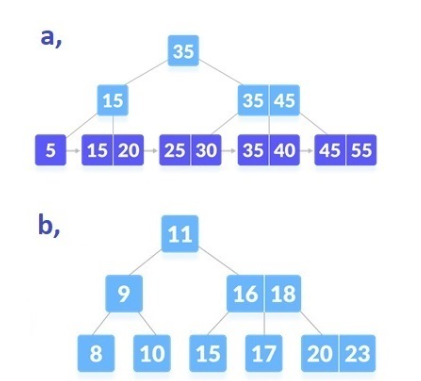
1. Các cây nào sau đây không hợp lệ

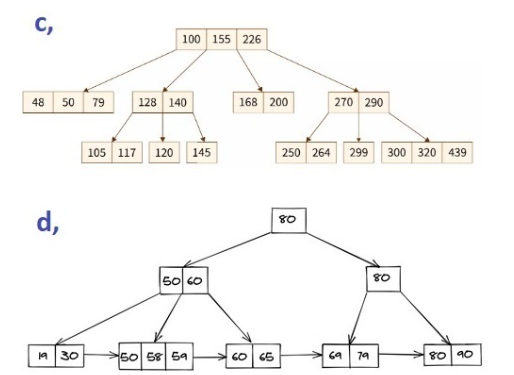
Các cây ở câu a, c, d là không hợp lệ

a: có 1 khóa 35 ở lá nhưng có 2 khóa 35 ở các node trên. Khóa ở node gốc đúng ra là 25 (khóa nhỏ nhất bên phải)

c: khóa 120 bé hơn khóa 128 nhưng lại bên phải khóa 128, các node lá không cùng mức

d: có 1 khóa 80 ở lá nhưng có 2 khóa 80 ở các node trên. Khóa ở node gốc đúng ra là 69 (khóa nhỏ nhất bên phải)



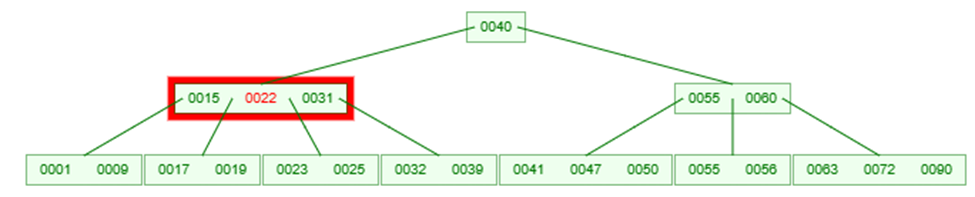


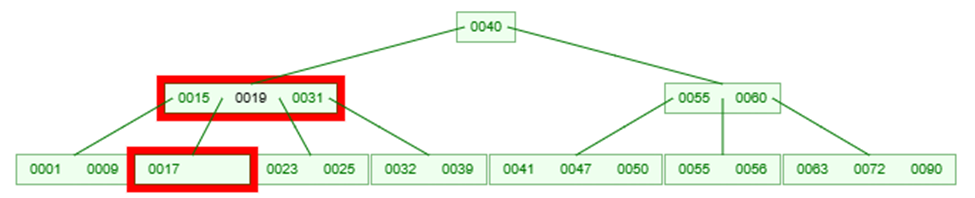
2. Vẽ cây sau khi đã xóa khóa 22

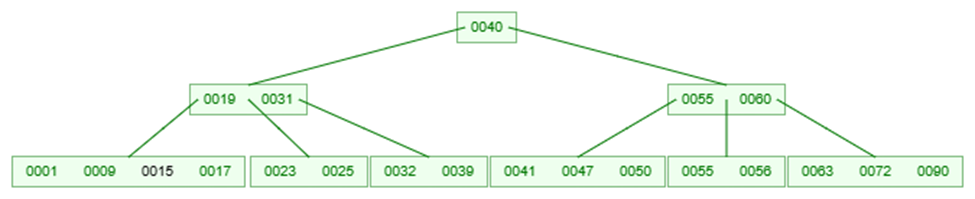
Số khóa tối thiểu (m-1)/2 = 2

Số khóa tối đa m-1 = 5

* B1: Xóa khóa 22. Node hiện tại có 4 node con mà chưa có đủ 4 – 1 = 3 khóa nên bị thiếu khóa (số node con luôn bằng số khóa cộng 1)
* B2: Tìm khóa predecessor (khóa lớn nhất bên trái) và truyền vào vị trí thiếu. Ở đây là khóa 19 của node con bên trái
* B3: Node con bên trái ấy chỉ còn 1 khóa, chuyền khóa 17 và 15 sang
* B4: Node trên thiếu khóa, bên trái là node ngoài cùng nên không tiếp tục lấy predecessor. Gộp 2 node con ở vị trí khóa thiếu là node 1 | 9 và 15 | 17







3. Các khóa gây ra split node:

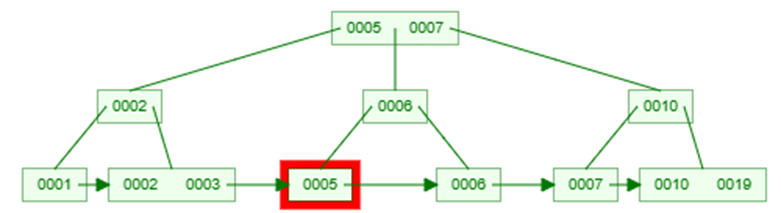
23, 1, 21, 5, 17, 25

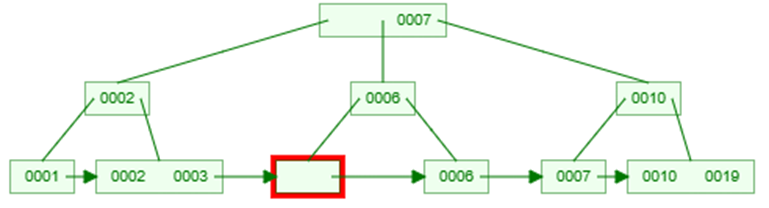
4. Vẽ cây sau khi xóa khóa 5

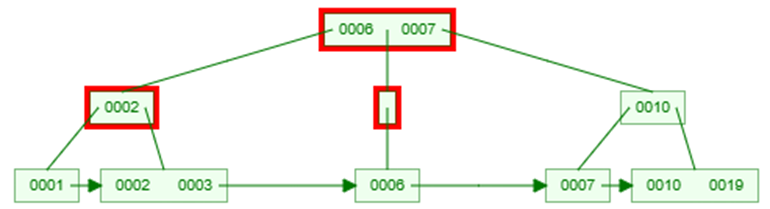
Số khóa tối thiểu (m-1)/2 = 1

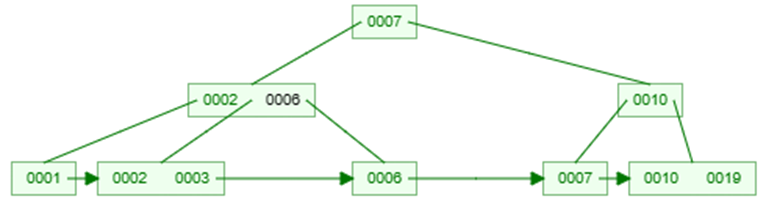
Số khóa tối đa m-1 = 2

* B1: Xóa khóa 5 ở node lá. Xóa khóa 5 ở node trên chứa nó. Node hiện tại có 0 khóa nên bị thiếu khóa
* B2: Kiểm tra node bên cạnh chỉ có node chứa 1 khóa 6, là số khóa tối thiểu nên không thể mượn khóa
* B3: Gộp node thiếu khóa node với node bên cạnh. Điền khóa 6 vào node gốc ở vị trí vừa xóa khóa 5. Node trên giờ chỉ có 1 node con, xử lí như bị thiếu khóa
* B4: Node trên bị thiếu lấy 1 khóa 6 ở node cha
* B5: Node gốc thiếu 1 khóa, không thể lấy khóa predecessor nên gộp 2 node con









5. Giả sử một khóa trong cây B-Tree sử dụng 32 bytes và một pointer đến node con cần sử dụng 4 bytes. Tổng dung lượng yêu cầu tối thiểu của một node trong một cây B-Tree cấp độ m là?

1 node chứa tối đa: (m-1) khóa và m node con

32(m -1) + 4m = 36m - 32 (bytes)

6.B+ tree so với B-Tree thì:

+ Độ bạt (fan-out) lớn hơn

+ Cho phép trùng lặp khóa

+ Xử lý range query tốt hơn

- Cài đặt phức tạp hơn.

- Tốn nhiều bộ nhớ hơn

7. Sử dụng Bảng băm và B+Tree trong Indexing.

B+Tree được sử dụng vì:

* Thao tác tìm kiếm, chèn và xóa nhanh, độ phức tạp O(log n)
* Có thể xử lí tìm kiếm trong khoảng
* Có thể xử lí tìm kiếm bằng một phần của khóa

Bảng băm được sử dụng vì: Độ phức tạp tìm kiếm O(1)