CẤU TRÚC DỮ LIỆU
CÂY ĐỎ ĐEN

Bùi Tiến Lên

01/01/2017

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Cây đỏ đen

Định nghĩa 1

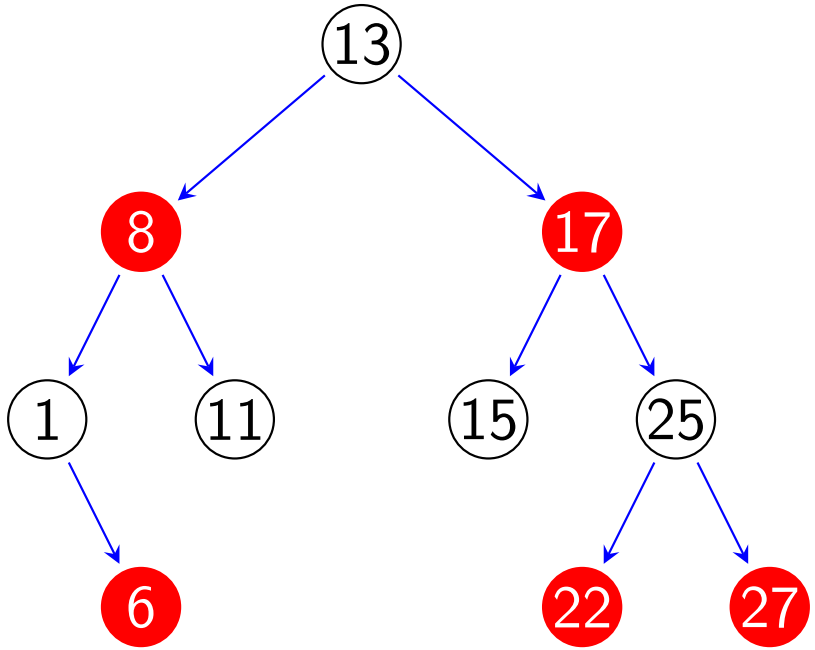
Cây đỏ đen (*red black tree*) được Rudolf Bayer phát minh và là một cây nhị phân tìm kiếm có các đặc điểm sau

1. Mọi nút phải là **nút đỏ** hoặc **nút đen**
2. Nút gốc là **nút đen**
3. Nếu một nút là **nút đỏ**, thì con của nó phải **nút đen**
4. Tất cả các đường đi từ nút gốc đến nút-0 (không có con) hoặc nút-1 (có 1 con) phải có cùng số lượng nút đen (điều kiện cân bằng)

Nhận xét

Cây đỏ đen là cây tổng quát của cây AVL

Cây đỏ đen (cont.)



Hình 1: Cây đỏ đen

Cấu trúc dữ liệu cho nút cây đỏ đen

Cấu trúc dữ liệu để lưu trữ cho nút cây đỏ đen

```
1  template <class T>
2  struct RBNode
3  {
4      T data;
5      int key;
6      NodeColor color;
7      RBNode *pLeft;
8      RBNode *pRight;
9      RBNode *pParent;
10 };
```

Tìm kiếm và duyệt

- Vì cây đỏ đen là một cây nhị phân tìm kiếm, do đó tìm kiếm và duyệt cây trên cây đỏ đen tương tự như trên cây nhị phân tìm kiếm

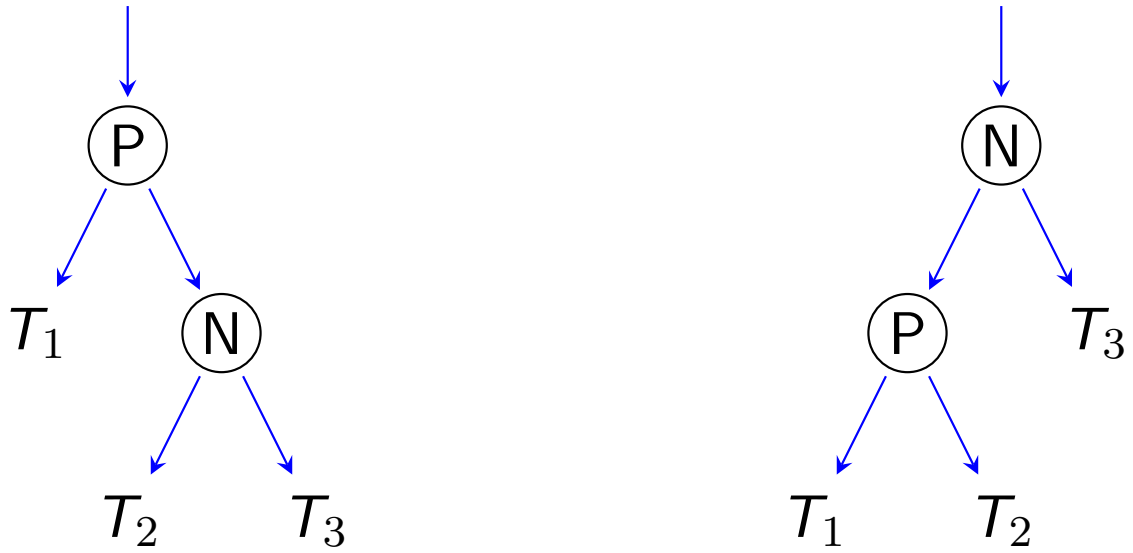
Các phép biến đổi cây đỏ đen cân bằng

Có ba phép biến đổi dùng để điều chỉnh cho cây đỏ đen cân bằng

- Thay đổi màu (*change color*)
- Thực hiện xoay trái (*left rotation*)
- Thực hiện xoay phải (*right rotation*)

Các phép biến đổi cây đỏ đen cân bằng (cont.)

Thực hiện xoay trái giữa hai nút Ⓟ và Ⓝ; trong đó, Ⓝ là nút con phải của Ⓟ

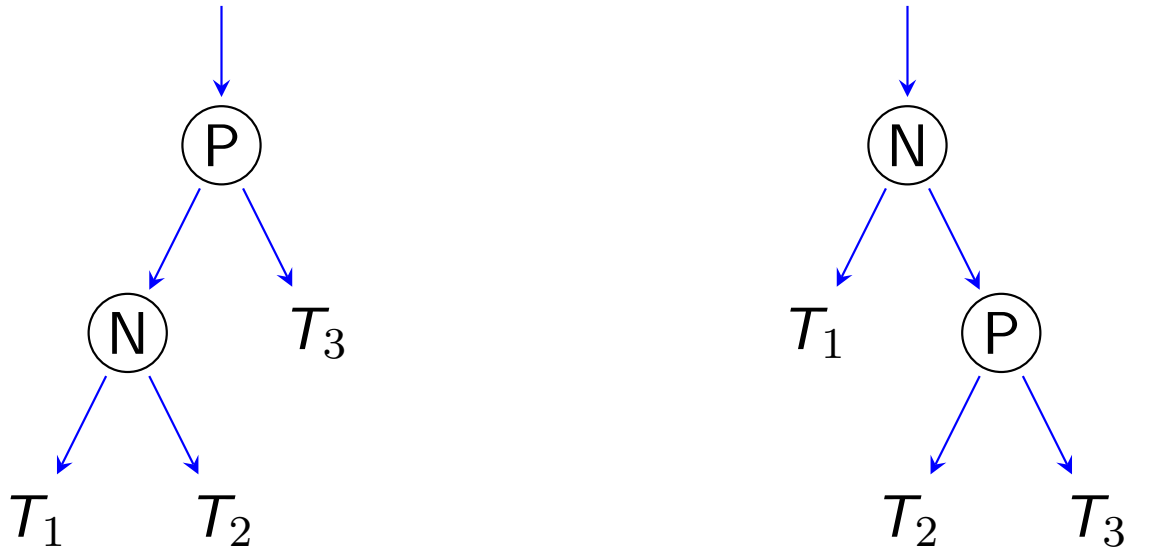


(a) trước khi xoay **(b)** sau khi xoay

Hình 2: Thao tác xoay trái

Các phép biến đổi cây đỏ đen cân bằng (cont.)

Thực hiện xoay phải giữa hai nút Ⓟ và Ⓝ; trong đó, Ⓝ là nút con trái của Ⓟ



(a) trước khi xoay **(b)** sau khi xoay

Hình 3: Thao tác xoay phải

Thêm một nút mới vào cây đỏ đen

- Sử dụng thuật toán thêm của cây nhị phân tìm kiếm để thêm nút mới
- Nút mới thêm luôn luôn là màu đỏ
- Duyệt từ nút vừa thêm trở về gốc để hiệu chỉnh cân bằng lại

Các tình huống xảy ra khi duyệt ngược

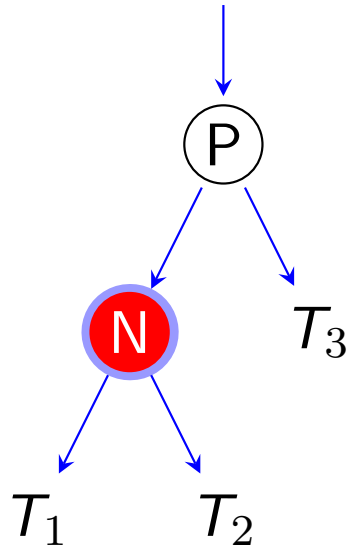
- **Trường hợp 1:** Nút đang xét Ⓝ là nút gốc có màu đỏ

(a) trước khi hiệu chỉnh **(b)** sau khi hiệu chỉnh

Hình 4: TH1 $\rightarrow$ Đổi màu nút Ⓝ thành màu đen. *Dừng hiệu chỉnh*

Các tình huống xảy ra khi duyệt ngược (cont.)

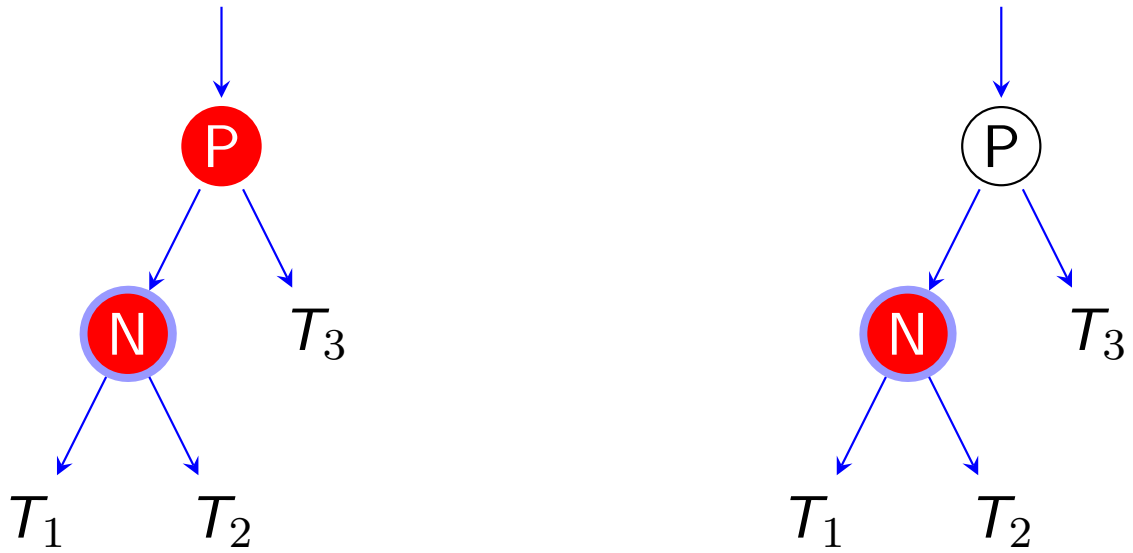
- **Trường hợp 2:** Nút đang xét Ⓝ là nút đỏ và nút cha Ⓟ nút đen



Hình 5: TH2 $\rightarrow$ *Dừng hiệu chỉnh*

Các tình huống xảy ra khi duyệt ngược (cont.)

- **Trường hợp 3:** Nút đang xét Ⓝ là nút đỏ và nút cha Ⓟ cũng là nút đỏ và là gốc

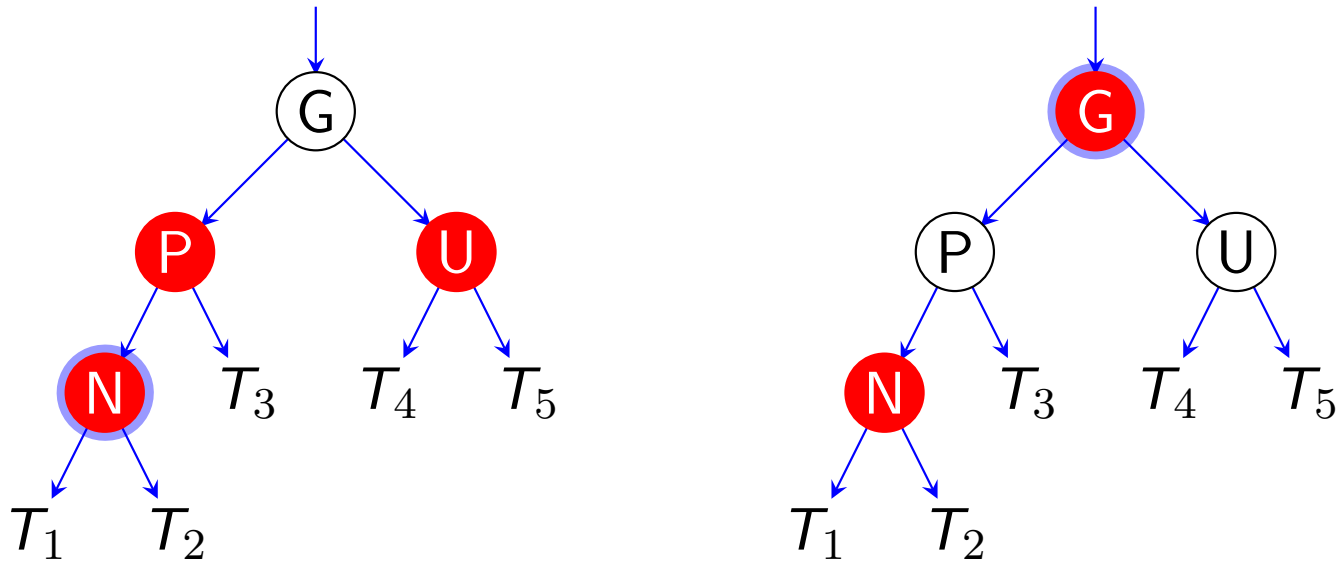


(a) trước khi hiệu chỉnh **(b)** sau khi hiệu chỉnh

Hình 6: TH3 $\rightarrow$ Đổi màu nút Ⓟ thành đen. *Dừng hiệu chỉnh*

Các tình huống xảy ra khi duyệt ngược (cont.)

- **Trường hợp 4:** Nút đang xét Ⓝ là nút đỏ và nút cha Ⓟ và nút chú Ⓤ cũng là nút đỏ

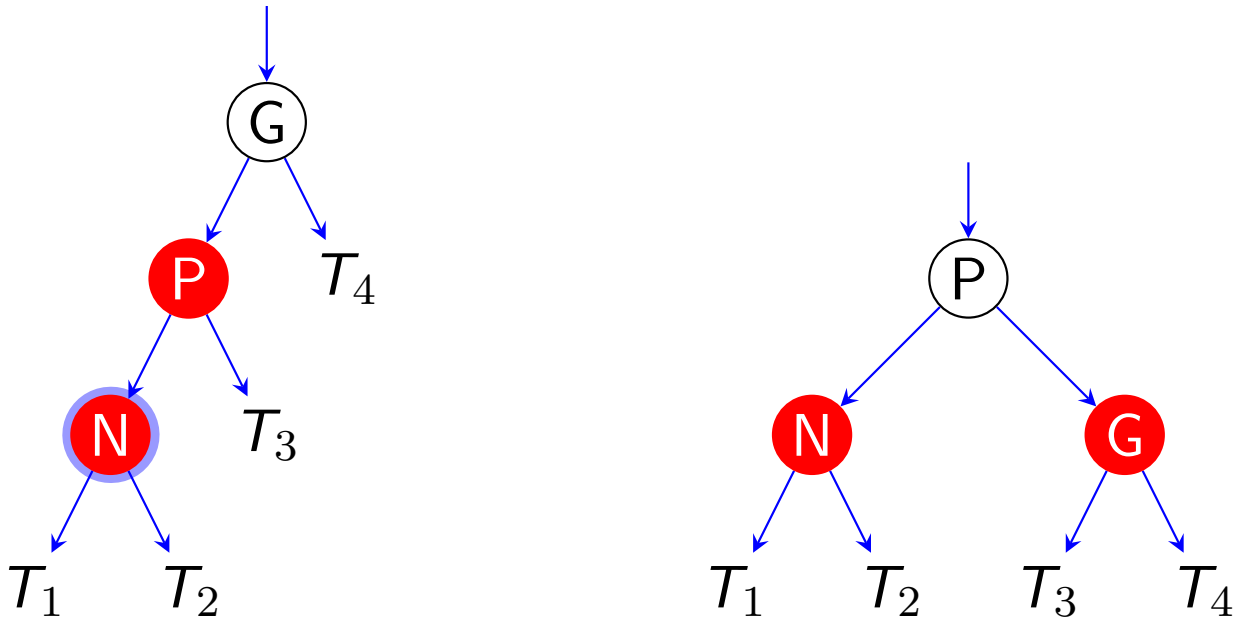


(a) trước xử lý **(b)** sau xử lý

Hình 7: TH4 → Đổi màu Ⓟ và Ⓤ thành đen, đổi màu Ⓖ thành đỏ. Tiếp tục xét nút Ⓖ

Các tình huống xảy ra khi duyệt ngược (cont.)

- **Trường hợp 5:** Nút đang xét Ⓝ là nút đỏ và nút cha Ⓟ, ...

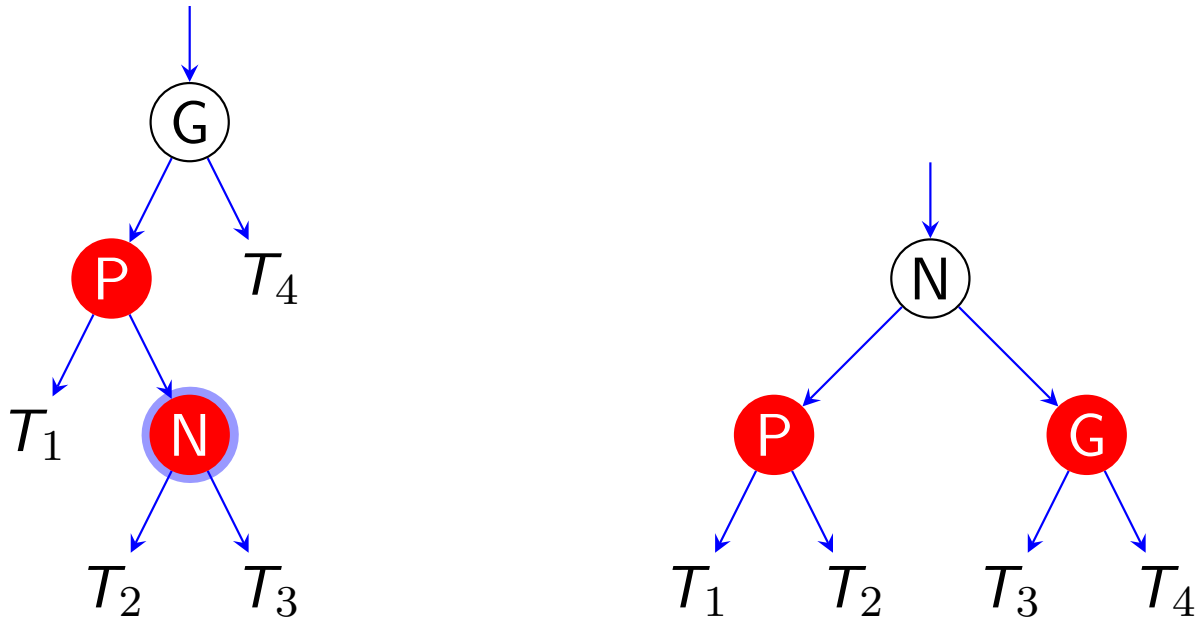


(a) trước xử lý **(b)** sau xử lý

Hình 8: TH5 (T_4 là cây rỗng hoặc gốc là nút đen) $\rightarrow$ Đổi màu Ⓟ thành đen, Ⓖ thành đỏ, xoay Ⓟ và Ⓖ. *Dừng hiệu chỉnh*

Các tình huống xảy ra khi duyệt ngược (cont.)

- **Trường hợp 6:** Nút đang xét Ⓝ là nút đỏ và nút cha Ⓟ, ...



(a) trước xử lý **(b)** sau xử lý

Hình 9: TH6 (T_4 là cây rỗng hoặc gốc là nút đen) $\rightarrow$ Đổi màu Ⓝ thành đen, Ⓖ thành đỏ, xoay Ⓝ và Ⓟ, xoay Ⓝ và Ⓖ. *Dừng hiệu chỉnh*

Các tình huống xảy ra khi duyệt ngược (cont.)

- Một số trường hợp còn lại là đối xứng của các trường hợp đã xét
- Sinh viên hãy liệt kê ra các trường hợp

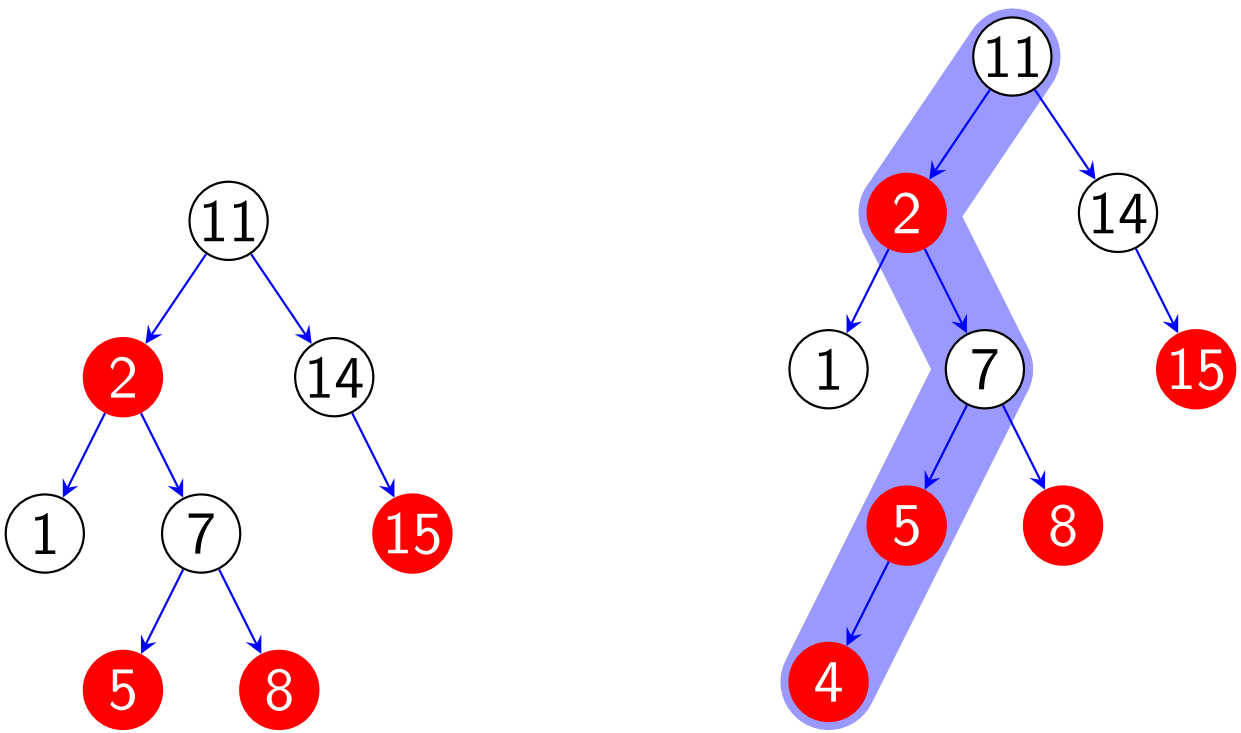
Các tình huống xảy ra khi duyệt ngược (cont.)

Bài tập

Sinh viên hãy cho biết sự thay đổi sau các xử lý

- Số nút đen
- Số nút đỏ
- Số lượng nút đen trên các đường đi từ gốc đến nút
- Chiều cao của cây

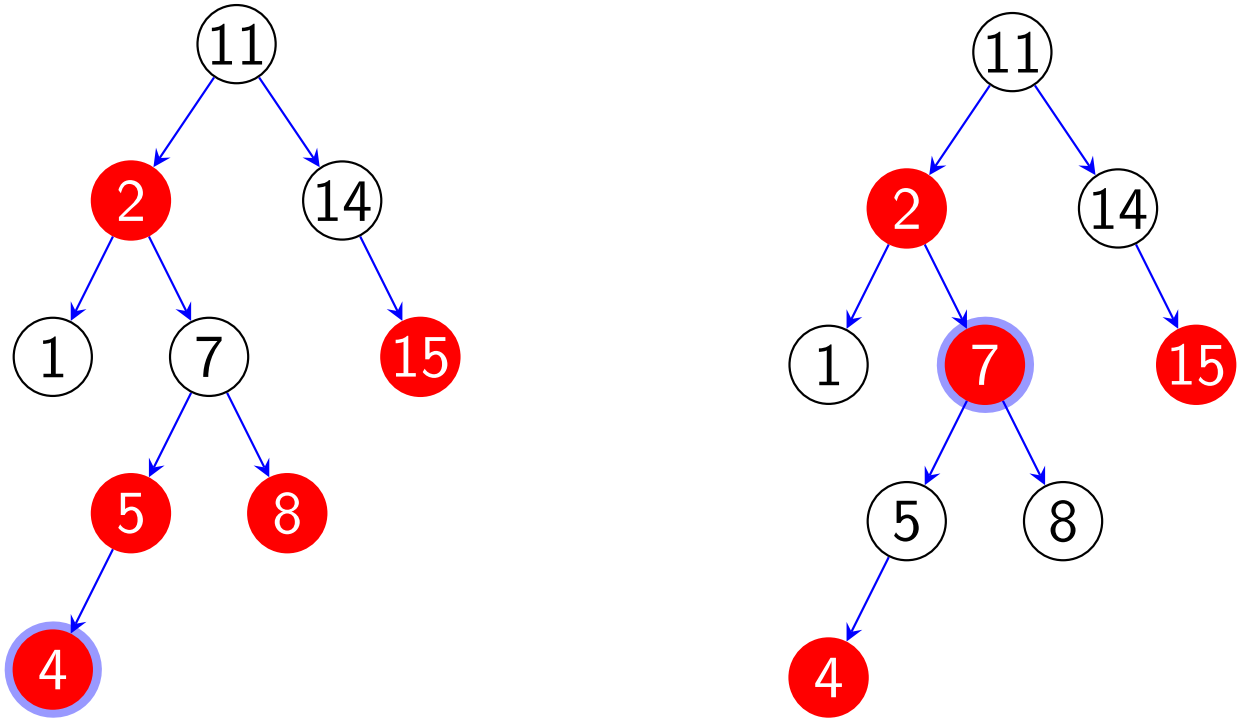
Minh họa thêm phần tử



(a) trước khi thêm **(b)** sau khi thêm

Hình 10: Thêm nút (4)

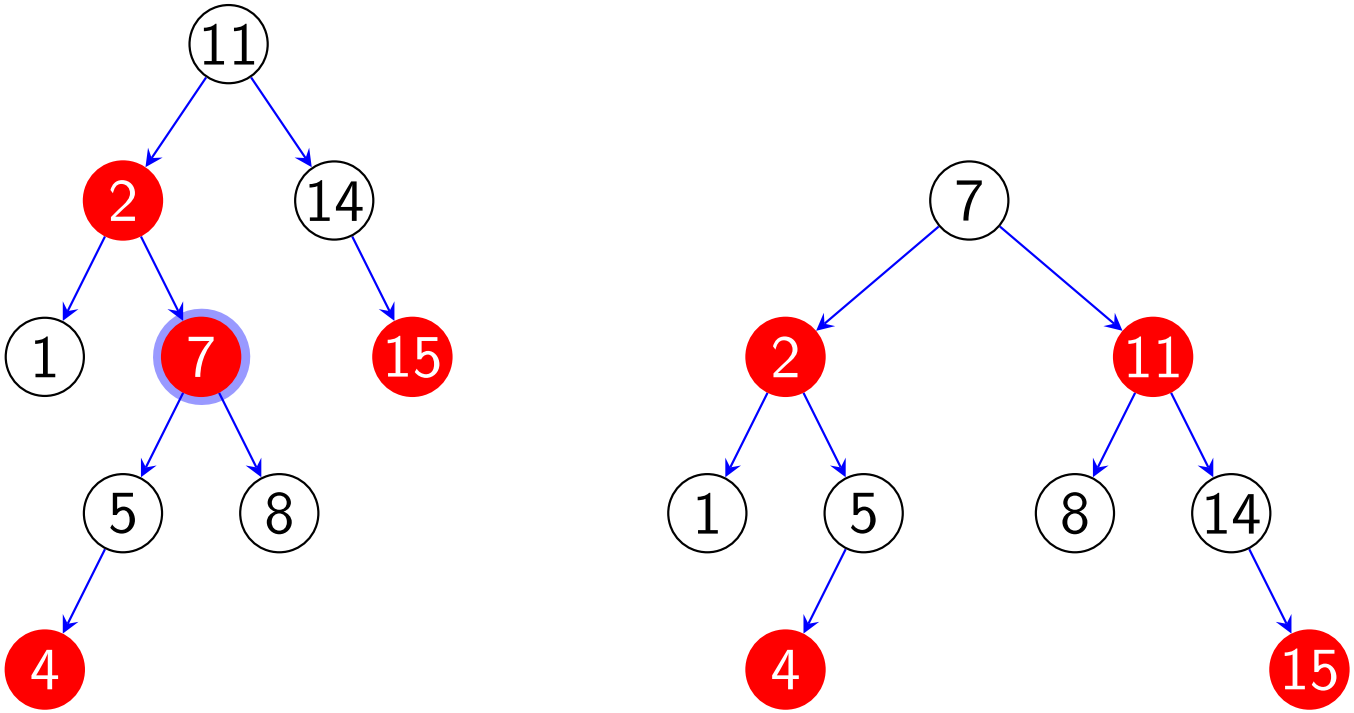
Minh họa thêm phần tử (cont.)



(a) trước xử lý **(b)** sau xử lý

Hình 11: Trường hợp 4

Minh họa thêm phần tử (cont.)



(a) trước xử lý **(b)** sau xử lý

Hình 12: Trường hợp 6

Xóa một nút

- Sử dụng thuật toán xóa một phần tử của cây nhị phân tìm kiếm để xóa phần tử của cây đỏ đen
- Nếu nút bị xóa có màu đỏ thì không cần phải làm gì thêm
- Nếu nút bị xóa có màu đen cần phải điều chỉnh sự cân bằng của cây
  1. Mọi nút phải **đỏ** hoặc **đen** ☑
  2. Nút gốc là **đen** ☑
  3. Nút cha và nút con không cùng **đỏ** ☑
  4. Mọi đường dẫn từ gốc đến nút-0 và nút-1 có cùng số lượng nút **đen** ☑

Bài luyện tập

Ví dụ 1

Hãy xây dựng cây đỏ đen từ dãy {5,1, 4, 3, 2, 8, 9, 7, 16, 11, 12, 15}

- Xóa các nút 8, 16
- Thêm các nút 6, 17

Định lý về chiều cao của cây đỏ đen

Định nghĩa 2

Một số thuật ngữ cho cây đỏ đen

- **Chiều cao** (*node height*) của một nút x là số nút của một đường đi dài nhất từ nút x đến một nút ngoài

$$h(x) = max\{CountNode(path(x, l))\} \ l \text{ là nút ngoài} \quad (1)$$

- **Chiều cao đen** (*black node height*) của một nút x là số nút đen trên đường đi từ nút x đến nút ngoài

$$h_b(x) = CountBlackNode(path(x, l)) \ l \text{ là nút ngoài} \quad (2)$$

Định lý về chiều cao của cây đỏ đen (cont.)

Định lý 1

1. *Chiều cao của cây nhỏ hơn hai lần chiều cao đen*

$$h \leq 2h_b$$

2. *Cây đỏ đen có n nút thì*

$$h \leq 2\log_2(n+1)$$

Đánh giá về cây đỏ đen

Phân tích chi phí thực hiện theo n (số lượng nút của cây)

	xấu nhất	**trung bình**	**tốt nhất**
tìm một phần tử	?	?	?
thêm một phần tử	?	?	?
xóa một phần tử	?	?	?