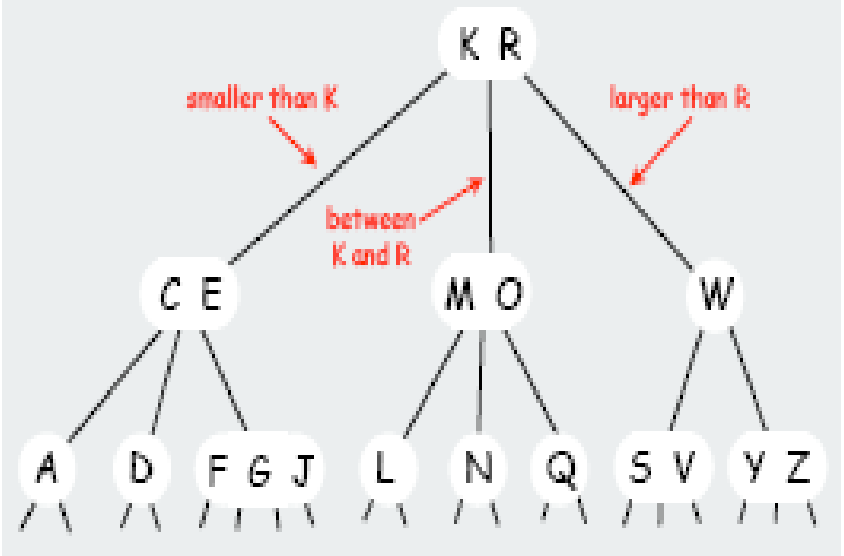
Cây đỏ - đen (Red Black tree)

Nội dung

- Cây 2-3-4
- Cây đỏ-đen lệch trái

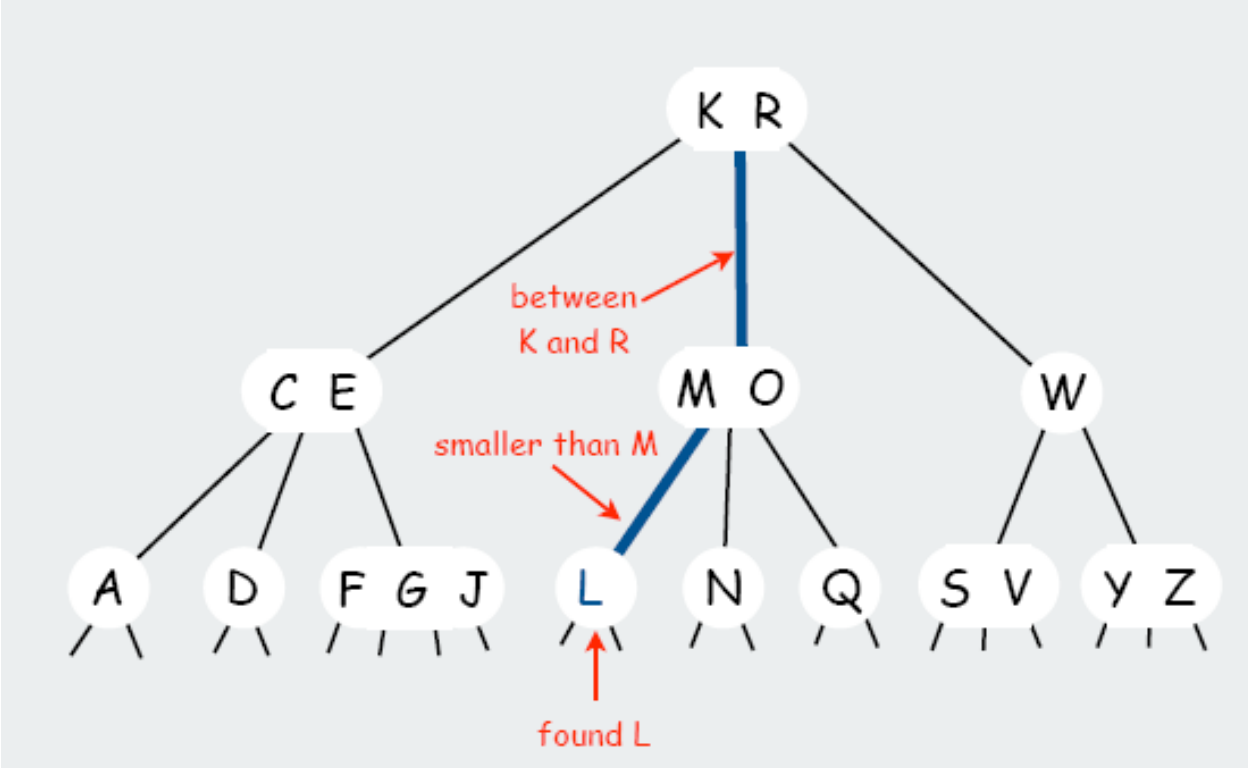
1. 2-3-4 tree

- 2-3-4 tree. Khởi tạo các nút, cho phép có nhiều giá trị khóa; giúp cho cây cân bằng
- Cân bằng hoàn hảo: mọi đường đi từ nút gốc (root) đến nút lá (leaf) có cùng độ dài
- Cho phép có 1, 2 hoặc 3 khóa mỗi nút
  - 2-node: 1 khóa, 2 con
  - 3-node: 2 khóa, 3 con
  - 4-node: 3 khóa, 4 con



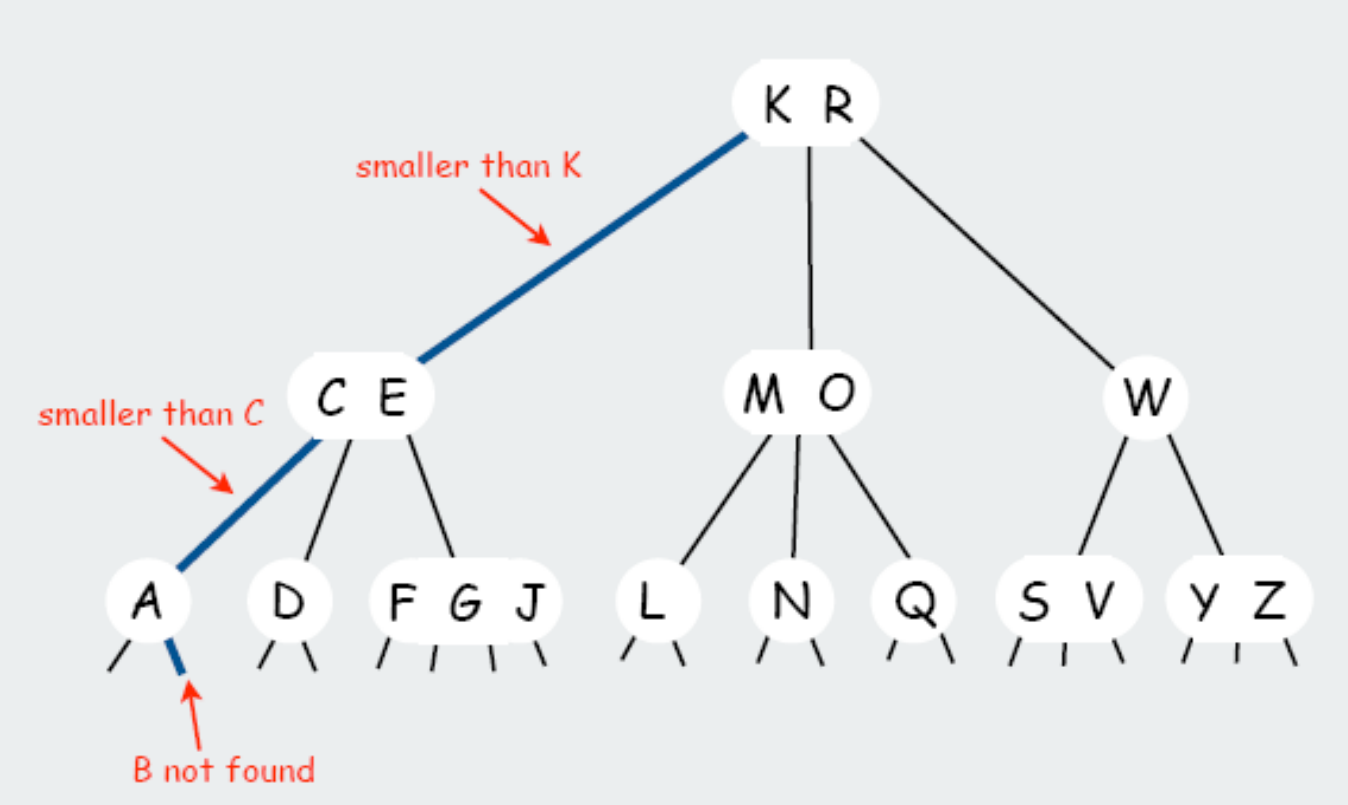
Search

- So sánh khóa cần tìm với khóa của nút
- Tìm nhánh chứa khóa liên quan
- Ví dụ: Search L



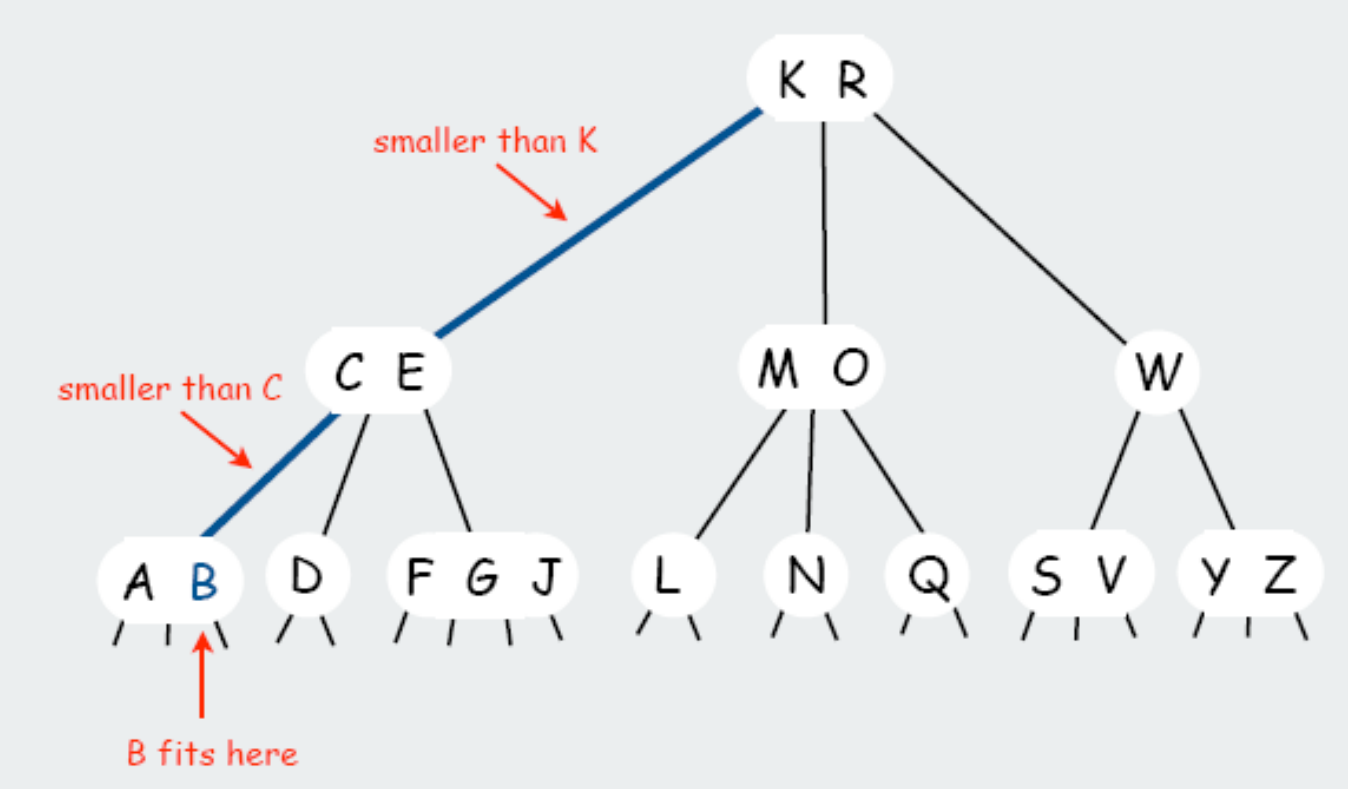
Insert (1)

- Tìm kiếm khóa cần chèn cho đến đáy (bottom)
- Ví dụ. Insert B



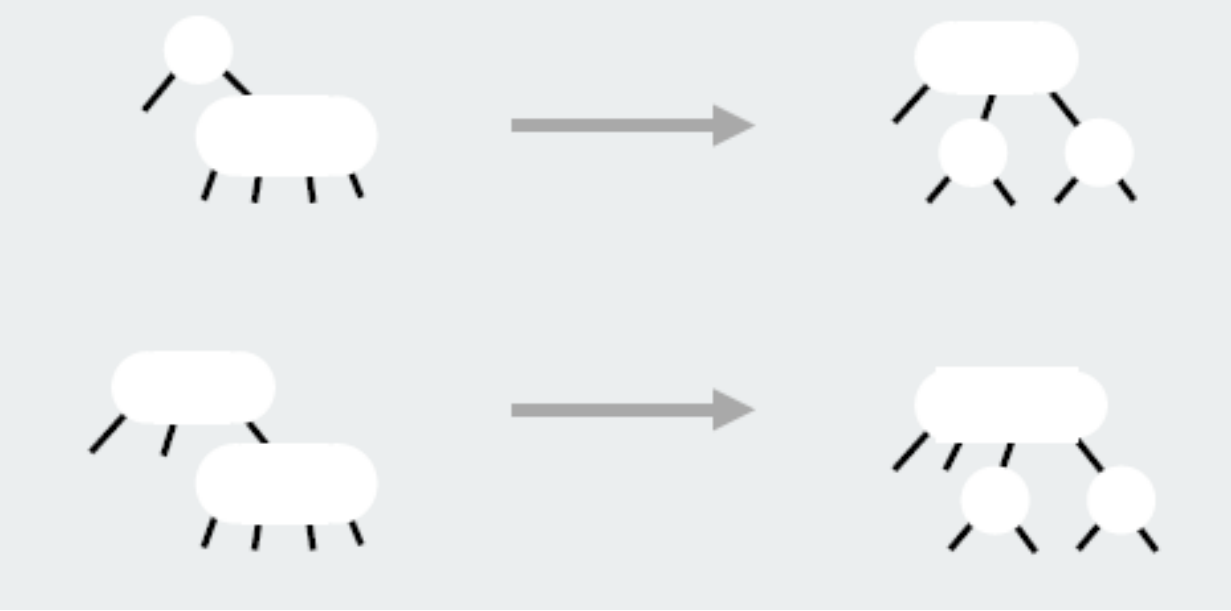
Insert (2)

- 2-node ở đáy (bottom): chuyển thành 3-node.
- 3-node ở đáy (bottom): chuyển thành 4-node.
- Ví dụ: insert B

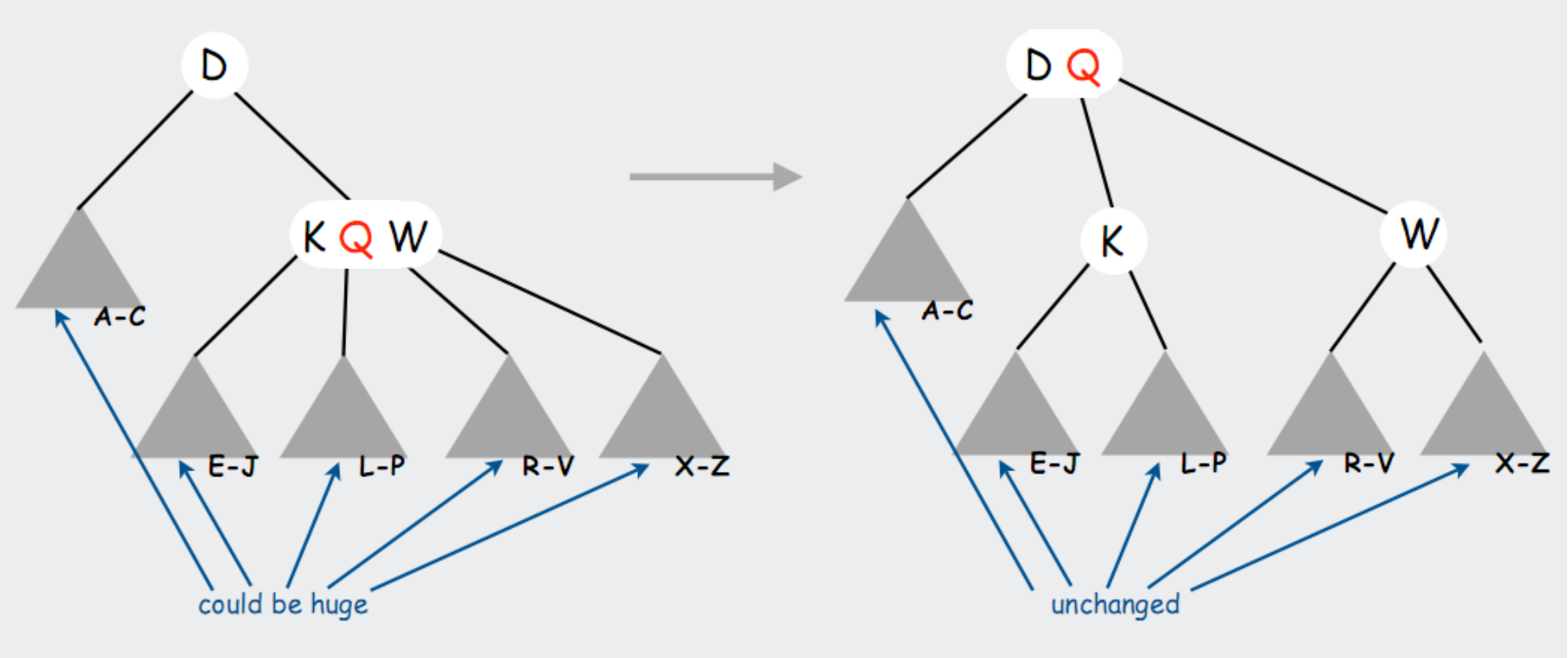


Dịch chuyển

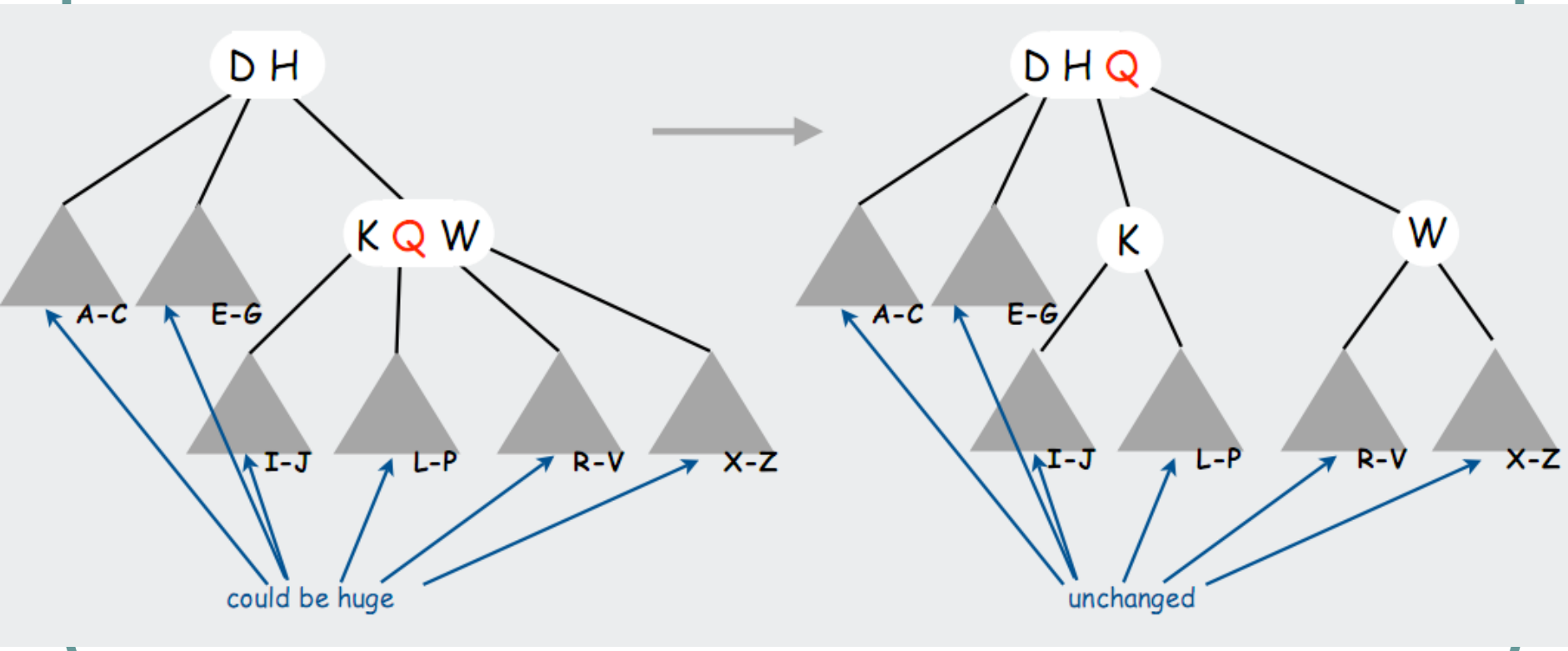
- Các dịch chuyển nên được thực thi để giữ cây cân bằng
- Đảm bảo rằng phần lớn các nút không phải là 4-node
- Các dịch chuyển để tách 4-node



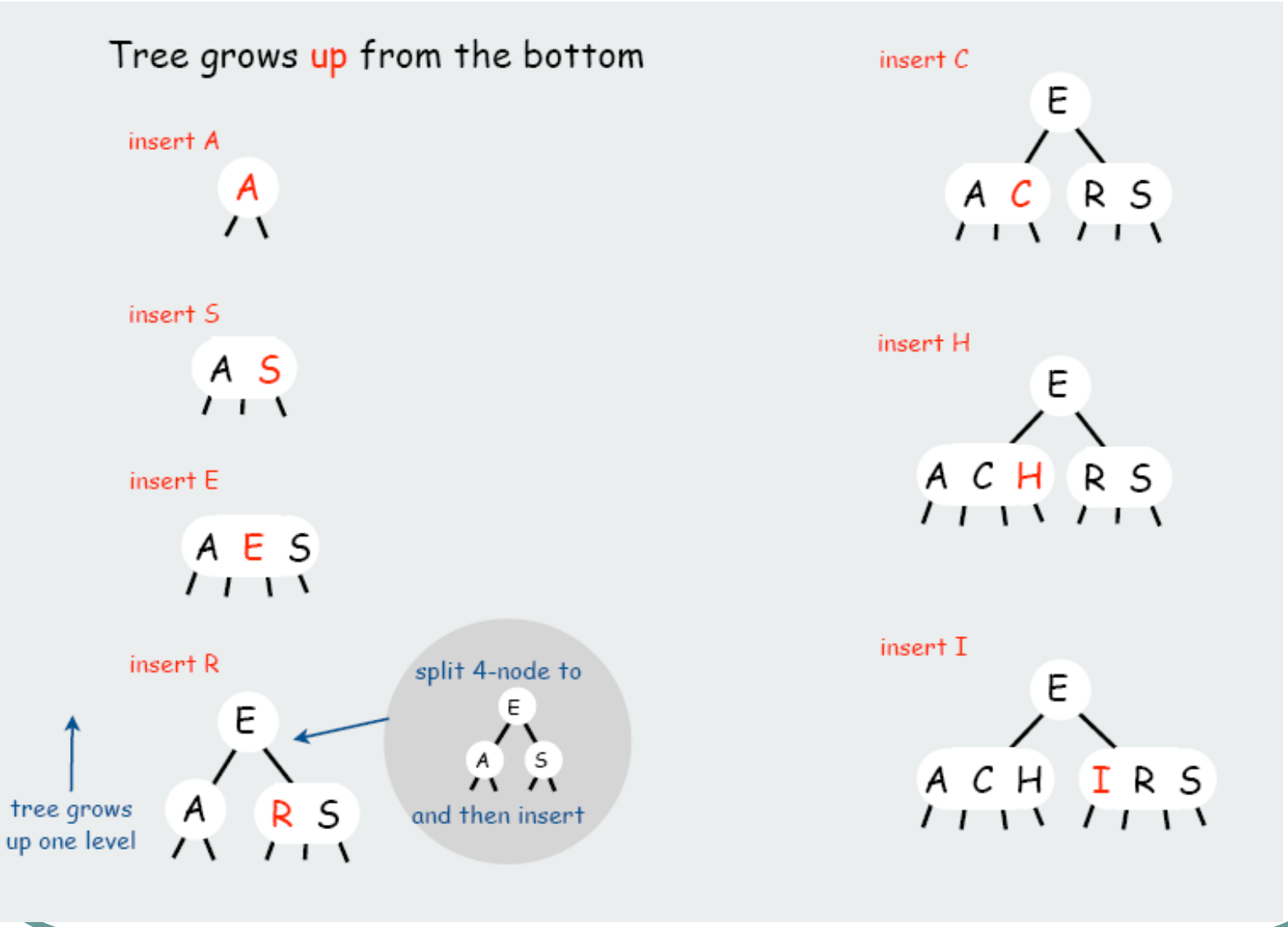
Dịch chuyển – 1



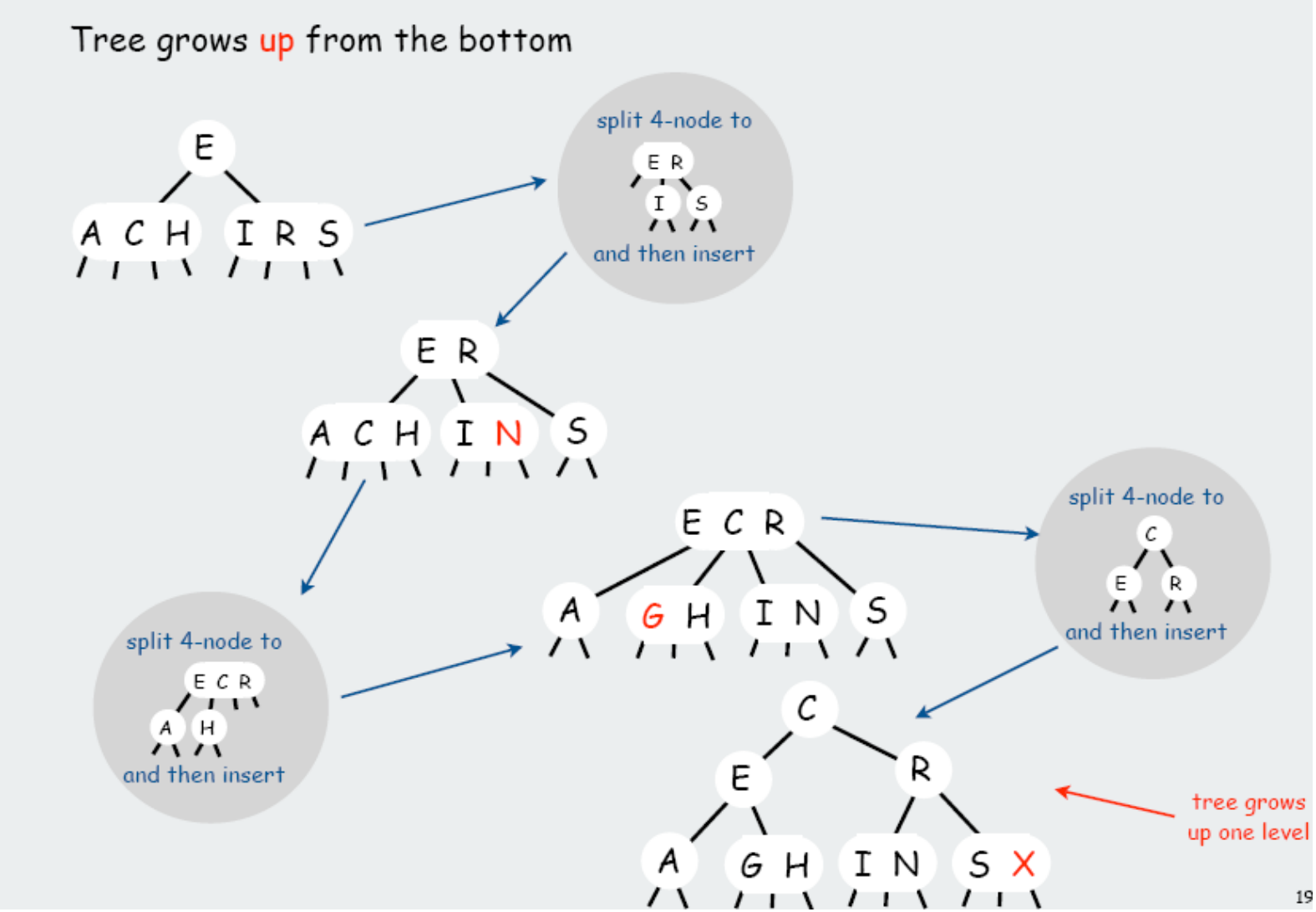
Dịch chuyển – 2



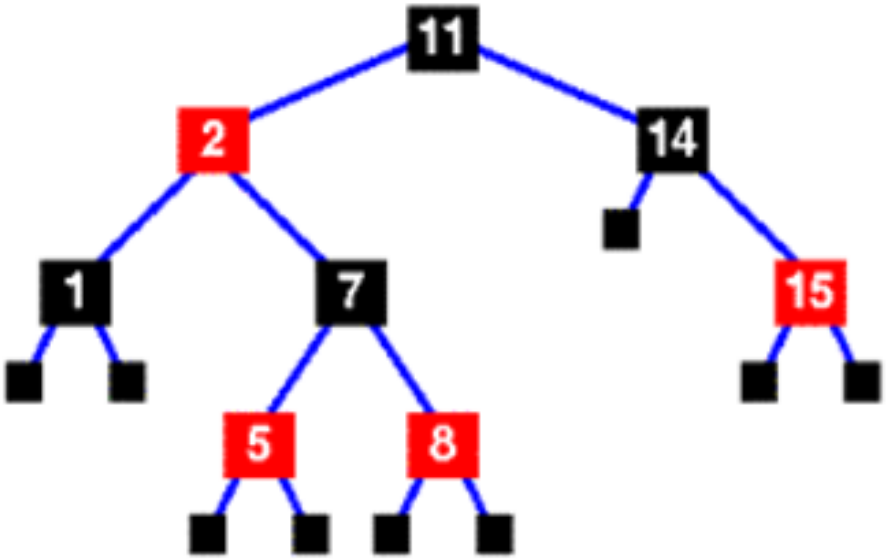
Sự gia tăng của cây



Sự gia tăng của cây (2)



2. Cây đỏ - đen (red black tree)

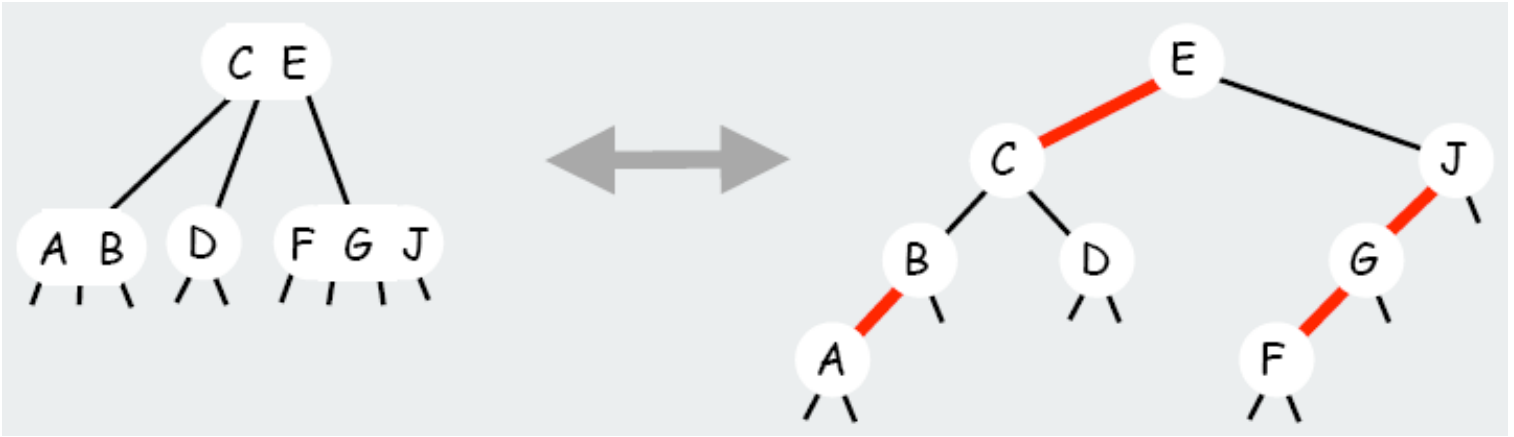


- Node: đỏ (red) hoặc đen (black)
- Nút gốc & lá (Root & leaf): đen (black)
- Node: đỏ (red) => con (child): đen (black)
- Đường từ nút gốc tới lá có cùng số lượng nút đen

2. Cây đỏ - đen

- Biểu diễn cây 2-3-4 dưới dạng cây đỏ đen
- Trong đó sử dụng các nút lệch trái cho nút 3 và 4

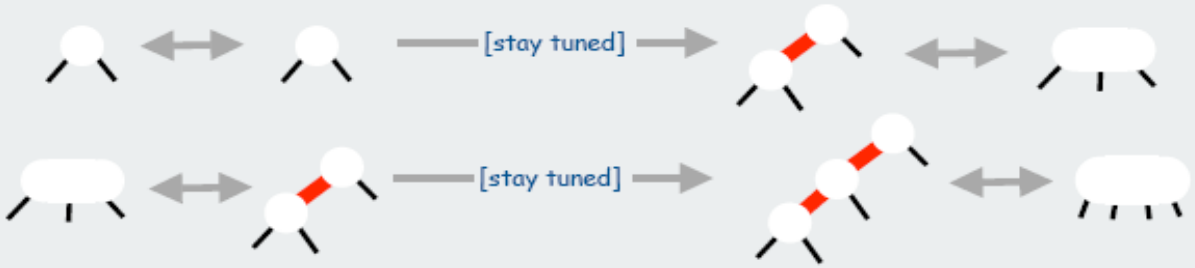
- Tương ứng 1-1 giữa cây 2-3-4 và cây đỏ đen lệch trái



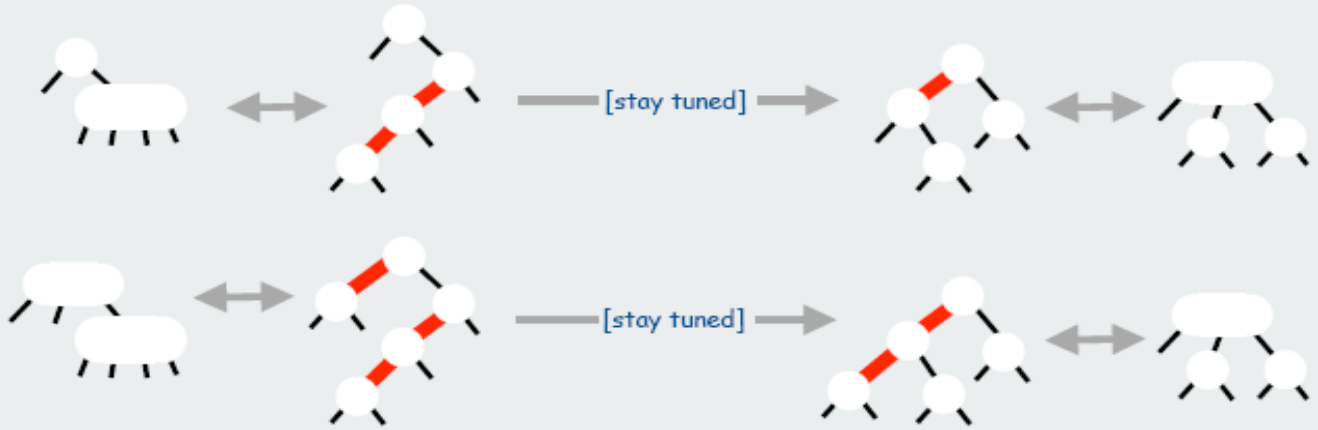
Thực hiện chèn

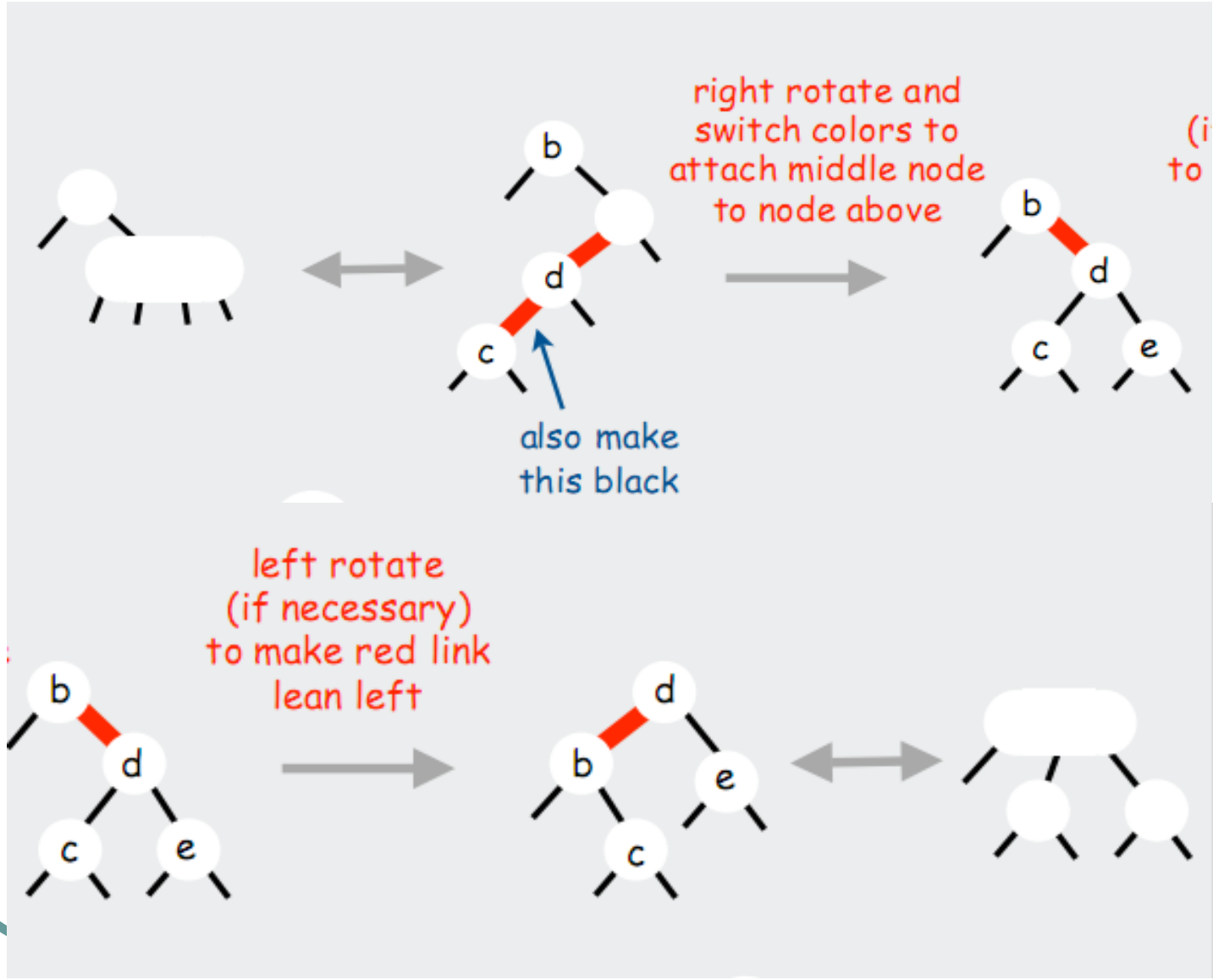
Basic idea: maintain 1-1 correspondence with 2-3-4 trees

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.




right rotate and
switch colors to
attach middle node
to node above
b
d
c
e
also make
this black
left rotate
(if necessary)
to make red link
lean left

Libfdr

- Libfdr là một thư viện chứa các cài đặt cho cây đỏ đen tổng quát trong C
- Link download

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

Cấu trúc dữ liệu Jval

- Là kiểu hợp (union) biểu diễn kiểu dữ liệu tổng quát

```
typedef union {
    int i;
    long l;
    float f;
    double d;
    void *v;
    char *s;
    char c;
    unsigned char uc;
    short sh;
    unsigned short ush;
    unsigned int ui;
    int iarray[2];
    float farray[2];
    char carray[8];
    unsigned char ucarray[8];
} Jval;
```

Sử dụng Jval

- Sử dụng Jval lưu trữ số nguyên

```
Jval j;
j.i = 4;
```

- Jval.h đã định nghĩa các hàm khởi tạo

```
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);
```

JRB

- JRB là một con trỏ, trỏ tới một nút trên cây

```
typedef struct jrb_node {
  unsigned char red;
  unsigned char internal;
  unsigned char left;
  unsigned char roothead;
  struct jrb_node *flink;
  struct jrb_node *blink;
  struct jrb_node *parent;
  Jval key;
  Jval val;
} *JRB;
```

JRB API (1)

- Tạo một cây mới
  - JRB make_jrb();
- Chèn một nút mới vào cây
  - JRB jrb_insert_str(JRB tree, char *key, Jval val);
  - JRB jrb_insert_int(JRB tree, int ikey, Jval val);
  - JRB jrb_insert_dbl(JRB tree, double dkey, Jval val);
  - JRB jrb_insert_gen(JRB tree, Jval key, Jval val, int (*func)(Jval,Jval));
- Tìm một nút qua giá trị
  - JRB jrb_find_str(JRB root, char *key);
  - JRB jrb_find_int(JRB root, int ikey);
  - JRB jrb_find_dbl(JRB root, double dkey);
  - JRB jrb_find_gen(JRB root, Jval, int (*func)(Jval, Jval));

JRB API (2)

- Giải phóng một nút
  - void jrb_delete_node(JRB node);
- Giải phóng cây
  - void jrb_free_tree(JRB root);
- Di chuyển trên cây
  - #define jrb_first(n) (n->flink)
  - #define jrb_last(n) (n->blink)
  - #define jrb_next(n) (n->flink)
  - #define jrb_prev(n) (n->blink)
  - #define jrb_empty(t) (t->flink == t)
  - #define jrb_nil(t) (t)
  - #define jrb_traverse(ptr, lst) \
    for(ptr = jrb_first(lst); ptr != jrb_nil(lst); ptr = jrb_next(ptr))

Bài 1

- Biên dịch & chạy các chương trình được đưa ra tại link

http://www.cs.utk.edu/~plank/plank/classes/cs360/360/notes/JRB/

Bài 2

- Sử dụng thư viện JRB, viết chương trình có các chức năng sau:
  - Insert. Nhập từ bàn phím một số nguyên, và chèn vào cây
  - Display. Hiển thị tất cả các số nguyên đã nhập
  - Destroy. Hủy bỏ vùng nhớ cấp phát cho các nút trên cây

Bài 3

- Sử dụng libfdr viết chương trình quản lý danh bạ điện thoại (có menu lựa chọn thêm, xóa, sửa số điện thoại, thoát).
  - Key: tên người
  - Val: số điện thoại
- Lưu ý: trong JRB, hàm insert luôn tạo ra một nút mới dù khóa (key) đã có trên cây. Do đó cần kiểm tra xem khóa cần chèn đã có trên cây chưa

Hướng dẫn

- Tạo danh bạ
  - JRB book = make_jrb();
- Chèn một thông tin điện thoại mới
  - jrb_insert_str(book, strdup(name), new_jval_l(number));
- Duyệt trên cây
  - jrb_traverse(node, book)

    /* code to do something on node */

Bài 4

- Mở rộng yêu cầu bài 3
- Thông tin danh bạ điện thoại được lưu trong file danhba.dat
- Đọc dữ liệu từ danh bạ vào cây, sau khi thực hiện các thao tác thêm, xóa, cập nhật điện thoại, trước khi xóa cây phải lưu lại vào file