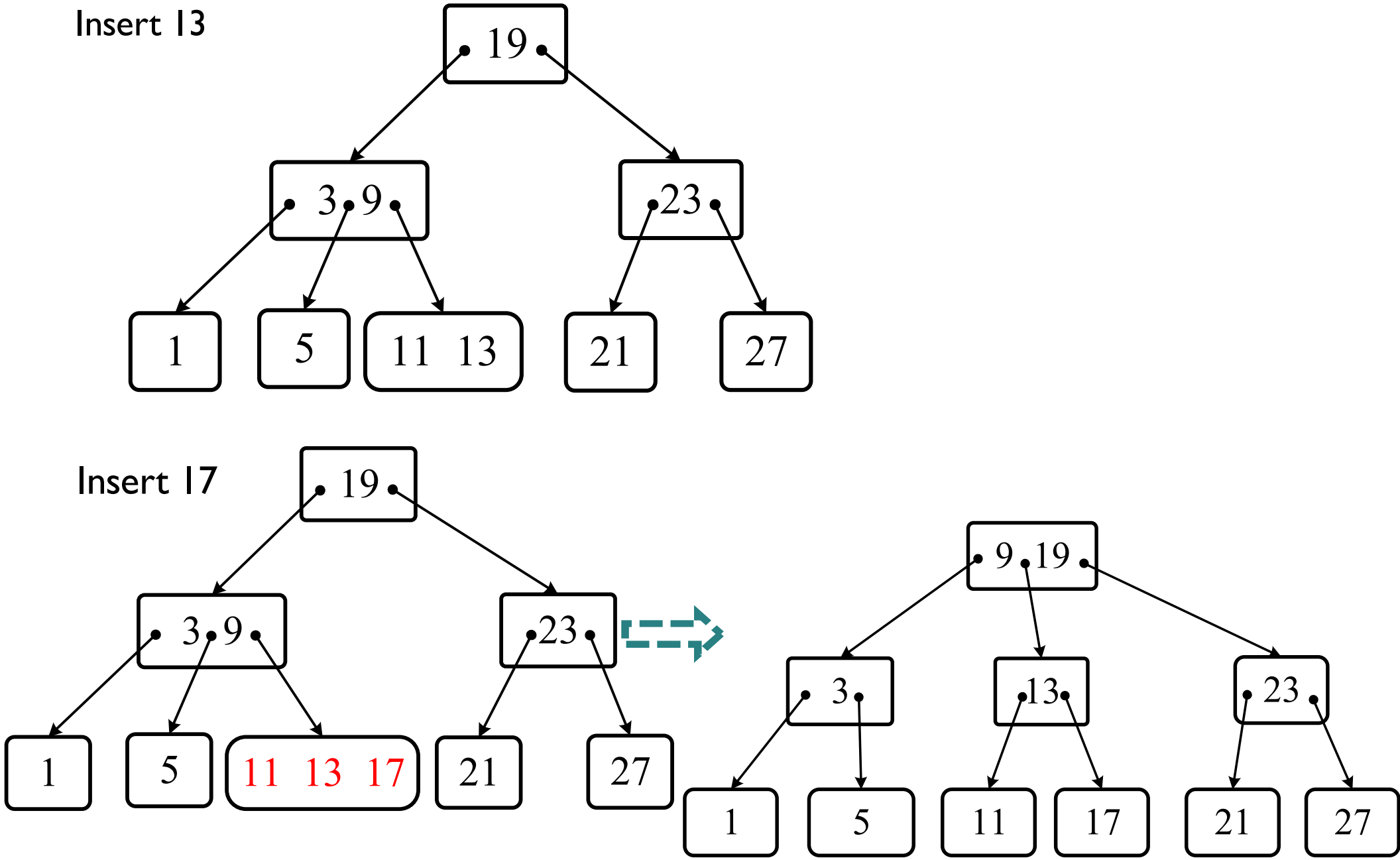
B-TREE-INSERTION:Ví dụ 2



B-TREE-INSERTION:Ví dụ 2

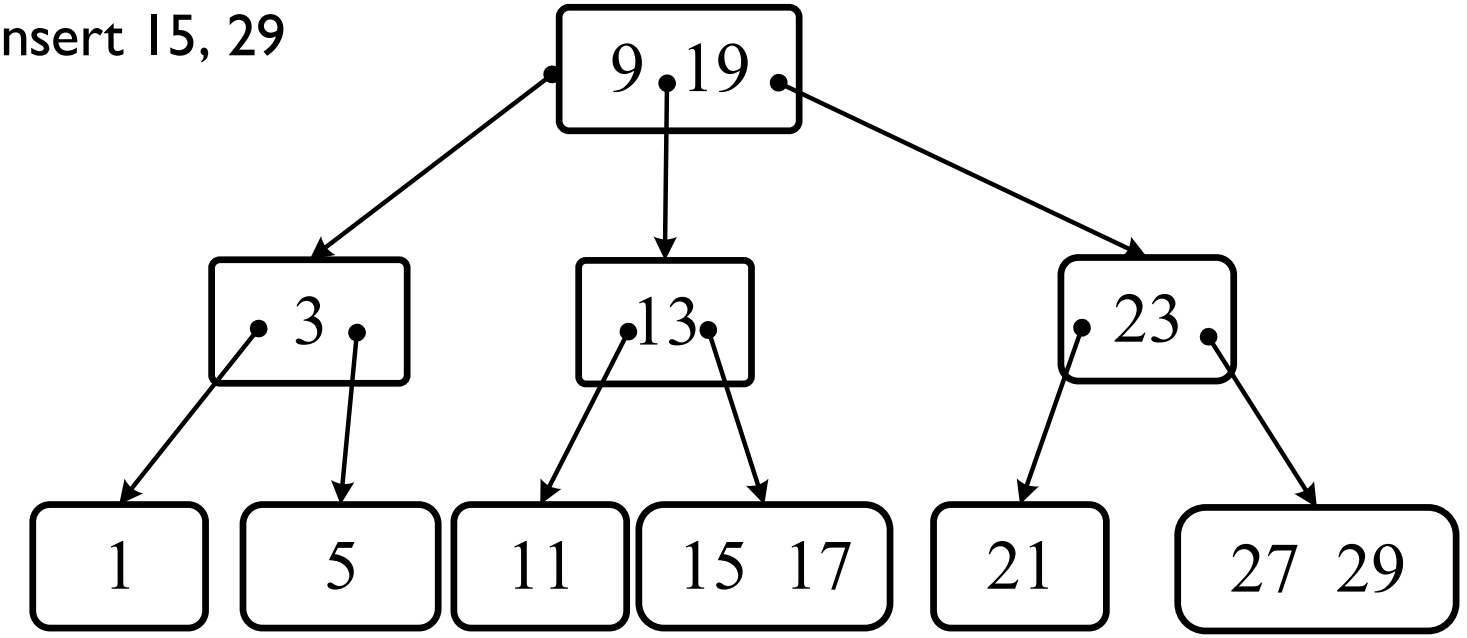


B-TREE-INSERTION:Ví dụ 2



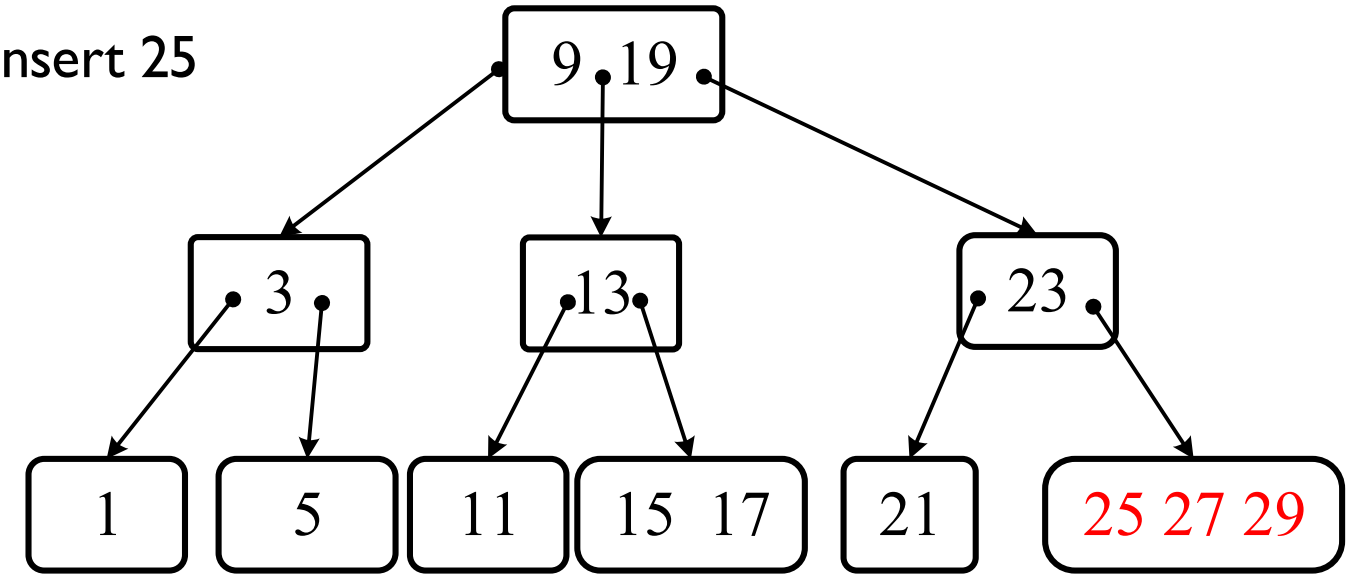
B-TREE-INSERTION: Ví dụ 2

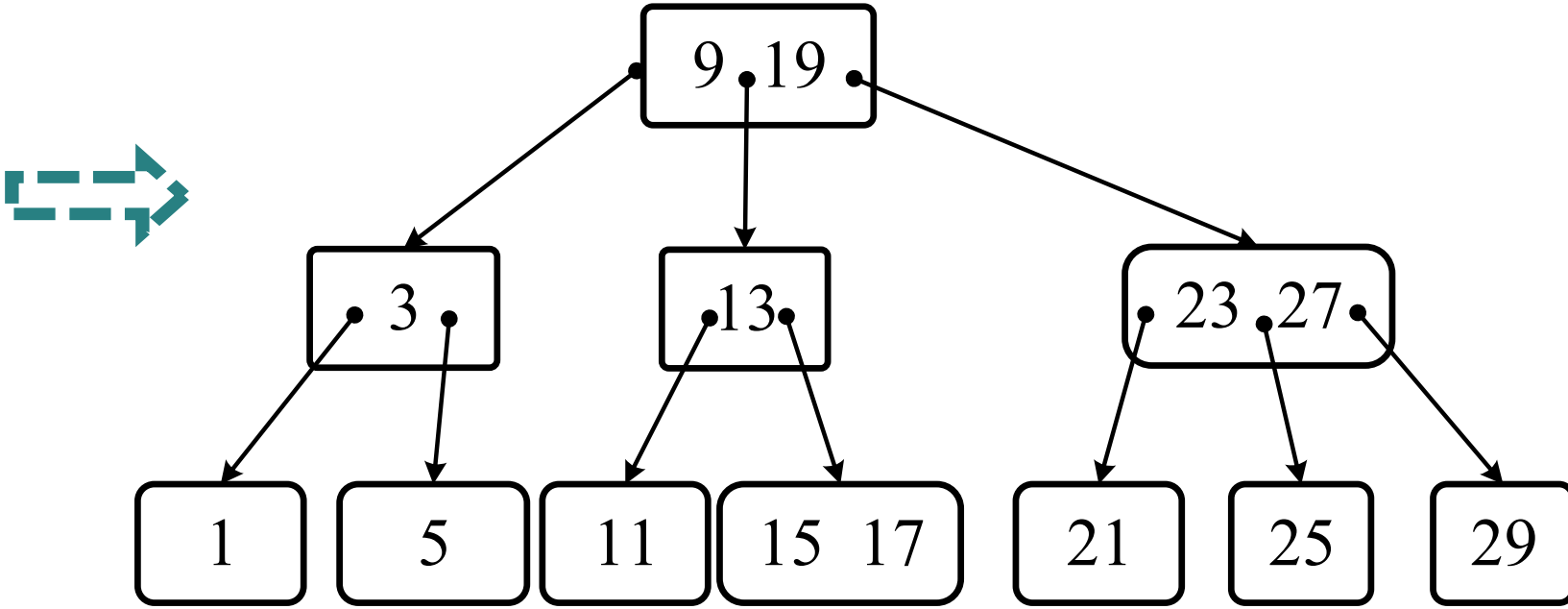
Insert 15, 29



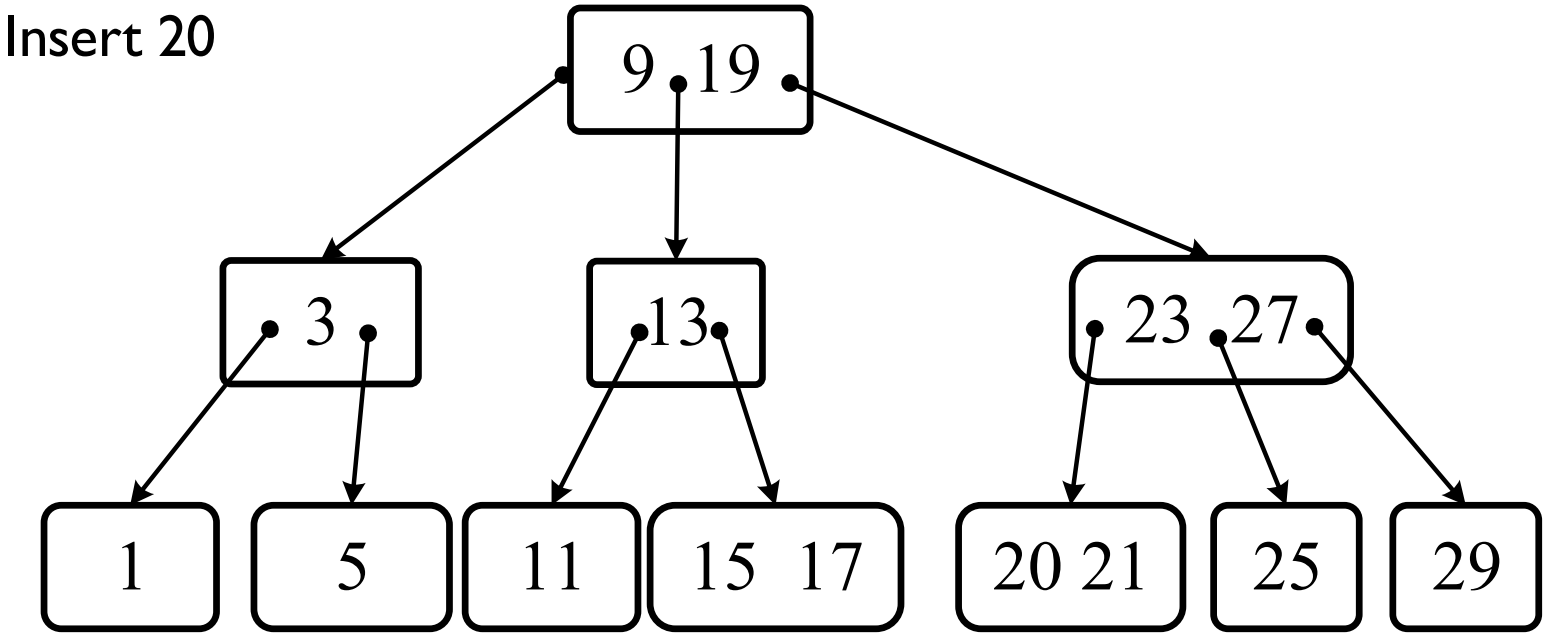
B-TREE-INSERTION:Ví dụ 2

Insert 25

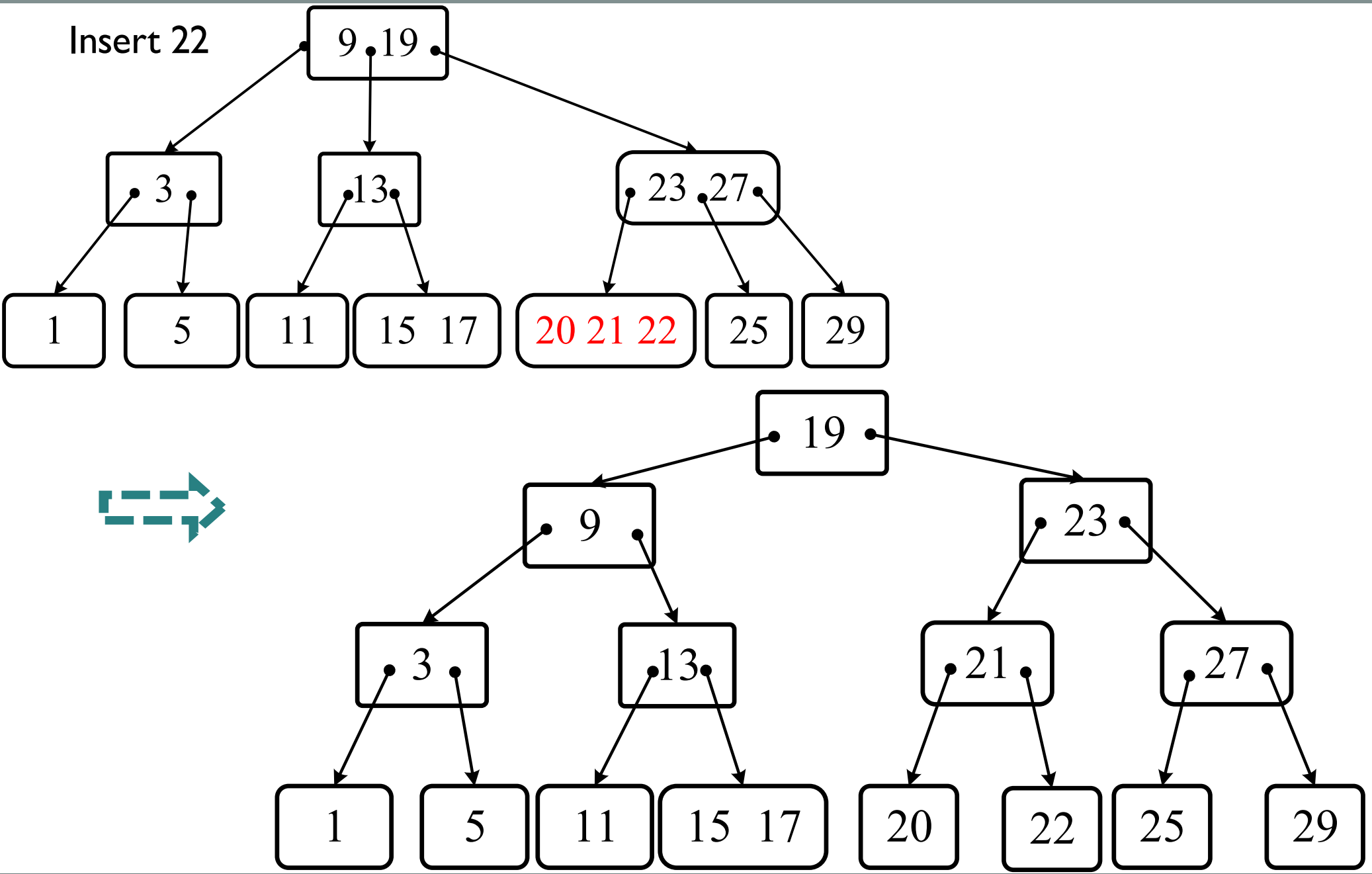




B-TREE-INSERTION:Ví dụ 2



B-TREE-INSERTION:Ví dụ 2



Nội dung

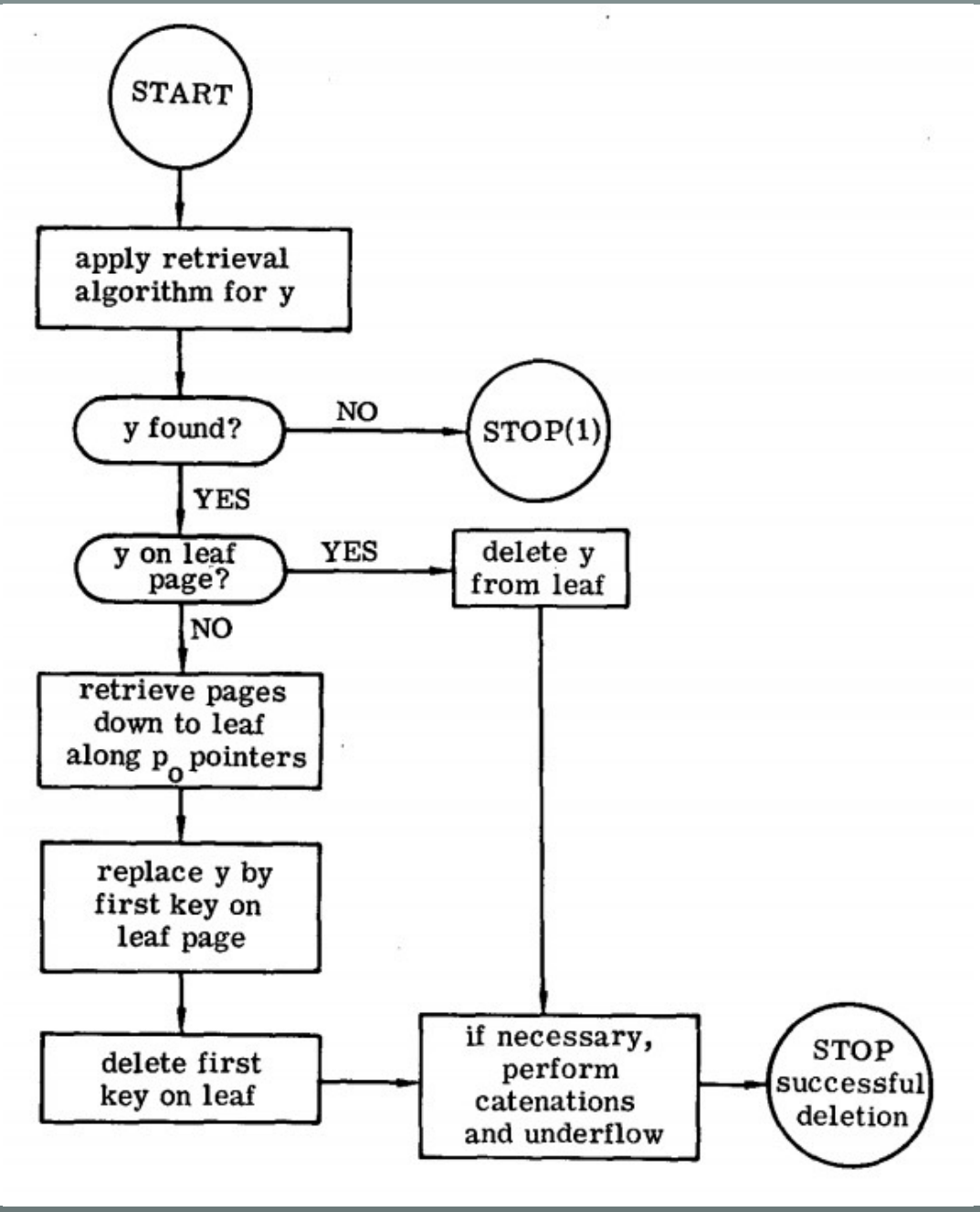
1. Định nghĩa using degree/order
2. Các phép toán cơ bản
   i. B-TREE-TRAVERSE
   ii. B-TREE-SEARCH
   iii. B-TREE-INSERTION
   iv. B-TREE DELETION

B-TREE DELETION

Có 2 trường hợp xóa:

- **Trường hợp 1**: Xóa khóa x của node lá
  - Xóa khóa x khỏi node lá
  - Nếu số lượng khóa trong node lá sau khi xóa x nhỏ hơn số lượng tối thiểu => Thực hiện Catenation hoặc Underflow
- **Trường hợp 2**: Xóa khóa x của node không phải node lá
  - Tìm khóa thế mạng là khóa lớn nhất trên cây con bên tay trái hoặc là khóa nhỏ nhất của cây con bên tay phải của node cần xóa
  - Hoán vị x với khóa thế mạng
  - Tiến hành xóa khóa x ở lá

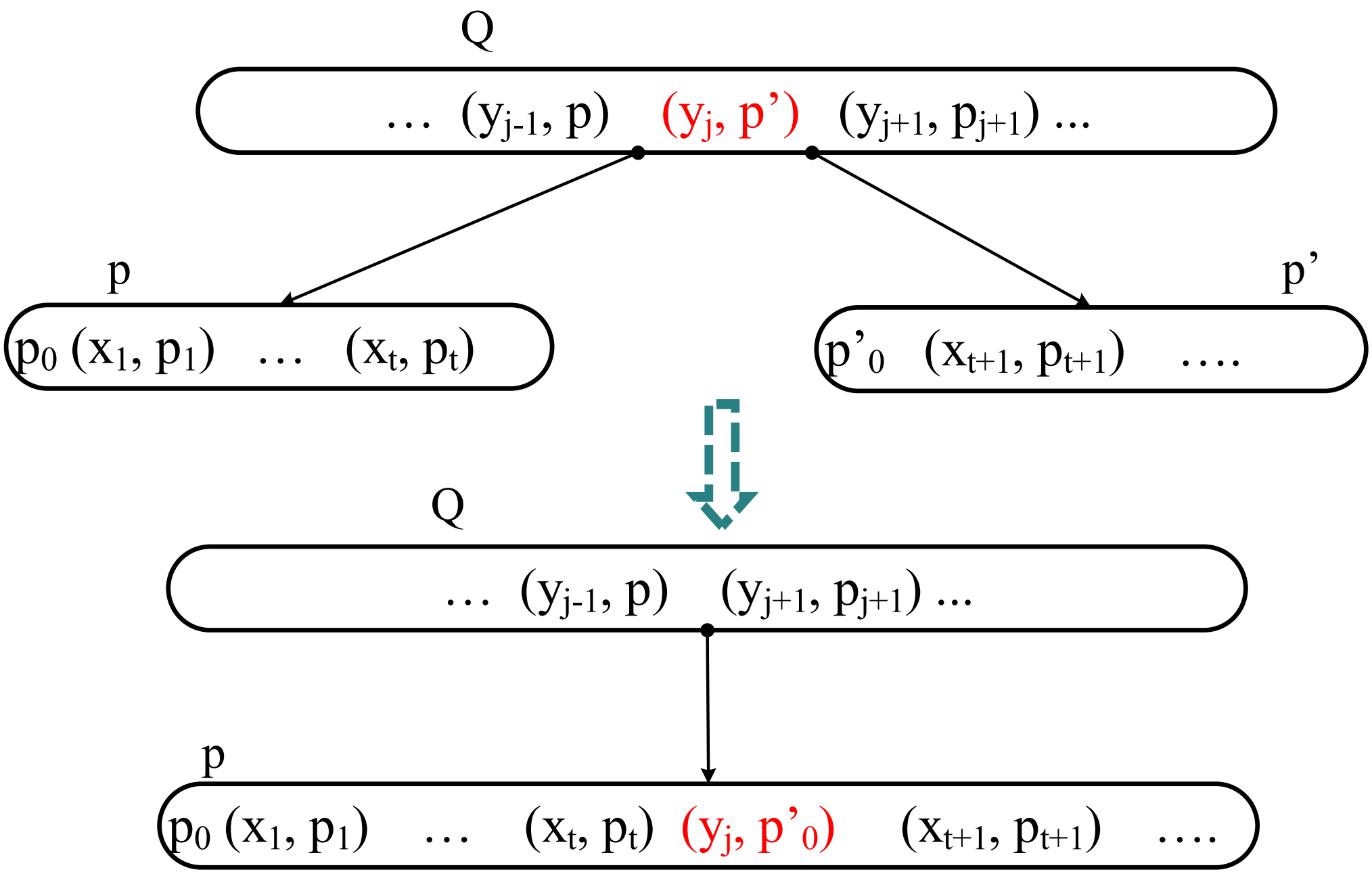
B-TREE DELETION



B-TREE DELETION - Catenation

- Cantenation: p và p' gọi là *adjacent brothers* vì có cùng cha Q và được trỏ bởi 2 con trỏ kề trong Q. 2 node p và p' có thể thực hiện Cantenation nếu khi kết hợp p và p' với nhau có số khóa **nhỏ hơn (<) số khóa tối đa** của node.
- Việc xóa khóa y_j khỏi Q có thể dẫn đến Q chứa ít hơn k khóa, như vậy ta phải xử lý tiếp khóa Q. Quá trình này **có thể lan truyền** tới tận gốc của cây.

B-TREE DELETION - Catenation

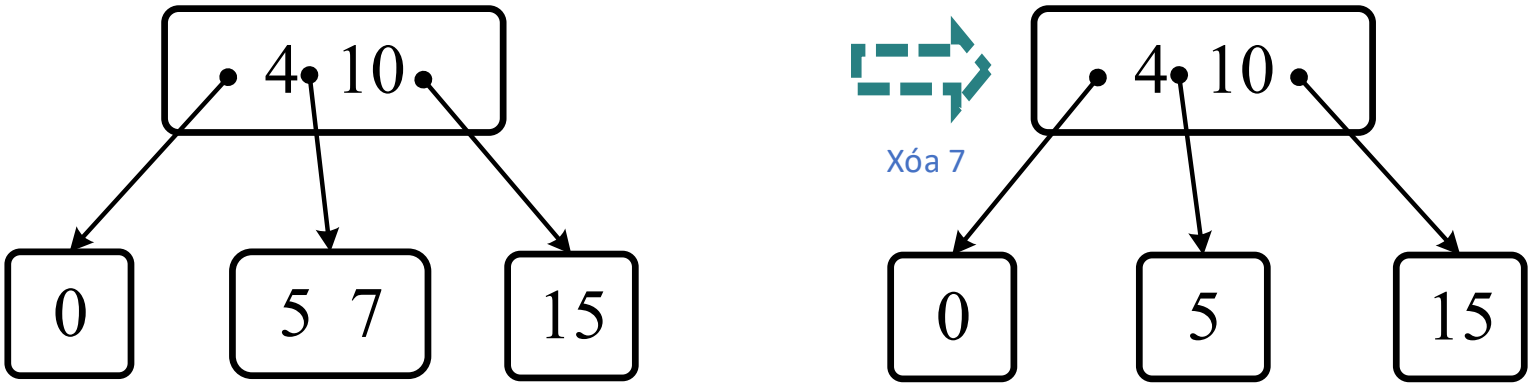


B-TREE DELETION – Underflow

- Underflow: Nếu tổng số lượng khóa của node p và p' **nhiều hơn** 2k (>=2k, với bậc m=2*k+1, hoặc số khóa tối đa), thì khóa trong p và p' có thể chia đều nhau, quy trình này gọi là underflow, được thực hiện như sau:
  - Catenation p và p' cho ra node p lớn
  - Split p lớn như thao tác split trình bày ở phần trên.
  - Chú ý: thao tác Underflow **không lan truyền**. Q được sửa đổi, nhưng số lượng khóa thì không thay đổi.

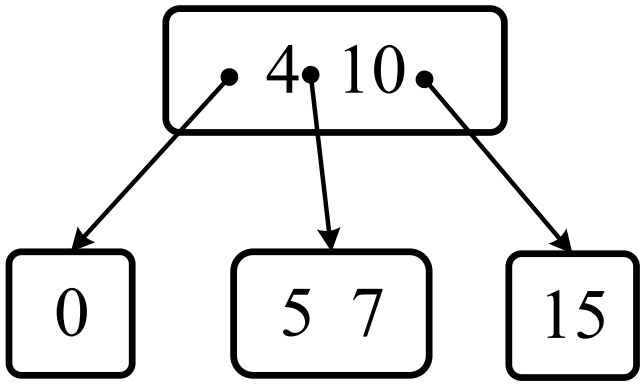
B-TREE DELETION - Ví dụ: Xóa khóa ở lá

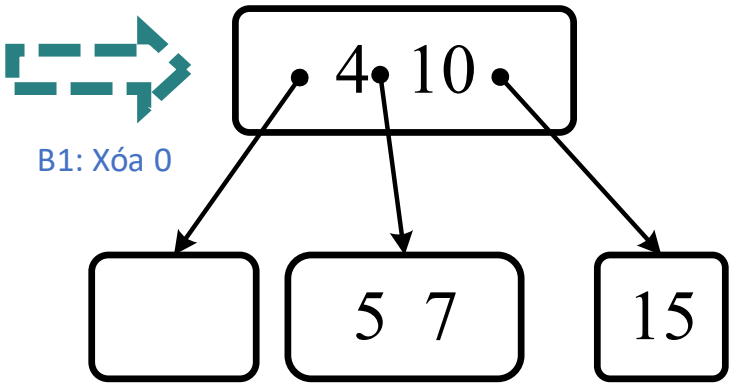
Xóa 7 trên cây B-Tree bậc m=3



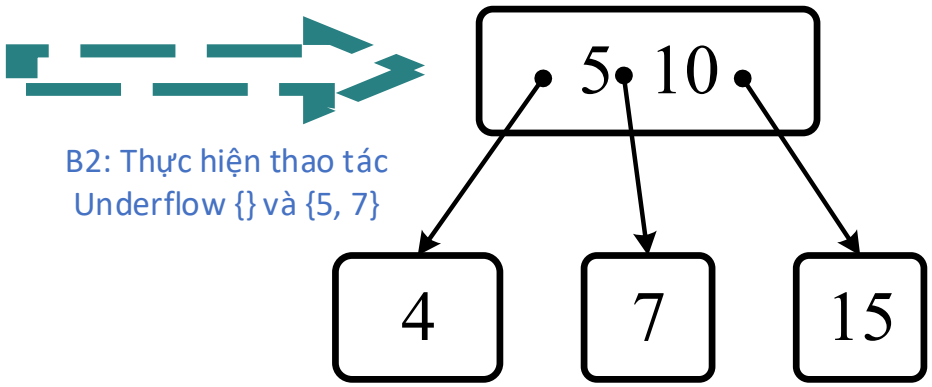
B-TREE DELETION - Ví dụ: Xóa khóa ở lá

Xóa 0 trên cây B-Tree bậc m=3



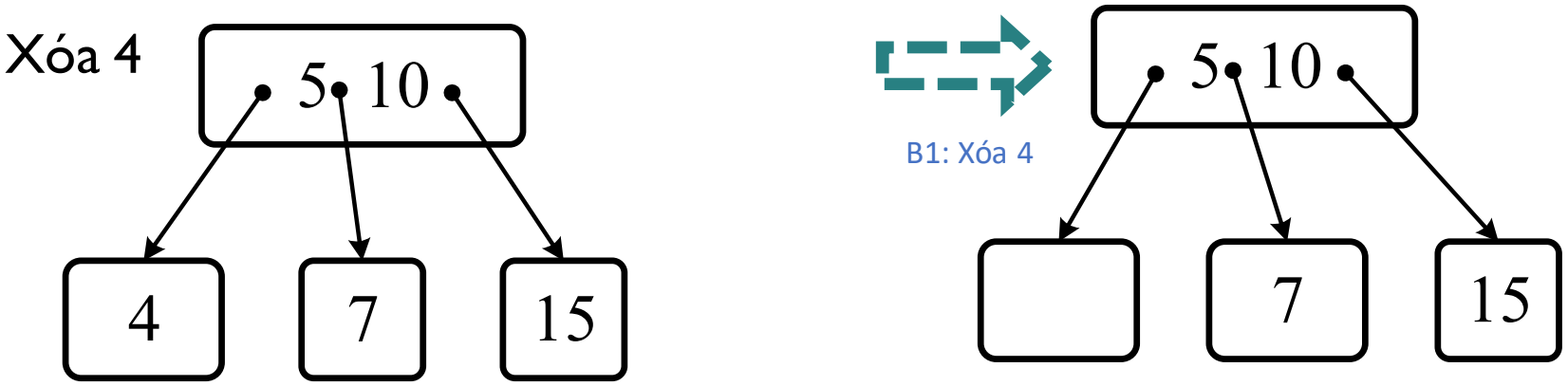


Tổng số khóa của {} và {5, 7} là 2 = số khóa Tối đa
=> Thực hiện thao tác Underflow {} và {5, 7}



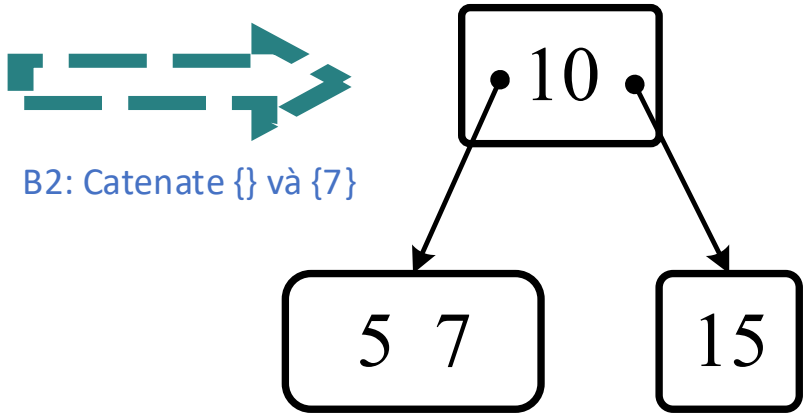
B-TREE DELETION - Ví dụ: Xóa khóa ở lá

Xóa 4 trên cây B-Tree bậc m=3



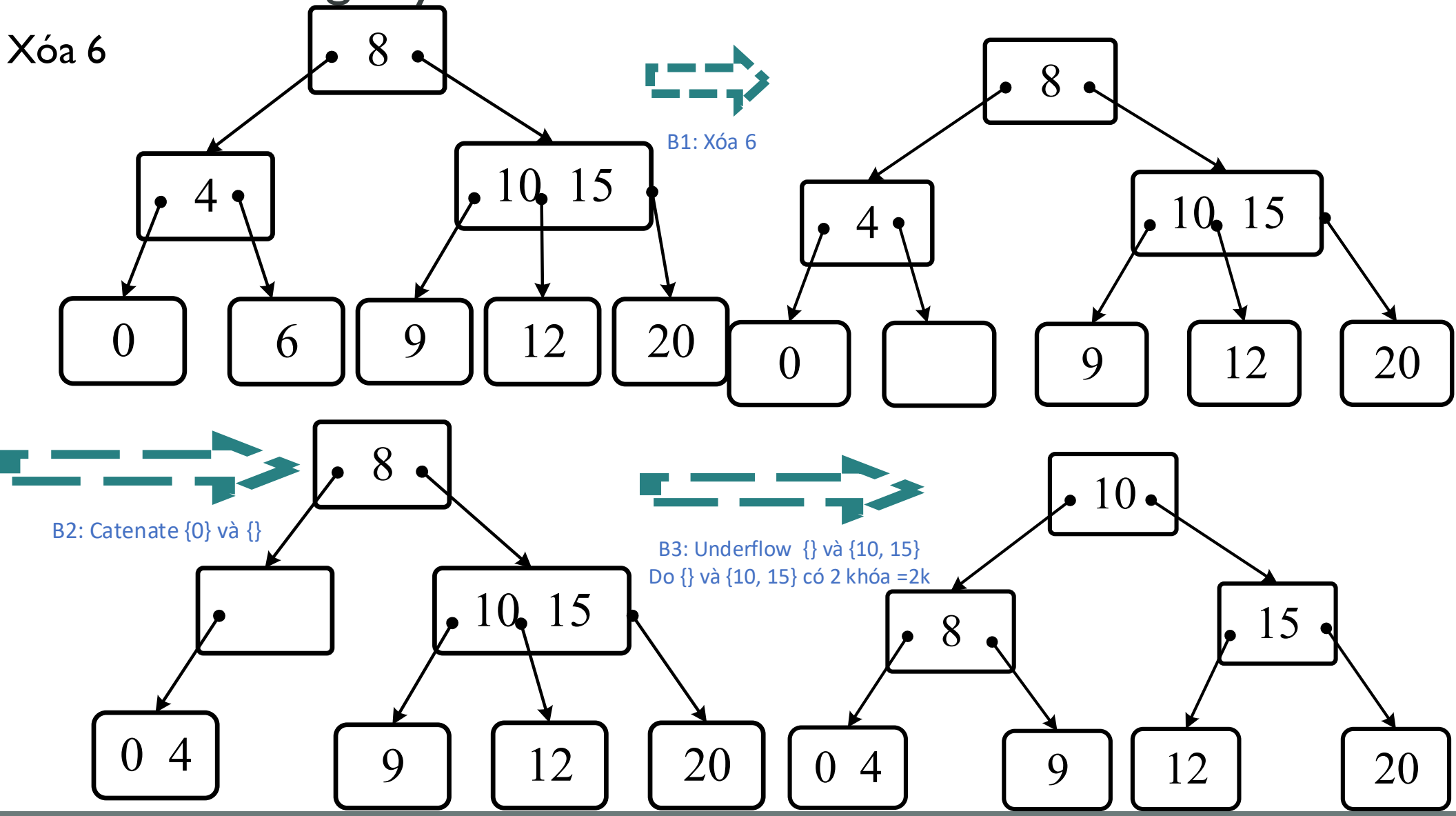
Tổng số khóa {} và {7} là 1 (<2k, với m=2k+1)

=> Thực hiện thao tác Catenate {} và {7}



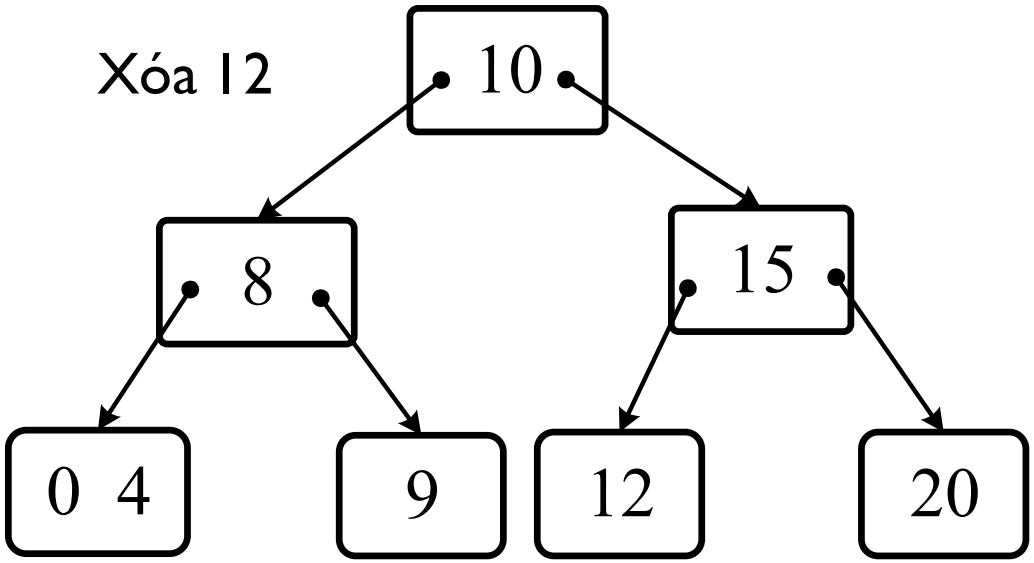
B-TREE DELETION - Ví dụ: Xóa khóa ở lá

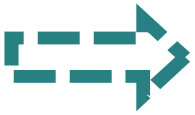
- Xóa 6 trong cây B-Tree m=3



B-TREE DELETION - Ví dụ: Xóa khóa ở lá

- Tiếp tục Xóa 12:

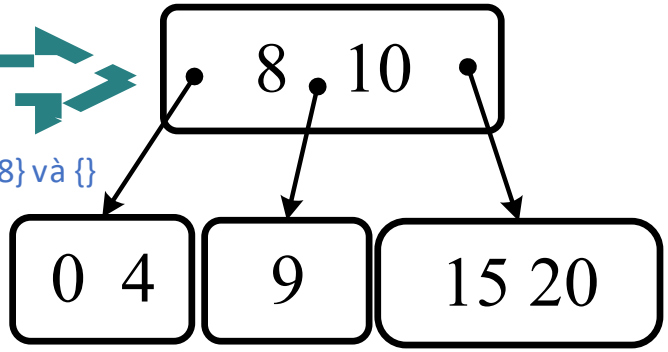




B1: Xóa 12

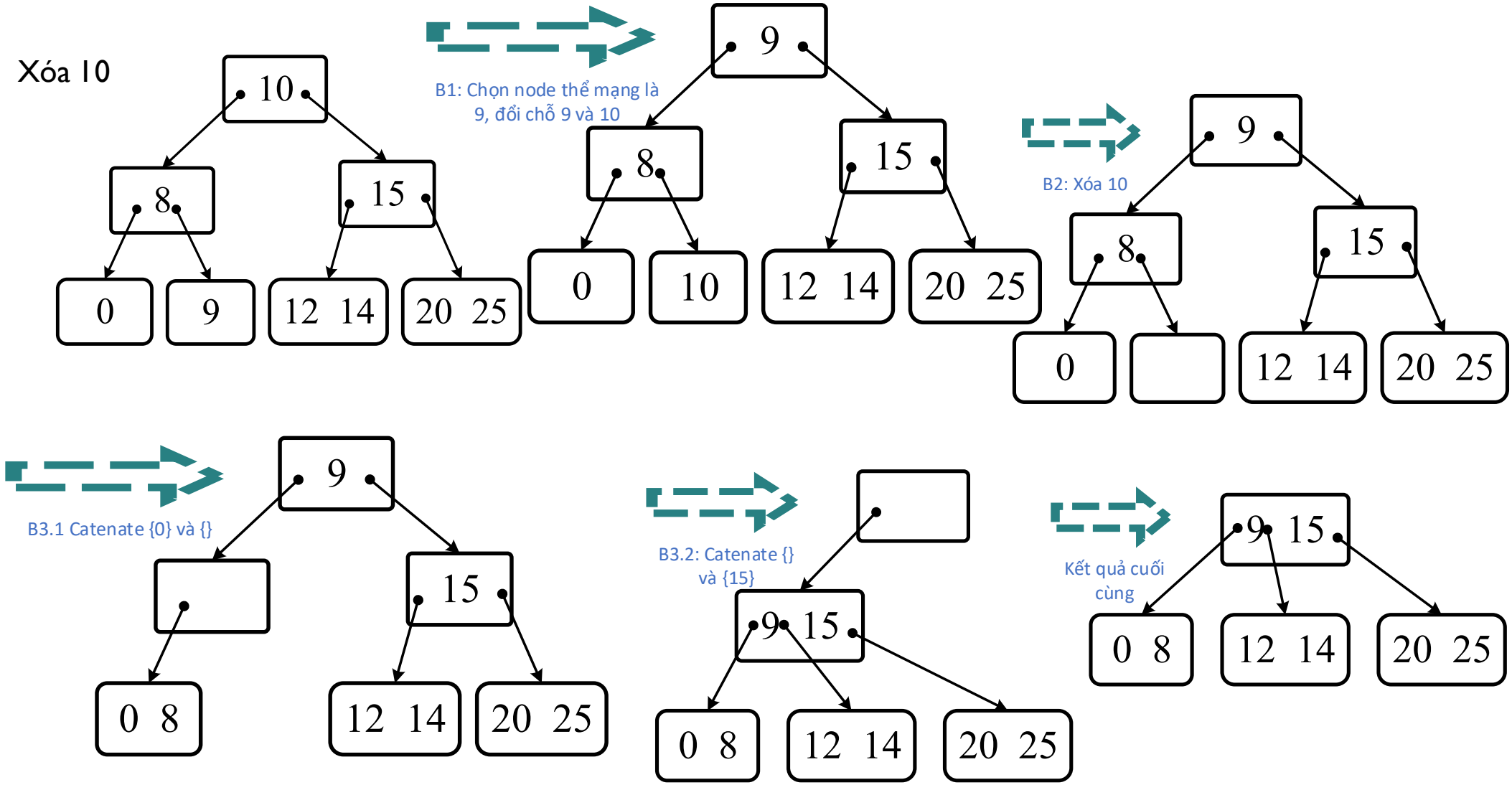
Catenate {} và {20}

Catenate {8} và {}



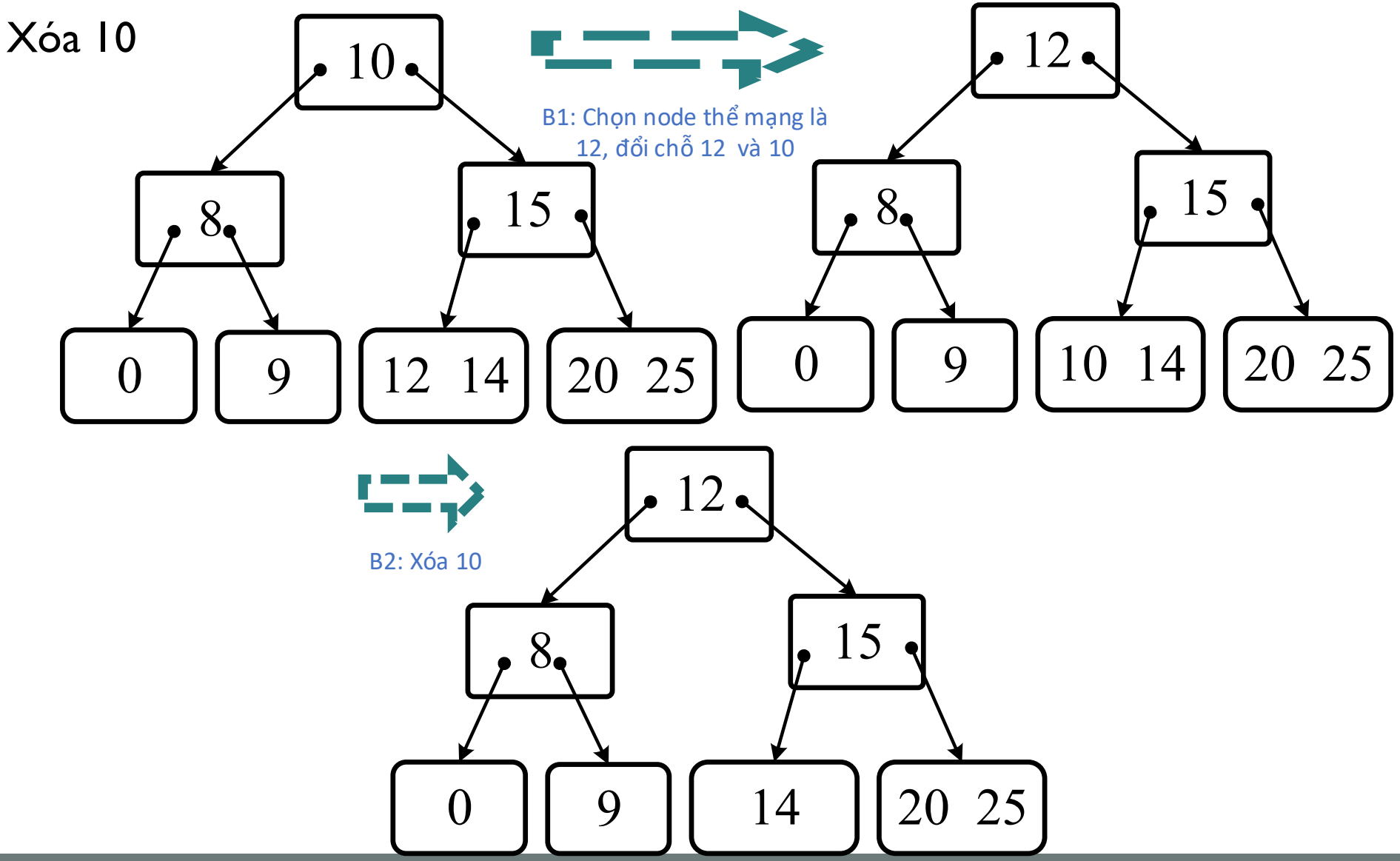
B-TREE DELETION - Vd: Xóa khóa node khác lá

- Xóa 10 trên cây B-Tree bậc m=3, chọn 9 làm key thế mạng



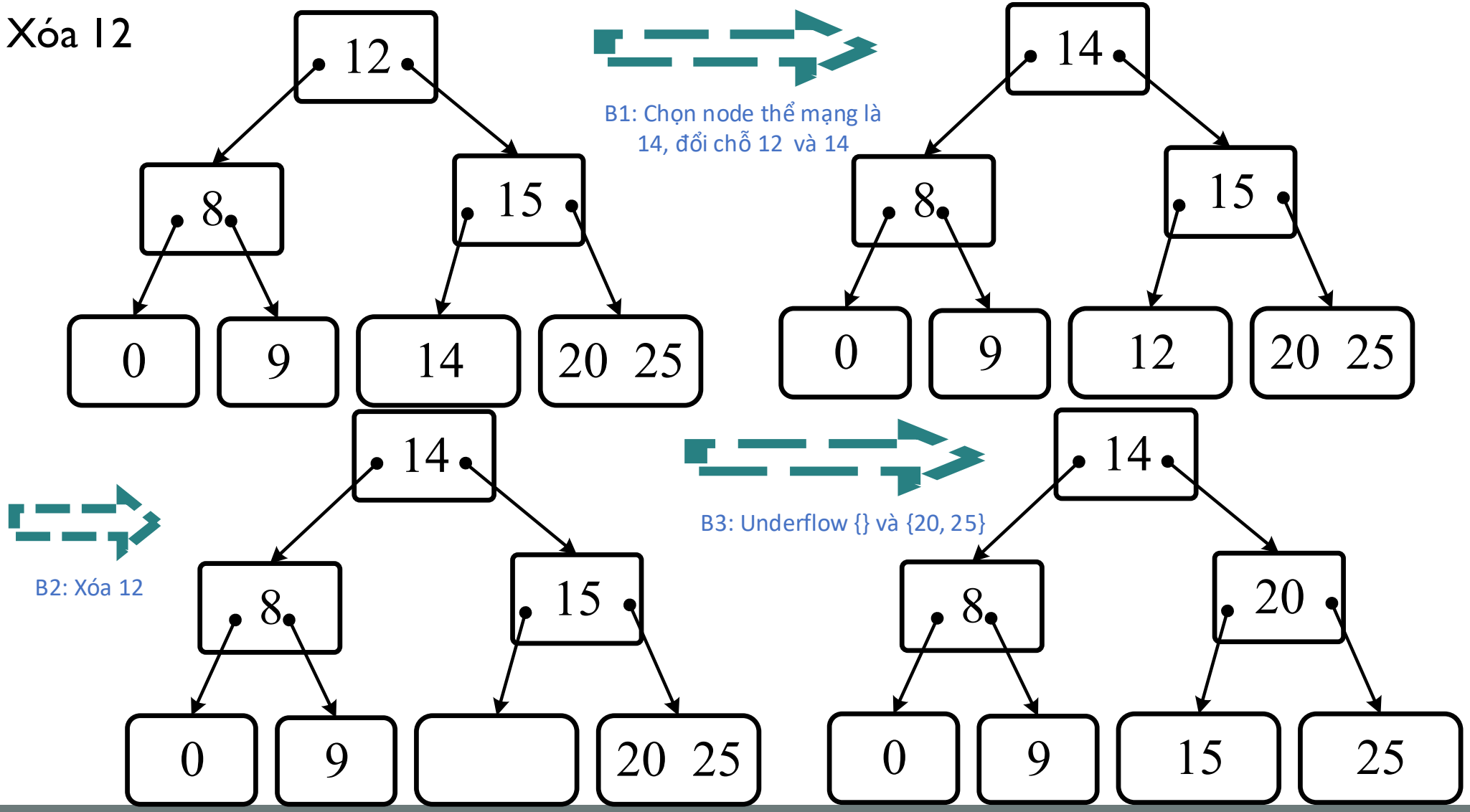
B-TREE DELETION - Vd: Xóa khóa node khác lá

- Xóa 10 trên cây B-Tree bậc m=3, chọn 12 là key thế mạng



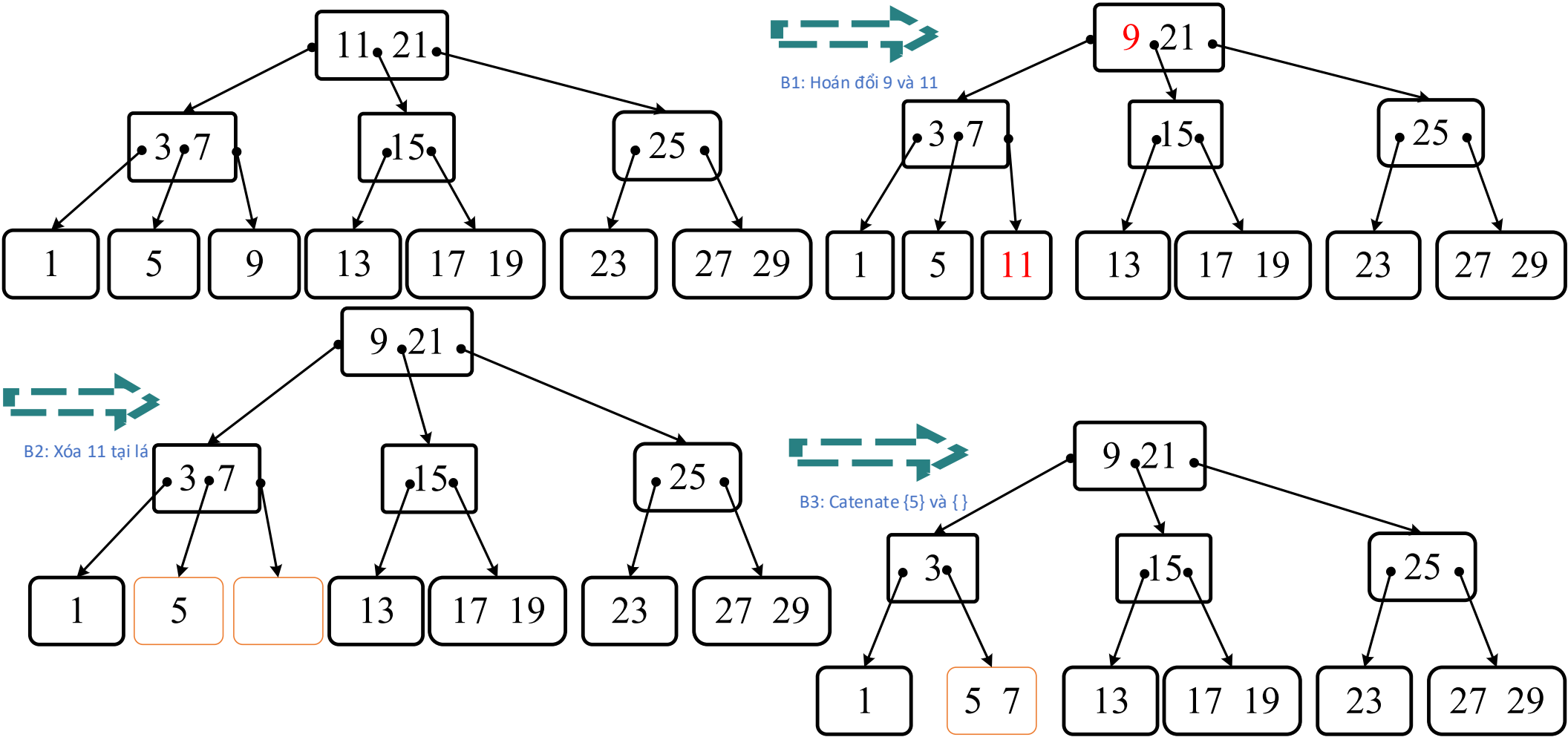
B-TREE DELETION - Vd: Xóa khóa node khác lá

- Xóa 12 trên cây B-Tree bậc m=3



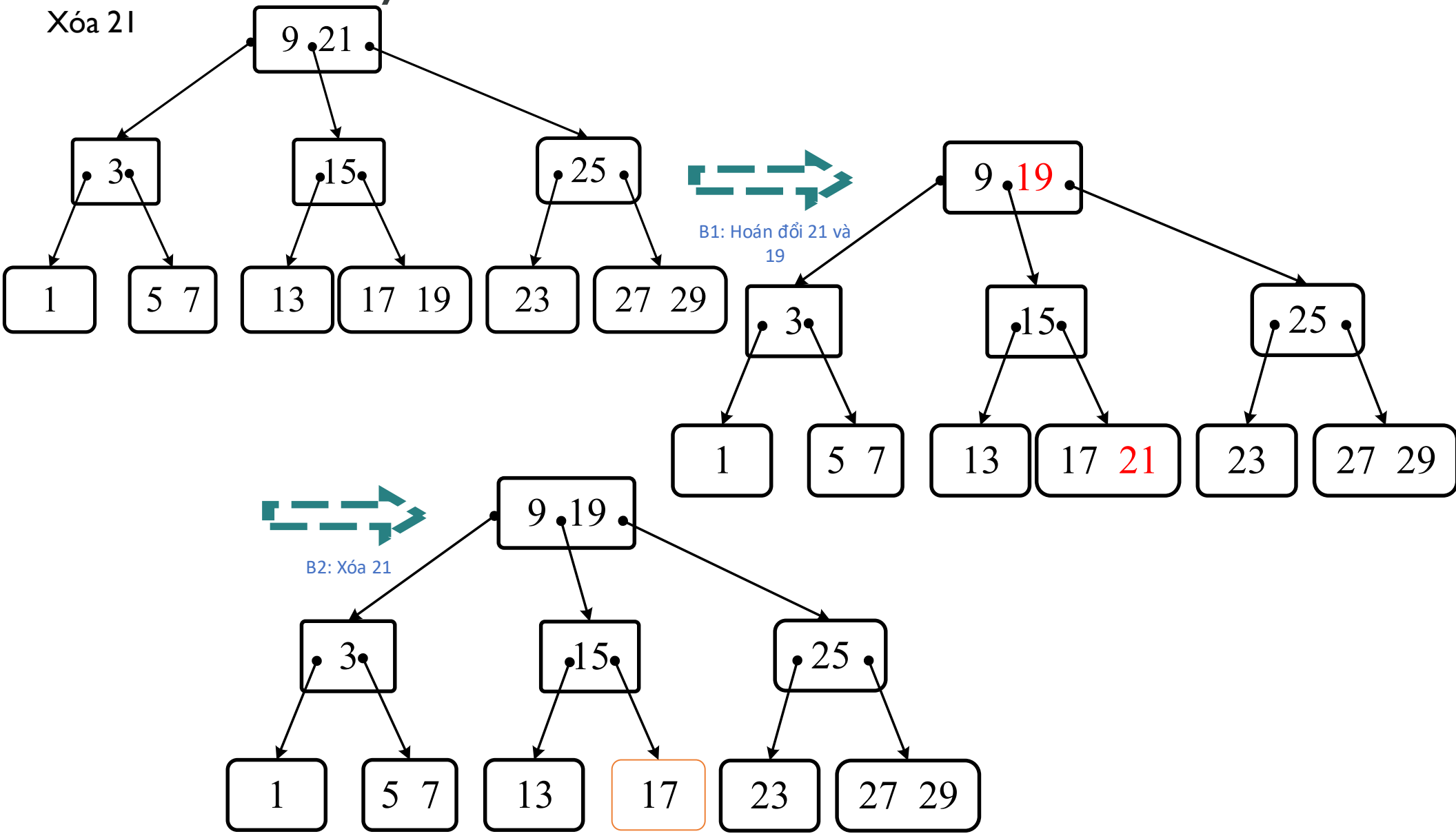
B-TREE DELETION - Vd: Xóa ở node gốc

- Xóa 11 trên cây B-Tree bậc 3, chọn khóa thay thế là khóa lớn nhất của cây con bên trái.



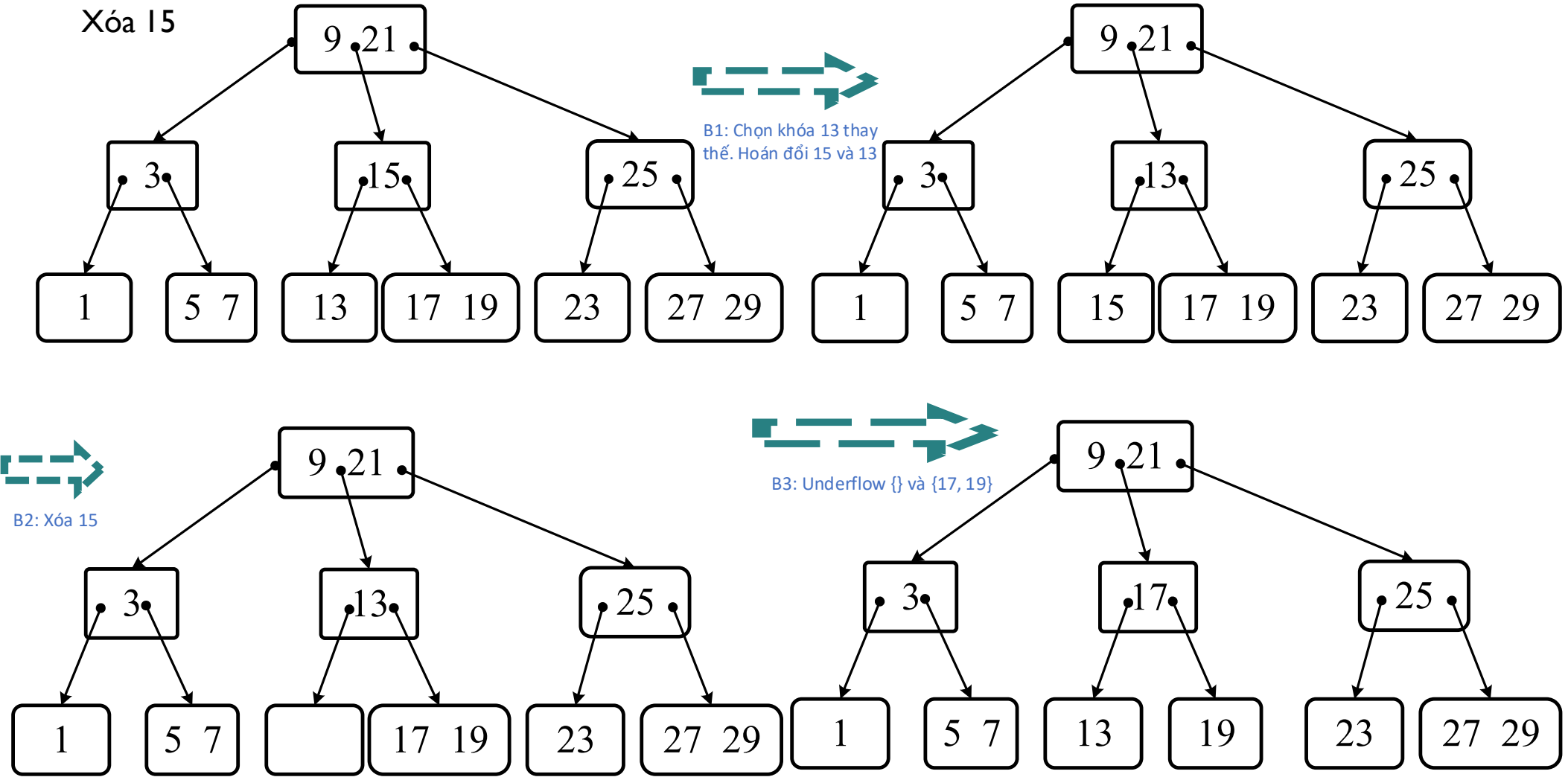
B-TREE DELETION - Vd: Xóa ở node gốc

Xóa 21 trên cây B-Tree bậc 3



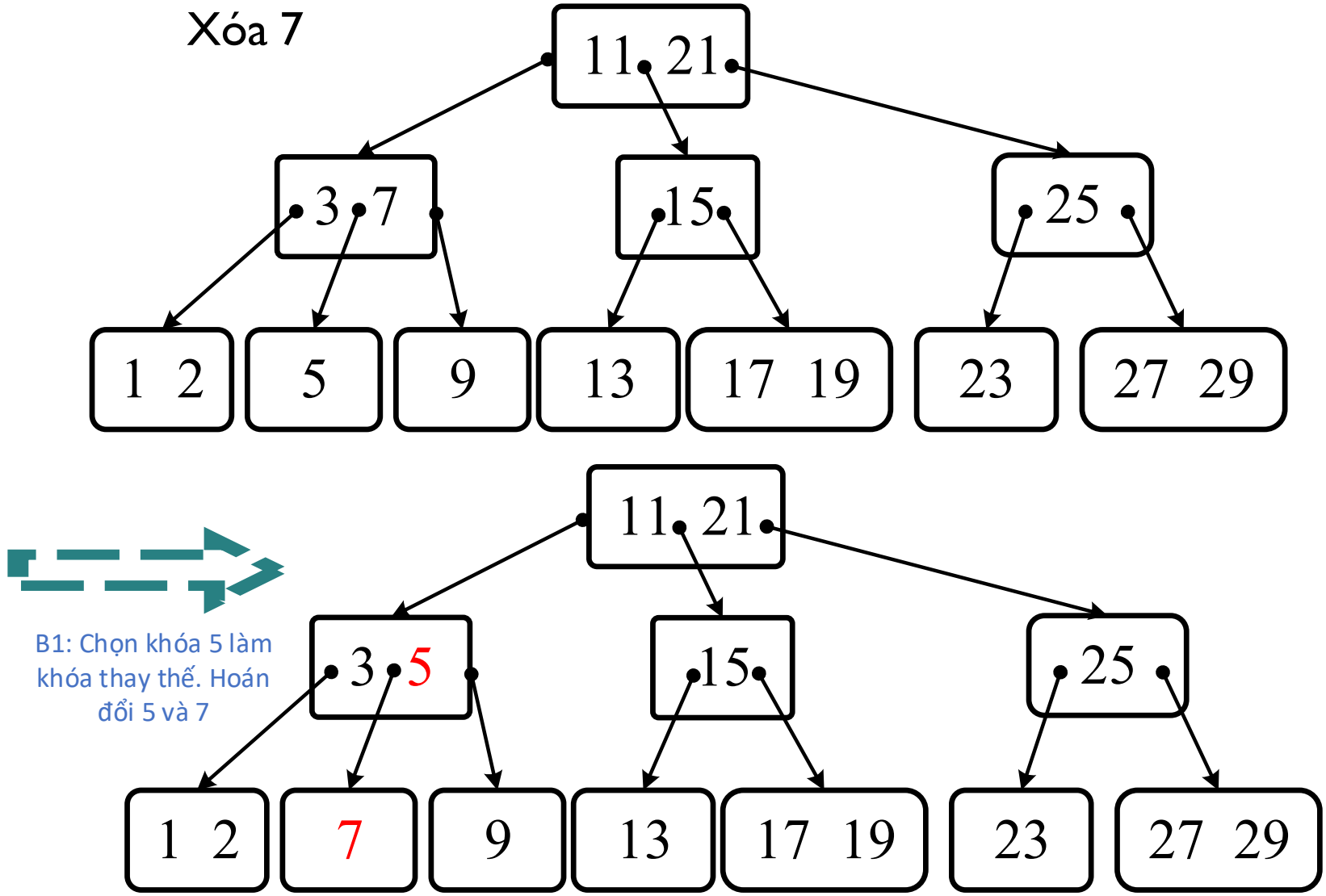
B-TREE DELETION - Vd: Xóa node khác lá

- Xóa 15 trên cây B-Tree bậc 3:

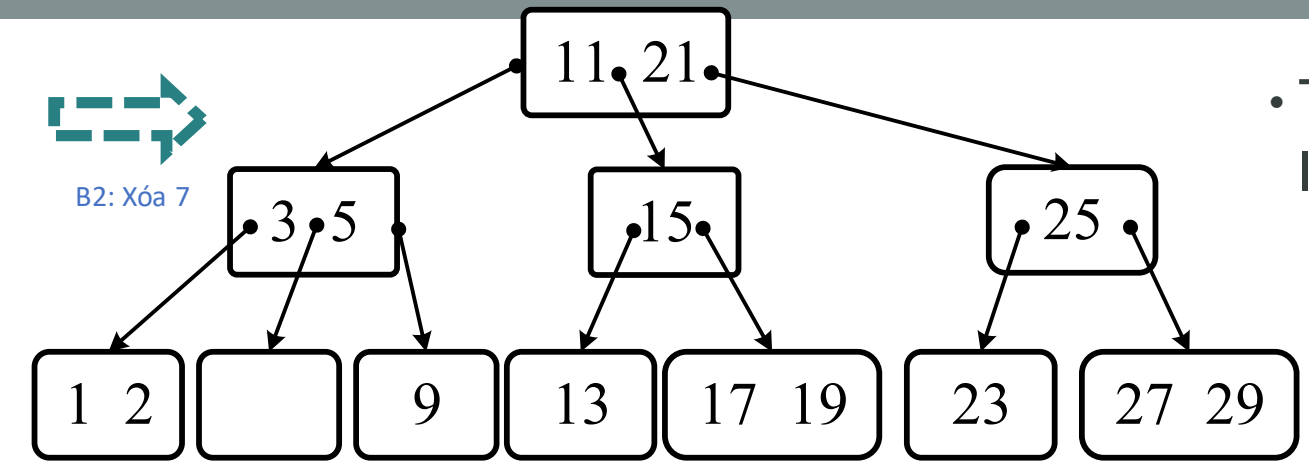


B-TREE DELETION - Vd: Xóa node khác lá

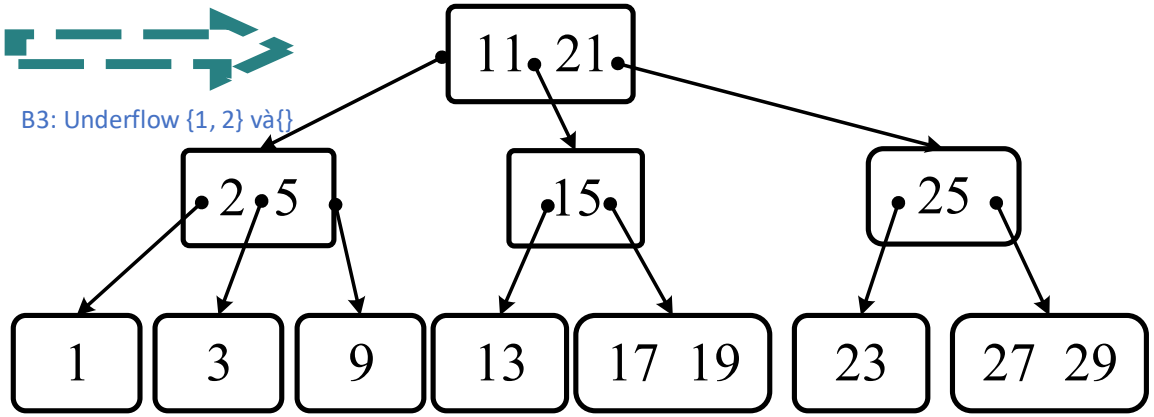
- Xóa 7 trên cây B-Tree bậc 3:

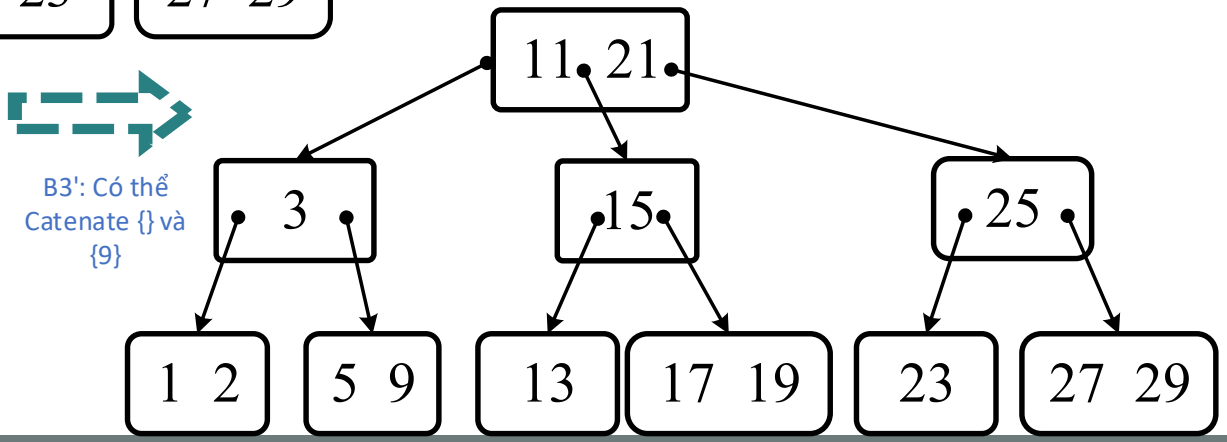


B-TREE DELETION - Vd: Xóa node khác lá

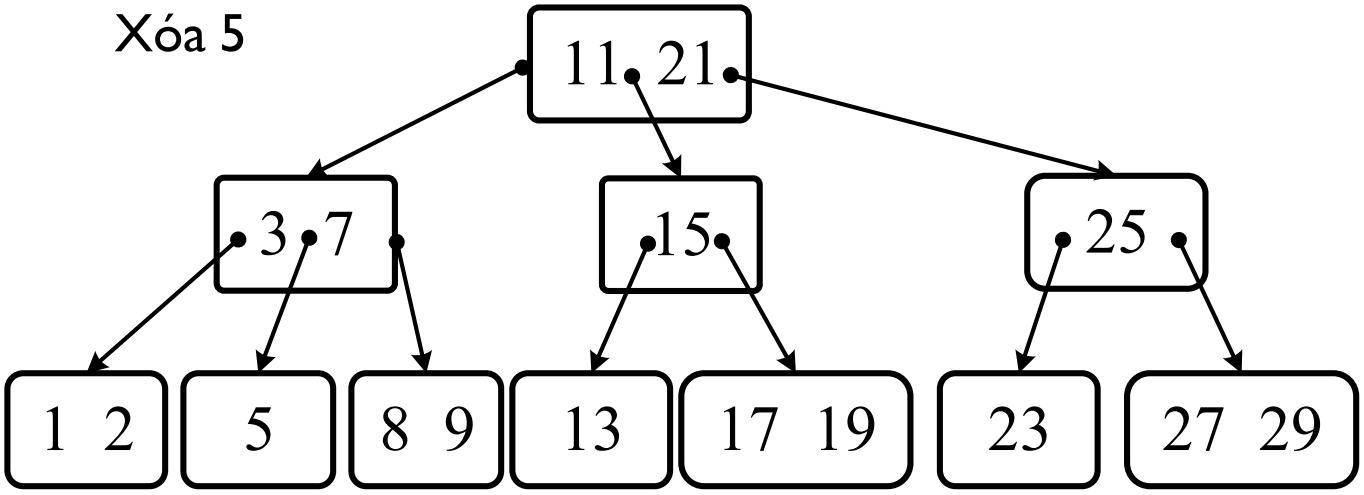


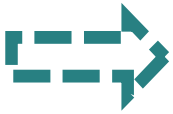
- Tại đây chọn 1 trong 2 cách xử lý sau:
  - Underflow {1, 2} và {}
  - Hoặc Catenate {} và {9}





B-TREE DELETION - Vd: Xóa node khác lá





B1: Xóa 5

Có 2 cách xử lý sử dụng Underflow ở đây!

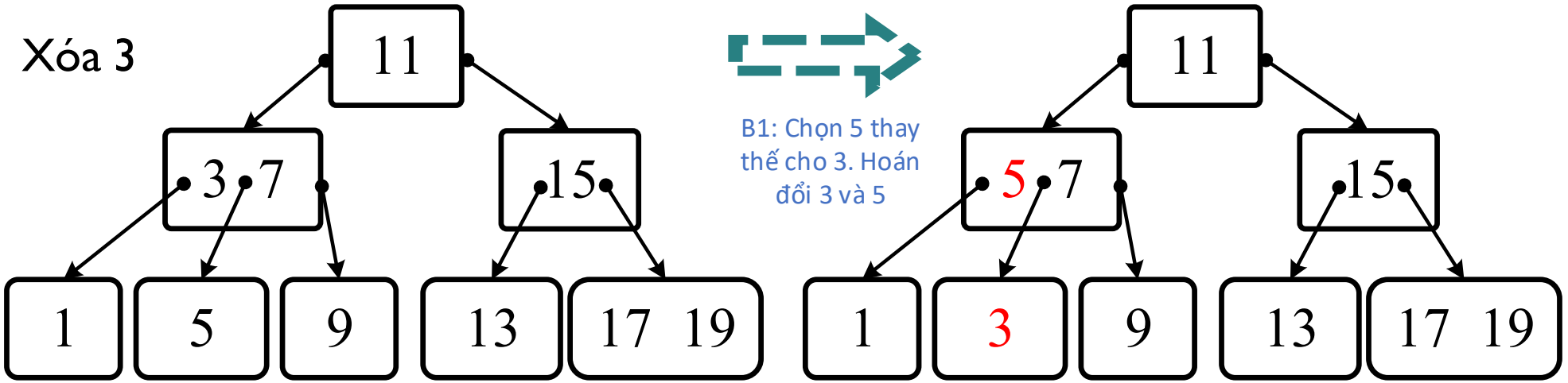
B-TREE DELETION - Vd: Xóa node khác lá

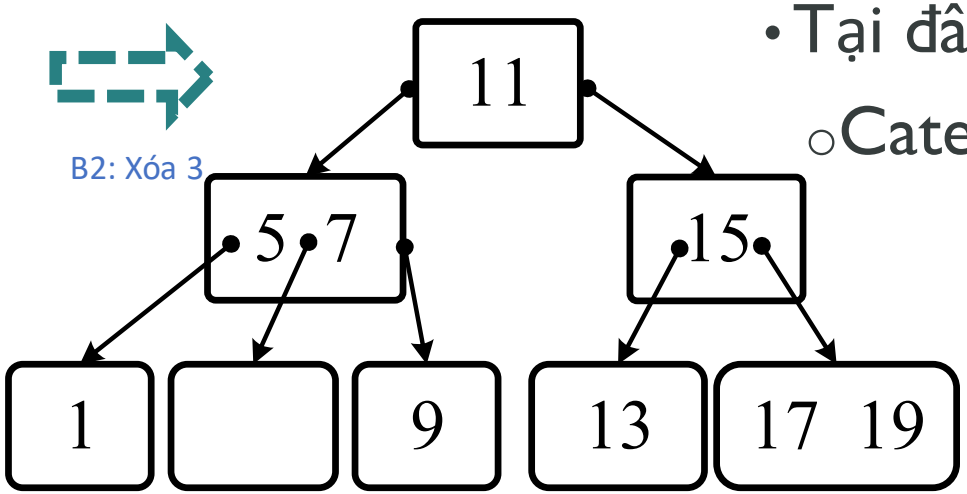
B2': Underflow {1, 2} và {}

Hoặc: B2': Underflow {} và {8 9}

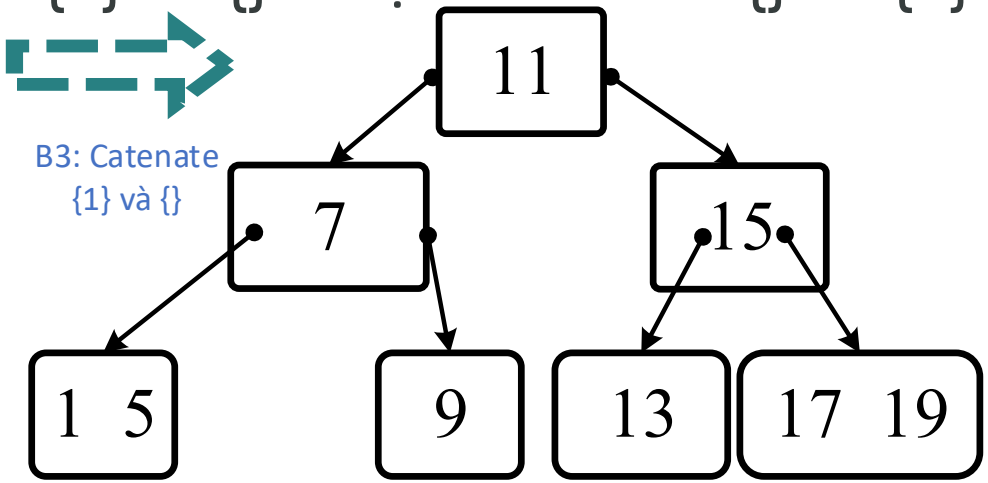
B-TREE DELETION - Vd: Xóa node khác lá

- Xóa 3 trên cây B-Tree bậc 3:

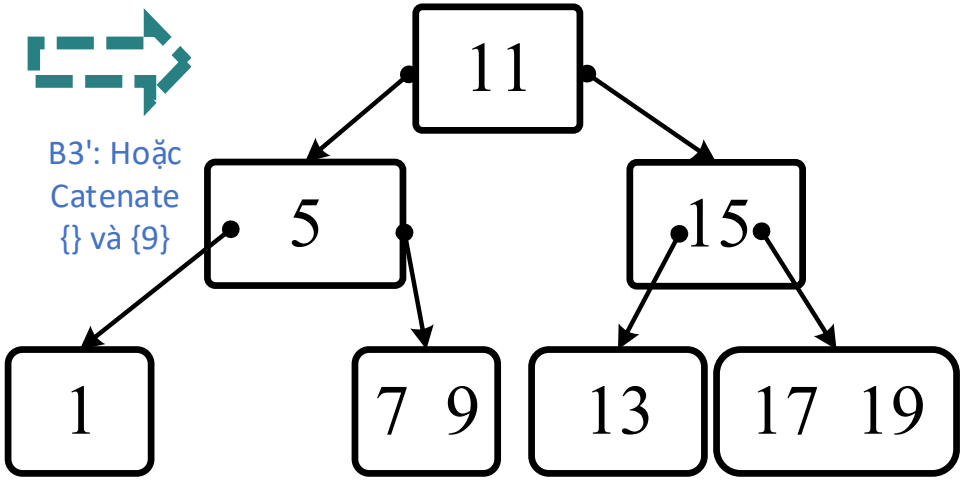




- Tại đây có 2 cách xử lý:
  - Catenate {1} và {} Hoặc Catenate {} và {9}

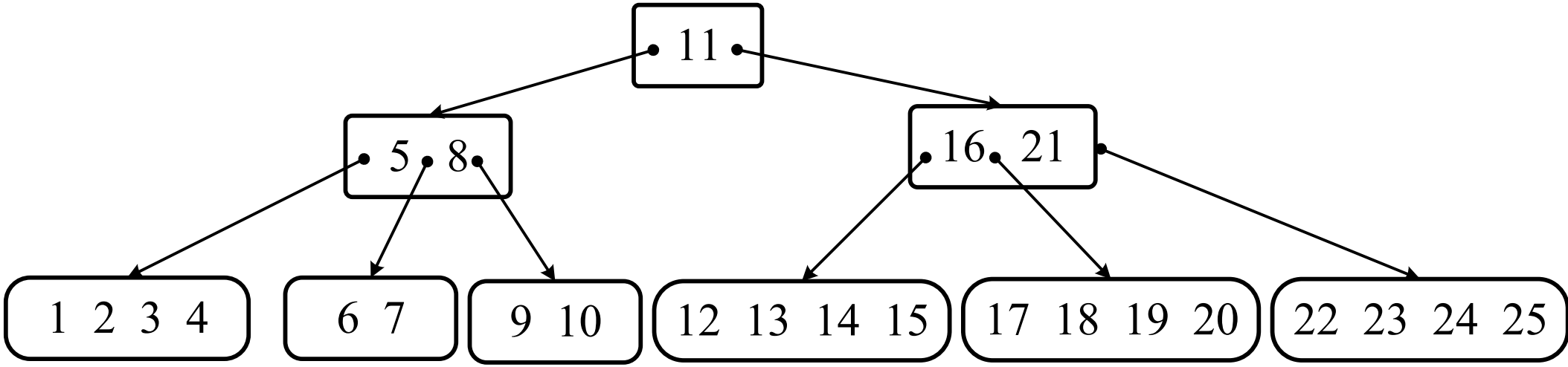


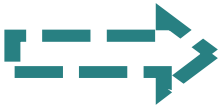
B-TREE DELETION - Vd: Xóa node khác lá



B-TREE DELETION - Ví dụ: Xóa ở node lá

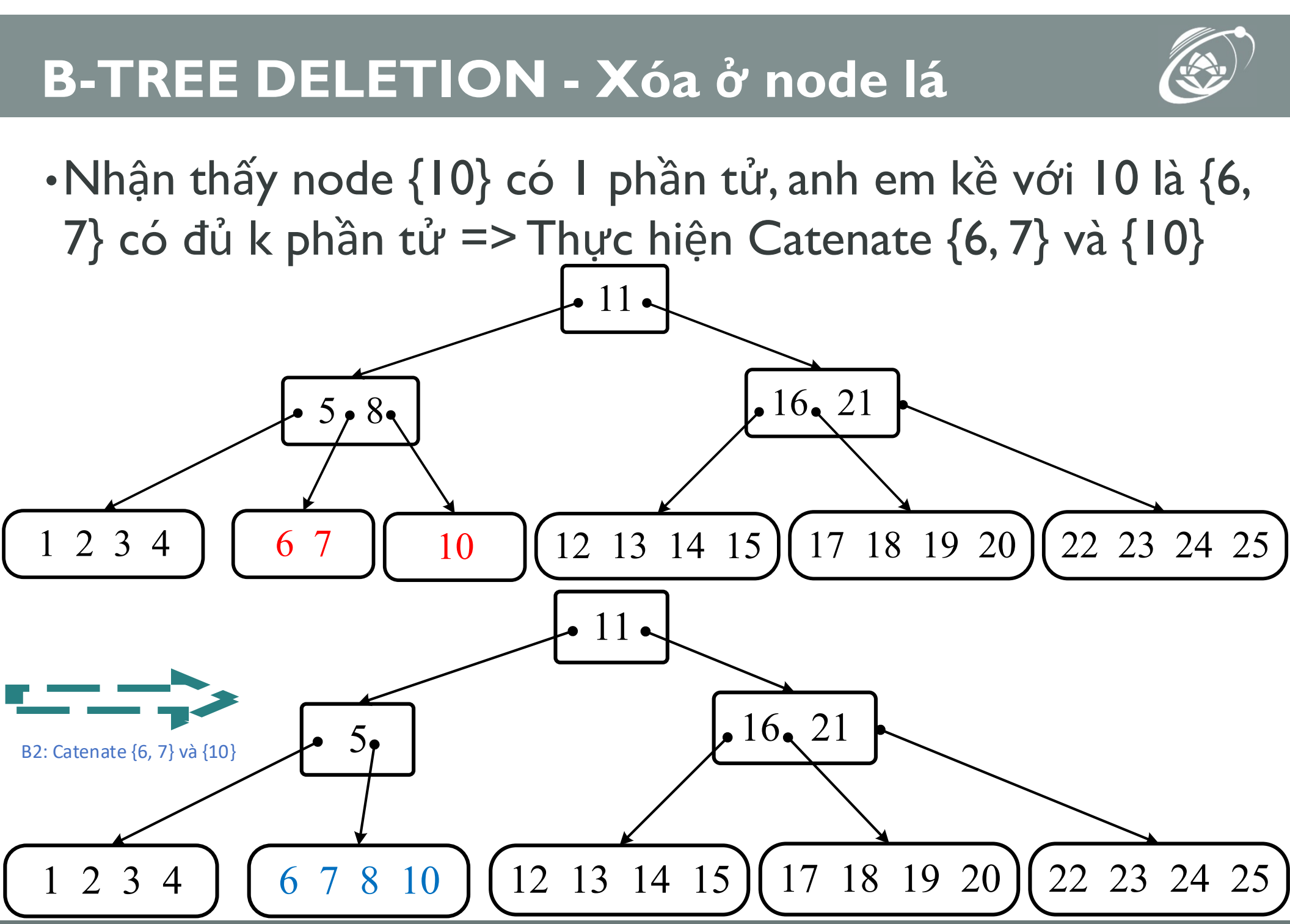
- Xóa 9 trên cây B-Tree bậc 5:





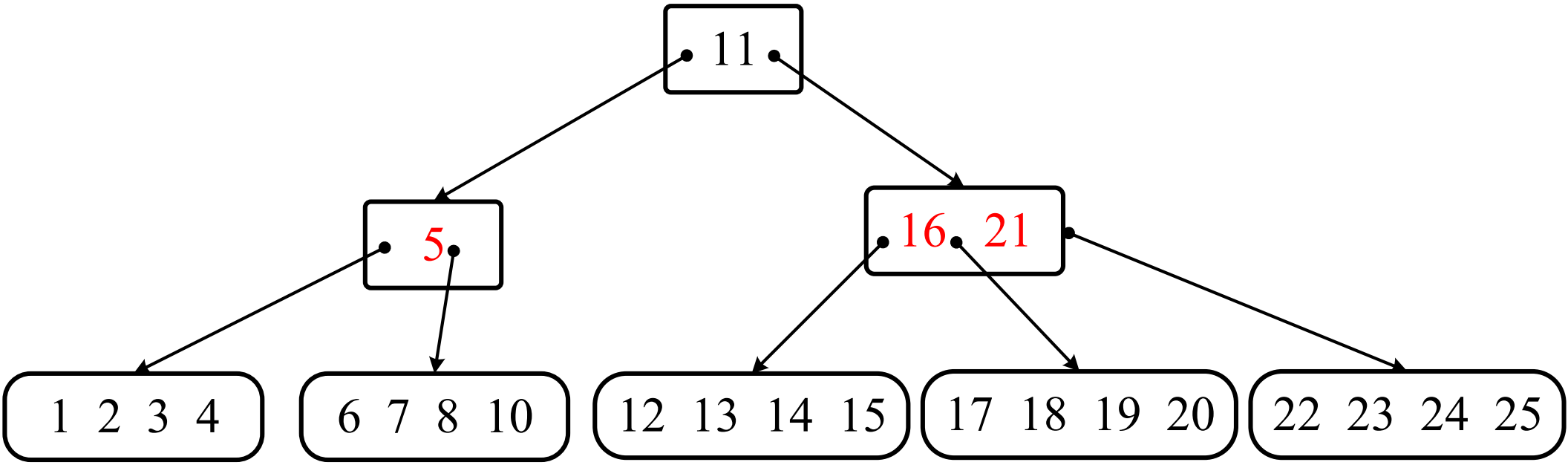
B1: Xóa 9 tại lá

B-TREE DELETION - Xóa ở node lá

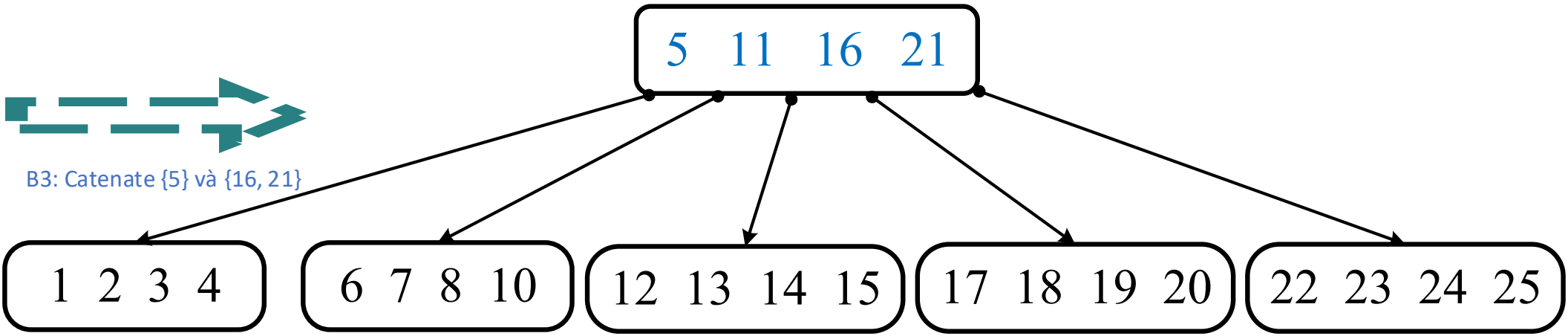


- Nhận thấy node {10} có 1 phần tử, anh em kề với 10 là {6, 7} có đủ k phần tử => Thực hiện Catenate {6, 7} và {10}

B-TREE DELETION - Xóa ở node lá

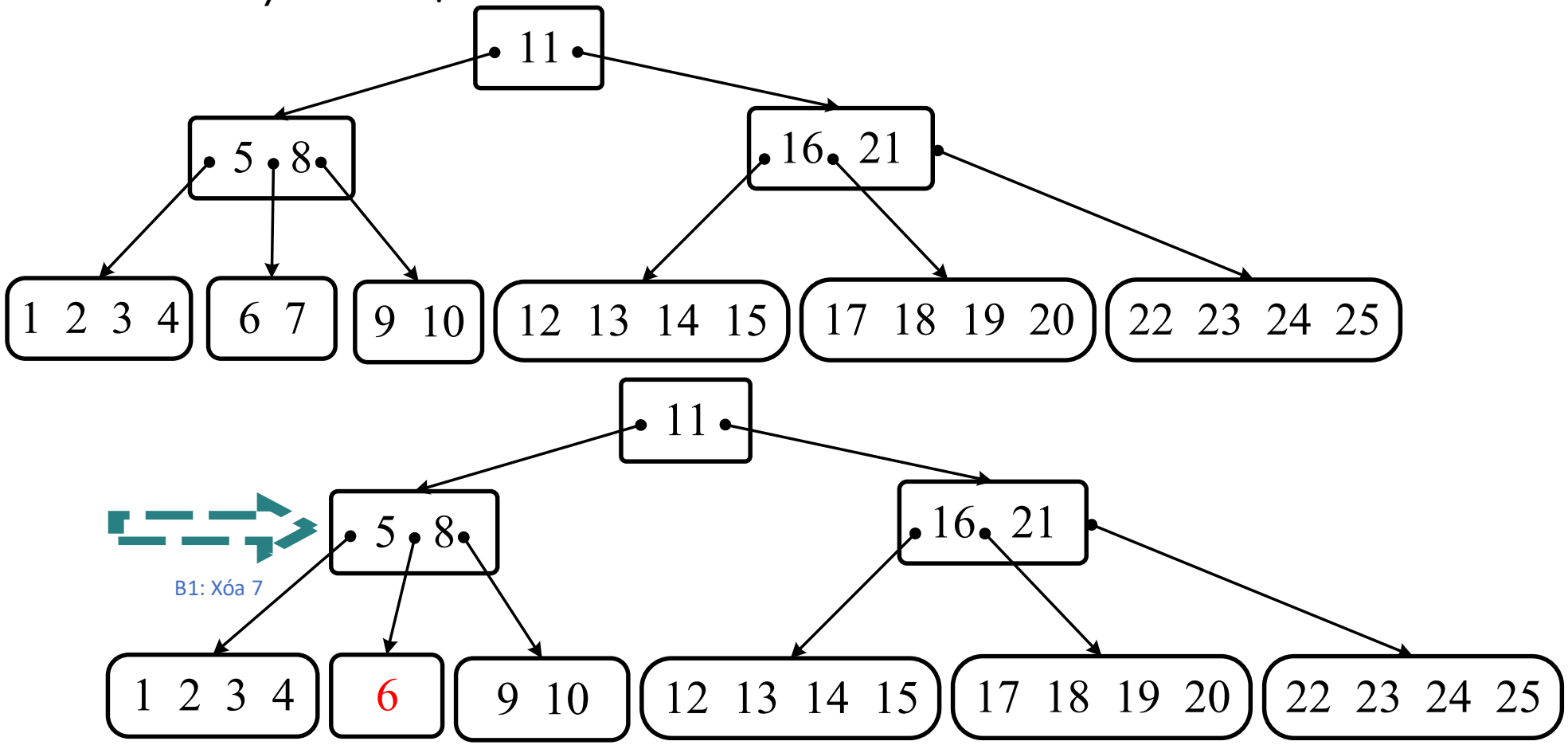


B3: Catenate {5} và {16, 21}



B-TREE DELETION - Xóa ở node lá

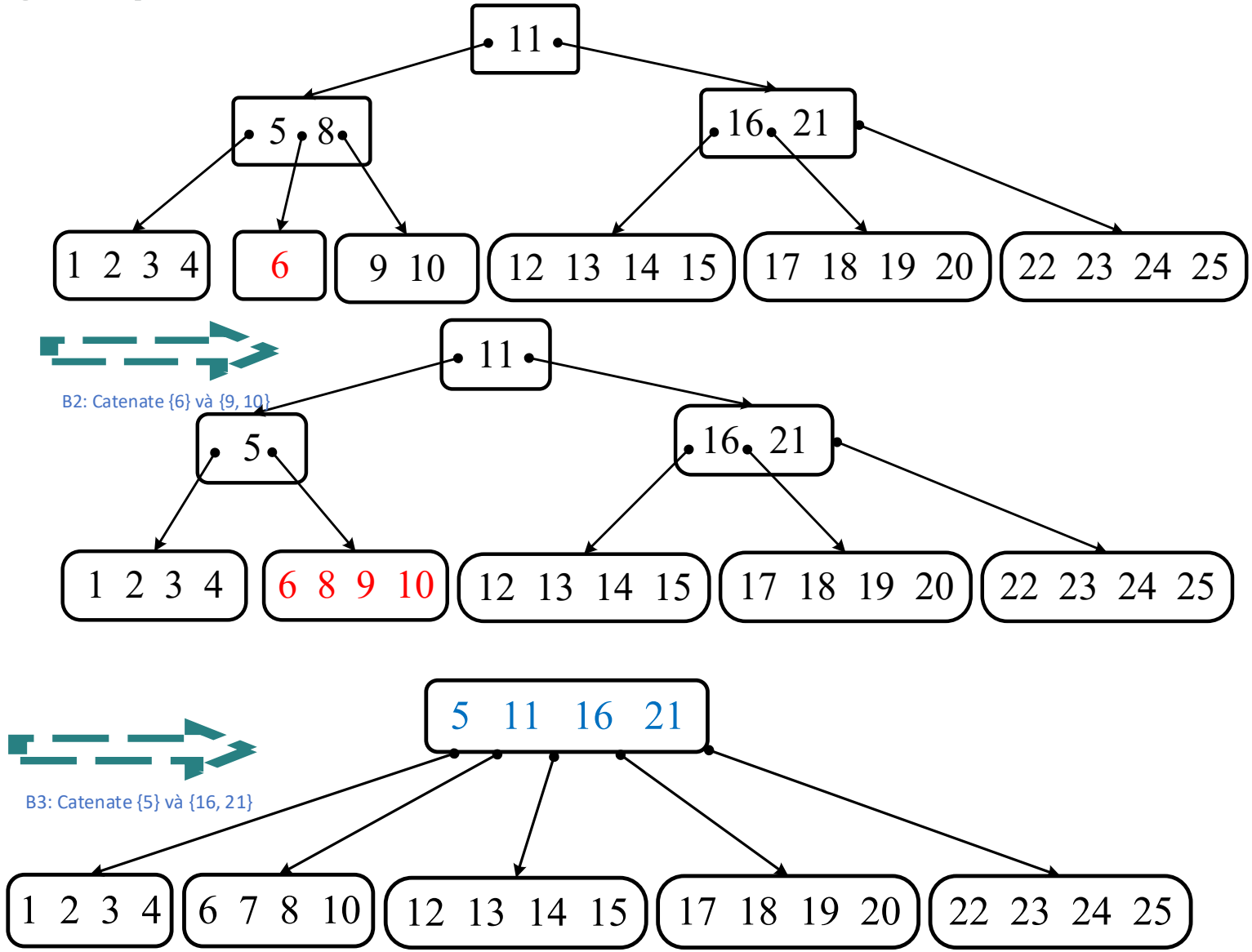
Xóa 7 trên cây B-Tree bậc 5



- Ở đây có thể xử lý
  - Underflow {1 2 3 4} và {6}
  - Hoặc Catenate {6} và {9 10}

B-TREE DELETION - Xóa ở node lá

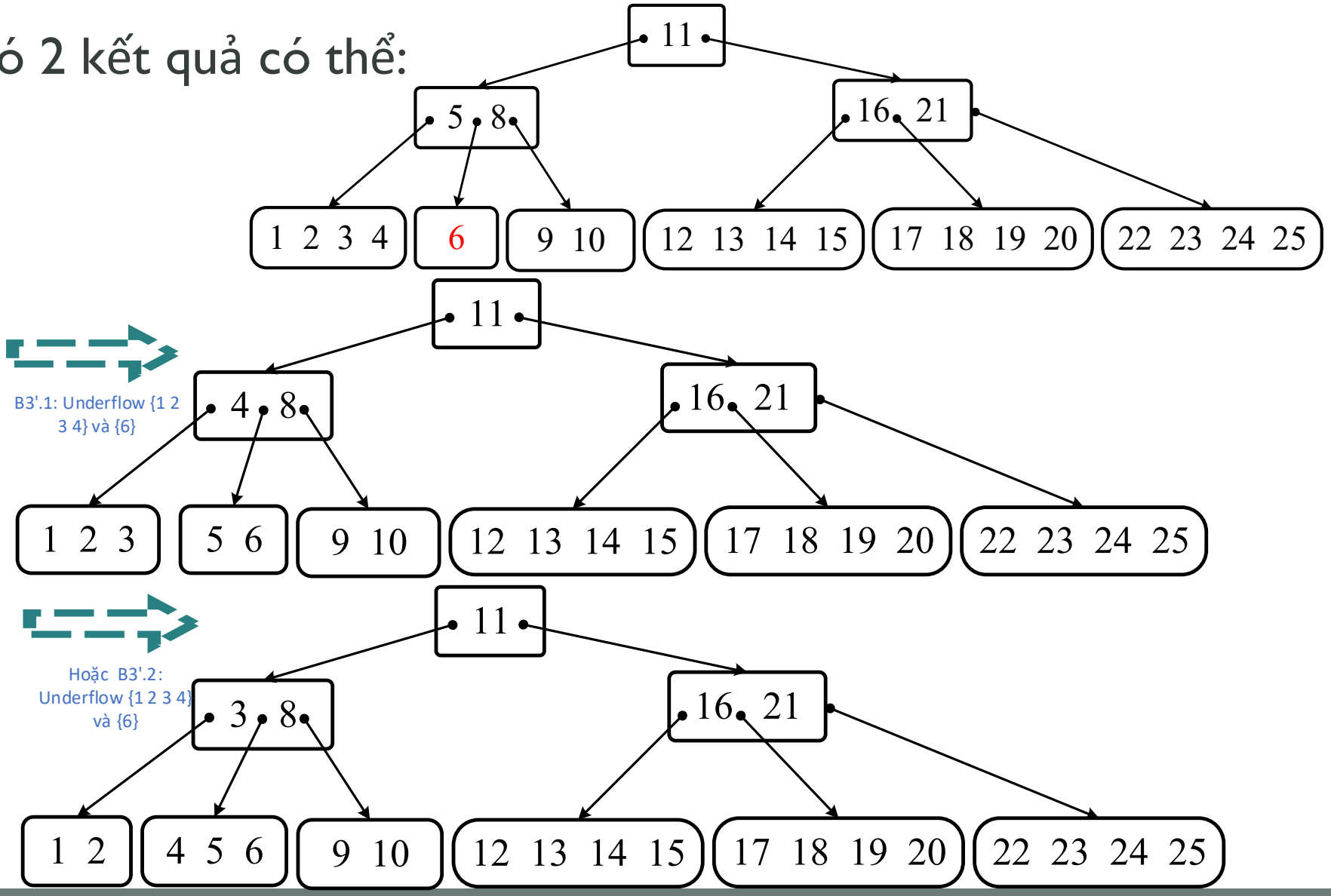
- Trường hợp: Catenate {6} và {9 10}



B-TREE DELETION - Xóa ở node lá

- Trường hợp: Underflow {1 2 3 4} và {6}

=> Có 2 kết quả có thể:



Chúc các em học tốt!