CHƯƠNG 5. CẤU TRÚC CÂY
CÂY NHỊ PHÂN TÌM KIẾM

ThS. Nguyễn Chí Hiếu

2021

NỘI DUNG

Giới thiệu cây nhị phân tìm kiếm

Cây nhị phân tìm kiếm - NPTK *(Binary Search Tree - BST)*

Là một *cây nhị phân* thỏa các điều kiện sau:

- Khóa của các nút thuộc cây con trái *nhỏ* hơn khóa nút gốc.
- Khóa của các nút thuộc cây con phải *lớn* hơn khóa nút gốc.
- Hai cây con trái và phải của nút gốc cũng là một cây NPTK.

Giới thiệu cây nhị phân tìm kiếm

Cây nhị phân tìm kiếm - NPTK *(Binary Search Tree - BST)*

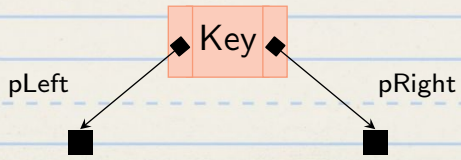
Đặc điểm cây NPTK

- Dữ liệu lưu trữ có thứ tự, hỗ trợ tìm kiếm tốt hơn danh sách liên kết, ngăn xếp, hàng đợi, ...
- Nếu tổng số nút trong cây NPTK là n thì chi phí tìm kiếm trung bình $log_2 n$.

Cách lưu trữ cây NPTK

Cấu trúc dữ liệu của một nút

- Thành phần dữ liệu: khóa *(key)* của một nút.
- Thành phần liên kết: con trỏ `pLeft` liên kết với *cây con trái* và con trỏ `pRight` liên kết với *cây con phải*.



Cấu trúc dữ liệu của một cây

- Chỉ cần một con trỏ trỏ đến nút gốc của cây.

Thao tác thêm nút vào cây NPTK

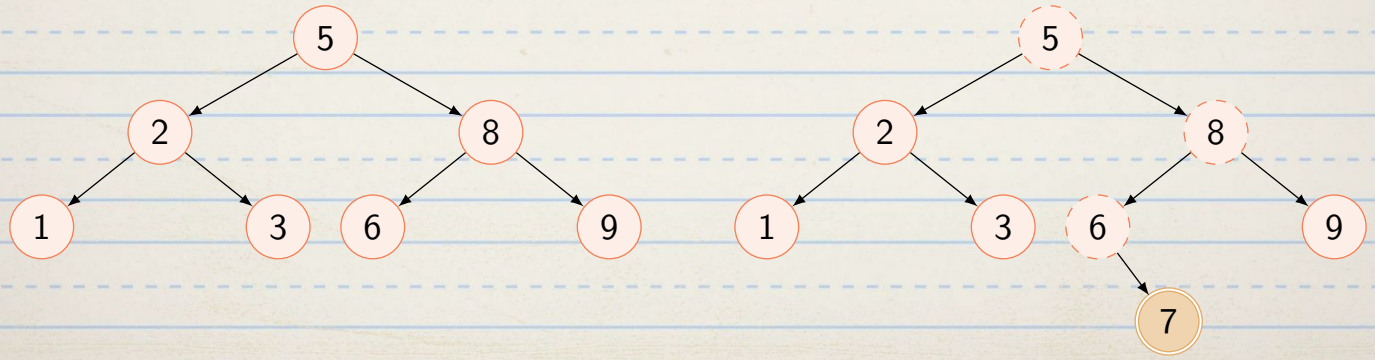
Thuật toán 1: `InsertNode(t, k)`
- Đầu vào: cây T và khóa k cần thêm.
- Đầu ra: cây T sau khi thêm k.

```
1   if cây rỗng // TH1. cay rong
2     Khởi tạo nút p có khóa k và cập nhật pLeft, pRight
3     pRoot trỏ đến p
4   else // TH2. cay khac rong
5     if pRoot->Key > k
6       Gọi đệ quy hàm InsertNode() với cây con bên trái
7     else if pRoot->Key < k
8       Gọi đệ quy hàm InsertNode() với cây con bên phải
```

Thao tác thêm nút vào cây NPTK

Ví dụ 1

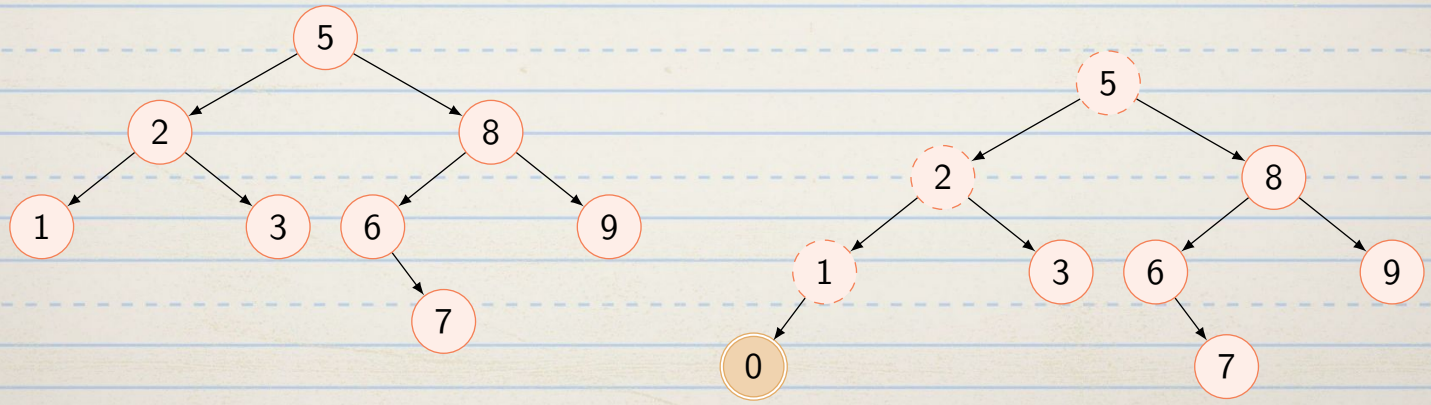
Cho cây NPTK, thêm nút có khóa là 7.



Thao tác thêm nút vào cây NPTK

Ví dụ 2

Cho cây NPTK, thêm nút có khóa là 0.



Thao tác thêm nút vào cây NPTK

```
1  public void Insert(ref Node root, Node p)
2  {
3      if (root == null)
4      {
5          root = p;
6      }
7
```


Thao tác thêm nút vào cây NPTK

```
8      else
9      {
10         if (root.Key > p.Key)
11         {
12             Insert(ref root.pLeft, p);
13         }
14         else if (root.Key < p.Key)
15         {
16             Insert(ref root.pRight, p);
17         }
18         else
19         {
20             return;
21         }
22     }
23 }
```

Thao tác tìm nút có khóa k trong cây NPTK

Thuật toán 2: SearchNode(t, k)

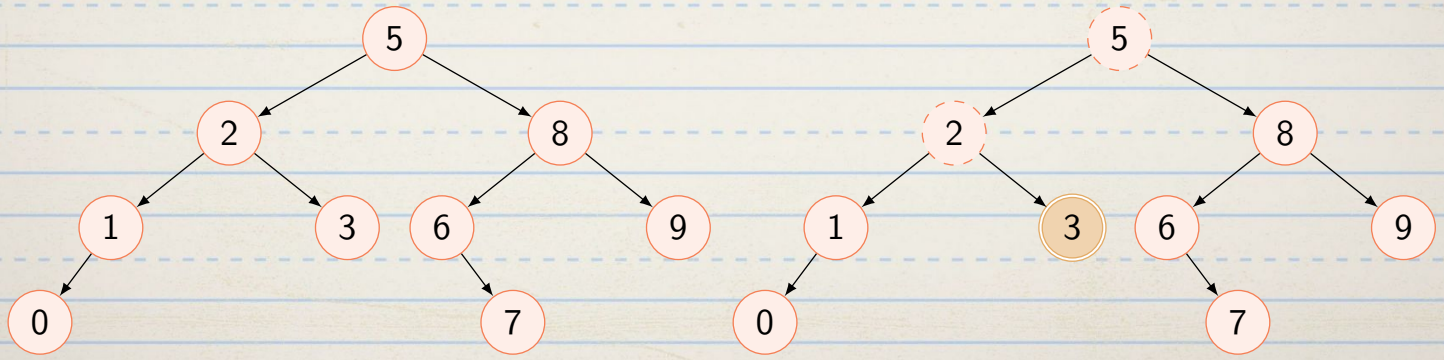
- Đầu vào: cây T và nút k cần tìm.
- Đầu ra: nút có khóa k hay NULL nếu không tìm thấy.

```
1   if cây khác rỗng
2     if pRoot->pKey = k
3       Trả về nút pRoot
4     else if pRoot->pKey > k
5       Gọi đệ quy hàm SearchNode() với cây con bên trái
6     else
7       Gọi đệ quy hàm SearchNode() với cây con bên phải
8   Trả về không tìm thấy
```


Thao tác tìm nút có khóa k trong cây NPTK

Ví dụ 3

Cho cây NPTK, tìm nút có khóa là 3.



Thao tác tìm nút có khóa k trong cây NPTK

```
1  public Node SearchNode(ref Node root, int k)
2  {
3    if (root != null)
4    {
5      if (root.Key == k) // TH1. Tim thay nut co khoa k
6        return root;
7      else if (root.Key > k) // Tim de quy cay con trai
8        return SearchNode(ref root.pLeft, k);
9      else // Tim de quy cay con phai
10       return SearchNode(ref root.pRight, k);
11   }
12   // TH2. Khong tim thay
13   return null;
14 }
```


Thao tác xóa một nút trong cây NPTK

Chia 3 trường hợp

- Trường hợp 1: nút khóa k là *nút lá*.
- Trường hợp 2: nút khóa k chỉ có *1* cây con trái hay phải.
- Trường hợp 3: nút khóa k chứa đầy đủ *2* cây con, thực hiện thao tác tìm *nút phải nhất của cây con trái* hay *nút trái nhất của cây con phải*.

Thao tác xóa một nút trong cây NPTK

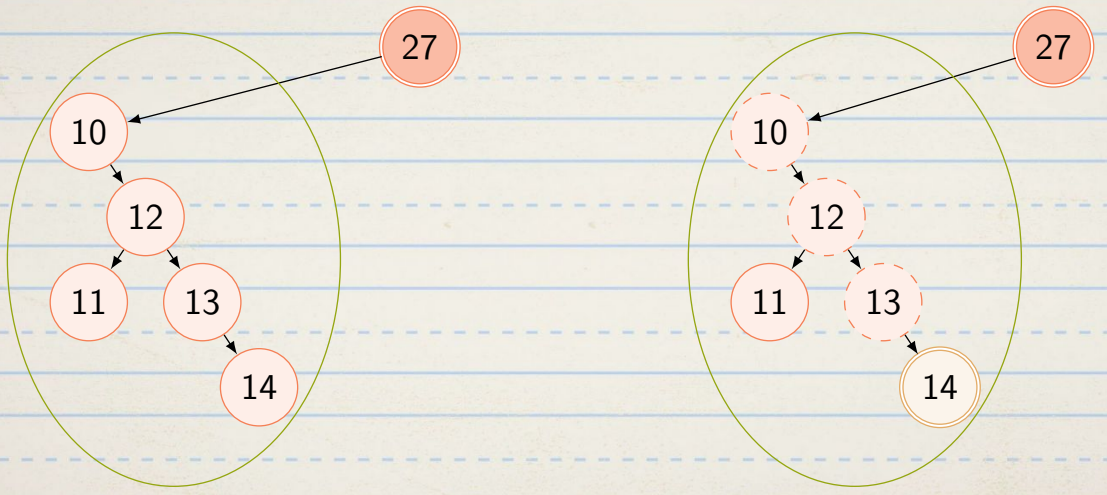
Tìm nút thay thế là nút phải nhất cây con trái

Thuật toán 3: SearchStandFor(t, p)

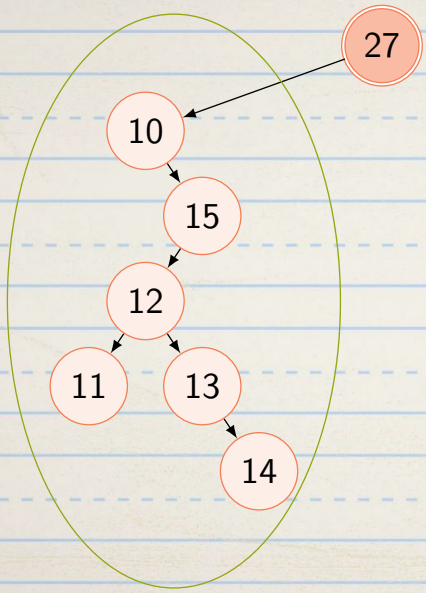
- Đầu vào: cây T.
- Đầu ra: nút p là nút thay thế (nút phải nhất/lớn nhất của cây T).

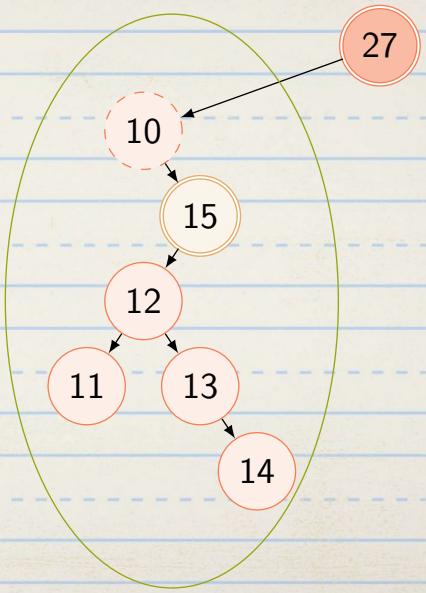
```
1    // Tìm theo nhánh bên phải của cây
2    if cây con phải nút đang xét khác rỗng
3      Gọi đệ quy hàm SearchStandFor() với cây con phải
4    else // Tim phan tu thay the
5      Chép dữ liệu của pRoot vào nút p ...
6      Lưu lại nhánh con trái (trường hợp nút p có cây con trái)
```


Thao tác xóa một nút trong cây NPTK



Thao tác xóa một nút trong cây NPTK





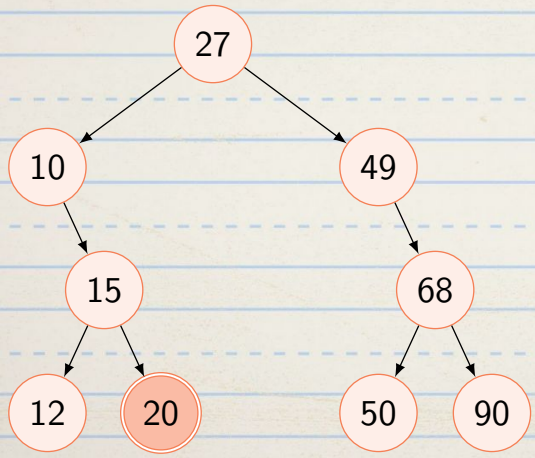
Thao tác xóa một nút trong cây NPTK

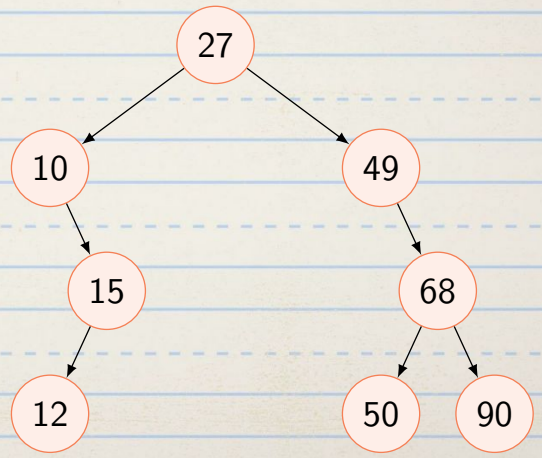
Tìm nút thay thế là nút phải nhất cây con trái

```
 1  public void SearchStandFor(ref Node root, Node p)
 2  {
 3    if (root.pRight != null)
 4      SearchStandFor(ref root.pRight, p);
 5    else
 6    {
 7      p.Key = root.Key;
 8      p = root;
 9      root = root.pLeft;
10    }
11  }
```

Thao tác xóa một nút trong cây NPTK

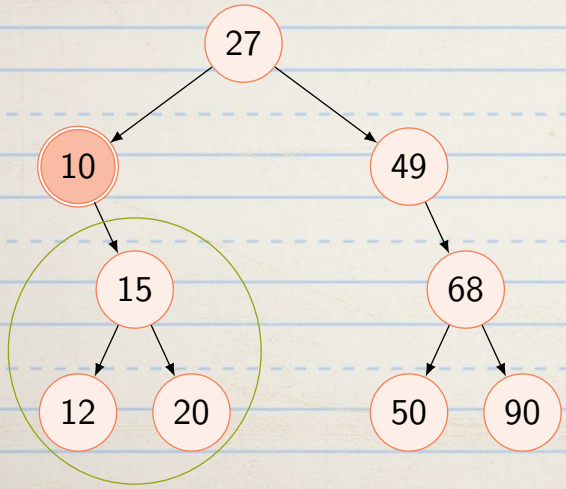
Trường hợp 1: nút k là *nút lá*.

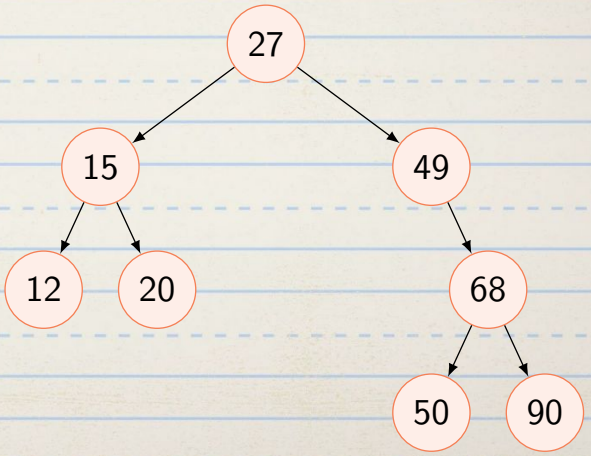




Thao tác xóa một nút trong cây NPTK

Trường hợp 2: nút k chỉ có *1* cây con trái hay phải





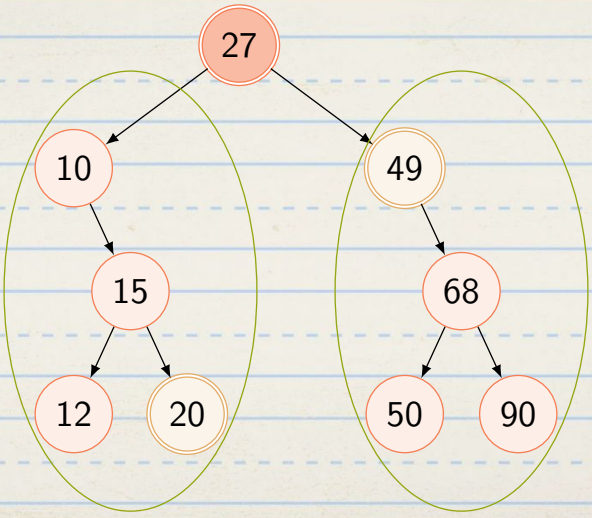
Thao tác xóa một nút trong cây NPTK

Trường hợp 3: nút k có đầy đủ *2* cây con

Thực hiện thao tác tìm *nút thay thế* trong 2 nút.

- Nút *phải nhất/lớn nhất* của *cây con trái*
- Nút *trái nhất/nhỏ nhất* của *cây con phải*.

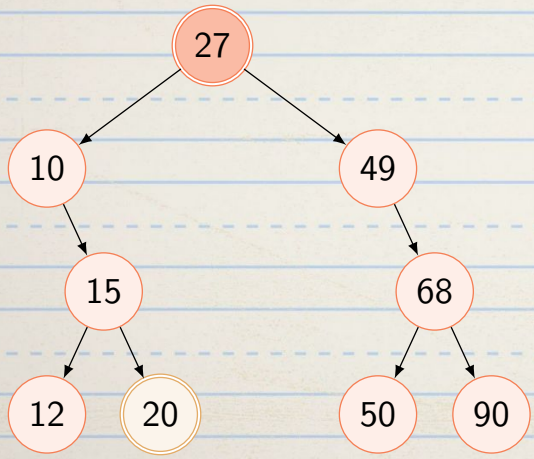
Thao tác xóa một nút trong cây NPTK

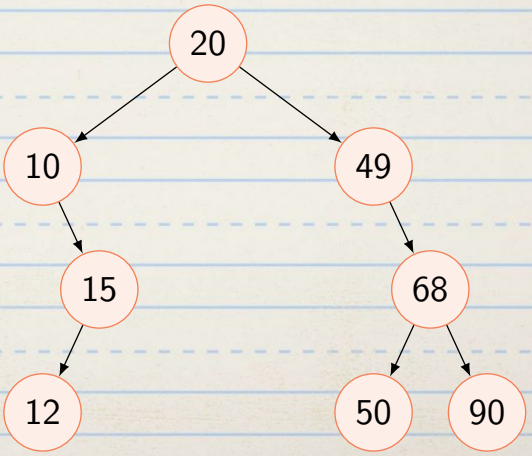


Thao tác xóa một nút trong cây NPTK

Nút k có đầy đủ *2* cây con

- Cách 1. Thực hiện thao tác tìm *nút phải nhất* của *cây con trái*.

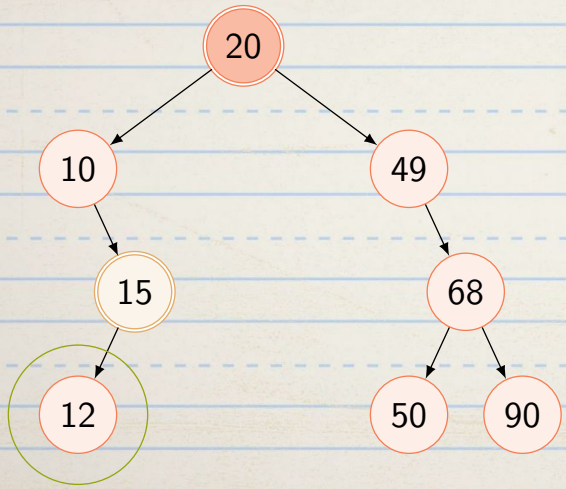


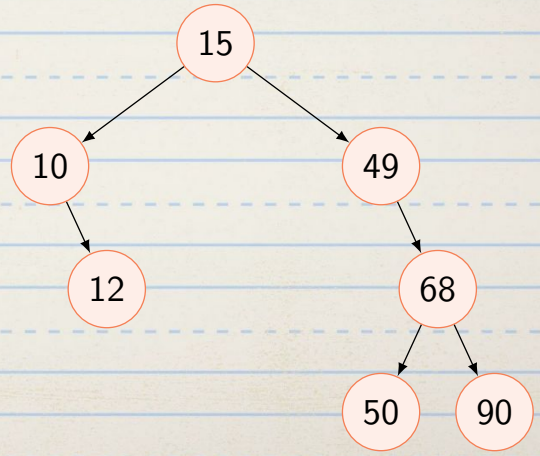


Thao tác xóa một nút trong cây NPTK

Nút k có đầy đủ *2* cây con

- Cách 1. Thực hiện thao tác tìm *nút phải nhất* của *cây con trái*.

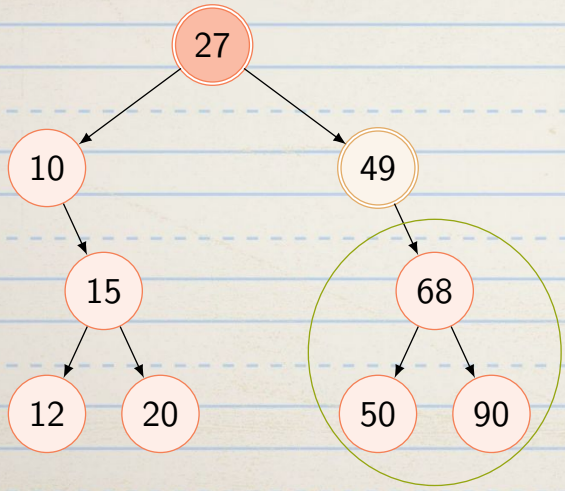


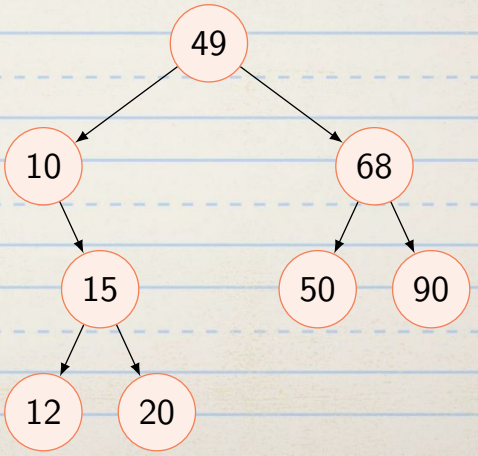


Thao tác xóa một nút trong cây NPTK

Nút k có đầy đủ *2* cây con

- Cách 2. Thực hiện thao tác tìm *nút trái nhất* của *cây con phải*.

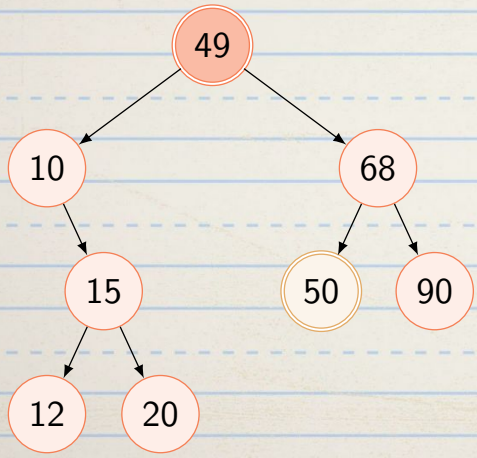


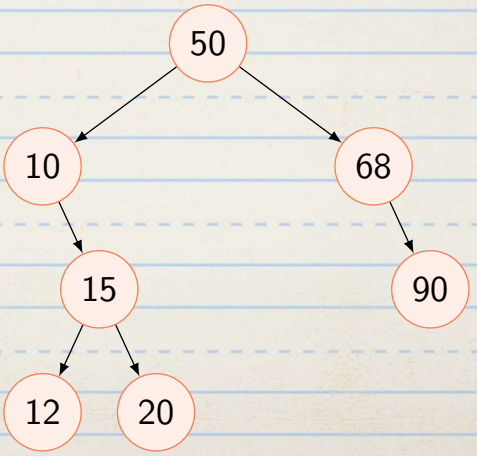


Thao tác xóa một nút trong cây NPTK

Nút k có đầy đủ *2* cây con

- Cách 2. Thực hiện thao tác tìm *nút trái nhất* của *cây con phải*.





Thao tác xóa một nút trong cây NPTK

Thuật toán 4: RemoveNode(t, k)

- Đầu vào: cây T và nút có khóa k cần xóa.
- Đầu ra: cây T sau khi xóa nút có khóa k.

```
1  if cây rỗng // TH1. Khong tim thay khoa k
2  ___Dừng thuật toán
3  if pRoot->Key > k
4  ___Gọi đệ quy hàm RemoveNode() với cây con trái
5  else if pRoot->Key < k
6  ___Gọi đệ quy hàm RemoveNode() với cây con phải
7  // TH2. Tim thay nut pRoot co khoa k
```


Thao tác xóa một nút trong cây NPTK

```
8  else
9  ___// Xoa nut p tuong ung 3 truong hop
10 ___// TH2.1. nut la
11 ___Khai báo nút p trỏ đến pRoot
12 ___// TH2.2. nut co 1 cay con trai/phai
13 ___if p chỉ có 1 cây con trái // nut co 1 cay con trai
14 ___ ___pRoot trỏ đến cây con trái của p
15 ___else if p chỉ có 1 cây con phải // nut co 1 cay con phai
16 ___ ___pRoot trỏ đến cây con phải của p
17 ___else // TH2.3. p co 2 cay con
18 ___ ___Gọi hàm tìm nút thay thế trước khi xóa nút có khóa k ...
19 ___Xóa nút p
```

Thao tác xóa một nút trong cây NPTK

```
1  public void RemoveNode(Node root, int k)
2  {
3      Node p = new Node();
4      if (pRoot == null) // TH1. Khong tim thay nut co khoa k
5          return;
6      if (root.Key > k)
7          RemoveNode(root.pLeft, k);
8      else if (root.Key < k)
9          RemoveNode(root.pRight, k);
```


Thao tác xóa một nút trong cây NPTK

```
10     else // TH2. Tim thay nut co khoa k
11     {
12         // TH2.1. nut la
13         p = root;
14         // TH2.2 nut co 1 cay con trai/phai
15         if (p.pRight == null) // nut co 1 cay con trai
16             root = p.pLeft;
17         else if (p.pLeft == null) // nut co 1 cay con phai
18             root = p.pRight;
19         // TH2.3. nut co 2 cay con
20         else
21             SearchStandFor(ref root.pLeft, p);
22
23         p = null;
24     }
25 }
```

Bài tập

1. Cho dãy gồm 9 phần tử: 5, 9, 3, 1, 8, 7, 4, 6, 2.
   - Lần lượt thêm các phần tử trên vào cây NPTK.
   - In cây nhị phân tìm kiếm theo 3 phương pháp duyệt cây: NLR, LNR, LRN.
   - Thêm vào phần tử 10.
   - Xóa phần tử 5.
2. Khử đệ quy các thuật toán duyệt cây NLR, LNR, LRN *(sử dụng phương pháp lặp)*.
3. Cài đặt thuật toán tìm kiếm phần tử nhỏ nhất/lớn nhất trong cây NPTK.
4. Ứng dụng cây nhị phân tìm kiếm (BST) xây dựng cấu trúc dữ liệu từ điển có chức năng lưu trữ từ tiếng Anh và nghĩa tiếng Việt tương ứng của nó. Viết các hàm thực hiện thao tác:
   - Thêm từ vựng mới vào từ điển.
   - **In tất cả các từ vựng có trong từ điển theo thứ tự tăng dần của chữ cái.**
   - **Tra từ điển.**

Tài liệu tham khảo

Donald E. Knuth.
The Art of Computer Programming, Volume 3.
Addison-Wesley, 1998.

Dương Anh Đức, Trần Hạnh Nhi.
Nhập môn Cấu trúc dữ liệu và Thuật toán.
Đại học Khoa học tự nhiên TP Hồ Chí Minh, 2003.

Niklaus Wirth.
Algorithms + Data Structures = Programs.
Prentice-Hall, 1976.

Robert Sedgewick.
Algorithms in C.
Addison-Wesley, 1990.