CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

B-Tree

Giới thiệu

- Do R.Bayer và E.M.McCreight đưa ra năm 1972.
- B-tree là cấu trúc dữ liệu phù hợp cho việc lưu trữ và truy xuất dữ liệu trên bộ nhớ ngoài (đĩa cứng)
- Hạn chế số thao tác đọc mỗi lần tìm kiếm trên cây:
  - Mỗi lần truy xuất đọc toàn bộ dữ liệu trong 1 node (mỗi node chứa M phần tử (items))
  - Sử dụng thuật toán tìm nhị phân để tìm phần tử x (giá trị cần tìm).
- Chiều cao cây $log_M N$, tăng M chiều cao cây giảm rất nhanh

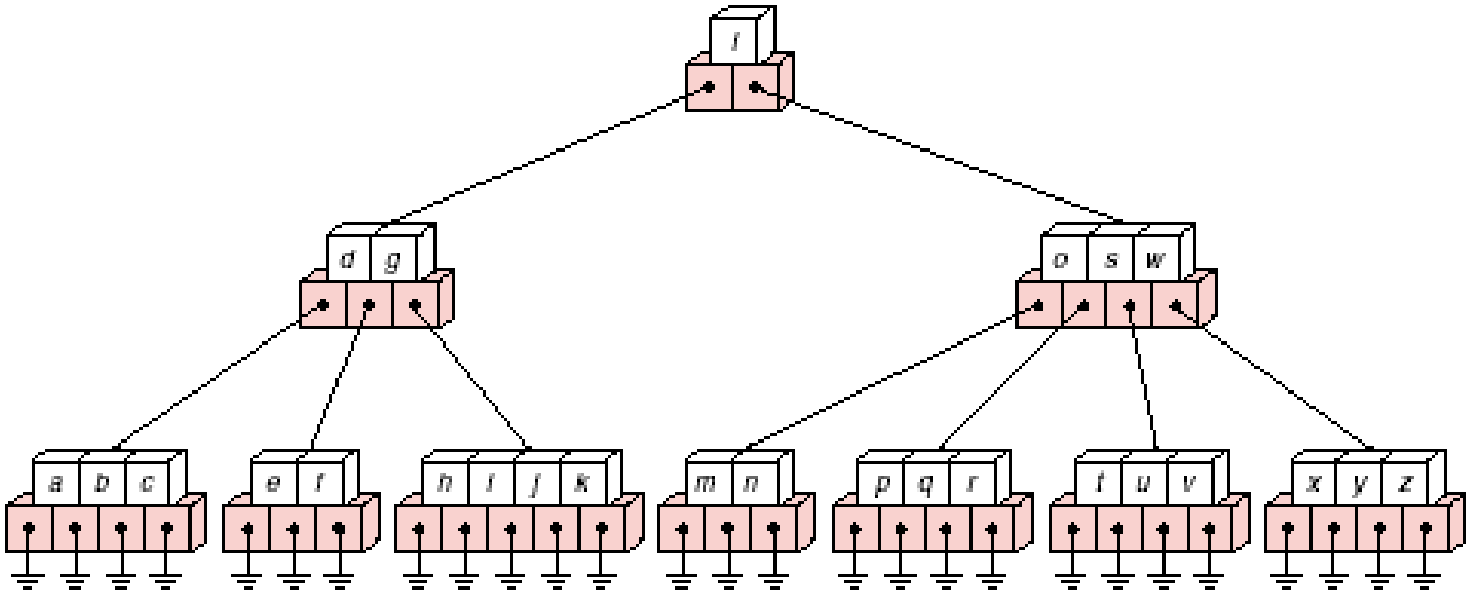
Định nghĩa

Một B-tree bậc m là cây m nhánh tìm kiếm thỏa các điều kiện sau:

Cho số tự nhiên *k* > 0, B-Trees bậc m với m = 2*k+1 là một cây thỏa mãn các tính chất:

- Tất cả node lá nằm trên cùng một mức
- Tất cả các node, trừ node gốc và node lá, có ***tối thiểu*** k+1 node con.
- Tất cả các node có ***tối đa*** m con
- Tất cả các node, trừ node gốc , có từ k cho đến m - 1 khóa (keys). Node gốc có từ 1 đến m-1 khóa.
- Một node không phải lá và có n khóa thì phải có n+1 node con.
- Tại mỗi khóa X bất kỳ, các khóa ở nhánh trái < X < các khóa ở nhánh phải

Định nghĩa



B-Tree bậc 5 có 3 mức

Khai báo cấu trúc

```
typedef struct
{
    int count;                  // số khoá của node hiện hành
    int Key[m-1];    // mảng lưu trữ các  khoá của node
    int *Branch[m];   /* các con trỏ chỉ đến các cây con,
m-Bậc của cây*/
} BNode;                        // Kiểu dữ liệu của node
typedef struct BNode *pBNode;       // con trỏ node
pBNode *Root;                       // con trỏ node gốc
```

Thêm

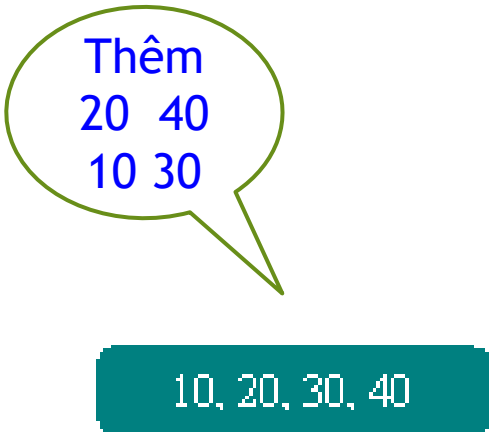
- Khóa mới sẽ được thêm vào node lá
  - Nếu chưa đầy -> thêm hoàn tất
  - Nếu đầy -> tách node:
    - Khóa giữa node được lan truyền ngược lên node cha
      - Trong trường hợp đặc biệt lan truyền đến tận gốc của B-Tree
    - Phần còn lại chia thành 2 node cạnh nhau trong cùng 1 mức

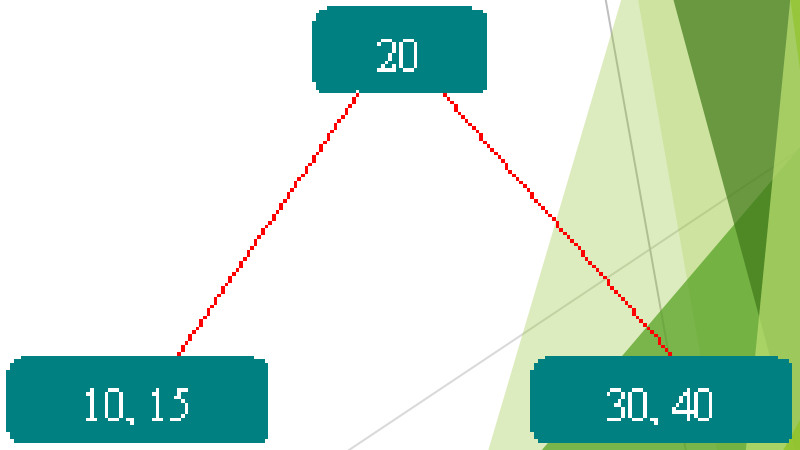
Thêm

Tạo B-Tree bậc 5 từ dãy các khóa sau: **20 40 10 30** 15 35 7 26 18 22 5 42 13 46 27 8 32 38 24 45 25

Thêm 15 -> đầy trang -> tách

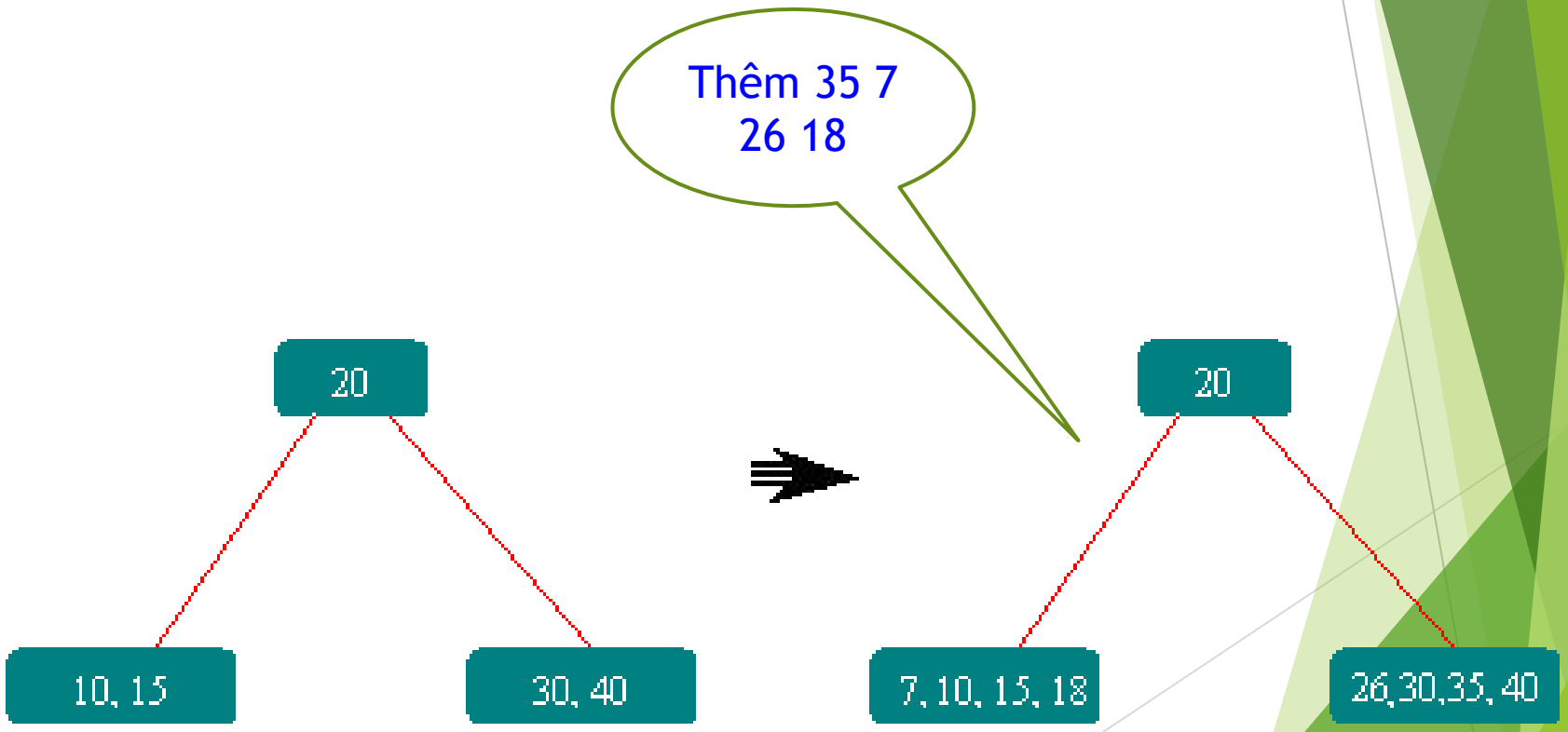
- Khóa giữa 20 đưa lên node cha
- 10, 15 là node con trái của 20
- 30, 40 là node con phải của 20





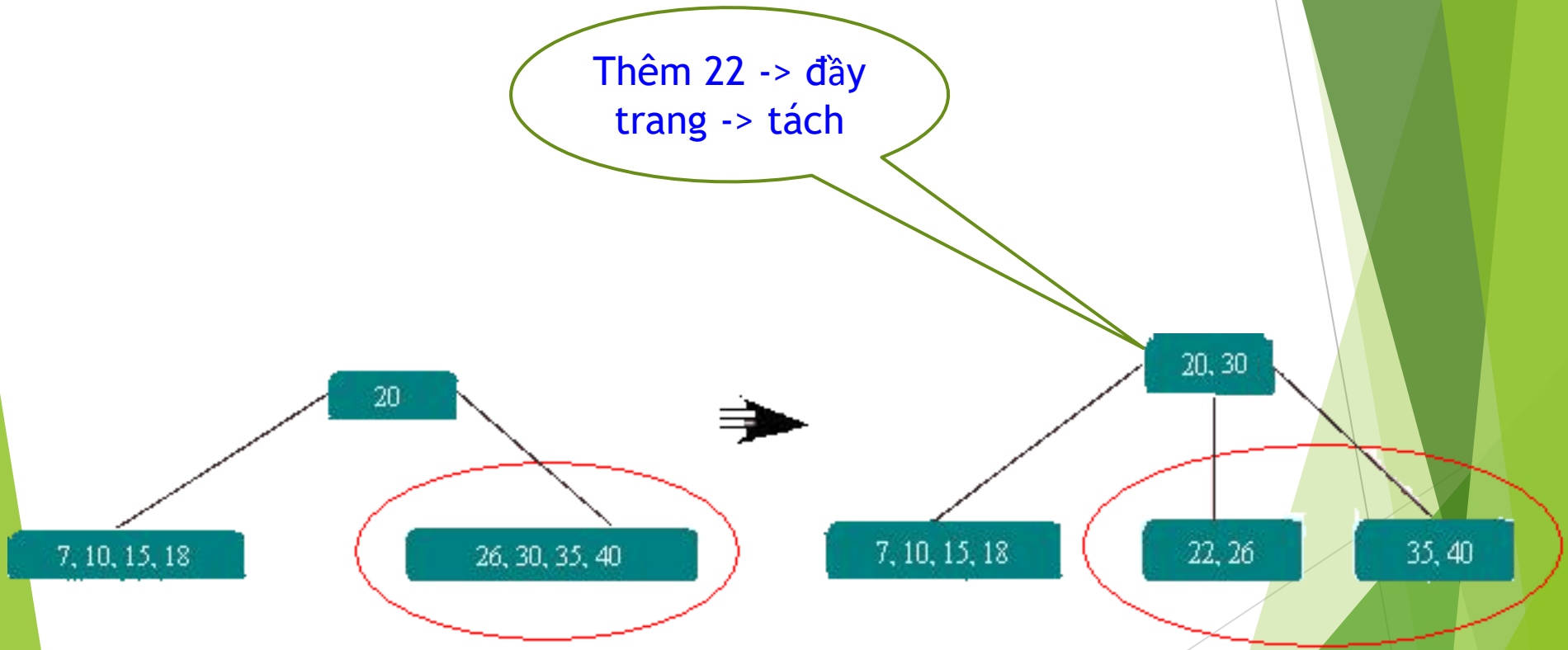
Thêm

- Tạo B-Tree bậc 5 từ dãy các khóa sau : 20 40 10 30 15 **35 7 26 18** 22 5 42 13 46 27 8 32 38 24 45 25



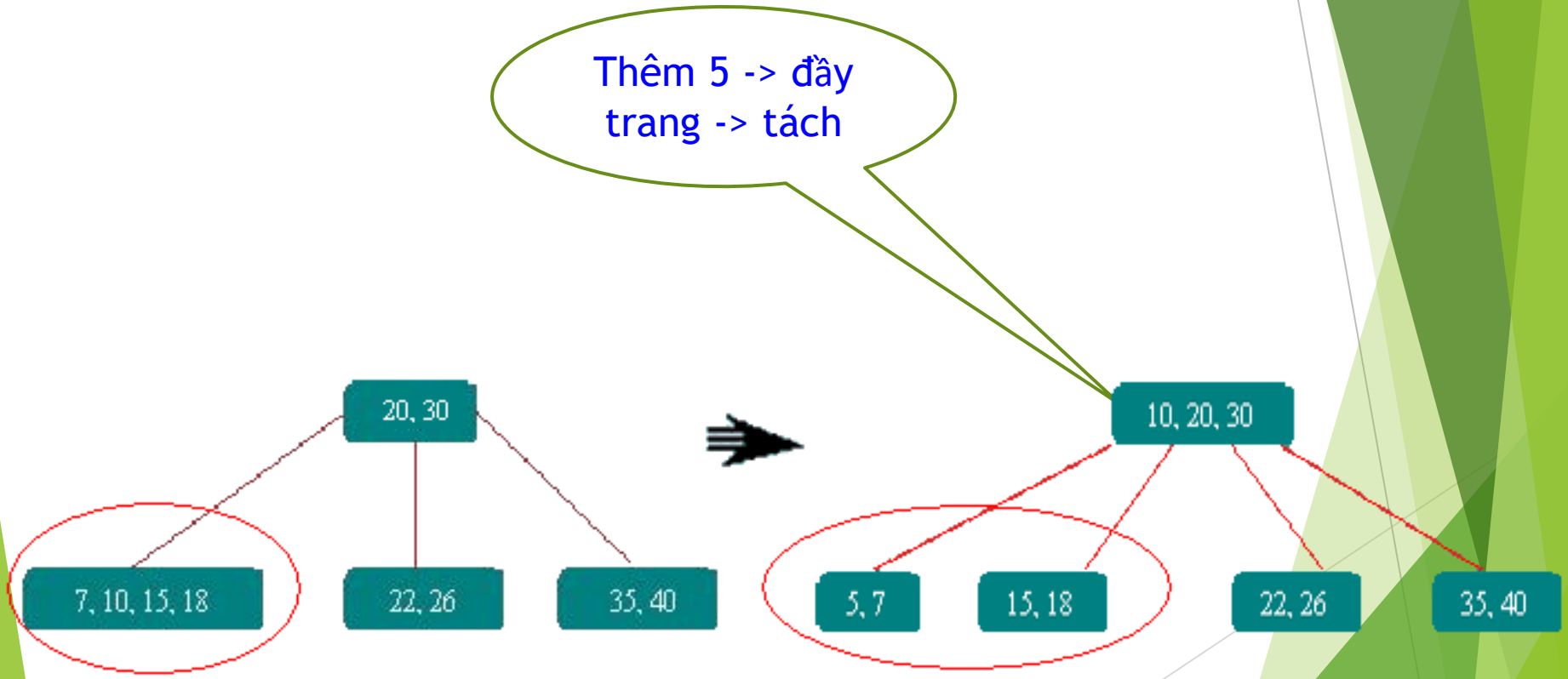
Thêm

- Tạo B-Tree bậc 5 từ dãy các khóa sau : 20 40 10 30 15 35 7 26 18 **22** 5 42 13 46 27 8 32 38 24 45 25



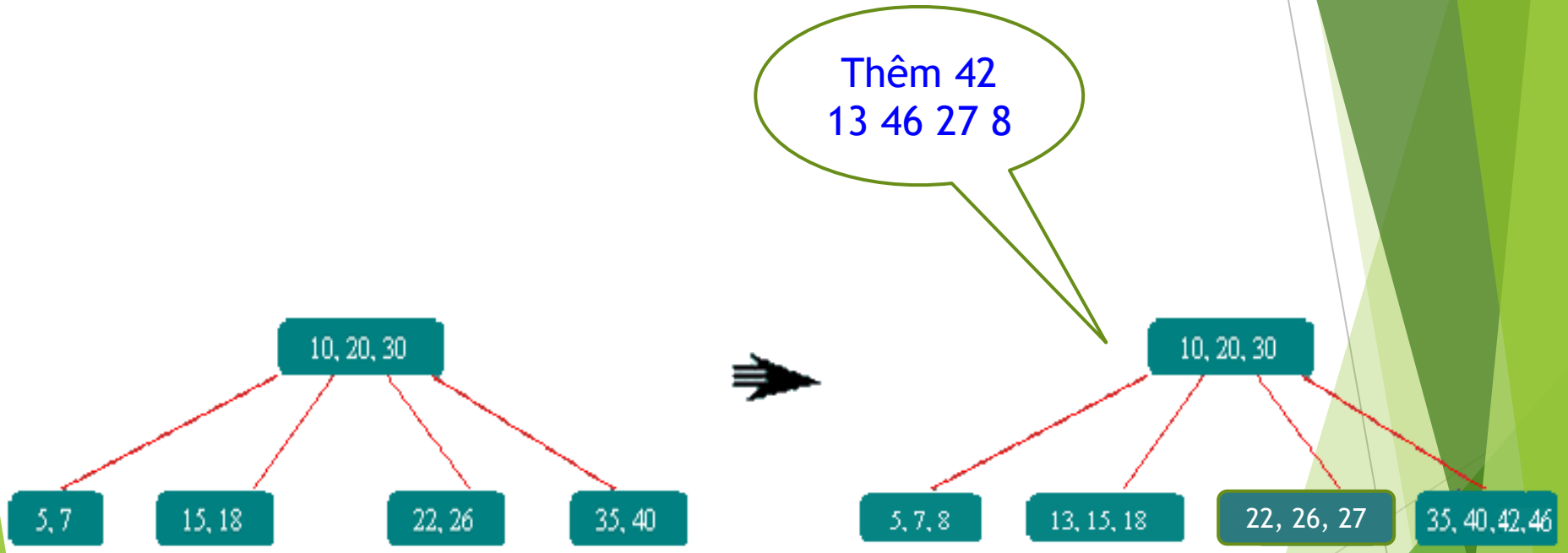
Thêm

- Tạo B-Tree bậc 5 từ dãy các khóa sau : 20 40 10 30 15 35 7 26 18 22 **5** 42 13 46 27 8 32 38 24 45 25



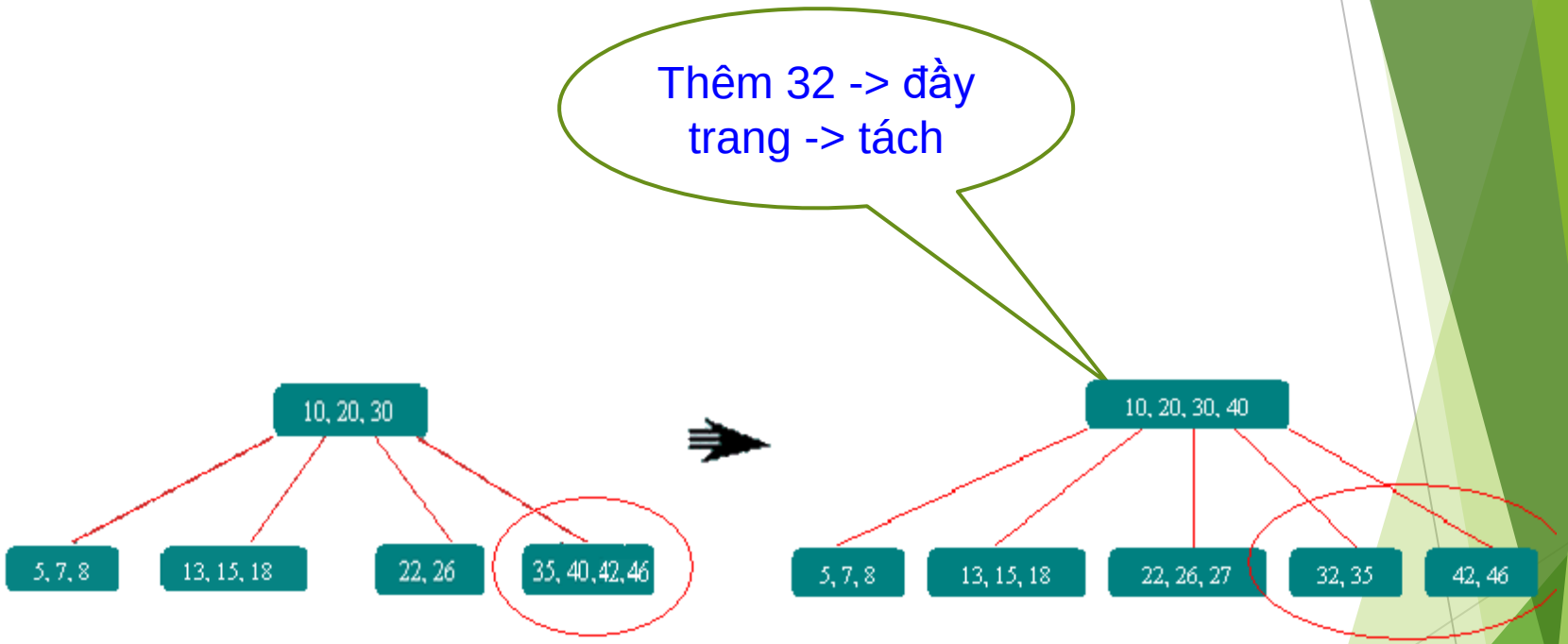
Thêm

- Tạo B-Tree bậc 5 từ dãy các khóa sau : 20 40 10 30 15 35 7 26 18 22 5 **42 13 46 27 8** 32 38 24 45 25



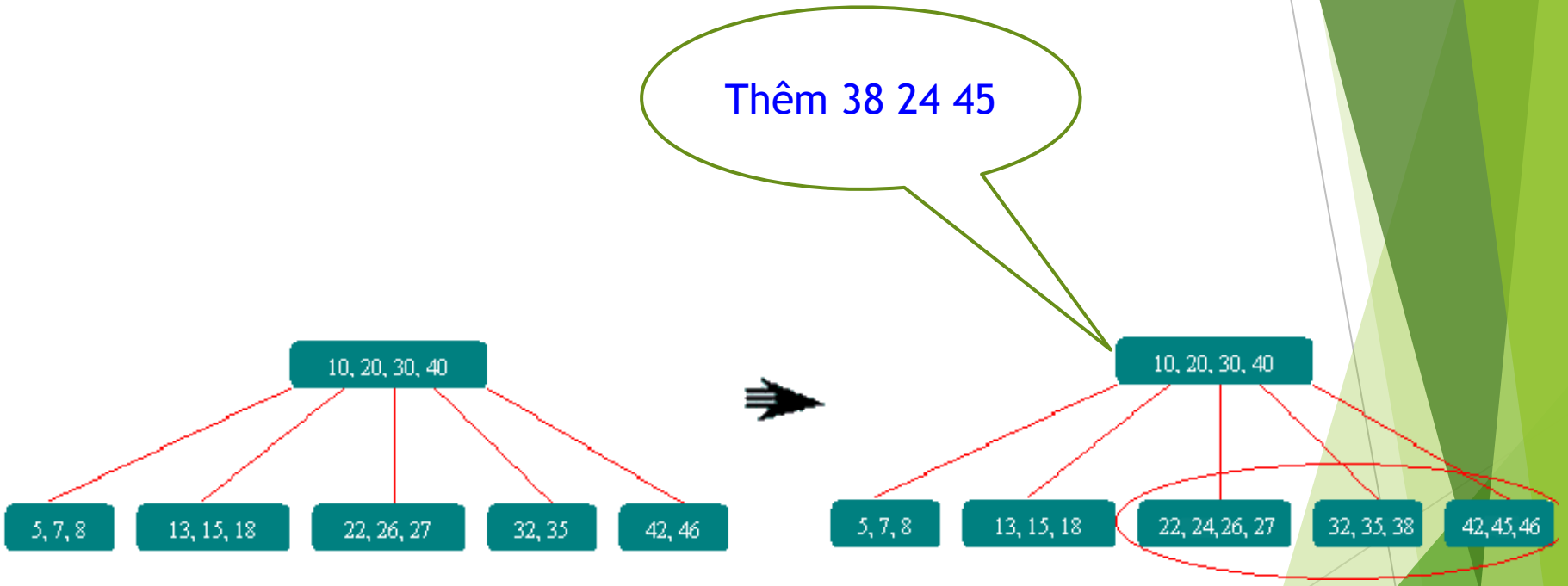
Thêm

- Tạo B-Tree bậc 5 từ dãy các khóa sau : 20 40 10 30 15 35 7 26 18 22 5 42 13 46 27 8 **32** 38 24 45 25



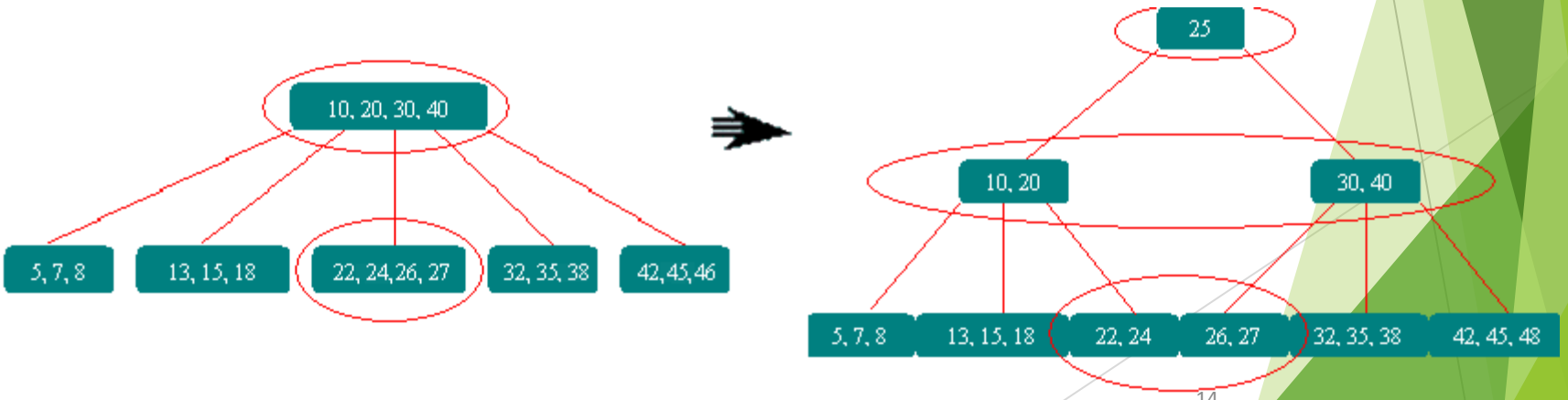
Thêm

- Tạo B-Tree bậc 5 từ dãy các khóa sau : 20 40 10 30 15 35 7 26 18 22 5 42 13 46 27 8 32 **38 24 45** 25



Thêm

- Tạo B-Tree bậc 5 từ dãy các khóa sau : 20 40 10 30 15 35 7 26 18 22 5 42 13 46 27 8 32 38 24 45 **25**
  - Thêm 25 vào node (22, 24 26, 27) làm node này bị đầy -> tách và 25 được đưa lên node cha (10, 20, 30, 40)
  - 25 được đưa lên node cha (10, 20, 30, 40) làm node này bị đầy -> tách thành 2 node và khoá giữa 25 được đưa lên thành cha (node gốc mới

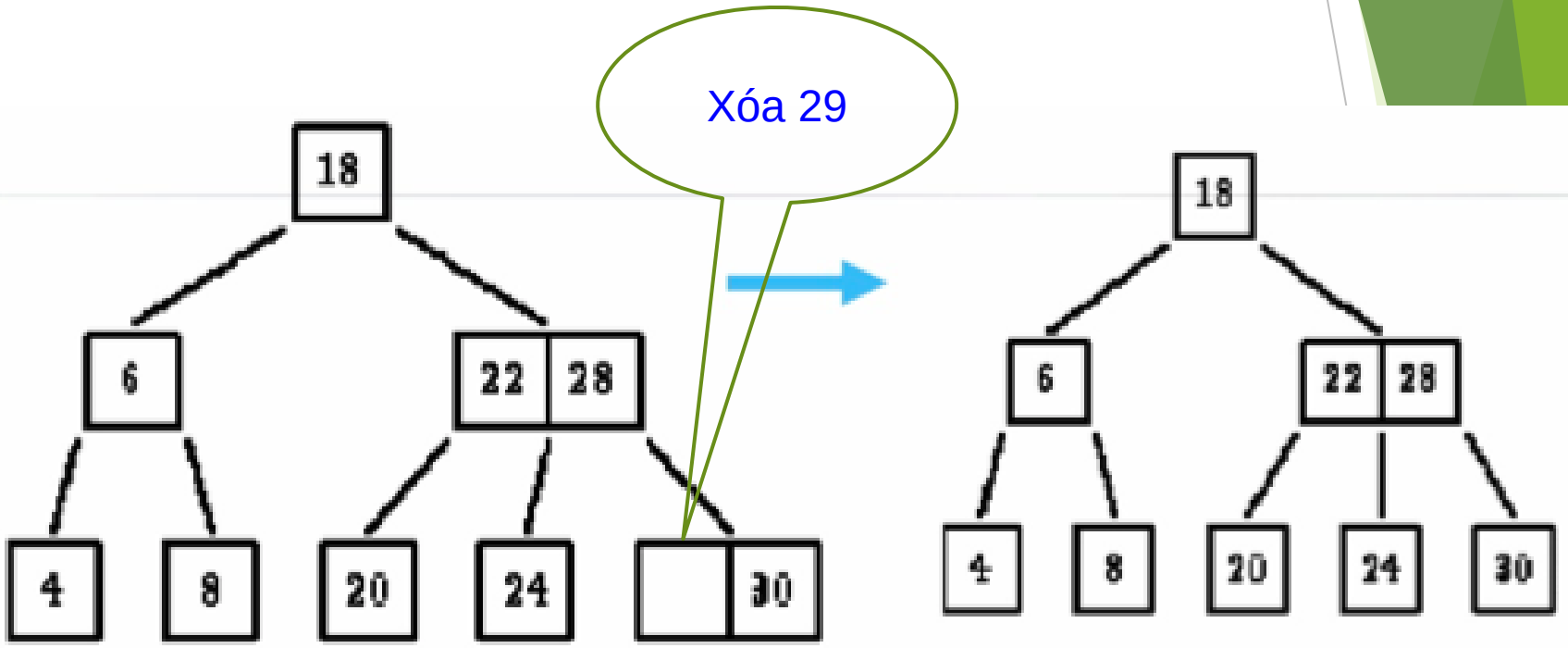


Xóa

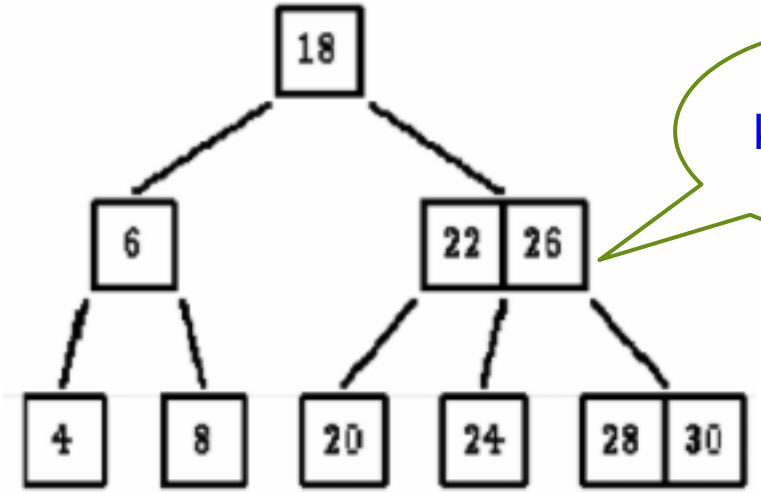
- Khóa cần xóa nằm trên node lá -> Xóa bình thường
- Khóa cần xóa không trên node lá:
  - Tìm phần tử thay thế: trái nhất ở cây con bên phải hoặc phải nhất trên cây con bên trái.

Xoá

- Xoá 1 khoá trên trang lá (B-Tree bậc 3)

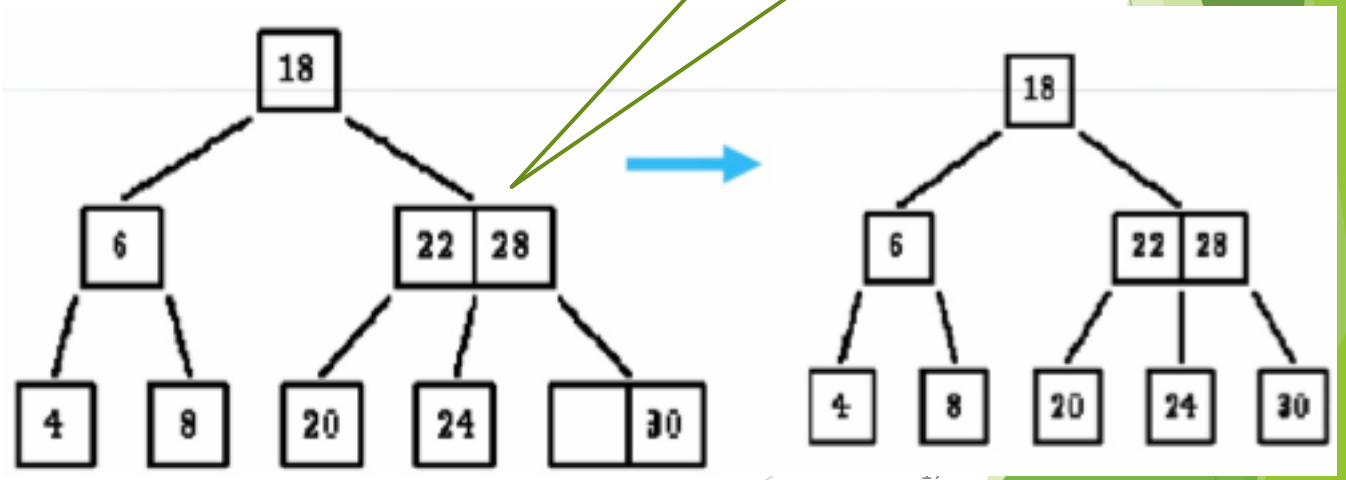


Xoá



Xóa 26, không trên trang lá

Phần tử thay thế 28: trái nhất trên cây con phải

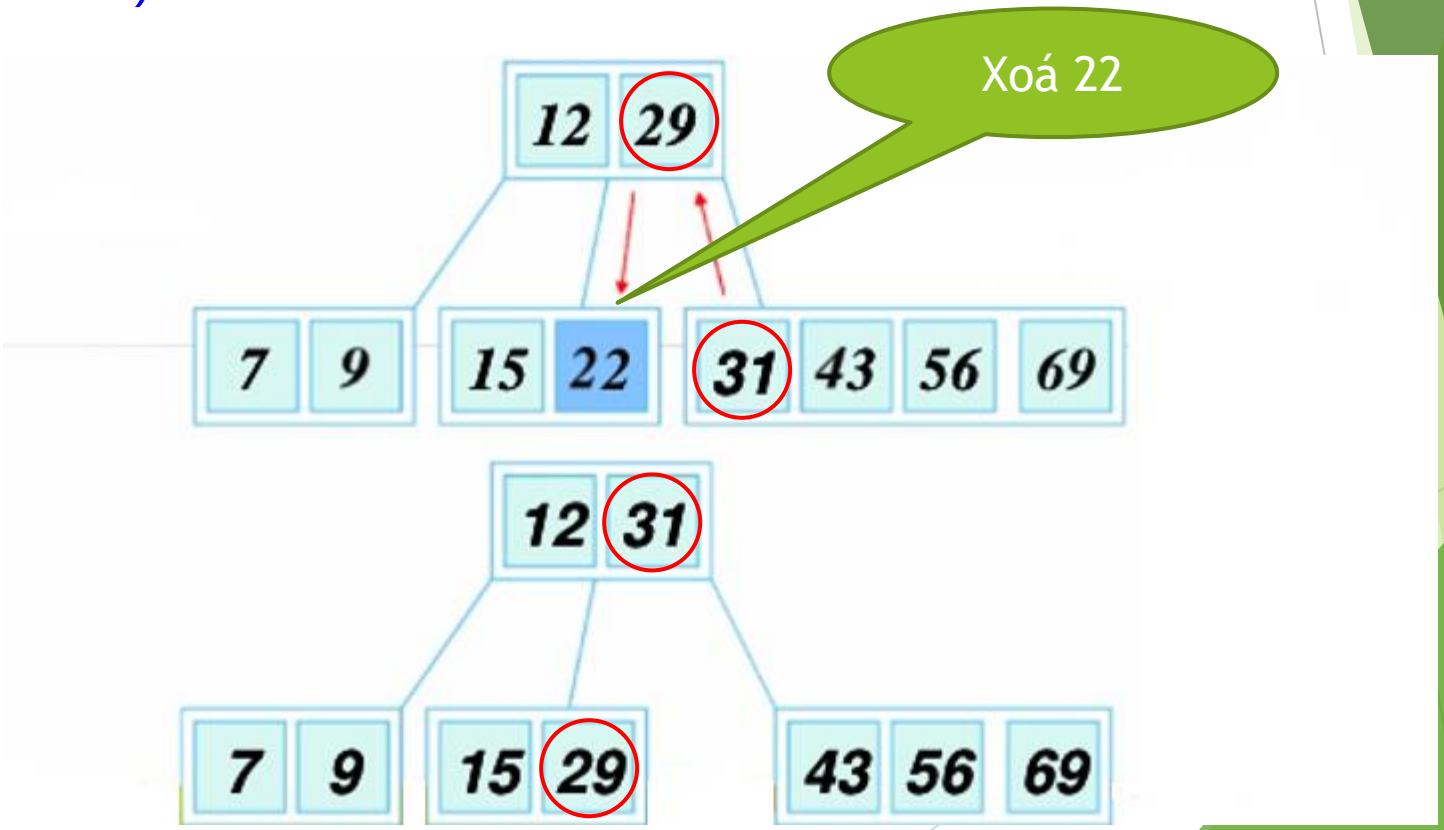


Cân bằng cây sau khi xoá

- Sau khi xóa, node bị thiếu (vi phạm điều kiện B-tree):
  - Hoặc chuyển dời phần tử từ node thừa
  - Hoặc ghép với node bên cạnh (trái/phải)

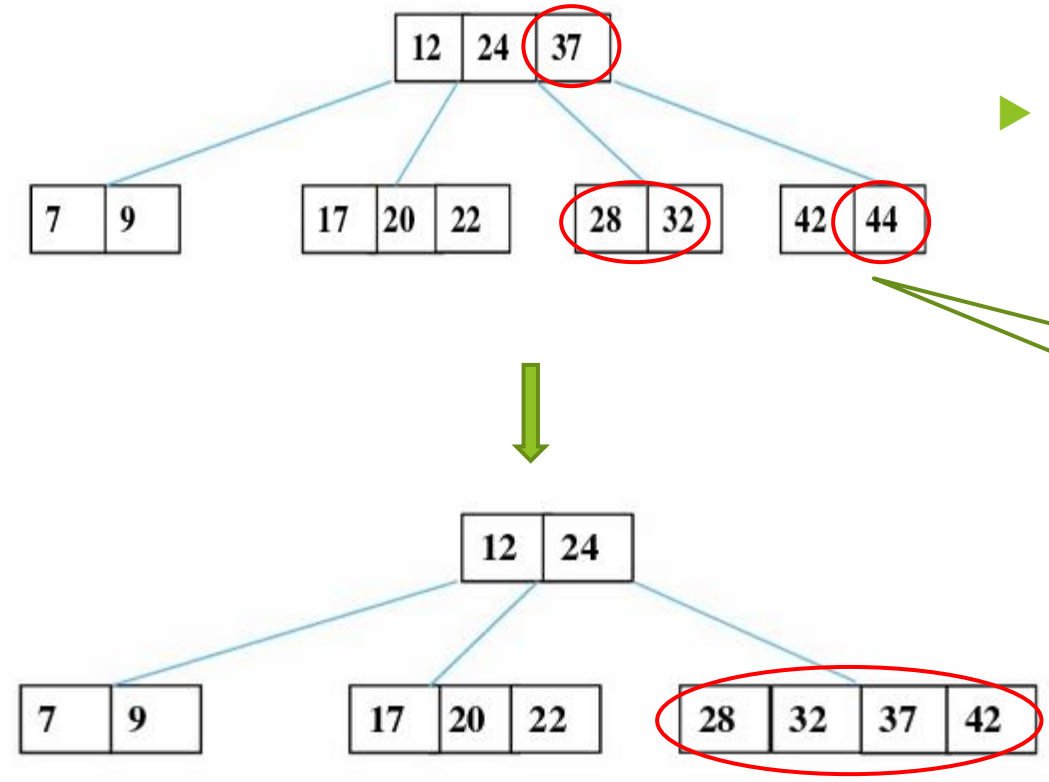
Cân bằng cây sau khi xoá

- Nếu một trong các **nút kế cận** nút đang xét (nút chứa khóa cần xóa) có số lượng khoá nhiều hơn số lượng tối thiểu:
  - Đưa 1 khoá của **nút kế cận** lên nút cha
  - Đưa 1 khoá ở nút cha xuống nút đang xét (thay thế khoá cần xoá)



Cân bằng cây sau khi xoá

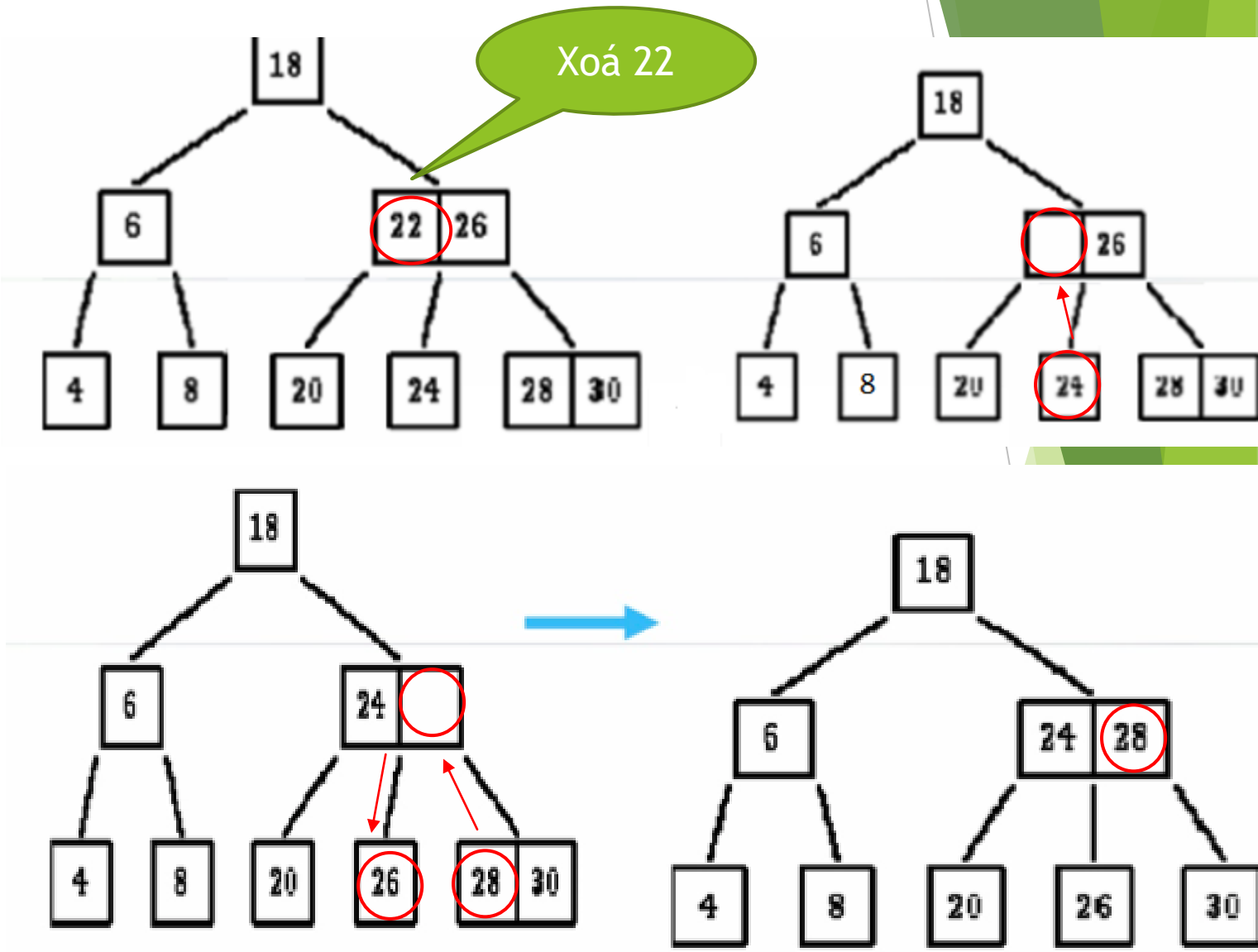
- Tất cả nút kế cận nút đang xét có số lượng khoá vừa đủ:
  - Chọn 1 nút kế cận để hợp nhất với nút đang xét và khoá tương ứng ở nút cha.
  - Nếu nút cha thiếu khoá, lặp lại quá trình này.



Xoá 44: hợp nhất 42, 37, 28 32

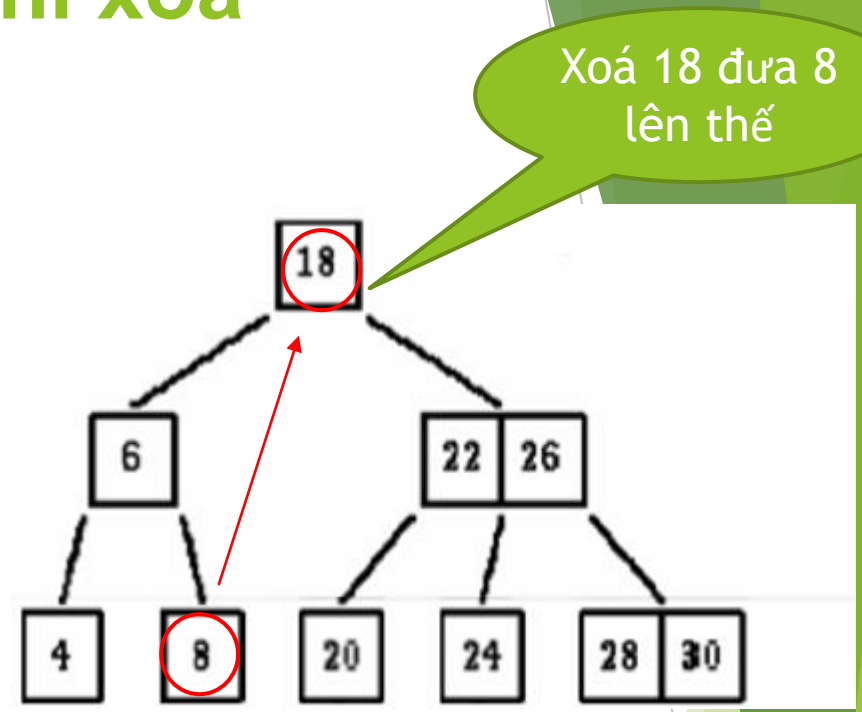
Cân bằng cây sau khi xoá

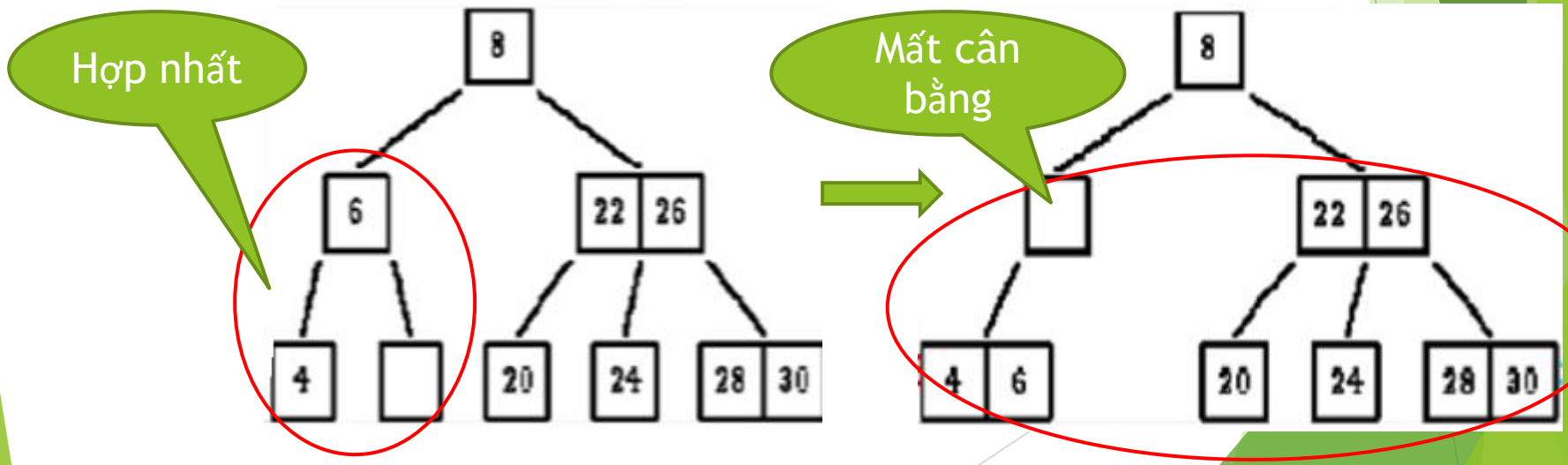
- Xoá 22 (b-tree bậc 3):
  - Nút thay thế là 24: trái nhất của nhánh phải
  - → thiếu lá.
  - Đưa 1 khoá của nút kế cận lên nút cha
  - Đưa 1 khoá ở nút cha xuống nút thiếu



Cân bằng cây sau khi xoá

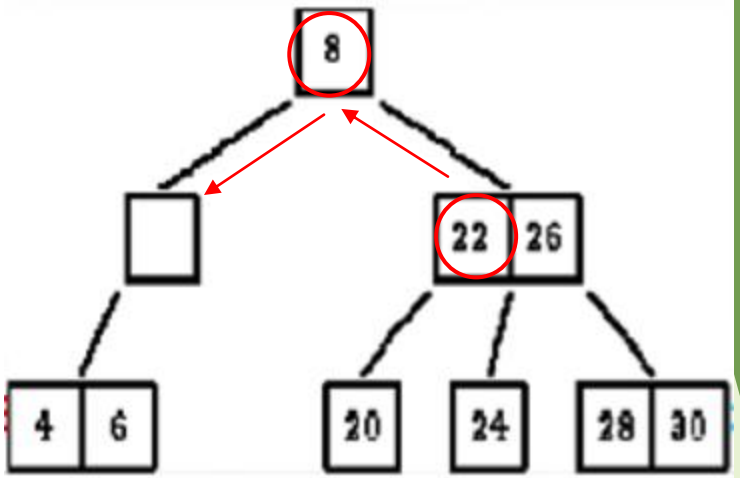
- Xoá 18:
  - **Đưa 8 lên thay** (phải nhất của cây con trái) -> thiếu lá
  - Các nút có số lượng khóa vừa đủ: hợp nhất nút kế cận (4), nút đang xét (null) và khoá tương ứng ở nút cha (6) → cây mất cân bằng

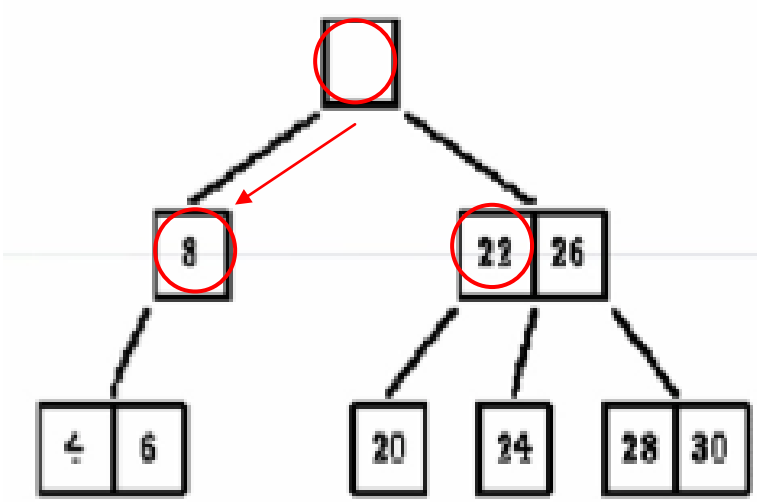


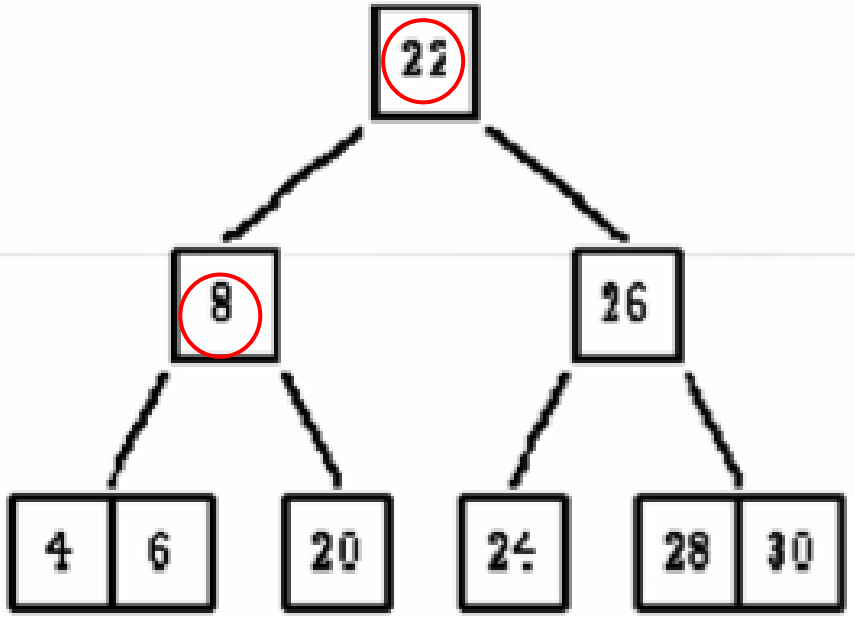


Cân bằng cây sau khi xoá

- Xoá 18 (tt)
  - Đưa 1 khoá của nút kế cân (22 26) lên nút cha
  - Đưa 1 khoá ở nút cha xuống nút thiếu

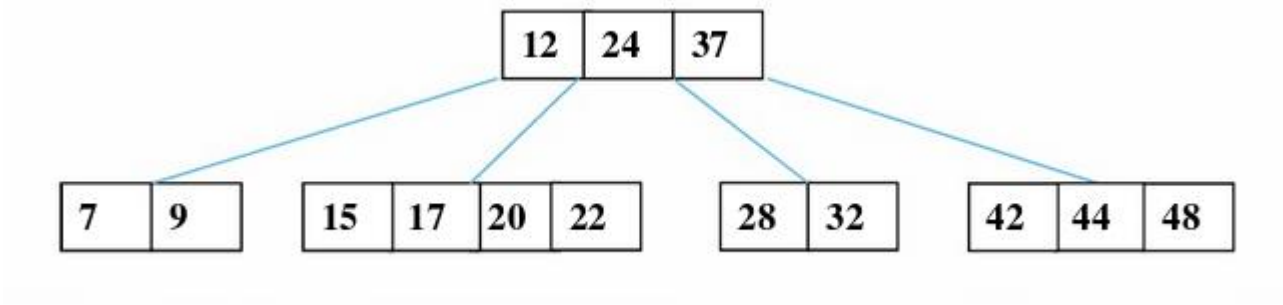




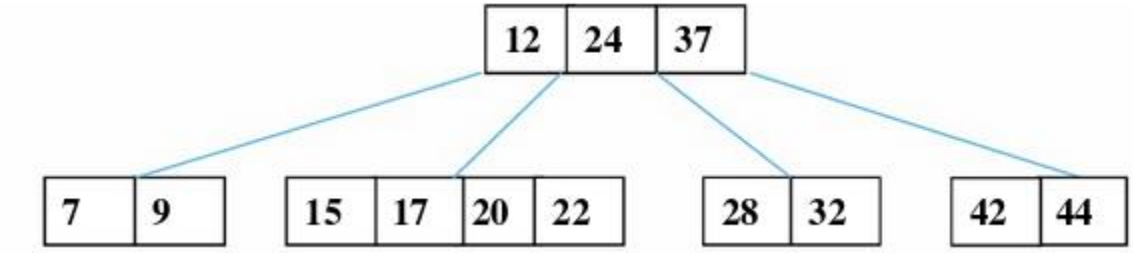


Xóa

- **B-tree bậc 5:**

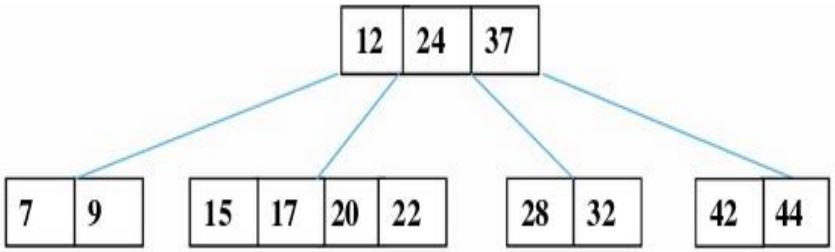


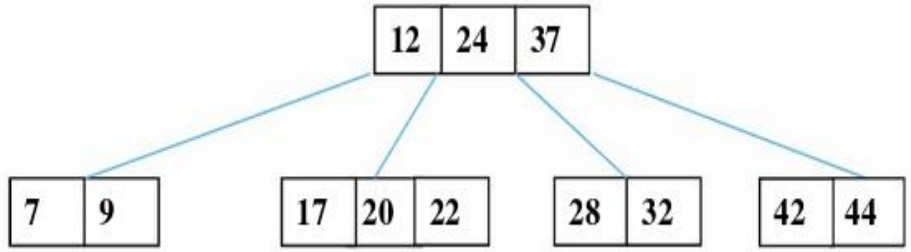
- **Xóa 48: trên trang lá**



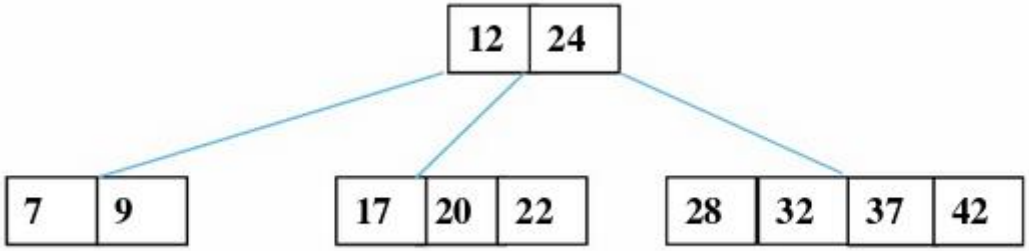
Xóa

- **Xóa 15: trên trang lá**



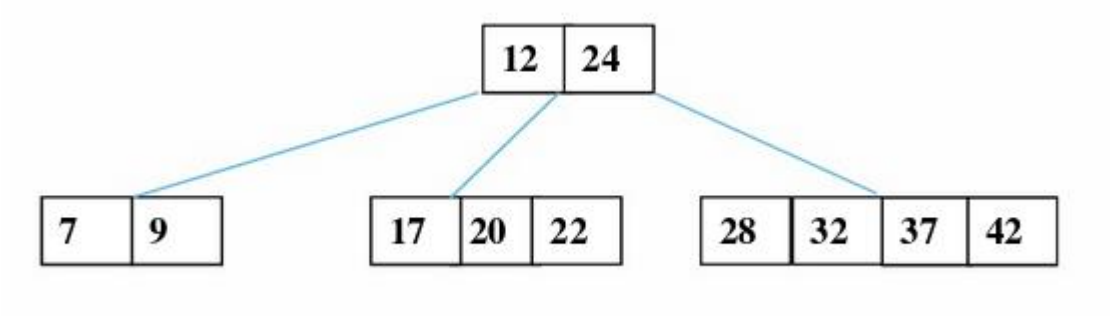


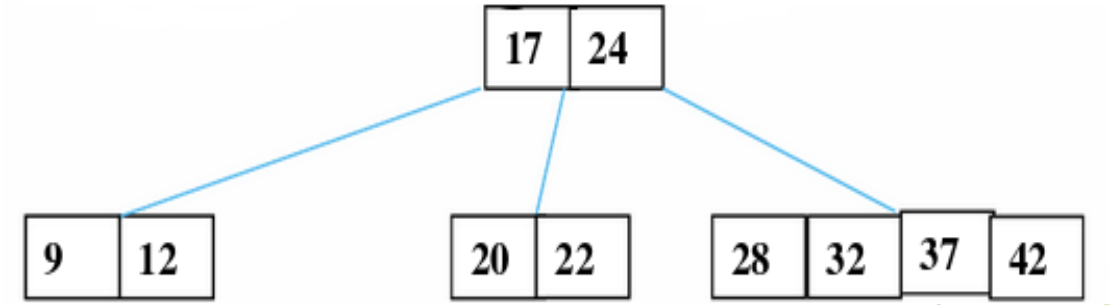
- **Xóa 44: gộp trang**



Xóa

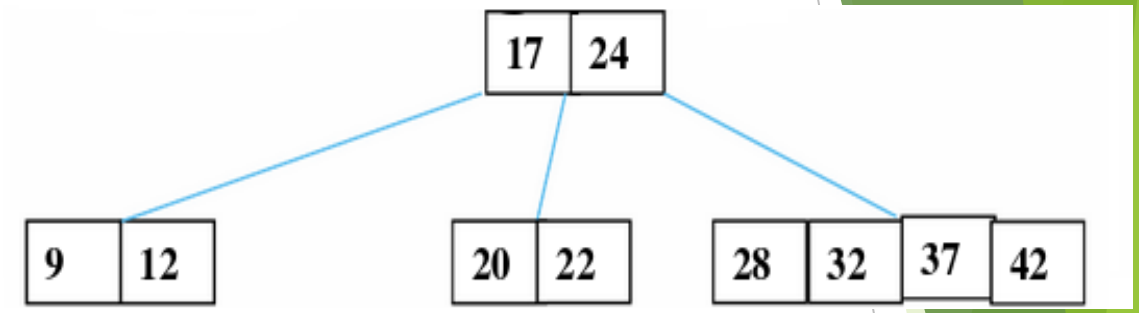
- **Xóa 7: mượn trang phải**

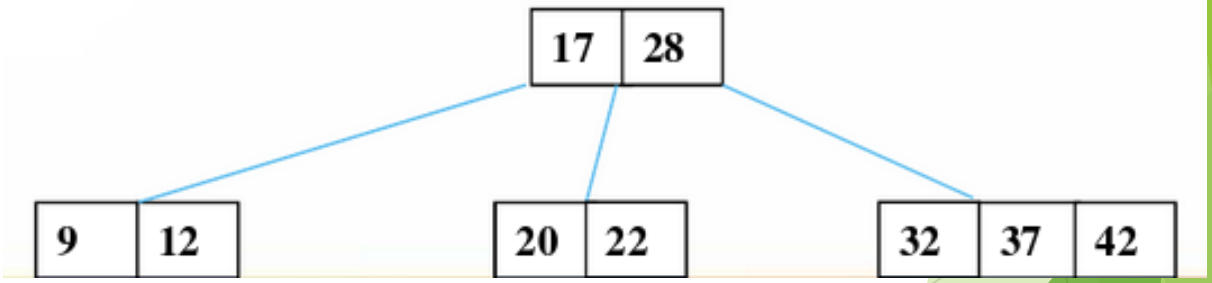




Xoá

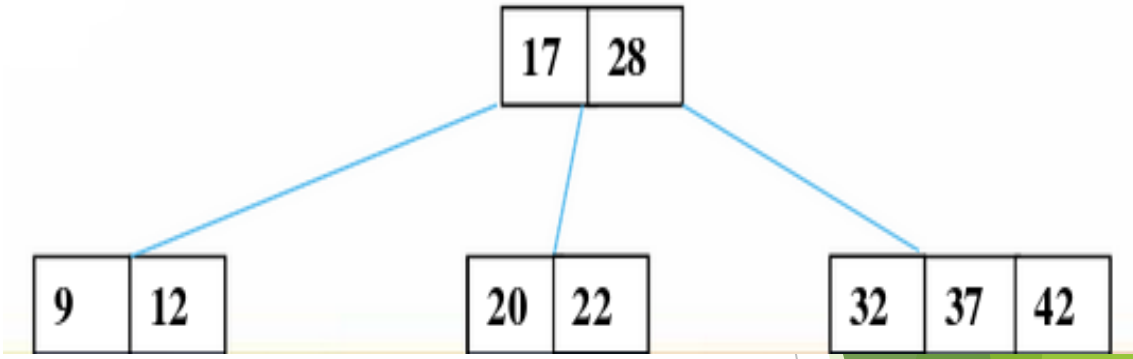
- **Xoá 24:**
  - Nếu đưa 22 lên thế, trang 22, 20 chỉ còn 1 phần tử (không hợp lệ)
  - → Đưa 28 lên thế

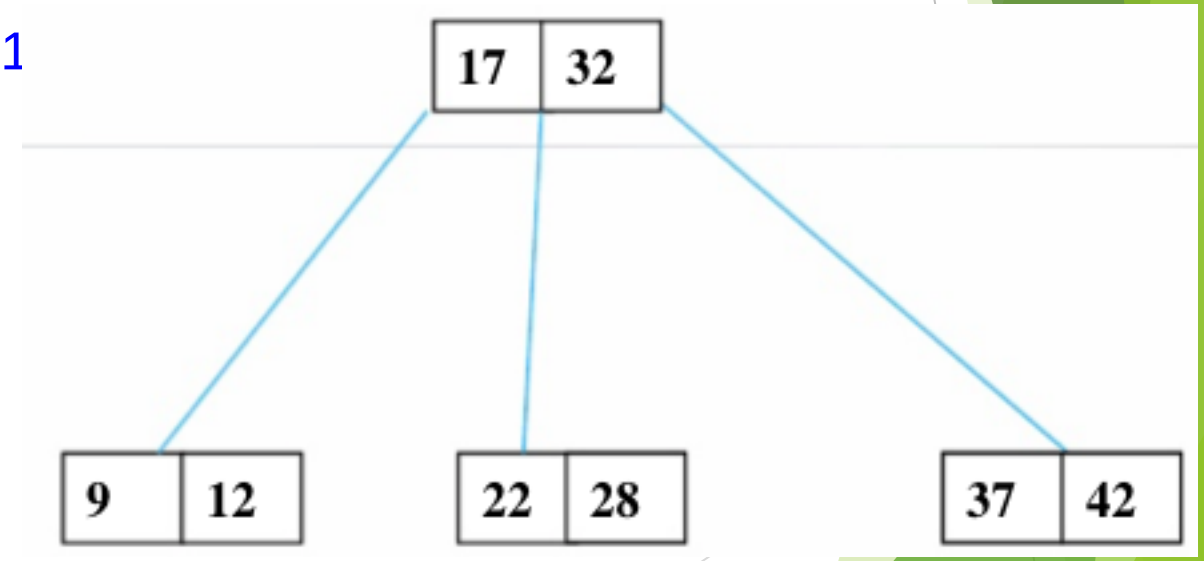




Xoá

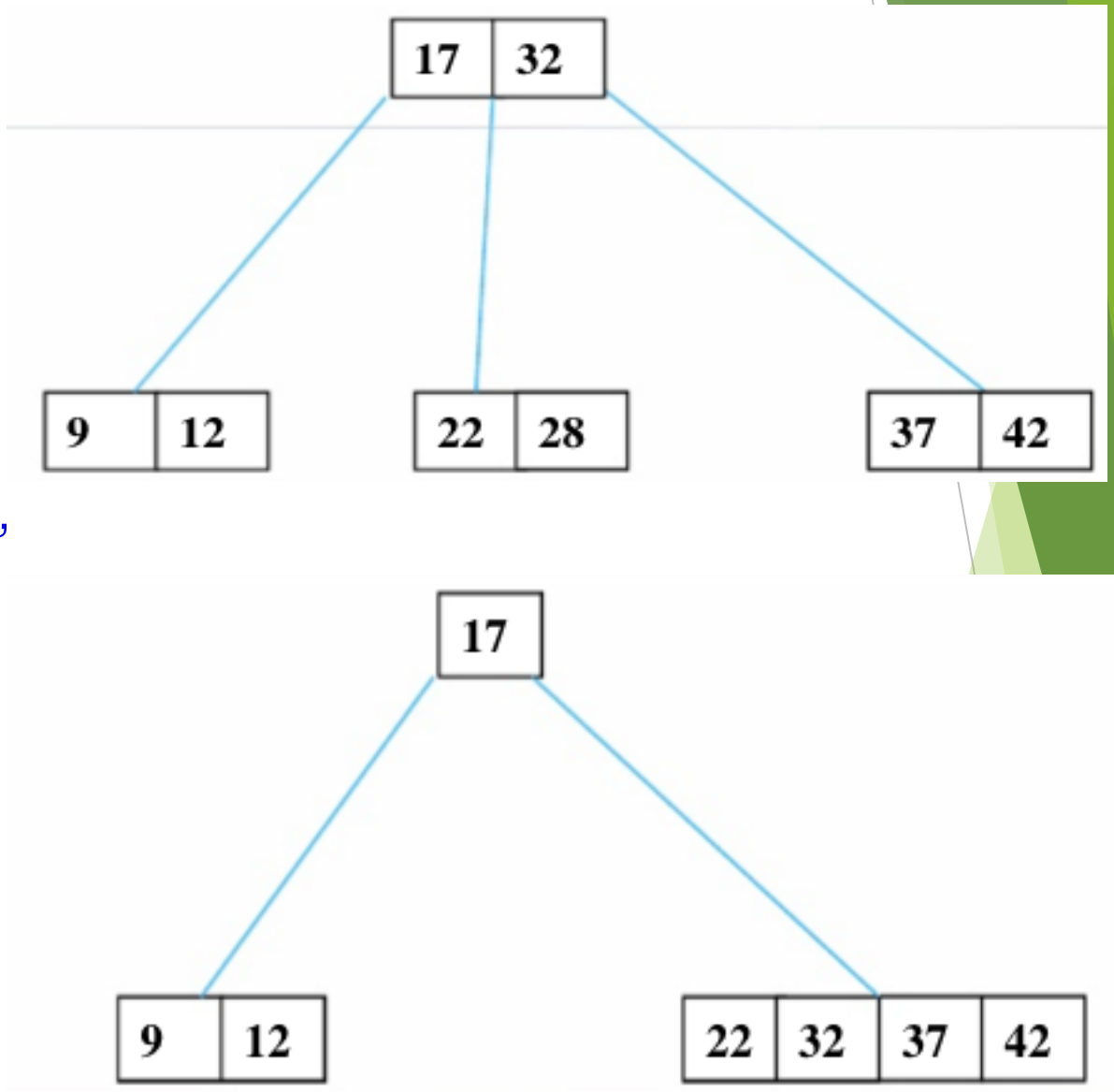
- Xoá 20:
  - Trang 22 còn 1 phần tử là không hợp lệ
  - Mượn trang phải 1 phần tử. Tức là mang 32 lên cha, đưa 28 xuống ghép với 22.





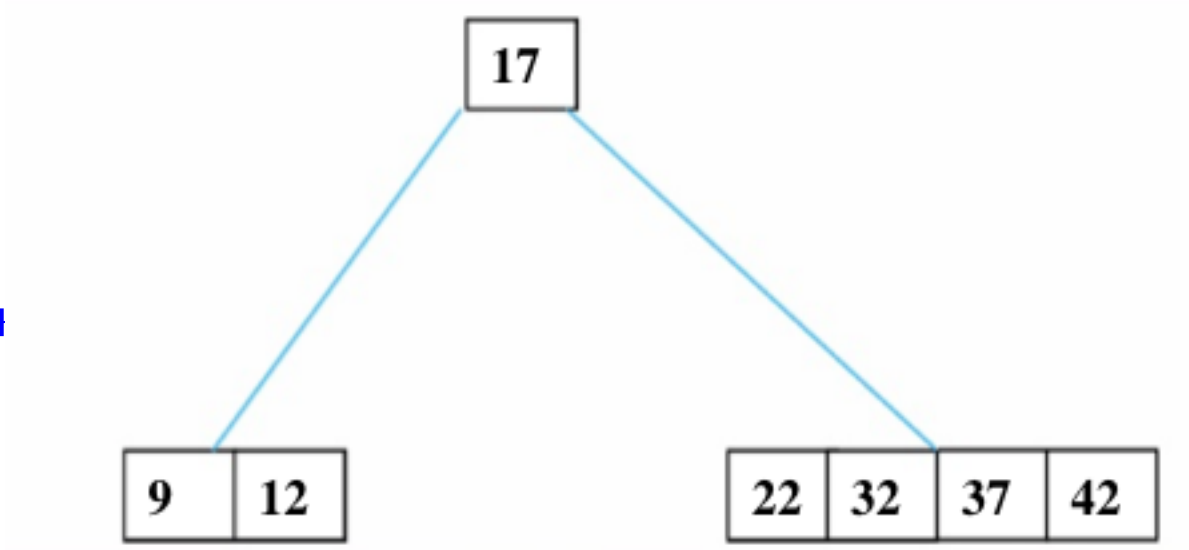
Xoá

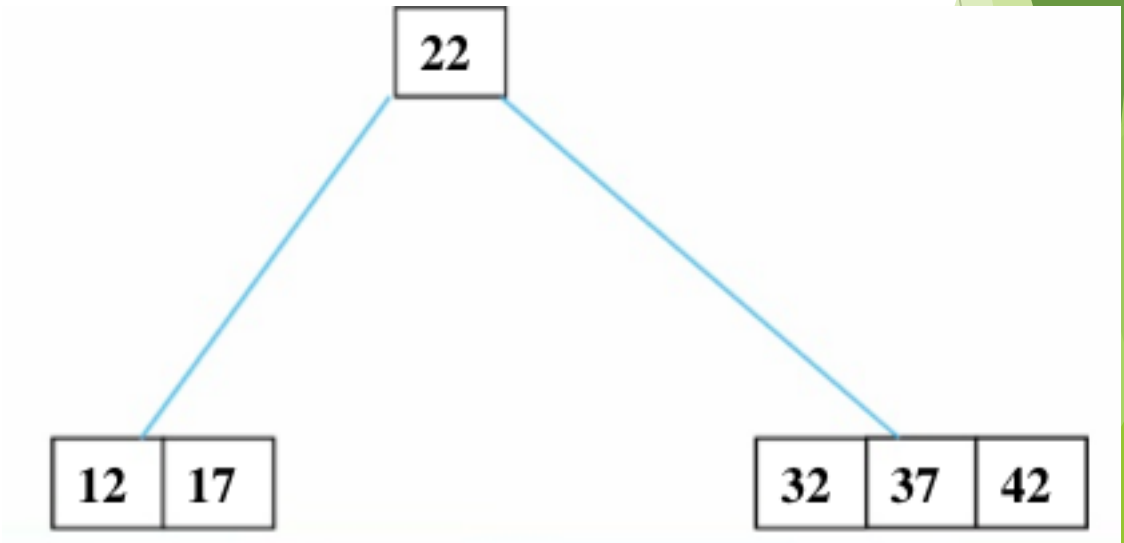
- **Xoá 28:**
- Trang 22 còn 1 phần tử là không hợp lệ.
- Trang bên phải cũng có số phần tử vừa đủ → không thể mượn.
- Do đó, phải gộp trang.



Xoá

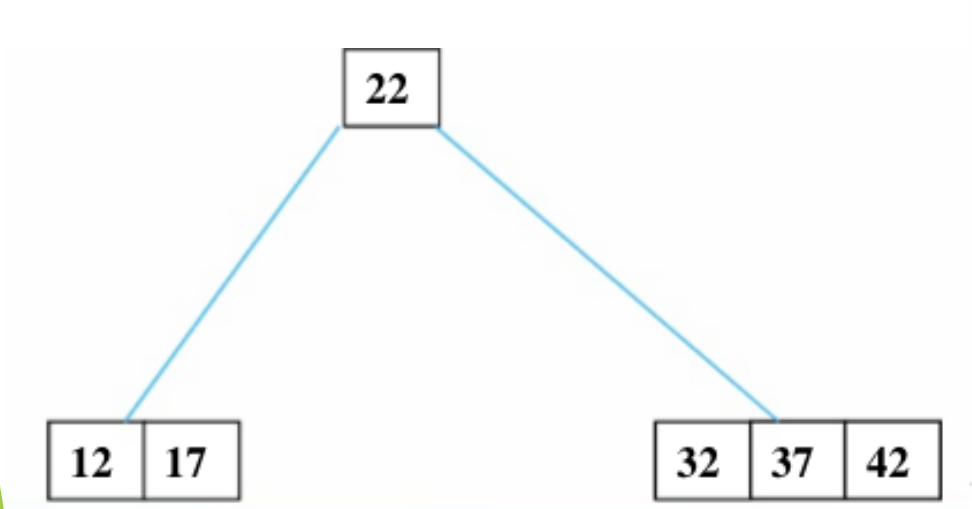
- **Xoá 9:**
- Mượn phần tử của trang bên phải.

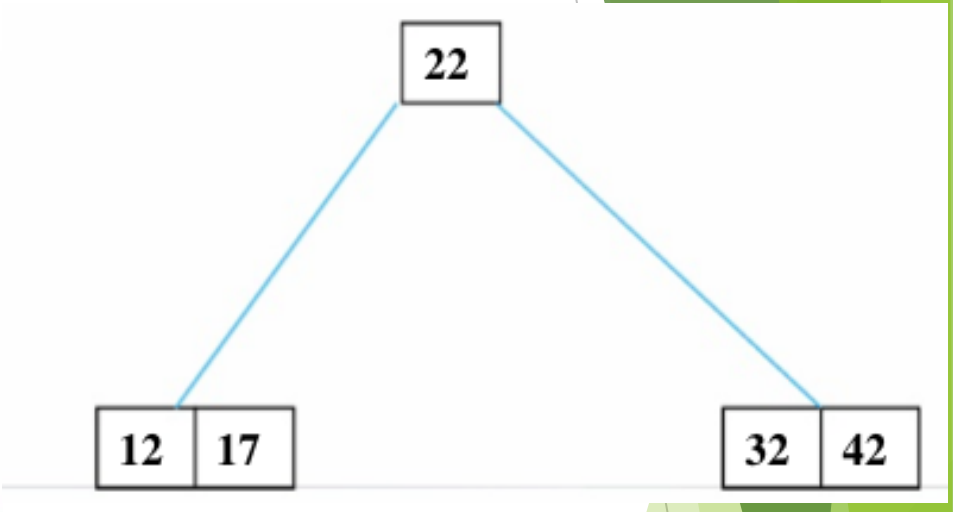




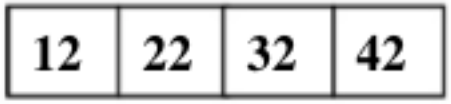
Xoá

- **Xóa 37:**





- **Xoá 17:** gôp trang.



Ứng dụng của B-Tree

- B-Tree có nhiều biến thể và cải tiến
  - B^+-Tree
  - B^*-Tree
- Quản lý dữ liệu trên đĩa cứng, dữ liệu lớn

Ứng dụng của B-Tree

- File system - Hệ thống quản lý file trên đĩa cứng
  - Danh sách các block còn trống
  - File x đang nằm ở block số mấy?
- Windows:
  - NTFS, FAT32,...
- MacOS
  - HFS+
- Linux
  - Btrfs, Ext, xFS
- Other:
  - HFS, Reiser4, HAMMER, ...