

## Red-black trees

Created by [anhtt-fit@mail.hut.edu.vn](mailto:anhtt-fit@mail.hut.edu.vn)  
Updated by [huonglt-fit@mail.hut.edu.vn](mailto:huonglt-fit@mail.hut.edu.vn)

## Nhắc lại khái niệm bảng ký hiệu

Bảng ký hiệu: cặp key-value trừu tượng.

- Chèn 1 value với key cho trước.
- Tìm 1 value với key cho trước.
- Xóa 1 value với key cho trước.
- Các kiểu cài đặt
  - Array
  - Linked list
  - BST (binary search tree)

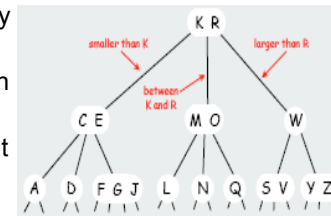
## Độ phức tạp

implementation	guarantee			average case			ordered iteration?
	search	insert	delete	search	insert	delete	
unordered array	N	N	N	N/2	N/2	N/2	no
ordered array	$\lg N$	N	N	$\lg N$	N/2	N/2	yes
unordered list	N	N	N	N/2	N	N/2	no
ordered list	N	N	N	N/2	N/2	N/2	yes
BST	N	N	N	$1.39 \lg N$	$1.39 \lg N$	?	yes
randomized BST	$7 \lg N$	$7 \lg N$	$7 \lg N$	$1.39 \lg N$	$1.39 \lg N$	$1.39 \lg N$	yes

- Randomized BST.
  - Đảm bảo độ phức tạp  $O(\lg N)$  với 1 phép toán (probabilistic).
  - Cần đếm số cây con tại mỗi nút
  - Cần các số ngẫu nhiên cho các thao tác chèn/xóa

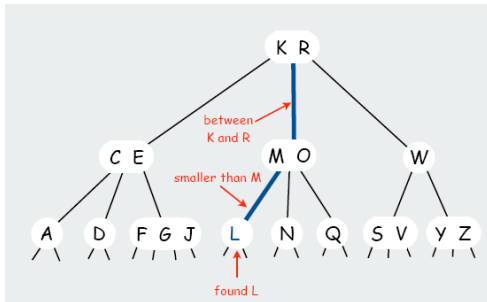
## 2-3-4 tree

- 2-3-4 tree. Tổng quát hóa nút để cho phép nhiều khóa  $\rightarrow$  để cân bằng cây
- Độ cân bằng hoàn hảo. Mọi đường đi từ gốc đến lá có cùng độ dài
- Cho phép 1,2,3 khóa/nút
  - 2-node: 1 khóa, 2 con
  - 3-node: 2 khóa, 3 con
  - 4-node: 3 khóa, 4 con.



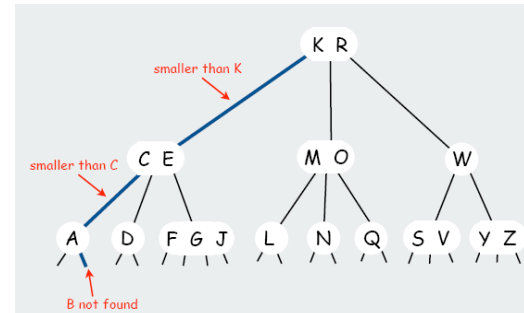
## Tìm kiếm

- So sánh khóa tìm kiếm với các khóa trên mỗi nút
- Tìm nhánh chứa khóa tìm kiếm
- Ví dụ: Tìm L



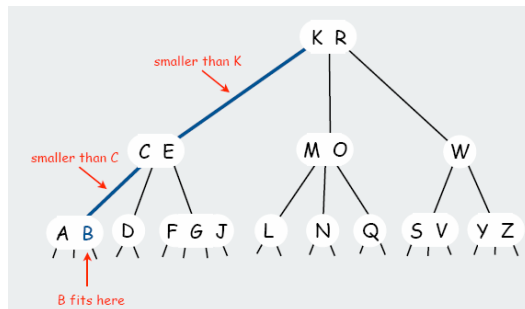
## Insert (1)

- Tìm khóa từ đáy
- Ví dụ: Chèn B



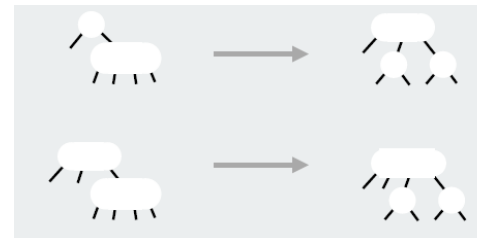
## Insert (2)

- 2-node từ đáy: chuyển thành 3-node.
- 3-node từ đáy: chuyển thành 4-node.
- Ví dụ: chèn B

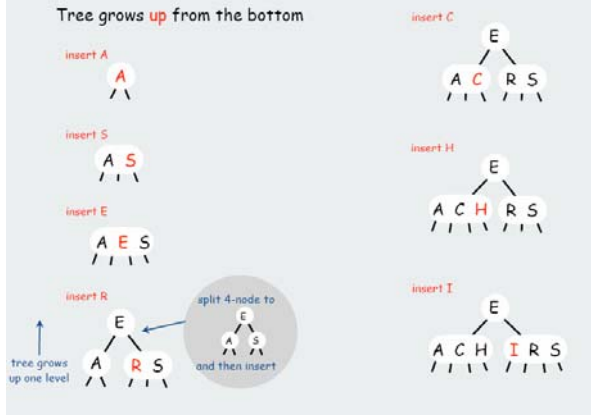


## Phép chuyển đổi

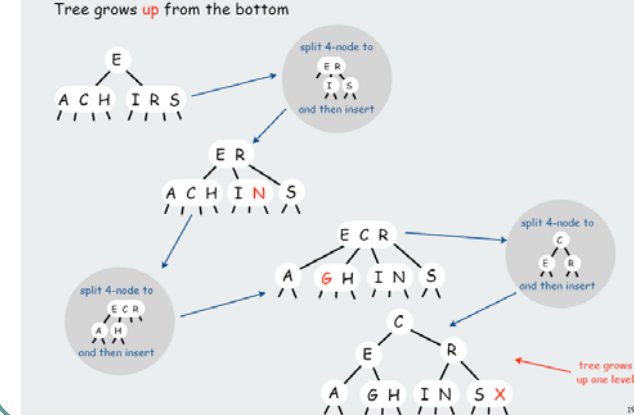
- Các phép chuyển đổi cục bộ cần được áp dụng để giữ cho cây cân bằng.
- Cần chắc chắn rằng phần lớn các nút không phải là 4-node.
- Các phép chuyển đổi để tách 4-nodes:



## Phát triển cây



## Phát triển cây



## Độ phức tạp (Complexity)

implementation	guarantee			average case			ordered iteration?
	search	insert	delete	search	insert	delete	
unordered array	N	N	N	N/2	N/2	N/2	no
ordered array	lg N	N	N	lg N	N/2	N/2	yes
unordered list	N	N	N	N/2	N	N/2	no
ordered list	N	N	N	N/2	N/2	N/2	yes
BST	N	N	N	1.38 lg N	1.38 lg N	?	yes
randomized BST	7 lg N	7 lg N	7 lg N	1.38 lg N	1.38 lg N	1.38 lg N	yes
2-3-4 tree	c lg N	c lg N	c lg N	c lg N	c lg N	c lg N	yes

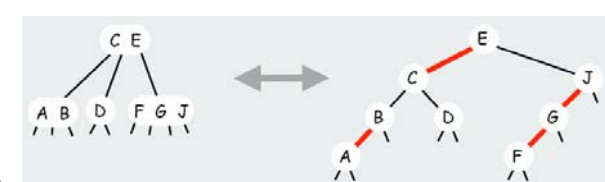
- TH xấu nhất: lg N [all 2-nodes]
- TH tốt nhất:  $\log_4 N = 1/2 \lg N$  [all 4-nodes]
- Giữa 10 và 20 với 1 triệu nút
- Giữa 15 và 30 với 1 tỉ nút

## Cây đỏ đen (Red-black tree)

- Biểu diễn cây 2-3-4 dưới dạng BST.
- Sử dụng các nhánh trong nghiêng bên trái ("internal" left-leaning)



- Sự tương ứng 1-1 giữa cây 2-3-4 và cây nhánh trong nghiêng trái



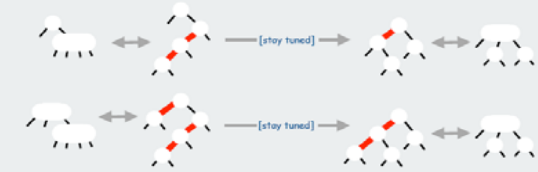
## Cài đặt phép chèn

Basic idea: **maintain 1-1 correspondence with 2-3-4 trees**

1. If key found on recursive search reset value, as usual
2. If key not found **insert a new red node at the bottom**



3. **Split 4-nodes** on the way **DOWN** the tree.



## Complexity

implementation	guarantee			average case			ordered iteration?
	search	insert	delete	search	insert	delete	
unordered array	N	N	N	N/2	N/2	N/2	no
ordered array	lg N	N	N	lg N	N/2	N/2	yes
unordered list	N	N	N	N/2	N	N/2	no
ordered list	N	N	N	N/2	N/2	N/2	yes
BST	N	N	N	1.38 lg N	1.38 lg N	?	yes
randomized BST	7 lg N	7 lg N	7 lg N	1.38 lg N	1.38 lg N	1.38 lg N	yes
2-3-4 tree	c lg N	c lg N		c lg N	c lg N		yes
red-black tree	3 lg N	3 lg N	3 lg N	lg N	lg N	lg N	yes

## Libfdr

- Libfdr là thư viện có chứa hàm cài đặt của cây đỏ đen trong C
- Tải và biên dịch tại

<http://www.cs.utk.edu/~plank/plank/classes/cs360/360/notes/Libfdr/>

## Jval datatype

- Biểu diễn kiểu dữ liệu tổng quát  
Ví dụ: Dùng Jval để lưu trữ số nguyên  
Jval j;  
j.i = 4;
- Jval.h khai báo tập các mẫu (prototypes) cho hàm khởi tạo (constructor functions)  
extern Jval new\_jval\_i(int);  
extern Jval new\_jval\_f(float);  
extern Jval new\_jval\_d(double);  
extern Jval new\_jval\_v(void \*);  
extern Jval new\_jval\_s(char \*);  
Ví dụ:  
Jval j = new\_jval\_i(4);

## RB tree routines

- Tạo cây
  - JRB make\_jrb();
- Chèn cây
  - JRB jrb\_insert\_str(JRB tree, char \*key, Jval val);
  - JRB jrb\_insert\_int(JRB tree, int key, Jval val);
  - JRB jrb\_insert\_dbl(JRB tree, double key, Jval val);
  - JRB jrb\_insert\_gen(JRB tree, Jval key, Jval val, int (\*func)(Jval, Jval));
- Tìm khóa
  - jrb\_find\_str(), jrb\_find\_int(), jrb\_find\_dbl() or jrb\_find\_gen()

## Bài 1

- Dịch và chạy một số chương trình mẫu (sử dụng libfdr) tại <http://www.cs.utk.edu/~plank/plank/classes/cs360/360/notes/JRB/>

## Bài 2

- Sử dụng libfdr để viết chương trình phone book (add, delete, insert, modify phone numbers). Phone book được lưu trong file.