Red-black trees

Instructor: Thanh-Chung Dao
(chungdt@soict.hust.edu.vn)
Slides by Dr. Ta Tuan Anh

1

Content

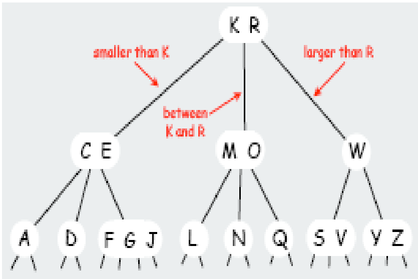
This lecture:

- 2-3-4 tree
- Left – leaning red black tree

2

1. 2-3-4 tree

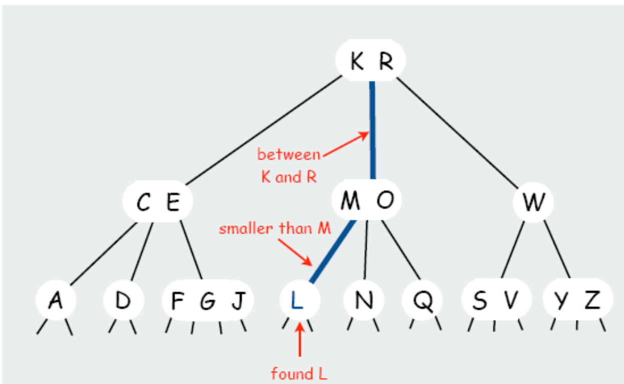
- 2-3-4 tree. Generalize node to allow multiple keys; help to keep tree balanced.
- Perfect balance. Every path from root to leaf has same length.
- Allow 1, 2, or 3 keys per node.
  - 2-node: one key, two children.
  - 3-node: two keys, three children.
  - 4-node: three keys, four children.



3

Search

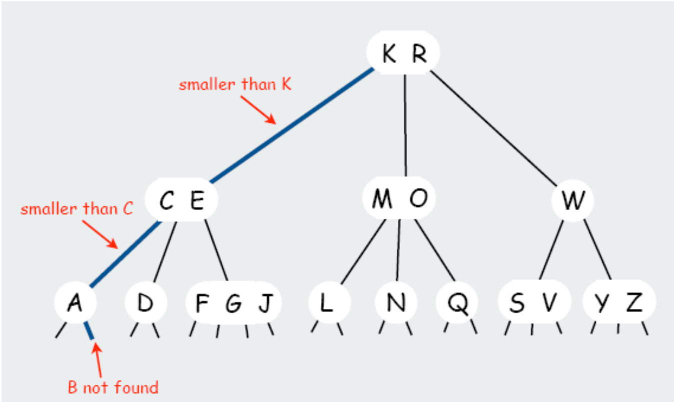
- Compare search key against keys in node.
- Find interval containing search key.
- Ex. Search for L



4

Insert (1)

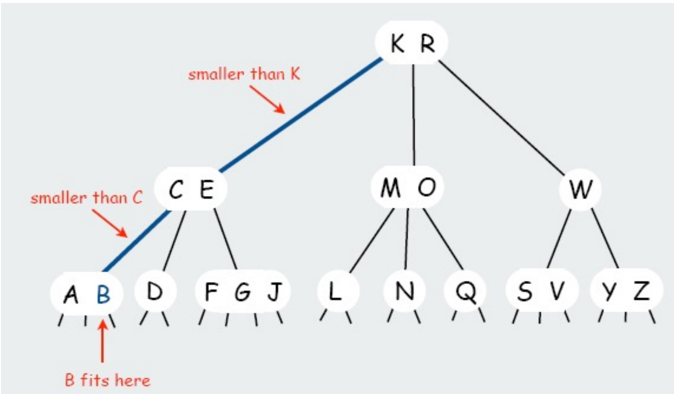
- Search to bottom for key.
- Ex. Insert B



5

Insert (2)

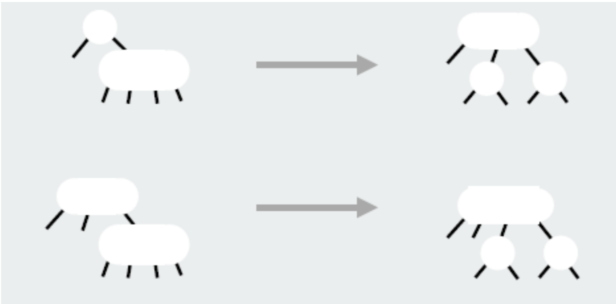
- 2-node at bottom: convert to 3-node.
- 3-node at bottom: convert to 4-node.
- Ex. Insert B



6

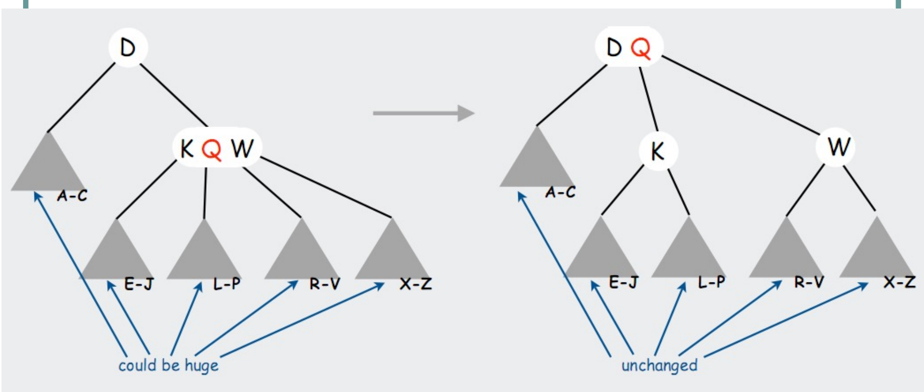
Transformation

- Local transformations should be applied to keep the tree balanced.
- Ensures that most recently seen node is not a 4-node.
- Transformations to split 4-nodes:



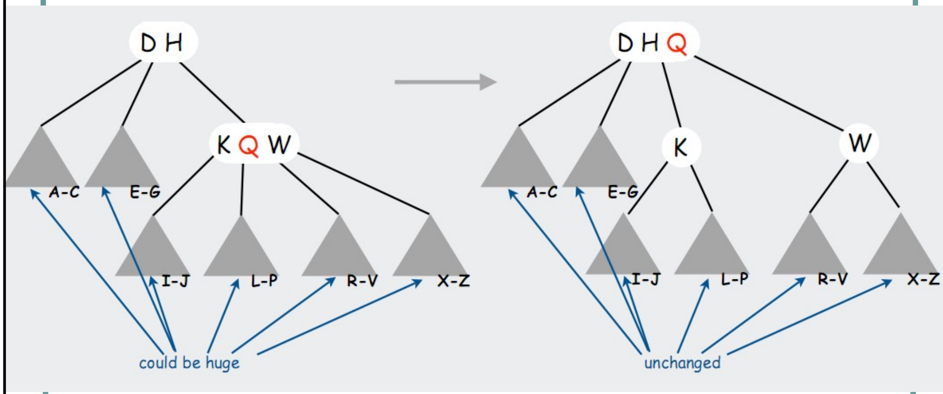
7

Transformation – 1



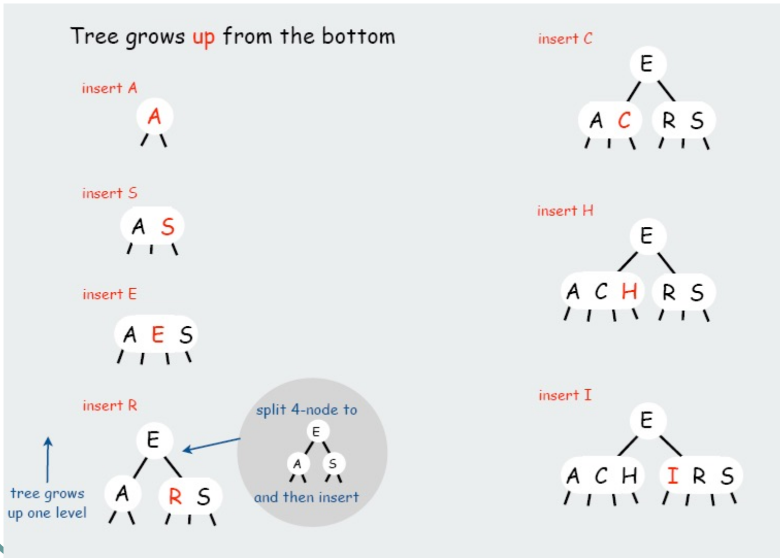
8

Transformation – 2



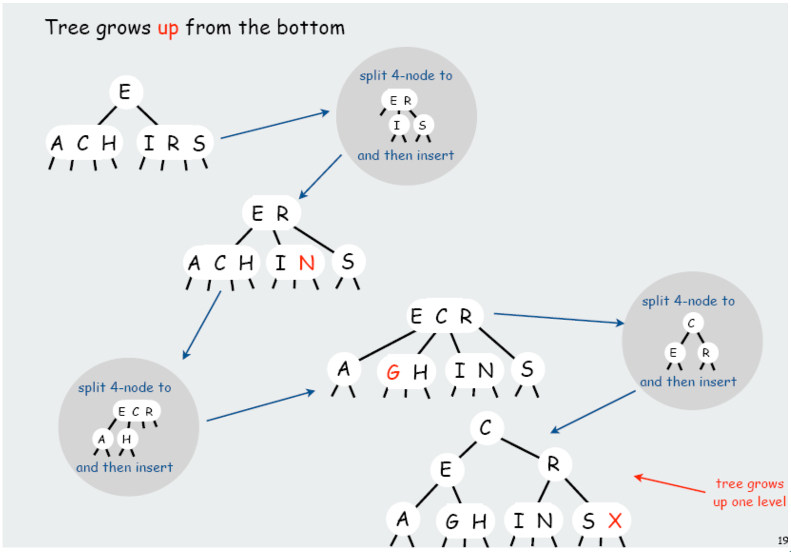
9

Growth of a tree



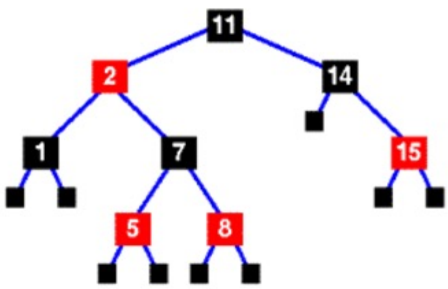
10

Growth of a tree (cont.)



11

2. Red black tree



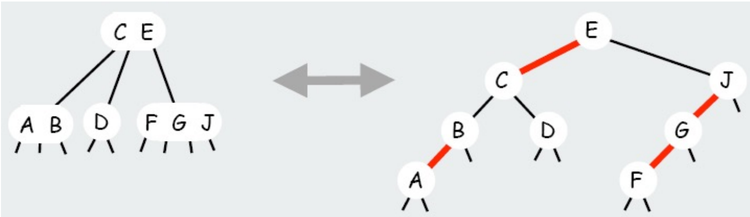
- Node: red or black
- Root: black
- Node: red => child: black and parent: black
- Way from root to leaf has same the number of black nodes

12

2. Red-black tree

- Represent 2-3-4 tree as a BST.
- Use "internal" left-leaning edges for 3- and 4- nodes.

- 1-1 correspondence between 2-3-4 and left-leaning red-black trees.



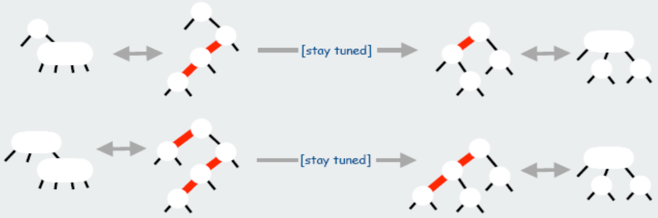
13

Insert implementation

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

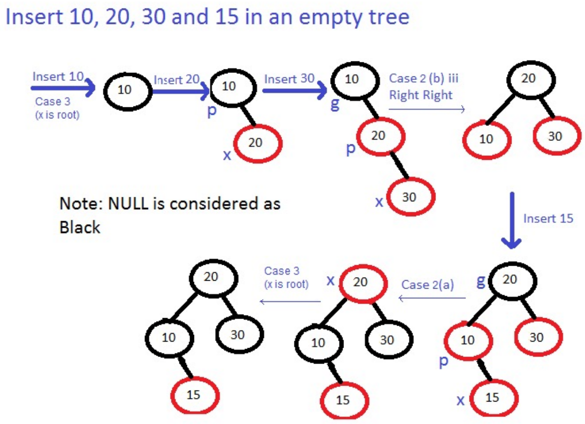
3. Split 4-nodes on the way DOWN the tree.



14

Insertion example

- https://www.geeksforgeeks.org/red-black-tree-set-2-insert/



15

Libfdr

- Libfdr is a library which contains an implementation for generic red-black trees in C
- Download and compile instructions at http://www.cs.utk.edu/~plank/plank/classes/cs360/360/notes/Libfdr/

On teams

- jrb.h jrb.c
- jval.h jval.c

16

Jval datatype

- A big union to represent a generic data type

```
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;
```

17

Jval usage

- Use Jval to store an integer

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

- Jval.h defines a whole bunch of prototypes for ``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 *);
```

Example:

```
Jval j = new_jval_i(4);
```

18

JRB datatype

- JRB is defined as a pointer to a node of the tree

```
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;
```

19

JRB API (1)

- Make a new tree
  - JRB make_jrb();
- Insert a new node to a tree
  - 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));
- Find a node via key
  - 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));

20

JRB API (2)

- Free a node (but not the key or val)
  - void jrb_delete_node(JRB node);
- Free all the tree
  - void jrb_free_tree(JRB root);
- Navigation in the tree
  - #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))

21

Quiz 1

- Use libfdr to write the phone book program (add, delete, modify phone numbers). The phone book should be stored in a file.
- NB: In the JRB, the insert function always creates a new node even the key exists already in the tree.
  - You should check the existence of a record before insert it in the tree

22

Instruction

- Create a phone book
  - JRB book = make_jrb();
- Insert a new entry
  - jrb_insert_str(book, strdup(name), new_jval_l(number));
    - You must allocate memory to store the name for the new node's key. This memory should to be free when we delete all the key.
- Navigation
  - jrb_traverse(node, book)

    /* code to do something on node */

23