

CANTHO UNIVERSITY 

CHƯƠNG 4


CÂY AVL

(G.M. Adelson-Velsky và E.M. Landis)

Bộ môn Công Nghệ Phần Mềm

Phân dành cho đơn vị www.ctu.edu.vn

1

 **Nội dung**

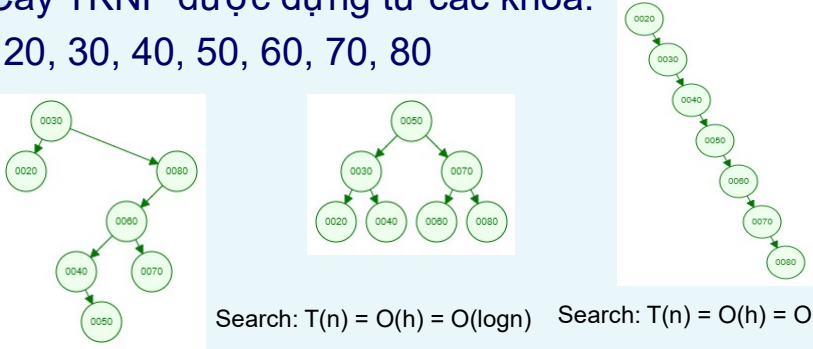
- Khái niệm về cây tìm kiếm nhị phân cân bằng
- Khái niệm về cây AVL
- Các thuật toán trên cây AVL
- Ý tưởng cài đặt cây bằng con trỏ

www.ctu.edu.vn

2

Khái niệm về cây tìm kiếm nhị phân cân bằng

• Cây TKNP được dựng từ các khóa:
20, 30, 40, 50, 60, 70, 80



Search: $T(n) = O(h) = O(\log n)$ Search: $T(n) = O(h) = O(n)$

• Nhận xét: Độ phức tạp trong các giải thuật trên cây tìm kiếm nhị phân trường hợp xấu nhất là $O(n)$ và trung bình $O(\log n)$.

www.ctu.edu.vn

3

Khái niệm về cây tìm kiếm nhị phân cân bằng

➤ Quá trình tìm kiếm khóa, thêm nút, xóa nút trên cây TKNP là quá trình di chuyển từ nút gốc ra nút lá. ➔ Cây càng cao thì giải thuật càng kém hiệu quả.

➤ Do vậy rất cần thiết phải xây dựng cây TKNP mà trong đó chiều cao của cây càng nhỏ càng tốt để cho các giải thuật được hiệu quả nhất. Cây ở dạng này được gọi là cây cân bằng.

www.ctu.edu.vn

4

Cây TKNP cân bằng hoàn toàn

- Cây TKNP **cân bằng hoàn toàn** là cây TKNP mà tại mỗi nút có **tổng số nút của cây con trái và con phải lệch nhau không quá một**.
- Cây nào là cây cân bằng hoàn toàn?

www.ctu.edu.vn

5

Cây TKNP cân bằng tương đối (cân bằng về chiều cao)

- Cây TKNP **cân bằng về chiều cao** là cây TKNP mà trong đó mỗi nút đều có chiều cao con trái và con phải lệch nhau tối đa là 1.
- Cây cân bằng về chiều cao còn gọi là **cân bằng tương đối**.
- Ví dụ:

www.ctu.edu.vn

6



CANTHO UNIVERSITY

Khái niệm về cây AVL

- Cây AVL được gọi theo tên của hai người đề xuất chúng, G.M. Adelson-Velsky và E.M. Landis, được công bố trong bài báo của họ vào năm 1962: "An algorithm for the organization of information." (Một thuật toán về tổ chức thông tin)
- **Cây AVL là cây TKNP mà chiều cao của hai cây con của mọi nút chênh lệch tối đa là 1.**

www.ctu.edu.vn

7


Khái niệm về cây AVL

- Là BST, trong đó sự khác biệt giữa chiều cao của con trái và con phải của bất kỳ nút nào không vượt quá 1.
- Hệ số cân bằng (Balanced factor):
 - $BF(p) = \text{height}(\text{left}(p)) - \text{height}(\text{right}(p))$
- Chiều cao của một nút là độ dài của đường dẫn từ nút đó đến nút rời xa nhất.
 - Nút **NULL** có chiều cao -1
 - Nút lá có chiều cao 0
 - Chiều cao nút p:

$$\text{height}(p) = 1 + \text{MAX}(\text{height}(\text{left}(p)), \text{height}(\text{right}(p)))$$
- Một nút được gọi là cân bằng chiều cao nếu hệ số cân bằng của nó là -1, 0 hoặc 1

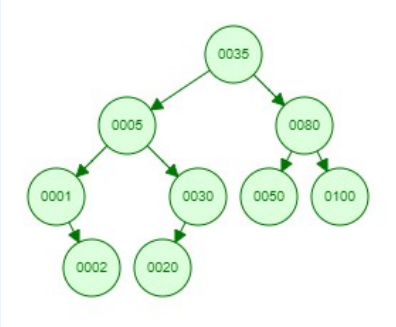
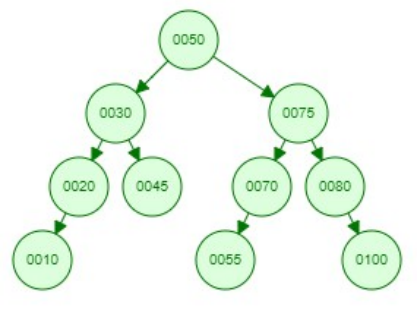
8

8




Khái niệm về cây AVL

- Ví dụ:

www.ctu.edu.vn

9




Các thao tác trên cây AVL

- Cây AVL là cây tìm kiếm nhị phân nên các thao tác cơ bản trên cây AVL tương tự như các thao tác cơ bản trên cây TKNP.
- Trong đó, thao tác thêm và xóa nút trên cây có thể làm cây mất cân bằng nên phải có thao tác cân bằng lại cây sau khi thực hiện thêm hoặc xóa nút.

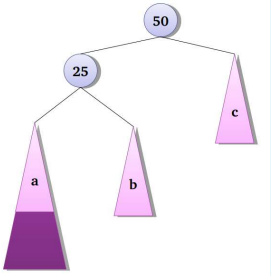
www.ctu.edu.vn

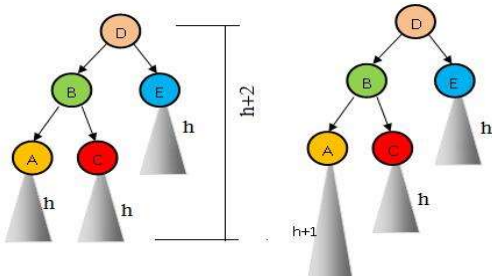
10



Các trường hợp cây mất cân bằng

- Trường hợp 1: Cây mất cân bằng bên trái của con trái (L-L)






1. Cây lệch trái trước khi chèn. Chiều cao cây là $h+2$,

2. Chèn một phần tử vào cây con trái của cây con trái làm cho cây mất cân bằng AVL

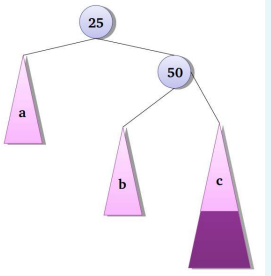
www.ctu.edu.vn

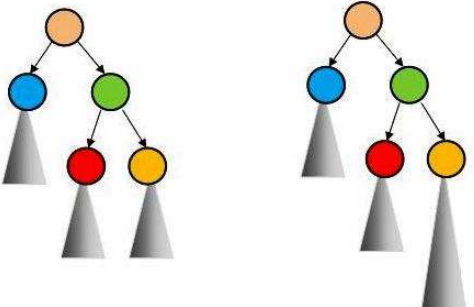
11



Các trường hợp cây mất cân bằng

- Trường hợp 2: Cây mất cân bằng bên phải của con phải (R-R)






1. Cây lệch phải

2. Sau khi chèn, cây con phải lệch phải

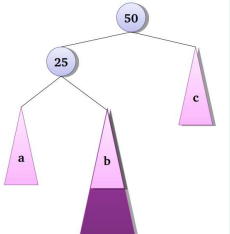
www.ctu.edu.vn

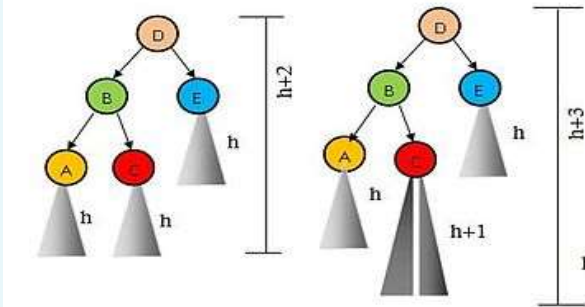
12



Các trường hợp cây mất cân bằng

- Trường hợp 3: Cây mất cân bằng bên phải của con trái (R-L)






1. Cây ban đầu lệch trái, cây con trái cân bằng. Chiều cao cây là $h+2$

2. Phép chèn vào cây con trái làm cho cây con trái lệch phải.

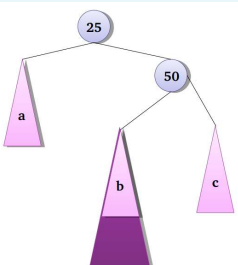
www.ctu.edu.vn

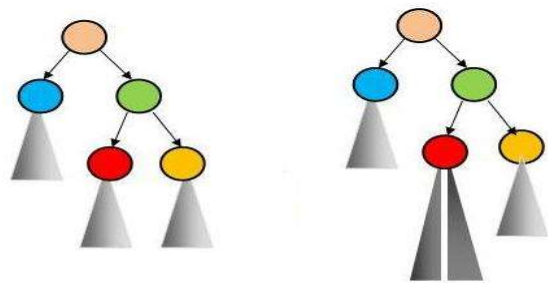
13



Các trường hợp cây mất cân bằng

- Trường hợp 4: Cây mất cân bằng bên trái của con phải (L-R)





1. Cây AVL lệch phải

2. Sau phép chèn vào cây con phải, cây con phải lệch trái.

www.ctu.edu.vn

14

CÁC QUY TẮC XỬ LÝ KHI CÂY MẤT CÂN BẰNG

• Đối với trường hợp 1 (L-L) ta thực hiện quay đơn qua phải (right rotate) như sau:

1. Cây lệch trái trước khi chèn. Chiều cao cây là $h+2$,

2. Chèn một phần tử vào cây con trái của cây con trái làm cho cây mất cân bằng AVL

3. Sau phép quay phải cây trở thành cân bằng, chiều cao cây vẫn là $h+2$

www.ctu.edu.vn

15

Steps for Right rotation

```

rightRotate(node):
    temp = node->left
    node->left = temp->right
    temp->right = node
    calculateHeight(node)
    calculateHeight(temp)
    return temp

```

$T(n) = O(1)$

16

16

CÁC QUY TẮC XỬ LÝ KHI CÂY MẤT CÂN BẰNG

• Trường hợp 2 (R-R): Ta thực hiện quay đơn sang trái (left rotate) như sau:

1. Cây lệch phải

2. Sau khi chèn, cây con phải lệch phải

3. Sau phép quay trái cây trở thành cân bằng AVL

www.ctu.edu.vn

17

Steps of left rotation

```

leftRotate(node):
    temp = node->right
    node->right = temp->left
    temp->left = node
    calculateHeight(node)
    calculateHeight(temp)
    return temp

```

$T(n) = O(1)$

18

18

CÁC QUY TẮC XỬ LÝ KHI CÂY MẮT CÂN BẰNG

Trường hợp 3 (R-L): Cây mất cân bằng bên phải của con trái ta thực hiện xoay kép trái - phải (left-right rotate) như sau:

1. Cây ban đầu lệch trái, cây con trái cân bằng. Chiều cao cây là $h+2$
2. Phép chèn vào cây con trái làm cho cây con trái lệch phải.
3. Phép xoay trái cây con trái đưa cây con trái thành cây lệch trái
4. Sau phép xoay phải cây, cây trở thành cân bằng AVL. Chiều cao vẫn là $h+2$

www.ctu.edu.vn

19

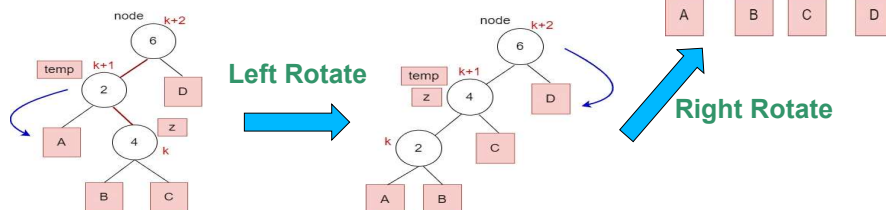
Steps for Left Right rotation

```

leftRightRotate(node):
    node->Left = leftRotate(node->left)
    return rightRotate(node)

```

$$T(n) = O(1)$$



20

20

CÁC QUY TẮC XỬ LÝ KHI CÂY MẤT CÂN BẰNG

Trường hợp 4 (L-R): Cây mất cân bằng bên trái của con phải. Trường hợp này ta thực hiện quay kép phải-trái (right - left rotate) như sau:

1. Cây AVL lệch phải
2. Sau phép chèn vào cây con phải, cây con phải lệch trái.
3. Sau phép quay phải cây con phải. Cây con phải trở thành lệch phải
4. Sau phép quay trái, cây trở thành cân bằng AVL.

www.ctu.edu.vn

21

Steps for Right Left rotation

```

rightleftRotate(node):
    node->right = rightRotate(node->right)
    return leftRotate(node)
  
```

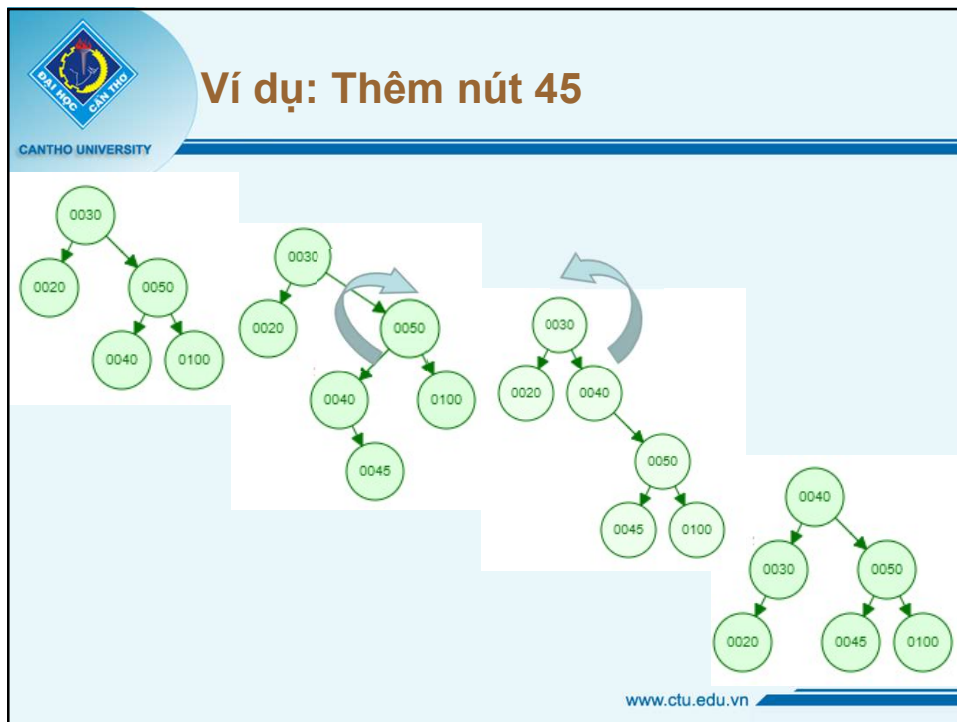
$T(n) = O(1)$

Right Rotate

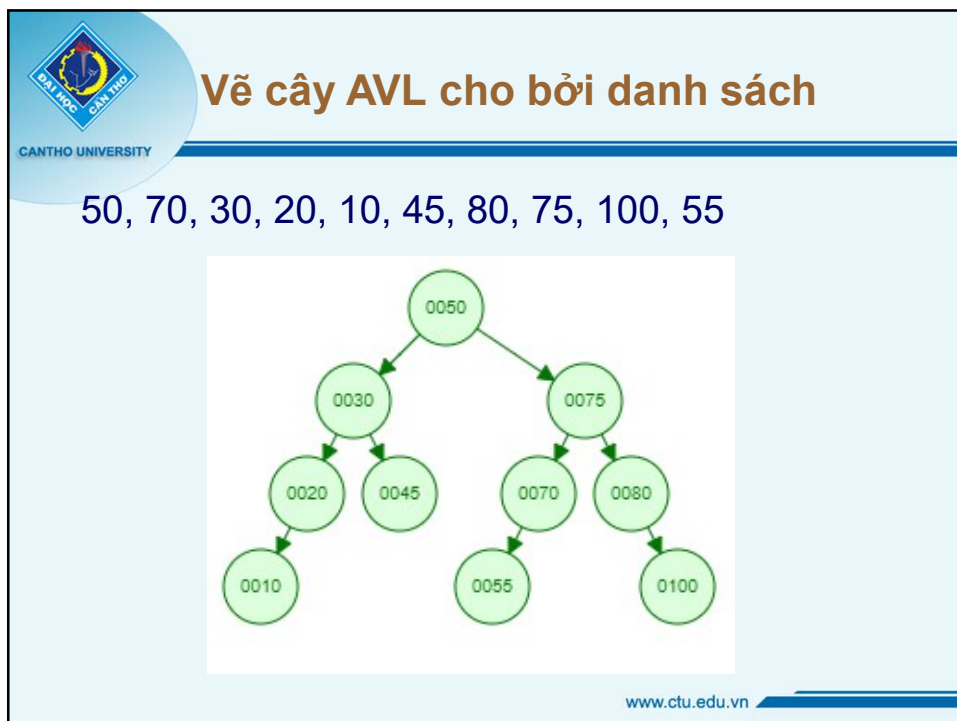
Left Rotate

22

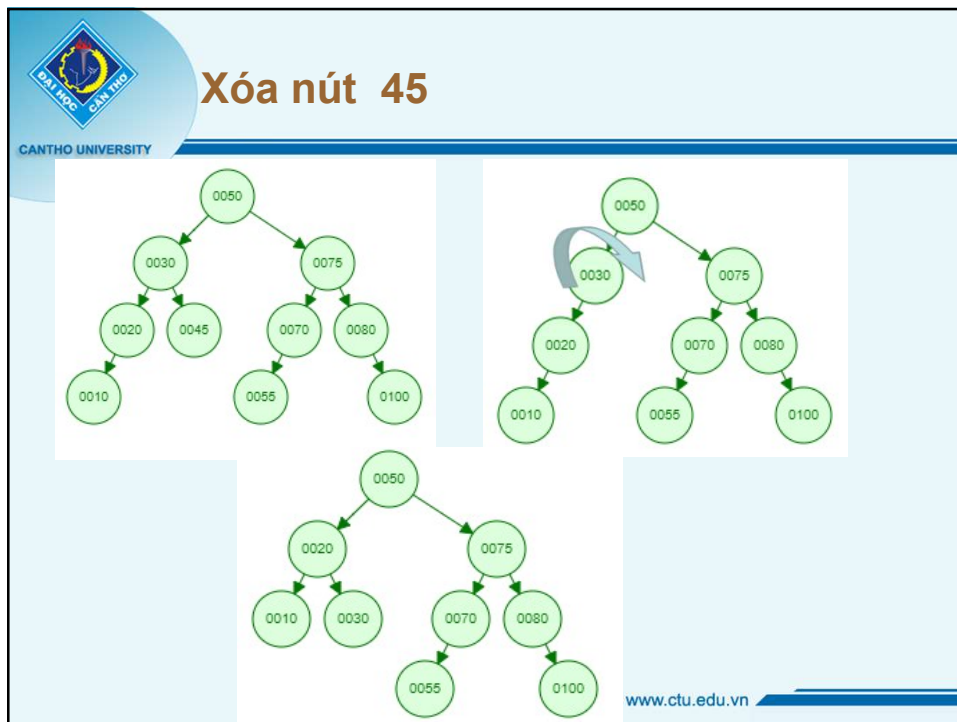
22



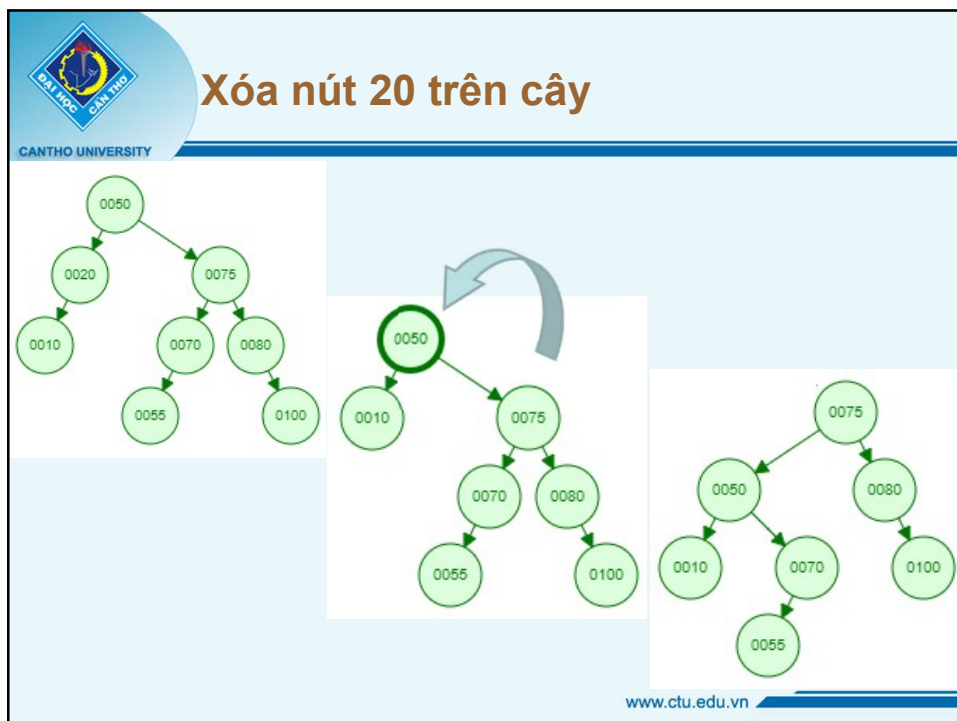
23



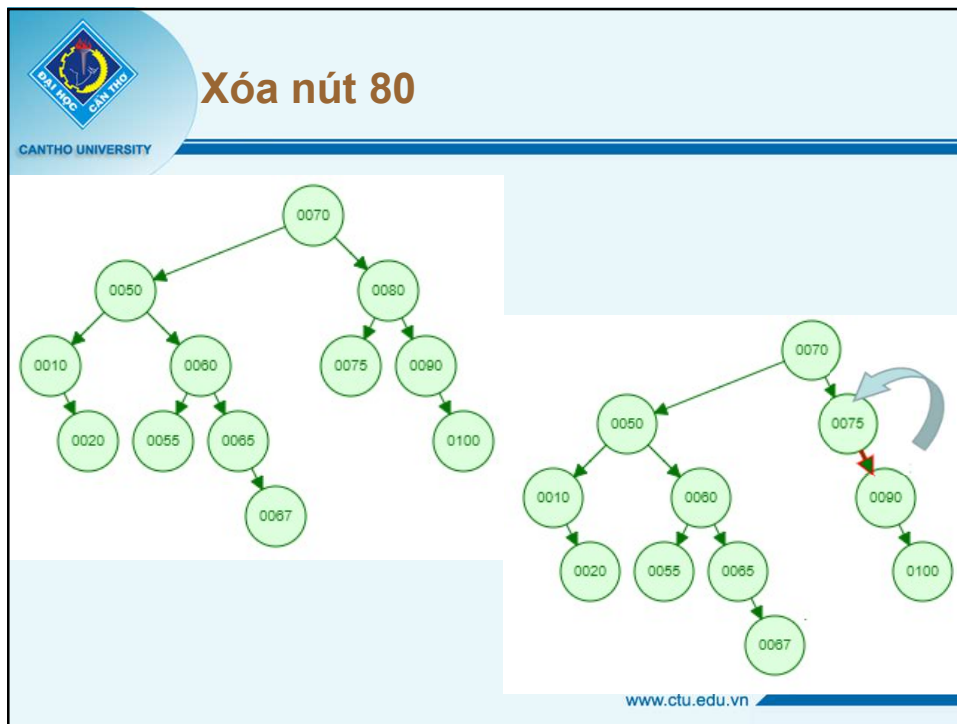
24



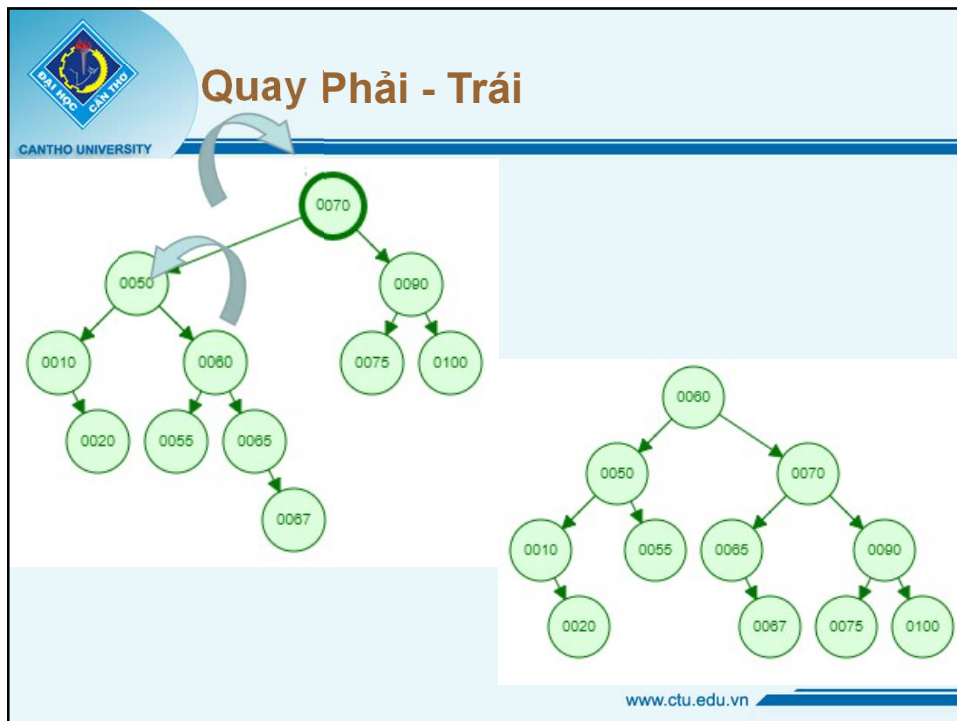
25



26



27



28



CANTHO UNIVERSITY

Ý tưởng cài đặt cây

Mỗi nút trên cây cần có các thông tin:

- Khóa của nút
- Cây con trái
- Cây con phải
- Chỉ số cân bằng với quy như sau:

Chỉ số là EH (=0) nếu cây cân bằng

Chỉ số là LH (=1) nếu cây lệch trái

Chỉ số là RH (=-1) nếu cây lệch phải

www.ctu.edu.vn

29



CANTHO UNIVERSITY

Khai báo cây AVL

```
typedef ... KeyType;
typedef struct Node
{
    KeyType Key;
    int Bal;
    Node *Left;
    Node *Right;
};
typedef Node * AVLTree;
```

www.ctu.edu.vn

30



CANTHO UNIVERSITY

Thêm nút vào cây AVL

- Thêm nút vào AVL tương tự như thêm nút vào cây TKNP thông thường.
- Sau đó ta xét xem cây có bị mất cân bằng hay không để thực hiện cân bằng lại cây theo các quy tắc xoay và cập nhật lại chỉ số cân bằng của các nút.

www.ctu.edu.vn

31

Insert to an AVL tree

- In all cases, after rotation at z , the whole tree becomes balanced.

```


insertNode(node, k):
    BSTInsert(node, k)
    calculateHeight(node)
    balance = getBalance(node)
    if (balance > 1):
        if (k < node->left->key): return rightRotate(node)
        else: return leftRightRotate(node)
    if (balance < -1):
        if (k > node->right->key): return leftRotate(node)
        else: return rightLeftRotate(node)
    return node

```

$T(n) = O(\log n)$

32

32



Xóa nút ra khỏi cây AVL

- Xóa nút ra khỏi AVL tương tự như **xóa nút ra khỏi cây TKNP** thông thường. Sau đó ta xét xem cây có bị mất cân bằng hay không để thực hiện cân bằng lại cây theo các quy tắc **xoay** và cập nhật lại chỉ số cân bằng của các nút.
- Quá trình xóa nút có thể dẫn đến cây mất cân bằng dây chuyền liên tiếp đến các nút tiền bối của nó. Theo đó, nguyên tắc thực hiện là phải cân bằng từ nút con trước rồi đến lên nút cha.

www.ctu.edu.vn

33



Tài liệu tham khảo

- [Adam Drozdek, Data structures and Algorithms Analysis in C++ 4th Edition, Cengage Learning, 2012](#)
- [Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein, MIT Press, 2012](#)
- <https://www.cs.usfca.edu/~galles/cs245S08/lecture/lecture23.pdf>
- <http://www.cse.chalmers.se/edu/year/2018/course/DAT037/slides/6c-avl-trees.pdf>
- <https://www.cs.usfca.edu/~galles/visualization/AVLtree.html>

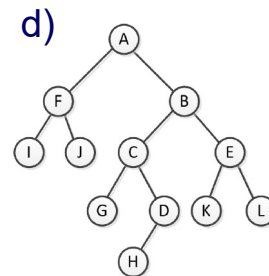
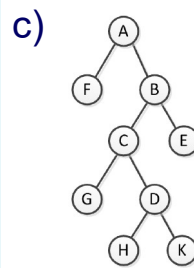
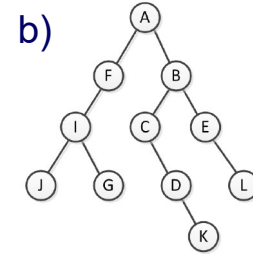
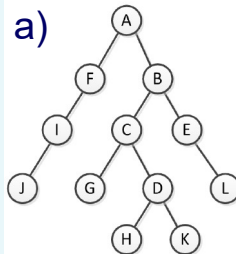
www.ctu.edu.vn

34



CANTHO UNIVERSITY

Chọn cây AVL?

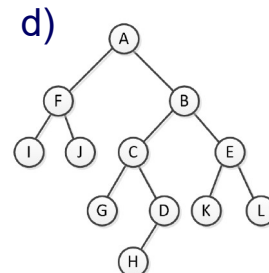
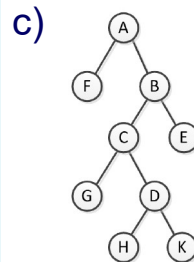
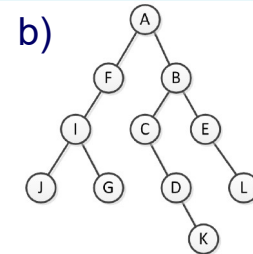
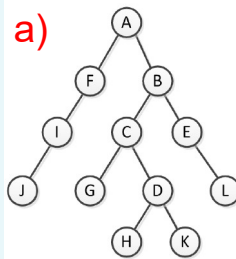


35



CANTHO UNIVERSITY

Chọn cây AVL?

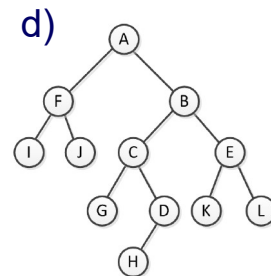
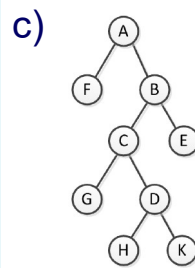
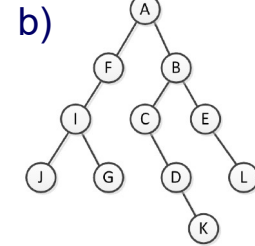
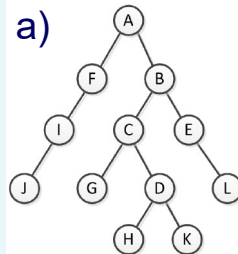


36



CANTHO UNIVERSITY

Chọn cây không là AVL và
chỉ ra nút sai phạm?

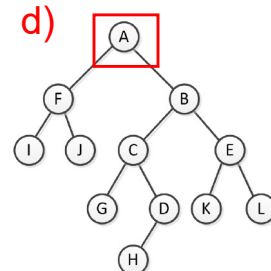
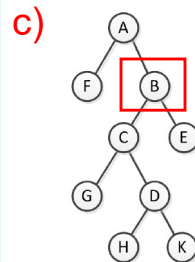
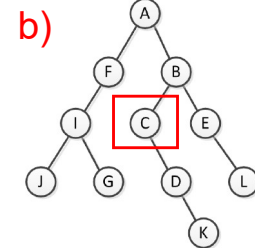
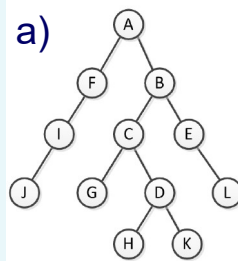


37




CANTHO UNIVERSITY

Chọn cây không là AVL và
chỉ ra nút sai phạm?



38


CANTHO UNIVERSITY


- Vẽ cây AVL tạo thành bằng cách thêm lần lượt các khóa sau (vẽ cây sau mỗi lần 1 khóa được thêm vào)

a) 3, 5, 7, 23, 12, 76, 87, 54, 19, 4

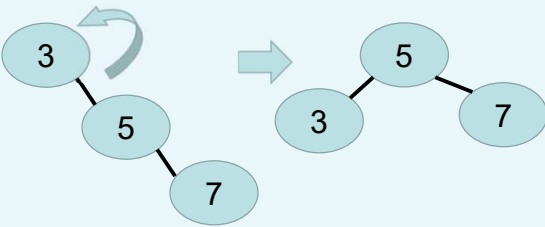
b) 12, 34, 53, 76, 15, 21, 18, 45, 16, 55, 11

www.ctu.edu.vn

39


CANTHO UNIVERSITY

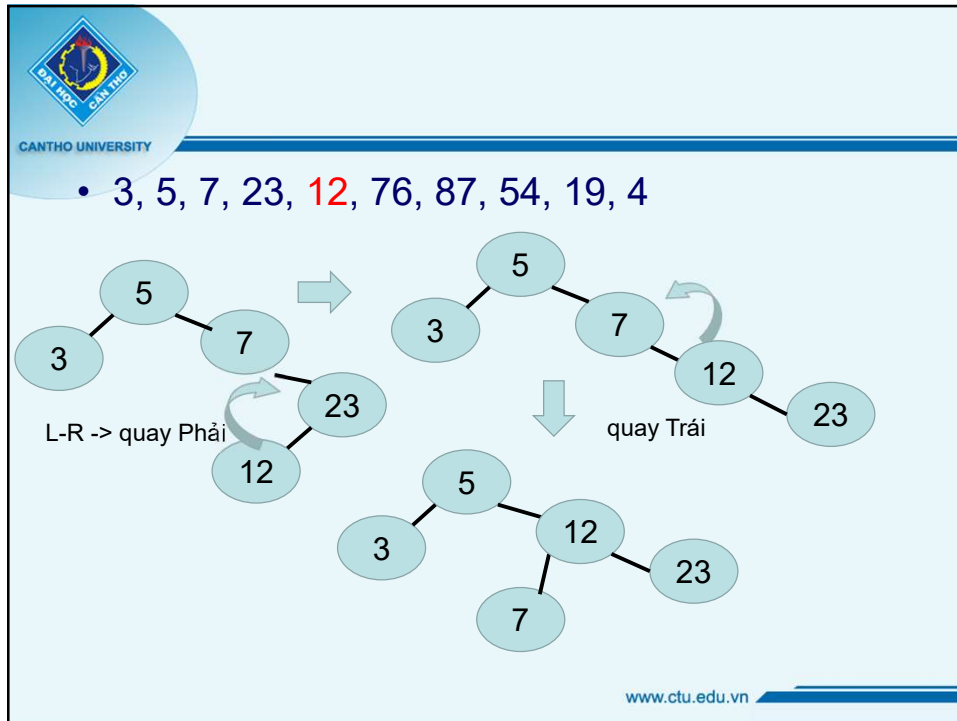
- 3, 5, **7**, 23, 12, 76, 87, 54, 19, 4



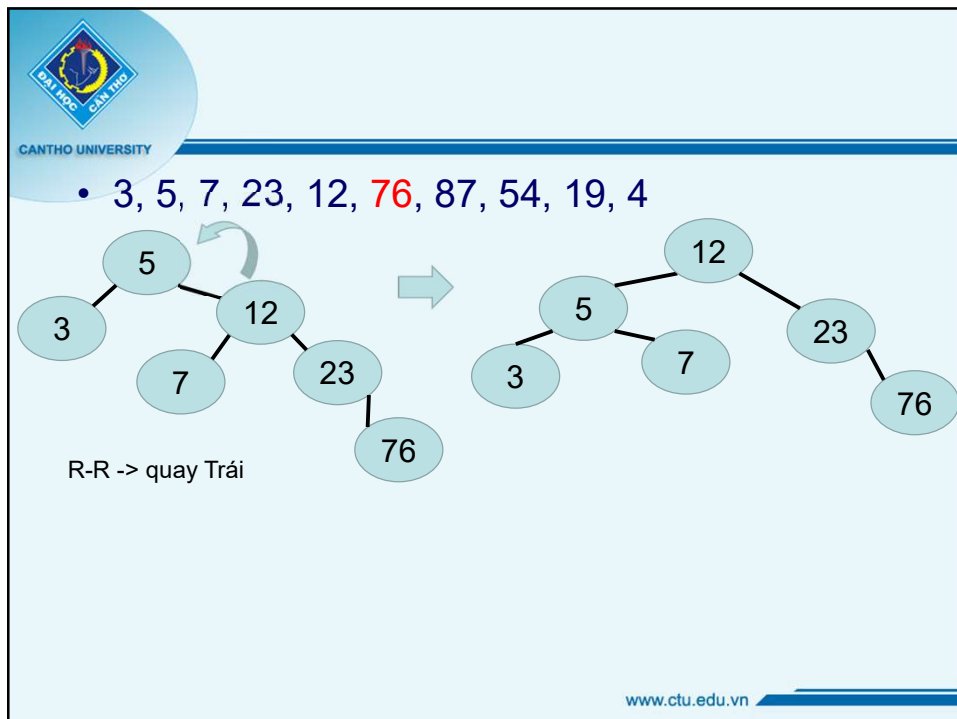
R-R -> quay Trái

www.ctu.edu.vn


40



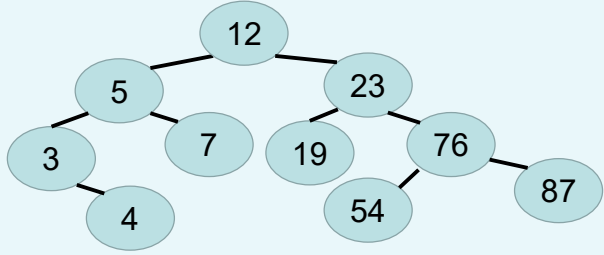
41



42

 CANTHO UNIVERSITY

- 3, 5, 7, 23, 12, 76, 87, 54, 19, 4



www.ctu.edu.vn

43

 CANTHO UNIVERSITY

 Q&A?

www.ctu.edu.vn

44