CẤU TRÚC DỮ LIỆU
CÂY

Bùi Tiến Lên

01/01/2017

cuu duong than cong . com

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

GIỚI THIỆU CÂY

cuu duong than cong . com

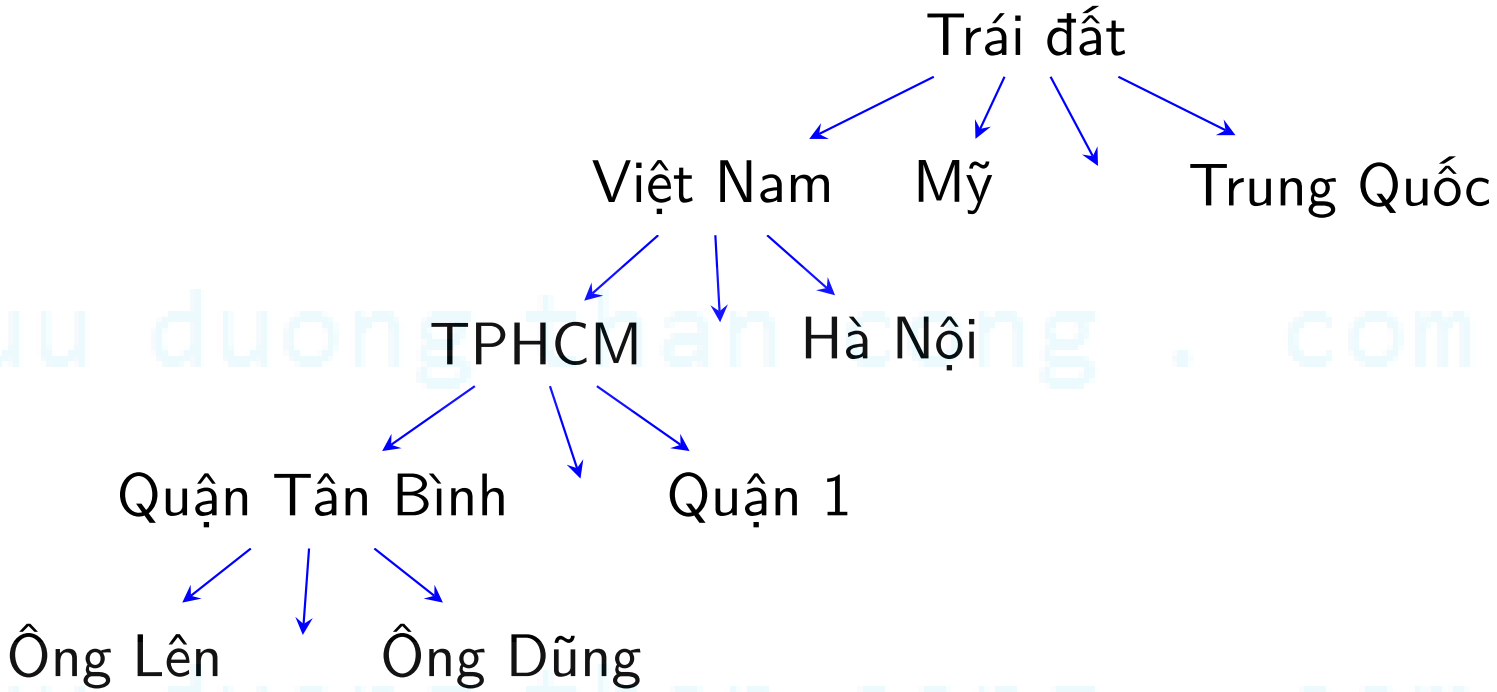
Ứng dụng của kiểu dữ liệu cây

Kiểu dữ liệu cây thể hiện tính "phân cấp", "kế thừa". Do đó có thể biểu diễn được những cấu trúc như

- Cây gia phả (trong các dòng họ)
- Cây phân cấp các loài (trong sinh học)
- Cây thư mục (trong máy tính)

Ứng dụng của kiểu dữ liệu cây (cont.)

- Hệ thống quản lý hành chính phân cấp toàn thế giới

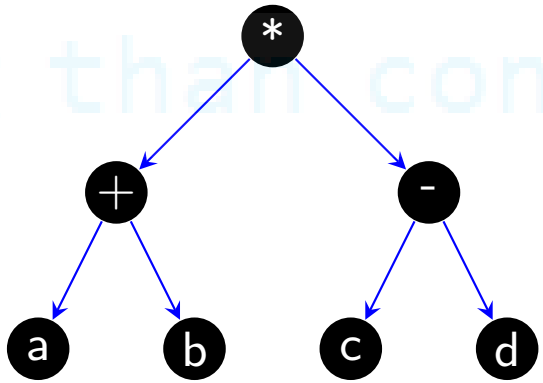


Hình 1: Quản lý hành chính toàn cầu

Ứng dụng của kiểu dữ liệu cây (cont.)

- Biểu thức toán học có thể được biểu diễn bằng cây. Ví dụ cây dưới đây dùng để biểu diễn biểu thức

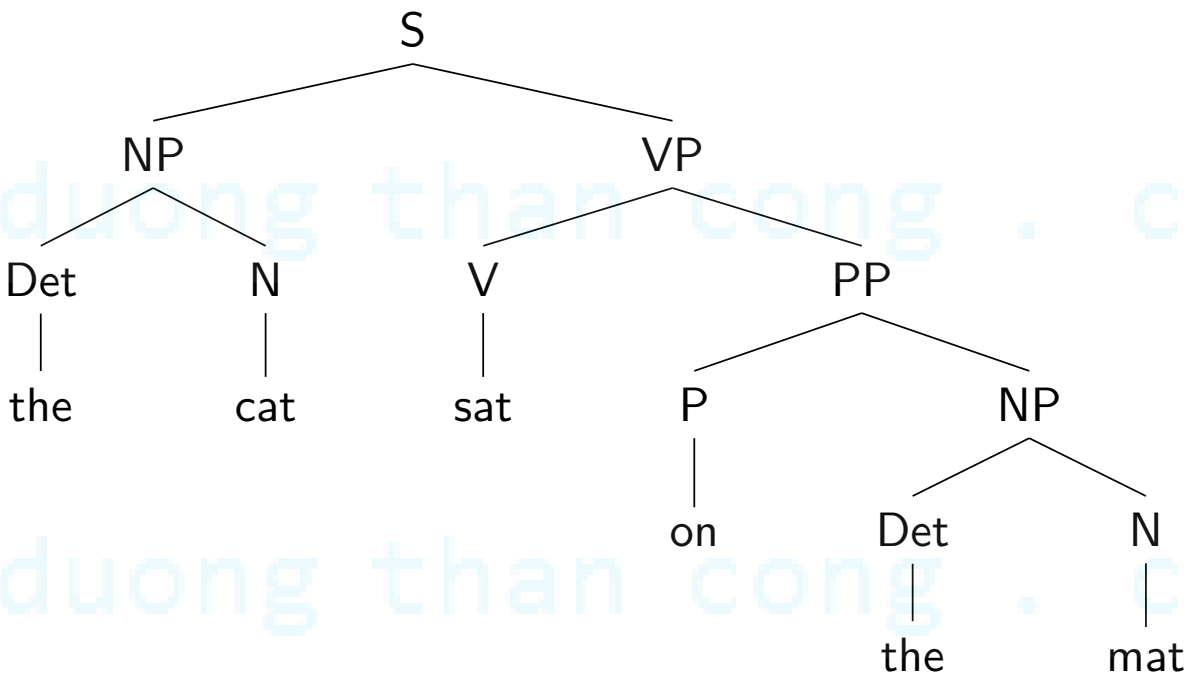
$$(a + b) * (c - d)$$



Hình 2: Cây biểu thức

Ứng dụng của kiểu dữ liệu cây (cont.)

- Các nhà ngôn ngữ học thường dùng cây ngữ pháp để biểu diễn cấu trúc ngữ pháp của một câu. Ví dụ sau đây dùng để biểu diễn câu *"the cat sat on the mat"*



Hình 3: Cây ngữ pháp

Kiểu dữ liệu cây

Định nghĩa 1

Cây (tree) là một cấu trúc phi tuyến. Được định nghĩa đệ qui như sau

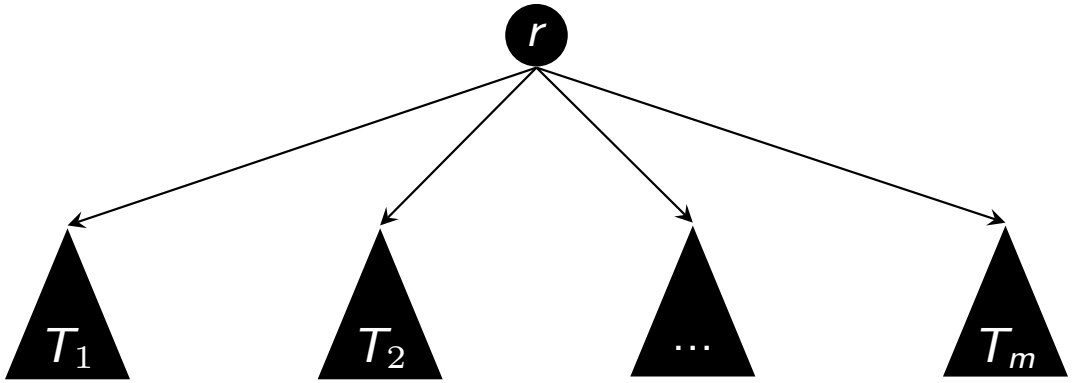
- **Cây** T là
  - **cây rỗng**

$$T = \emptyset$$

  - gồm nút gốc r và một tập các **cây con** có thứ tự $\{T_1, T_2, ..., T_m\}$

$$T = \{r \rightarrow \{T_1, , T_2, ..., T_m\}\}$$

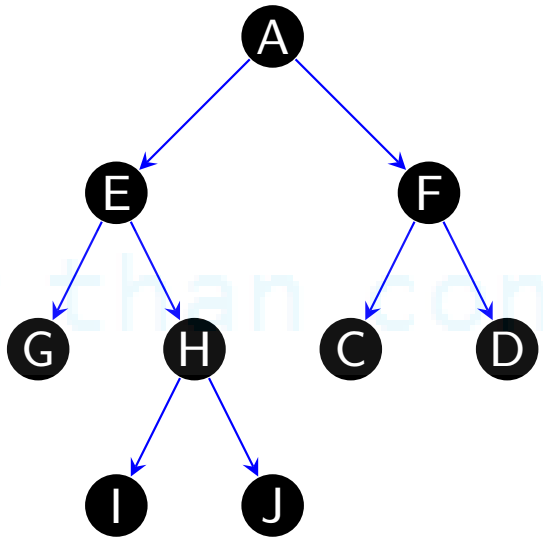
Kiểu dữ liệu cây (cont.)



Hình 4: Cây trong tin học

Các thuật ngữ liên quan đến cây

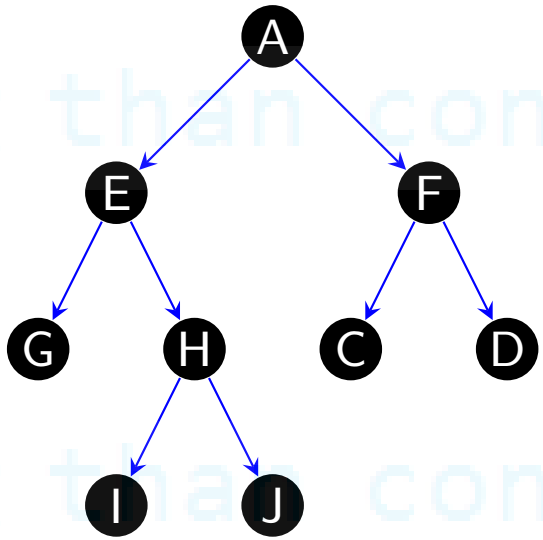
- **Nút** (*node*): là những phần tử trong cây



Hình 5: Cây có các nút {A, B, C, D, E, F, G, H, J}

Các thuật ngữ liên quan đến cây (cont.)

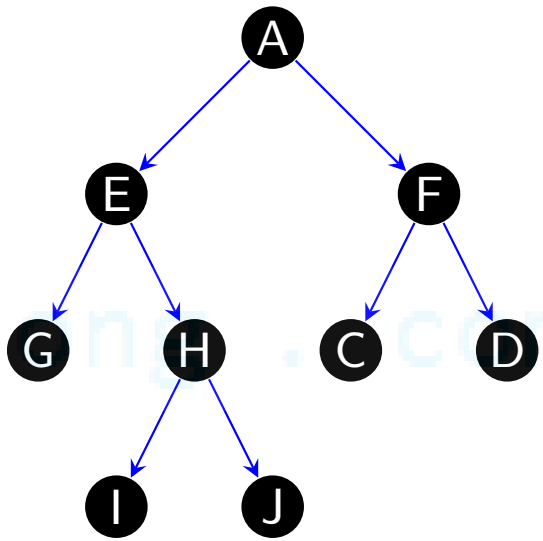
- **Nhánh** (*branch*): là cạnh mũi tên nối giữa hai nút trong cây
- **Nút cha** (*parent node*) và **nút con** (*child node*) là hai quan hệ được định nghĩa trên một cạnh, nút cha là nút đầu cạnh và nút con là nút cuối cạnh



Hình 6: Nút E là nút cha của H, nút H là nút con của E

Các thuật ngữ liên quan đến cây (cont.)

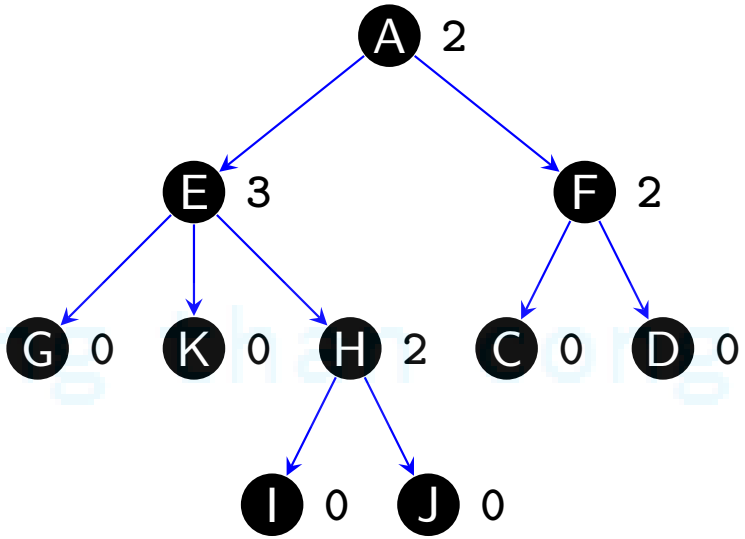
- **Nút gốc** (*root node*): là nút không có cha
- **Nút lá** (*leaf node*): là nút không có con
- **Nút nội** (*internal node*): là nút có cha và có con
- **Nút anh em** (*sibling node*): là những nút có cùng cha



Hình 7: Nút A là nút gốc; nút G, I, J, C, D là nút lá; nút E, H, F là nút nội; nút G và H là anh em

Các thuật ngữ liên quan đến cây (cont.)

- **Bậc của nút** (*node degree*): là tổng số nút con của nút này

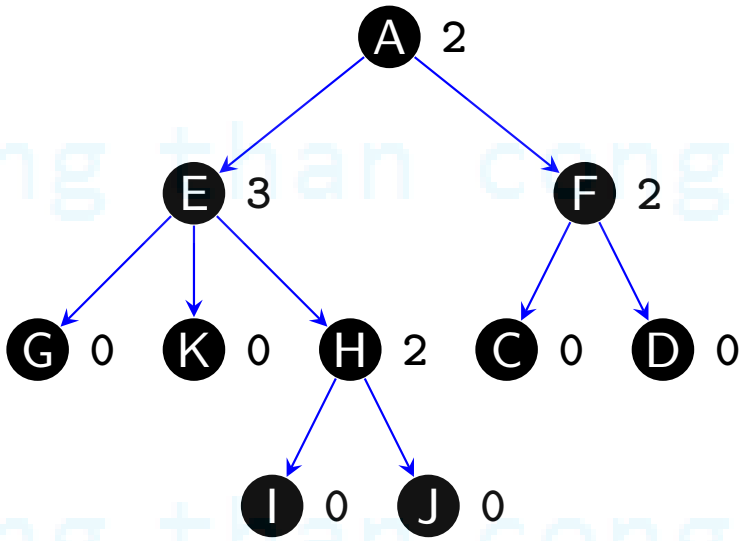


Hình 8: Cây và bậc của các nút

Các thuật ngữ liên quan đến cây (cont.)

- **Bậc của cây** (*tree degree*): là bậc lớn nhất của các nút của cây

$$\deg(T) = \max(\deg(p_i), p_i \in T) \quad (1)$$

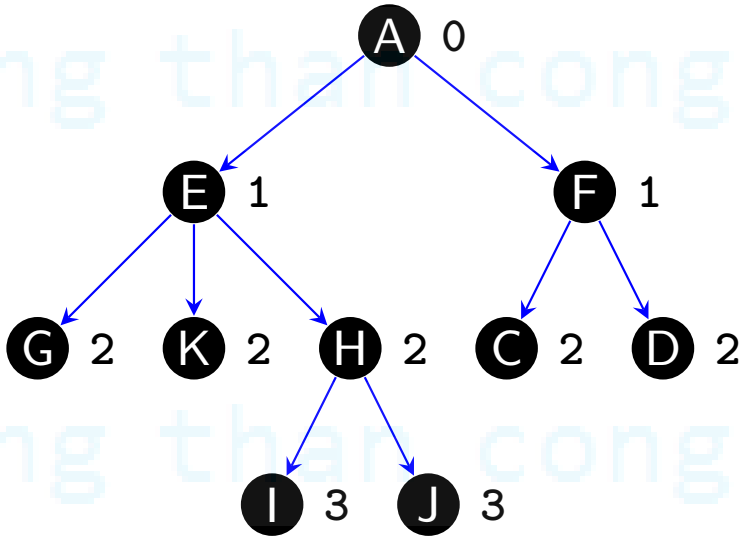


Hình 9: Bậc của cây là 3

Các thuật ngữ liên quan đến cây (cont.)

- **Mức của nút** (*node level*):

$$level\,(p) = \begin{cases} 0 & p = root \\ level\,(parent\,(p)) + 1 & p \neq root \end{cases} \quad (2)$$

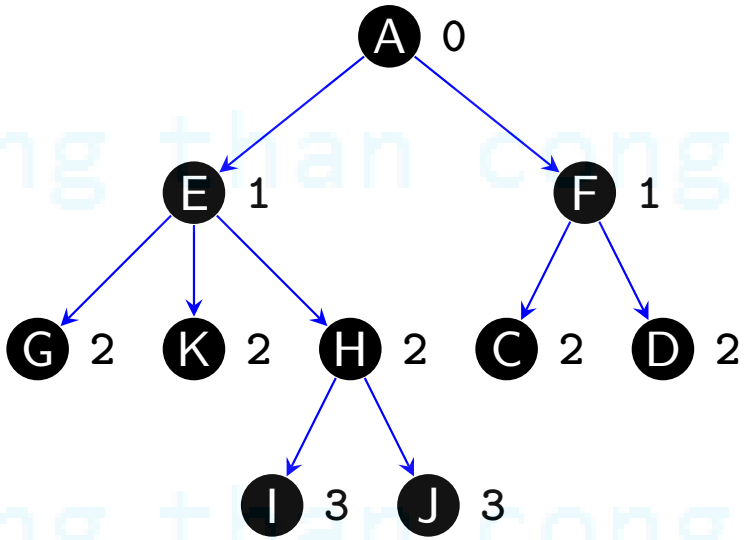


Hình 10: Cây và mức của các nút

Các thuật ngữ liên quan đến cây (cont.)

- **Chiều cao của cây** (*tree height*):

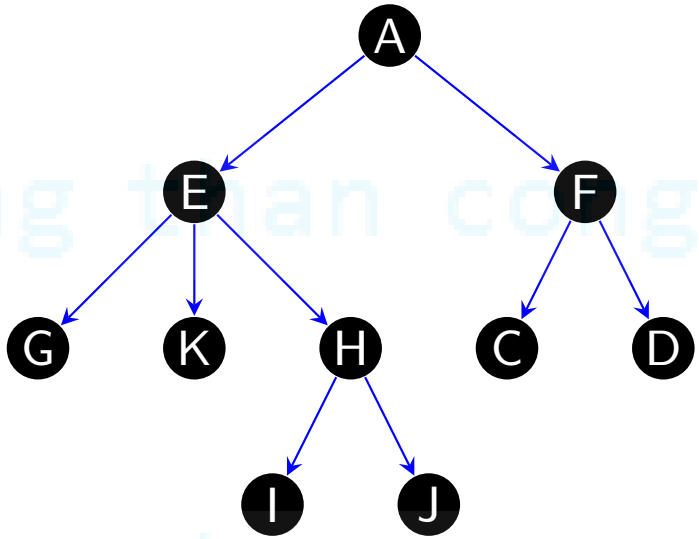
$$height\,(T) = \max\,(level\,(p_i) + 1, p_i \in T) \tag{3}$$



Hình 11: Chiều cao của cây là 4

Các thuật ngữ liên quan đến cây (cont.)

- **Đường đi** (*path*): là một chuỗi các nút khác nhau $\{p_1, p_2, ..., p_k\}$ sao cho giữa p_i, p_{i+1} có cạnh giữa chúng. Nút p_1 gọi nút đầu và p_k là nút cuối của đường đi.

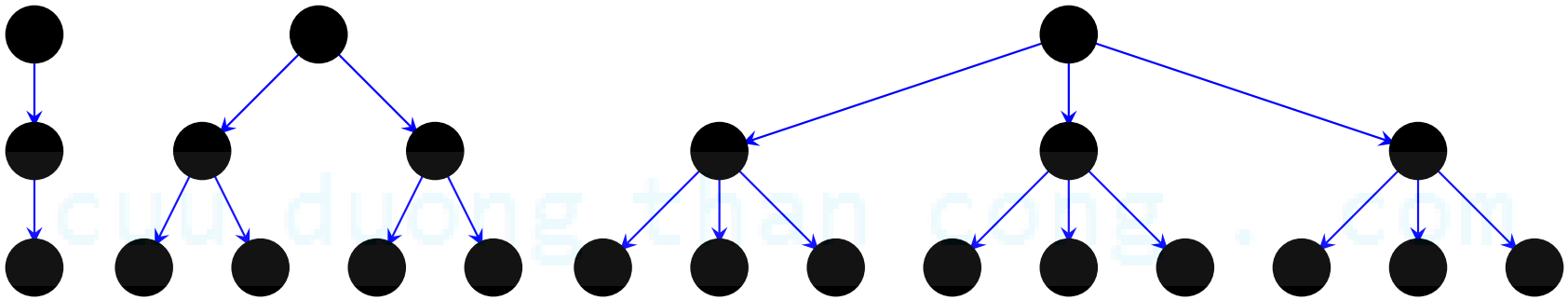


Hình 12: Dãy {A, E, H, I} là đường đi, dãy {A, E, C} không phải là đường đi

Phân loại cây

Định nghĩa 2

- **Cây tuyến tính** (*linear tree*): là cây có bậc bằng 1
- **Cây nhị phân** (*binary tree*): là cây có bậc bằng 2
- **Cây tam phân** (*ternary tree*): là cây có bậc bằng 3
- **Cây n-nhánh** (*n-ary tree*): là cây có bậc bằng n



Hình 13: Các loại cây

Một số loại cây nhị phân

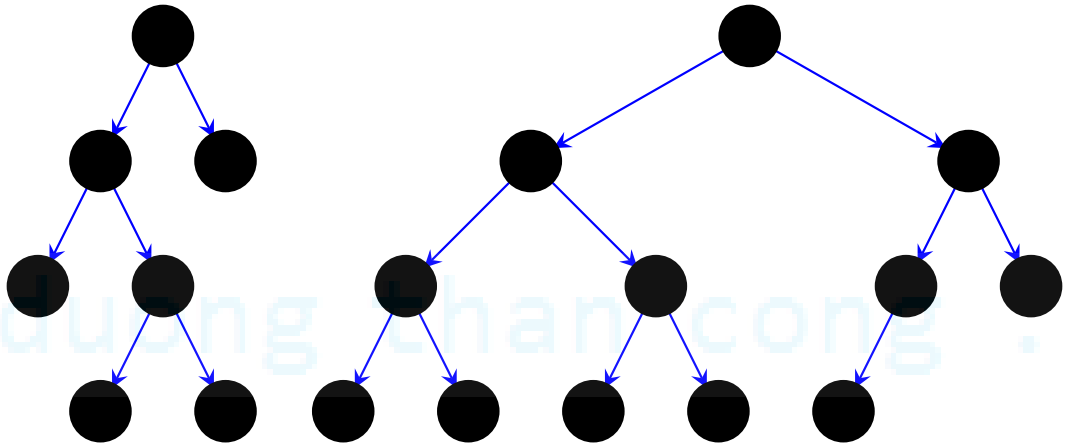
Định nghĩa 3

Một số cây nhị phân đặc biệt

- **Cây nhị phân đầy đủ** (*full binary tree*): là cây mà mỗi nút có 0 hoặc 2 nút con
- **Cây nhị phân hoàn chỉnh** (*complete binary tree*): là cây mà có
  1. Đầy đủ các nút từ mức 0 đến $h-1$ (h là chiều cao của cây)
  2. Riêng mức h thì các nút liên tiếp từ trái sang phải

Một số loại cây nhị phân (cont.)

Hình dưới minh họa cây đầy đủ và cây hoàn chỉnh.



Hình 14: Các loại cây đầy đủ và hoàn chỉnh

Các định lý về cây nhị phân

Định lý 1

1. *Nếu T là cây nhị phân thì sẽ không có quá* 2^k *nút có mức* $k \geq 0$
2. *Nếu T là cây nhị phân có chiều cao là h thì số nút lá tối đa của cây là* 2^{h-1}
3. *Nếu T là cây nhị phân có chiều cao là h thì số nút tối đa của cây là* $2^h - 1$
4. *Nếu T là một cây nhị phân có n nút thì chiều cao nhỏ nhất có thể của cây là là* $\log_2(n+1)$

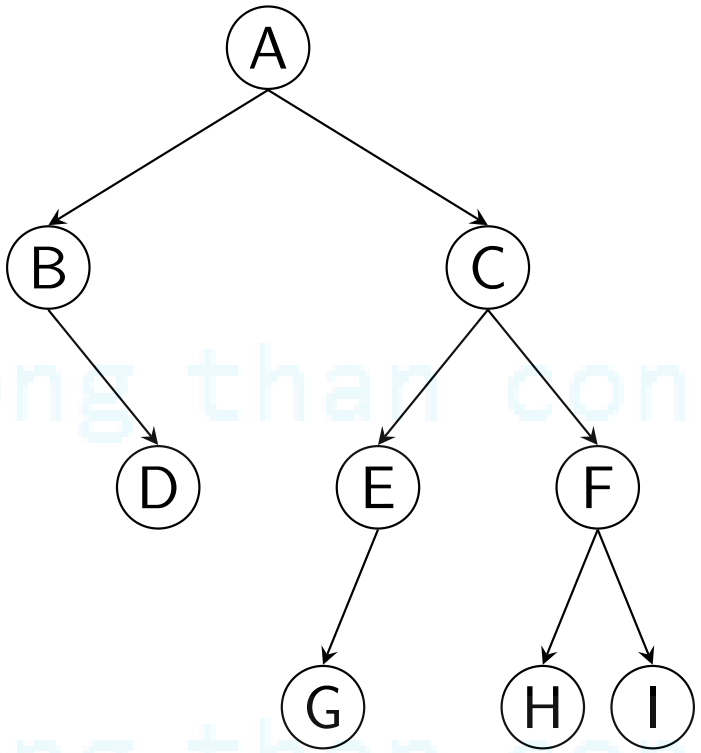
Các định lý về cây nhị phân (cont.)

Định lý 2

Cho T là một cây nhị phân đầy đủ. l là số nút lá và i là số nút nội

$$l = i + 2 \tag{4}$$

Cấu trúc dữ liệu biểu diễn cây



Hình 15: Biểu diễn vẽ cho một cây nhị phân

Cấu trúc dữ liệu biểu diễn cây (cont.)

Biểu diễn cây bằng mảng

Bảng 1: Biểu diễn mảng cho cây nhị phân

Chỉ số	Nút	Con trái	Con phải
0	A	1	2
1	B	-1	3
2	C	4	5
3	D	-1	-1
4	E	6	-1
5	F	7	8
6	G	-1	-1
7	H	-1	-1
8	I	-1	-1

Cấu trúc dữ liệu biểu diễn cây (cont.)

Mỗi nút của cây sẽ chứa một thông tin **định danh** (id) để phân biệt với các nút khác

Chương trình 1: cấu trúc dữ liệu nút

```
1  template <class T>
2  struct Node
3  {
4    T data;
5    int id;
6    Node<T> *left;
7    Node<T> *right;
8  };
```

Cấu trúc dữ liệu biểu diễn cây (cont.)

Chương trình 2: cấu trúc dữ liệu cây nhị phân

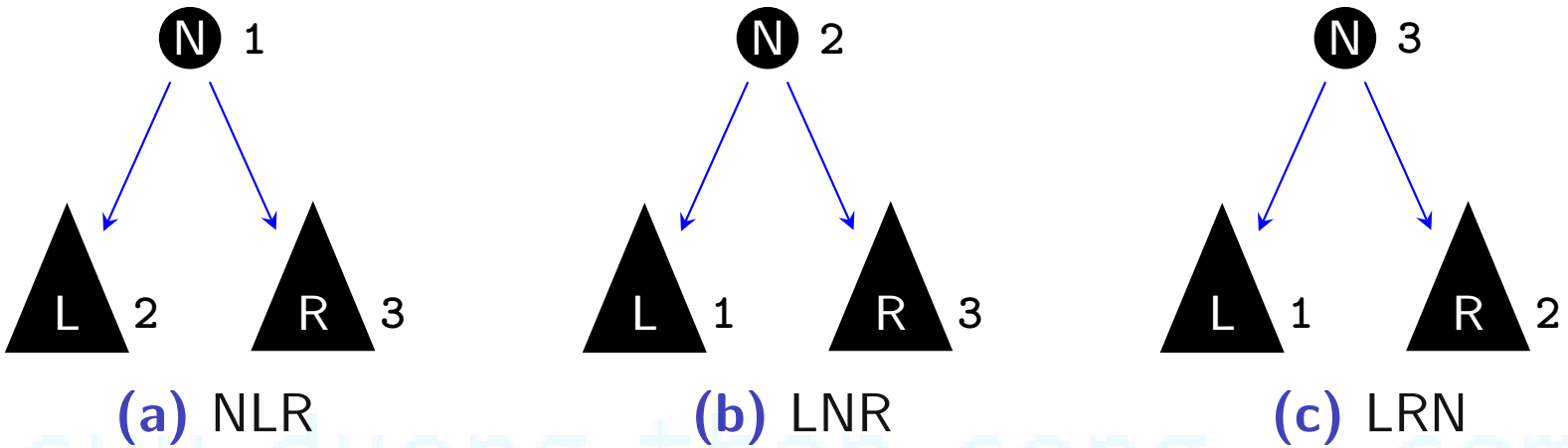
```
1  template <class T>
2  class BinaryTree
3  {
4  private:
5    Node<T> *root;
6
7  public:
8    Node<T> *search(int id);
9  };
```

Duyệt cây nhị phân

Đối với cây ta có các kỹ thuật duyệt cây như sau

- Duyệt các nút của cây theo thứ tự NLR; nghĩa là duyệt nút trước (N), sau đó duyệt cây con trái (L), cuối cùng duyệt cây con phải (R)
- Duyệt các nút của cây theo thứ tự LNR
- Duyệt các nút của cây theo thứ tự LRN

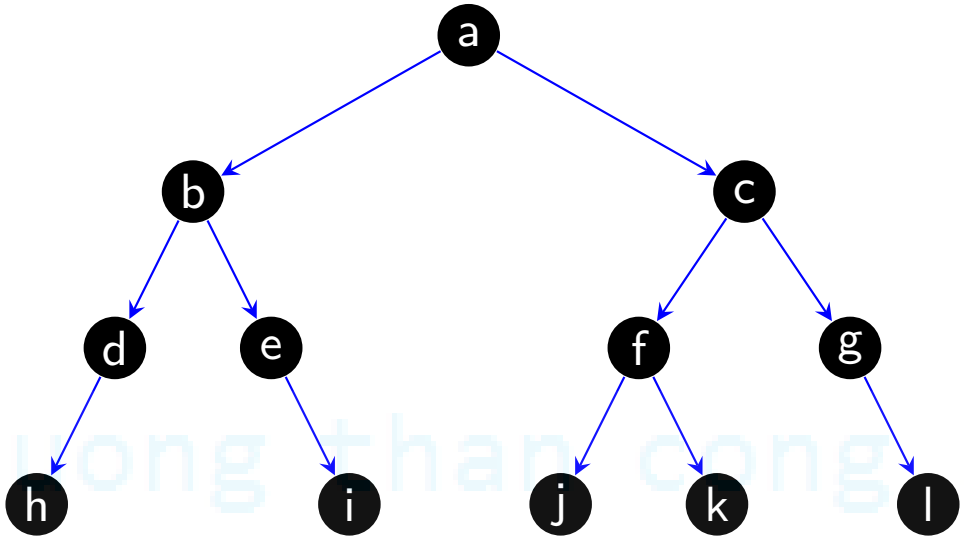
Duyệt cây nhị phân (cont.)



(a) NLR **(b)** LNR **(c)** LRN

Hình 16: Ba kiểu duyệt đặc thù của cây

Duyệt cây nhị phân (cont.)



Hình 17: Duyệt cây bằng 3 cách NLR, LNR, LRN

Duyệt cây nhị phân (cont.)

Chương trình 3: Duyệt cây NLR

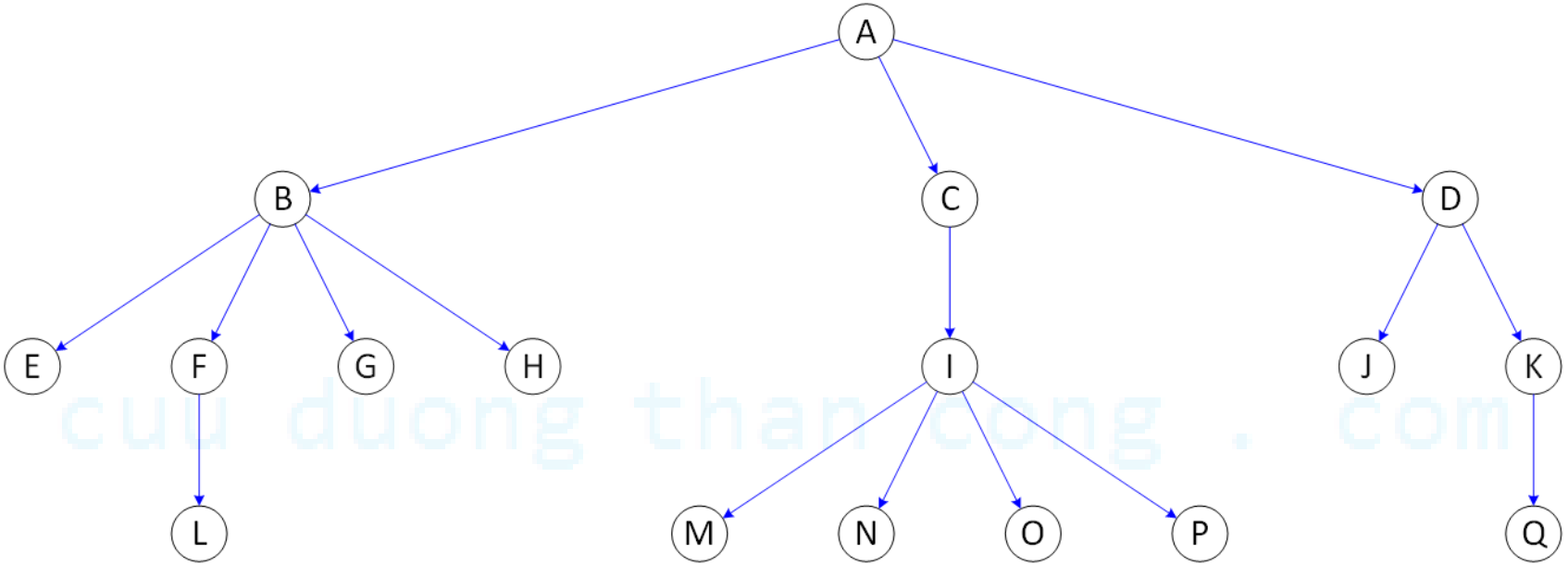
```
1  void NLR(Node<T> *p)
2  {
3    if (p == NULL)
4      return;
5    //do something for node
6    NLR(p->left);
7    NLR(p->right);
8  }
```

Cây n-nhánh

- Một node của cây n-nhánh có thể được biểu bằng tối đa n con trỏ

```
1  template <class T>
2  struct Node
3  {
4    T data;
5    int id;
6    Node<T> *childs[n];
7  };
```

Cây n-nhánh (cont.)



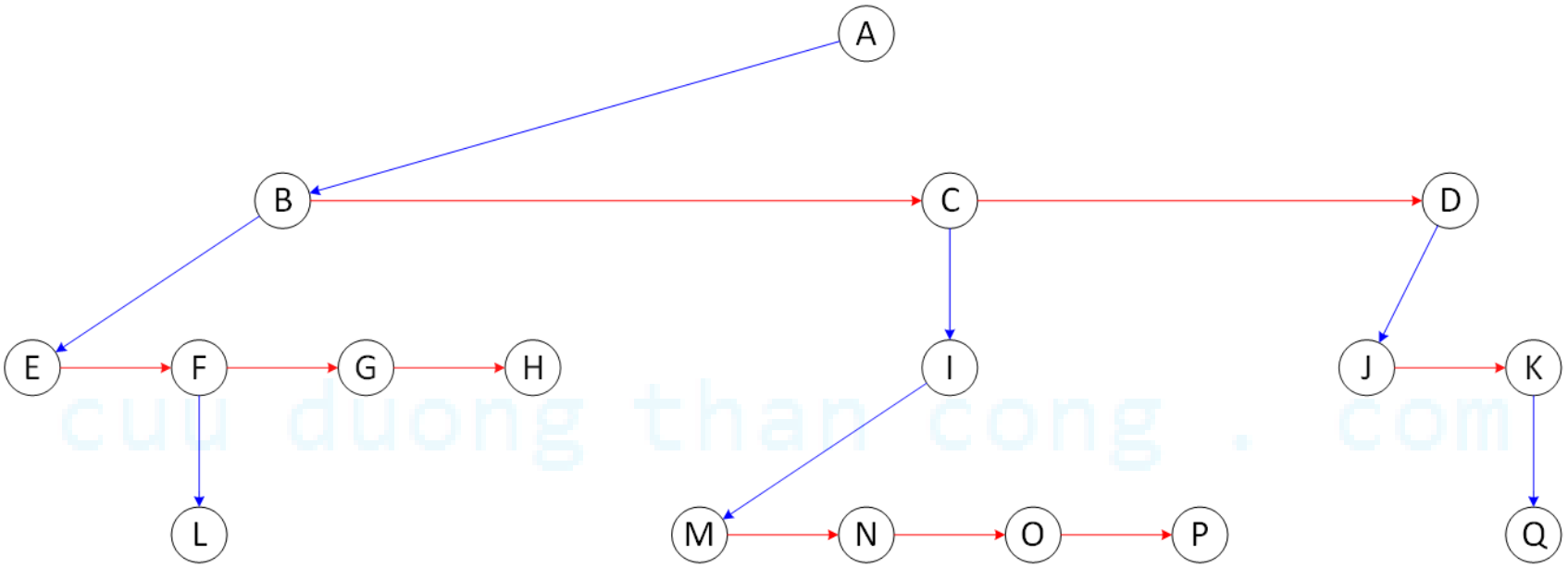
Hình 18: Cây n-nhánh

Cây n-nhánh (cont.)

- Một node của cây n-nhánh có thể được biểu diễn bằng mô hình "anh cả" chỉ cần sử dụng 2 con trỏ

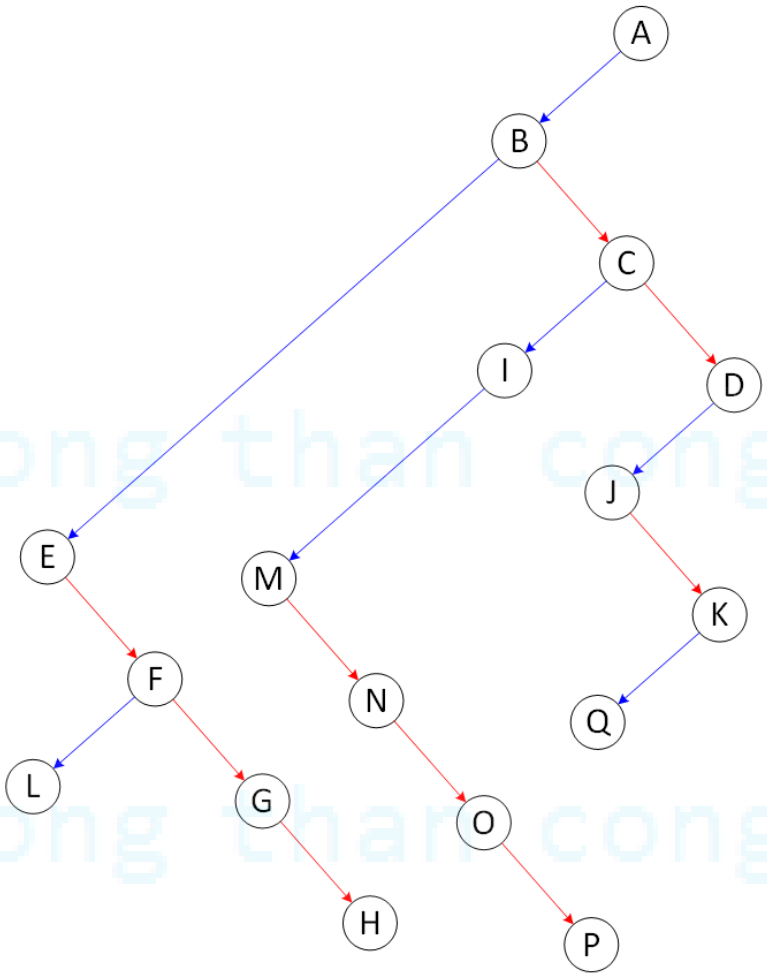
```
1  template <class T>
2  struct Node
3  {
4    T data;
5    int id;
6    Node<T> *nextSibling;
7    Node<T> *eldestChild;
8  };
```

Cây n-nhánh (cont.)



Hình 19: Cây n-nhánh: mũi tên màu xanh trỏ đến con cả, mũi tên màu đỏ trỏ đến em kế tiếp

Cây n-nhánh (cont.)



Hình 20: Mô hình "anh cả" được xoay 45 độ

CÂY NHỊ PHÂN TÌM KIẾM

cuu duong than cong . com

Cây nhị phân tìm kiếm

Định nghĩa 4

Cây nhị phân tìm kiếm T là cây mà mỗi nút của cây có một giá trị **khóa** (key)

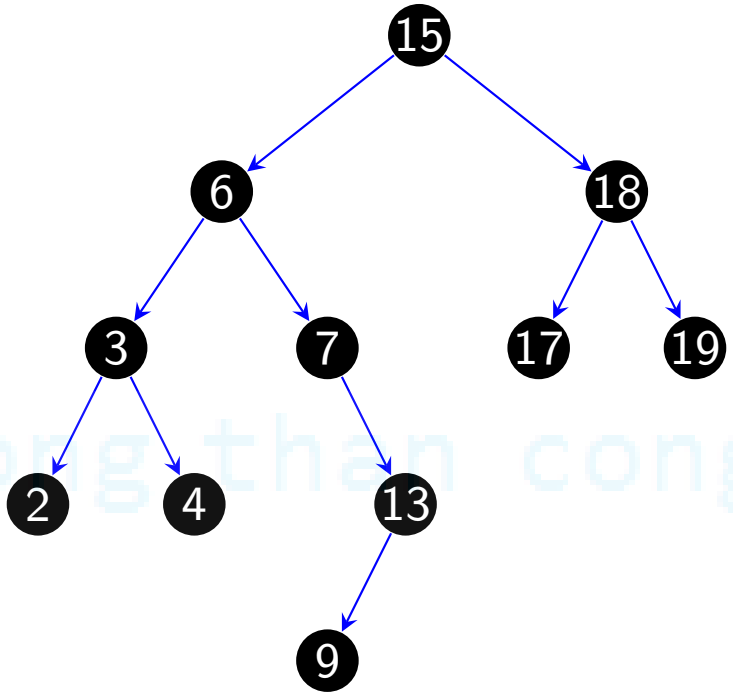
- Tại mỗi nút p
  1. Tất cả các nút của cây con trái đều có khóa nhỏ hơn khóa của p

$$\forall q \in (p \to left) : q \to key < p \to key$$

  2. Tất cả các nút của cây con phải đều có khóa lớn hơn khóa của p

$$\forall q \in (p \to right) : q \to key > p \to key$$

Cây nhị phân tìm kiếm (cont.)



Hình 21: Cây nhị phân tìm kiếm

cuu duong than cong . com

- Hãy xác định các nút có khóa nhỏ nhất và nhỏ nhất của cây
- Hãy xác định các nút có khóa đứng ngay trước và ngay sau nút 15

Cây nhị phân tìm kiếm (cont.)

Chương trình 4: Cấu trúc dữ liệu nút

```
1  template <class T>
2  struct BSTNode
3  {
4      T data;
5      int key;
6      Node<T> *left;
7      Node<T> *right;
8  };
```

Cây nhị phân tìm kiếm (cont.)

Trong nhiều tình huống, người lập trình có thể bổ sung thêm thông tin nút cha

Chương trình 5: Cấu trúc dữ liệu nút

```
1  template <class T>
2  struct BSTNode
3  {
4      T data;
5      int key;
6      Node<T> *left;
7      Node<T> *right;
8      Node<T> *parent;
9  };
```

Cây nhị phân tìm kiếm (cont.)

Chương trình 6: Cấu trúc dữ liệu cây nhị phân tìm kiếm

```
1  template <class T>
2  class BSTTree
3  {
4    private:
5      BSTNode<T> *root;
6
7    public:
8      BSTNode<T> *search(int key);
9      bool insert(int key, T data);
10     bool remove(int key);
11 };
```

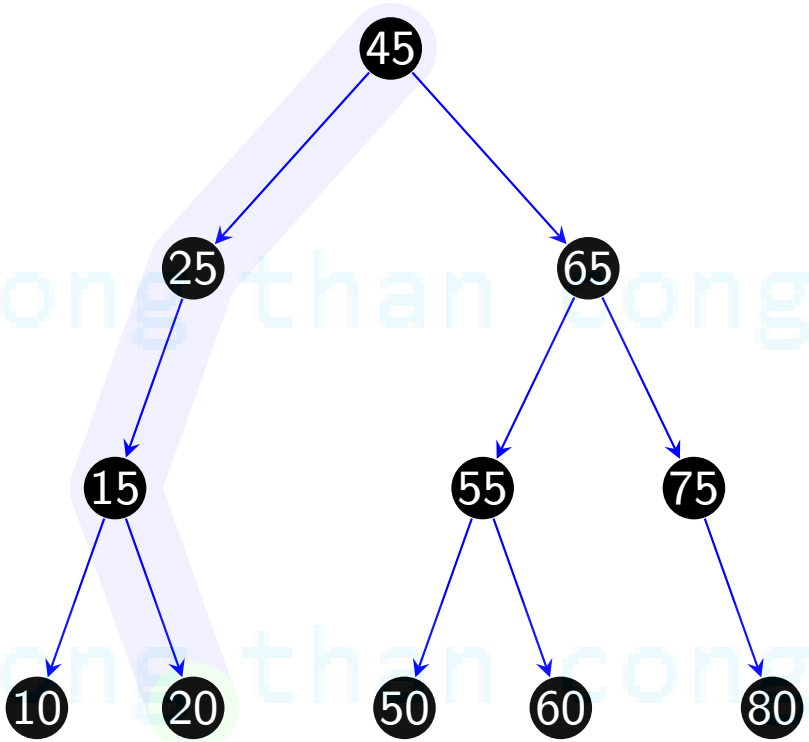
Tìm kiếm trên cây nhị phân tìm kiếm

Chương trình 7: Tìm kiếm khóa trên cây nhị phân tìm kiếm

```
1  BSTNode<T>* search(int key)
2  {
3      BSTNode<T> *p = root;
4      while (p) {
5          if (p->key==key) return p;
6          else if (p->key > key)
7              p = p->left;
8          else
9              p = p->right;
10     }
11     return NULL;
12 }
```

Minh họa tìm kiếm trên cây nhị phân tìm kiếm

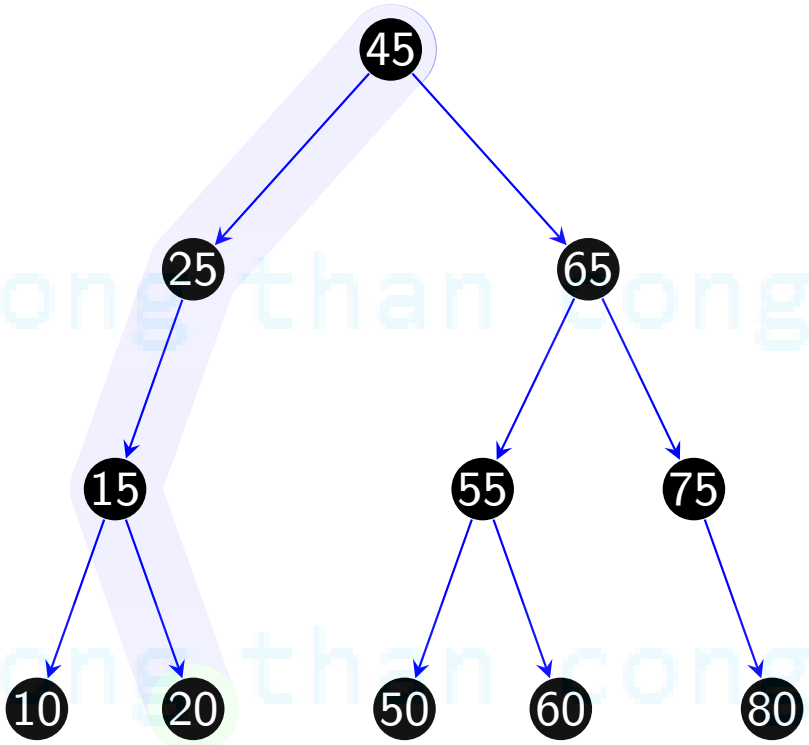
Tìm khóa 20



Hình 22: Cây nhị phân tìm kiếm

Minh họa tìm kiếm trên cây nhị phân tìm kiếm

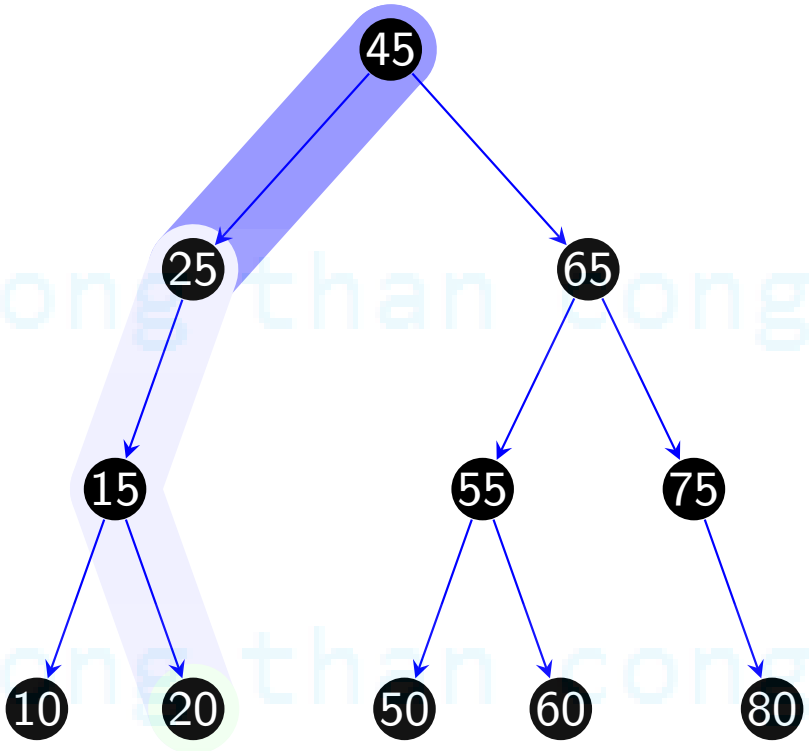
Tìm khóa 20



Hình 22: Cây nhị phân tìm kiếm

Minh họa tìm kiếm trên cây nhị phân tìm kiếm

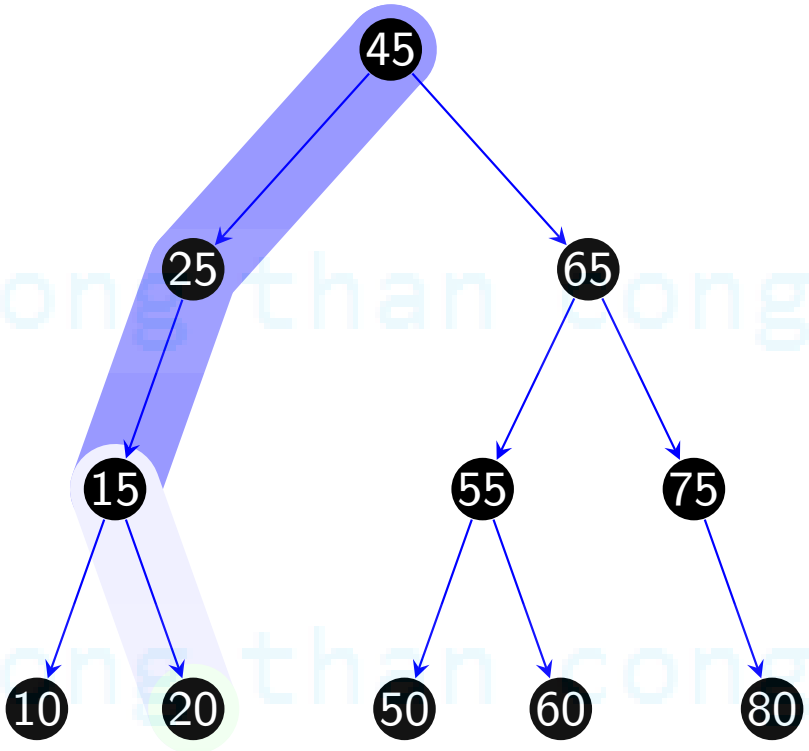
Tìm khóa 20



Hình 22: Cây nhị phân tìm kiếm

Minh họa tìm kiếm trên cây nhị phân tìm kiếm

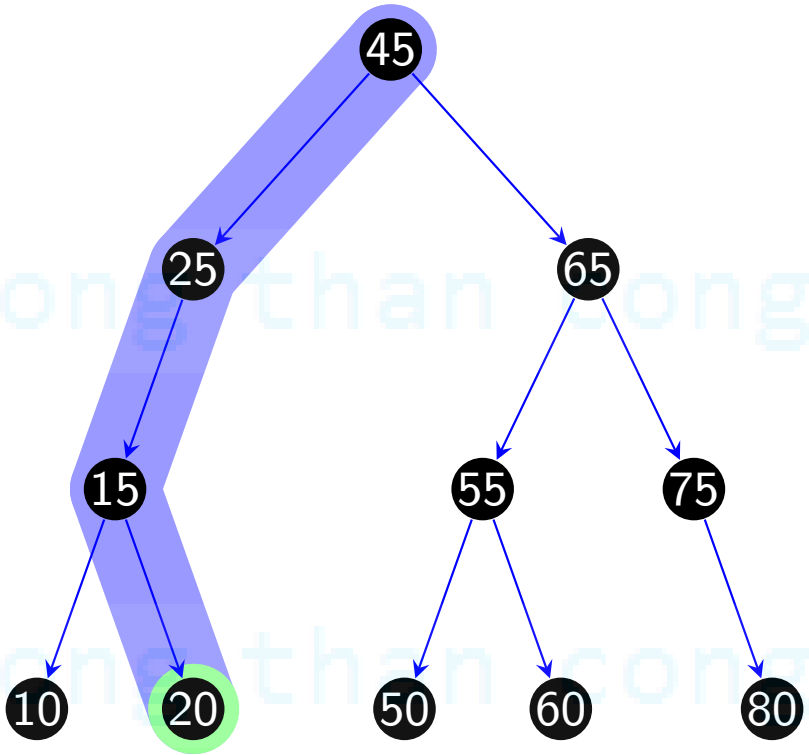
Tìm khóa 20



Hình 22: Cây nhị phân tìm kiếm

Minh họa tìm kiếm trên cây nhị phân tìm kiếm

Tìm khóa 20



Hình 22: Cây nhị phân tìm kiếm

Thêm một nút vào cây nhị phân tìm kiếm

Ý tưởng

Cho một cây T và một nút có khóa là *key*

- Tìm xem khóa *key* có tồn tại hay chưa
- Nếu tồn tại rồi thì dừng lại
- Nếu không tồn tại thì vị trí của nút lá cuối cùng sẽ là vị trí cần thêm vào

Thêm một nút vào cây nhị phân tìm kiếm (cont.)

Thêm khóa *key* và *data* vào cây nhị phân tìm kiếm

```
1  void insert(int key, T data)
2  {
3      BSTNode<T> *p = root;
4      while (p)
5      {
6          if (p->key == key)
7              return;
8          if (p->key > key)
9          {
10             if (p->left == null)
11             {
12                 p->left = new BSTNode(key, data);
13                 return;
14             }
15             p = p->left;
```

Thêm một nút vào cây nhị phân tìm kiếm (cont.)

```
16                  break;
17              }
18              if (p->key < key)
19              {
20                  if (p->right == null)
21                  {
22                      p->right = new BSTNode(key, data);
23                      return;
24                  }
25                  p = p->right;
26                  break;
27              }
28          }
29      }
```

Minh họa thêm nút

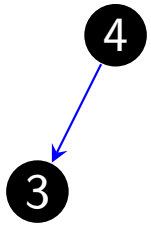
Ví dụ 1

Tạo một cây nhị phân tìm kiếm từ dãy số sau {4, 3, 5, 1, 2, 7, 9, 8}
Cây được khởi tạo là rỗng

Minh họa thêm nút (cont.)

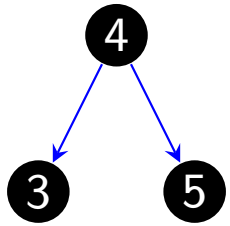
Hình 23: Thêm 4

Minh họa thêm nút (cont.)



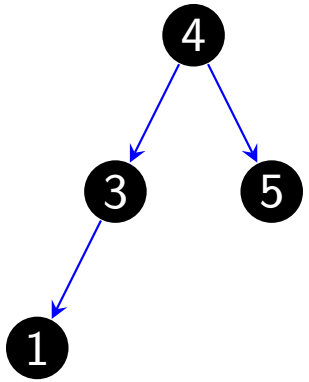
Hình 24: Thêm 3

Minh họa thêm nút (cont.)



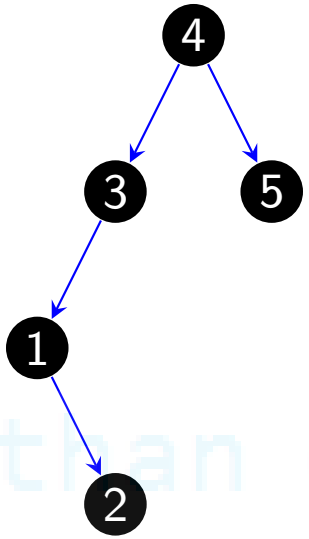
Hình 25: Thêm 5

Minh họa thêm nút (cont.)



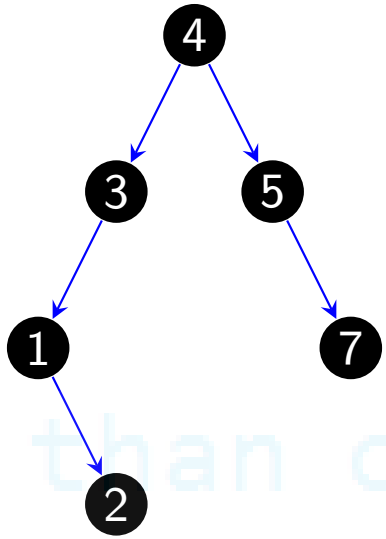
Hình 26: Thêm 1

Minh họa thêm nút (cont.)



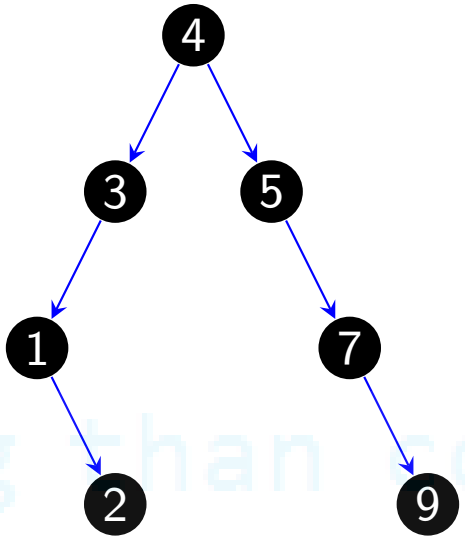
Hình 27: Thêm 2

Minh họa thêm nút (cont.)



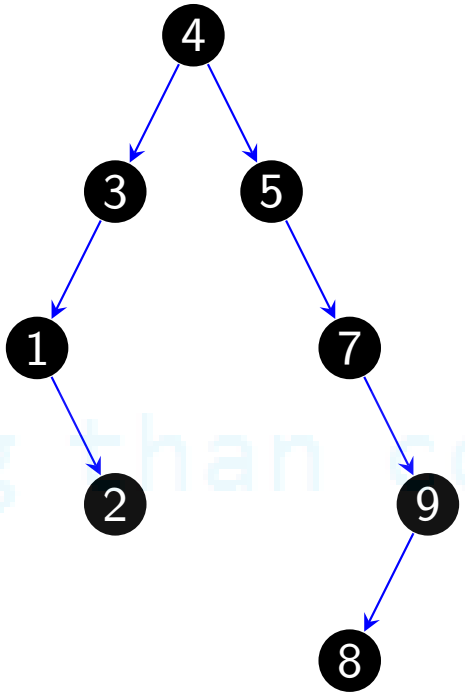
Hình 28: Thêm 7

Minh họa thêm nút (cont.)



Hình 29: Thêm 9

Minh họa thêm nút (cont.)



Hình 30: Thêm 8

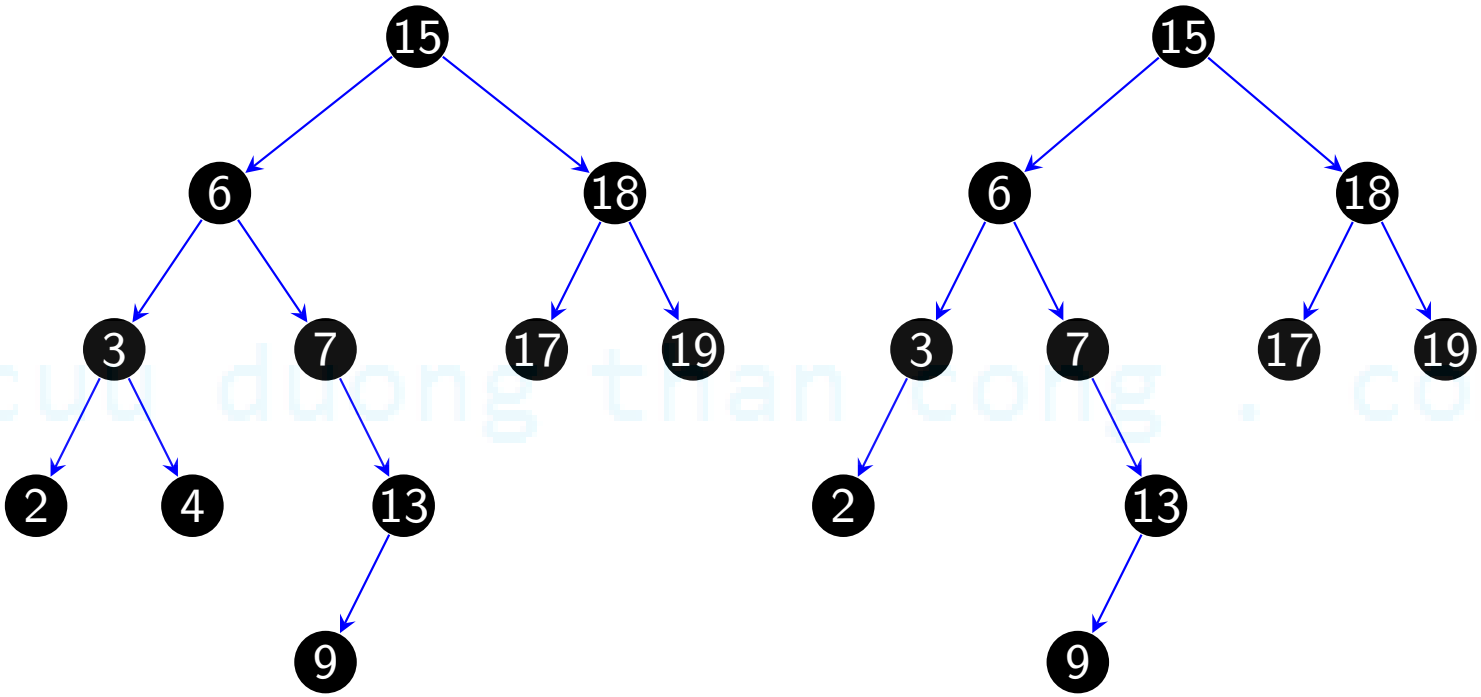
Xóa một nút khỏi cây nhị phân tìm kiếm

Các trường hợp

Có hai trường hợp xóa một nút của cây nhị phân tìm kiếm

- Xóa một nút lá: đơn giản
- Xóa một nút không phải lá: tìm phần tử thay thế
  - Phần tử lớn nhất bên cây con trái
    - Hoặc, phần tử nhỏ nhất bên cây con phải

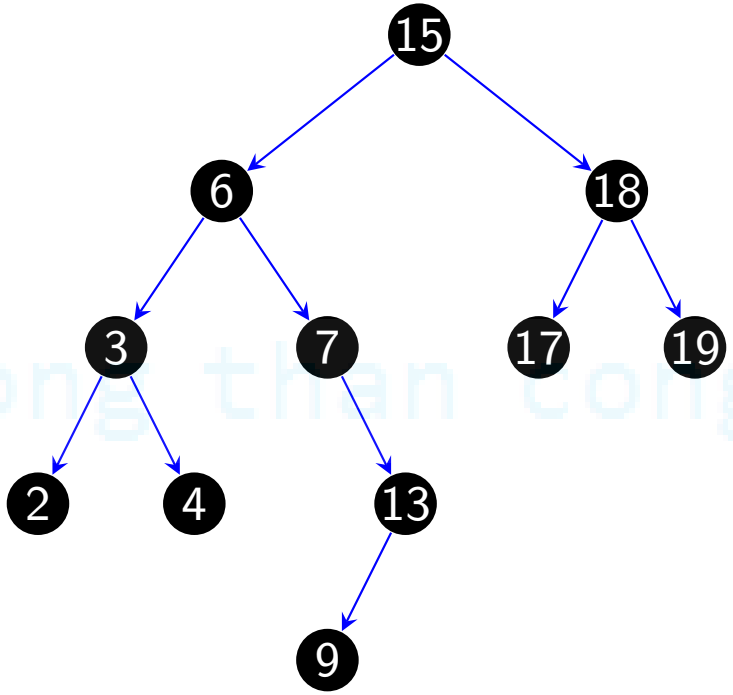
Minh họa xóa một nút lá



(a) cây trước khi xóa **(b)** cây sau khi xóa

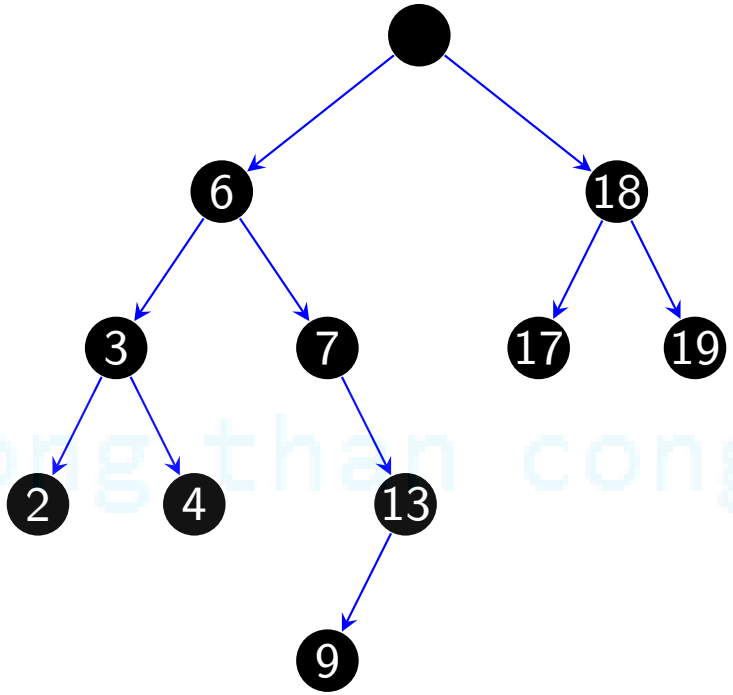
Hình 31: Xóa nút 4 khỏi cây

Minh họa xóa một nút không phải lá



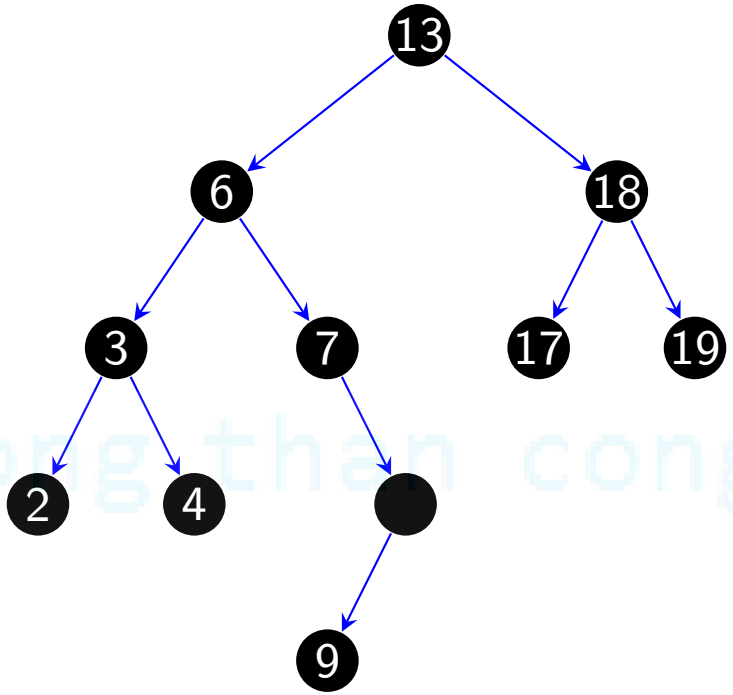
Hình 32: Hãy xóa nút 15 của cây

Minh họa xóa một nút không phải lá (cont.)



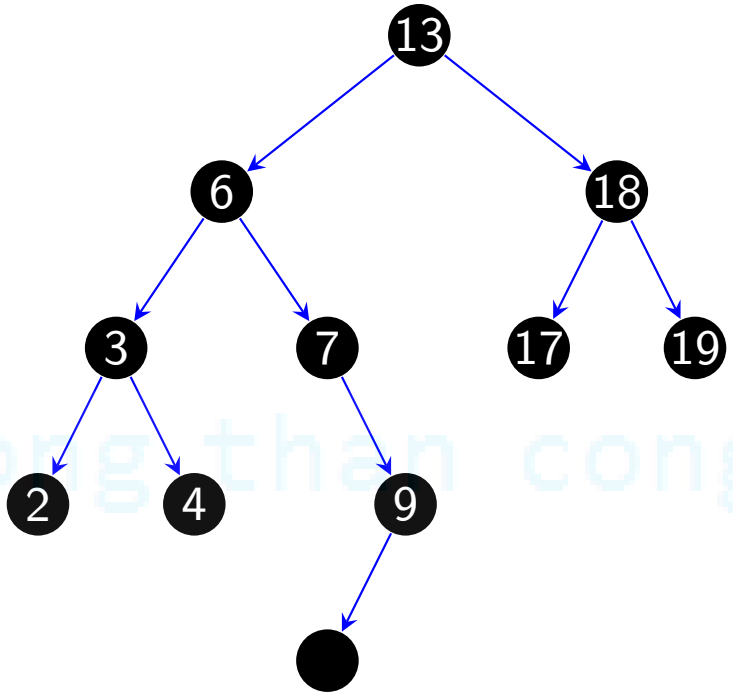
Hình 33: Xóa nút 15

Minh họa xóa một nút không phải lá (cont.)



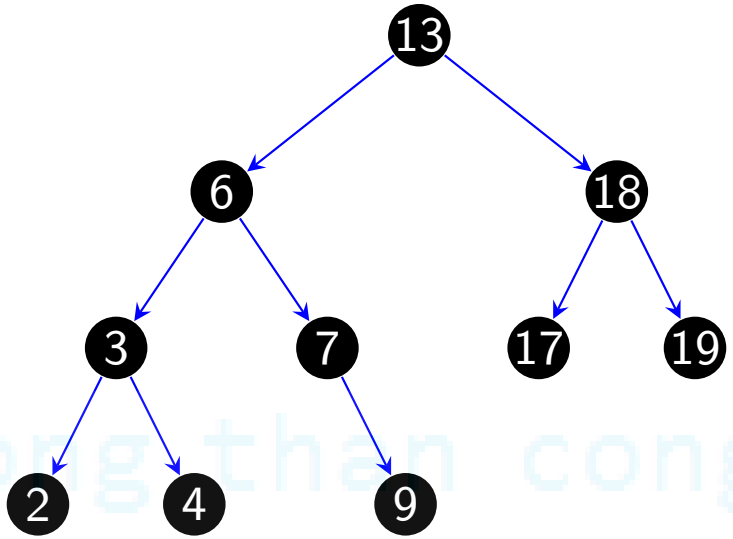
Hình 34: Nút 13 thế chỗ 15

Minh họa xóa một nút không phải lá (cont.)



Hình 35: Phần tử 9 thế chỗ 13

Minh họa xóa một nút không phải lá (cont.)



Hình 36: Xóa nút lá

Bài luyện tập

Ví dụ 2

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

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

Đánh giá về cây nhị phân tìm kiếm

Phân tích chi phí thực hiện theo h (chiều cao của cây)

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

Phân tích chi phí bộ nhớ theo n (số lượng nút của cây)

Tài liệu tham khảo

cuu duong than cong . com