

Red-black trees

anhtt-fit@mail.hut.edu.vn

Symbol Table Review

Symbol table: key-value pair abstraction.

- Insert a value with specified key.
- Search for value given key.
- Delete value with given key.
- Different implementations
 - Array
 - Linked list
 - BST (binary search tree)

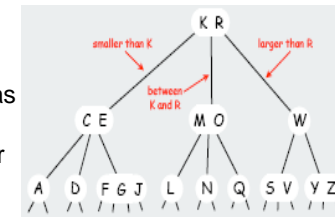
Complexity

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	$1.39 \lg N$	$1.39 \lg N$?	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.
 - Guarantee of $\sim c \lg N$ time per operation (probabilistic).
 - Need subtree count in each node.
 - Need random numbers for each insert/delete op.

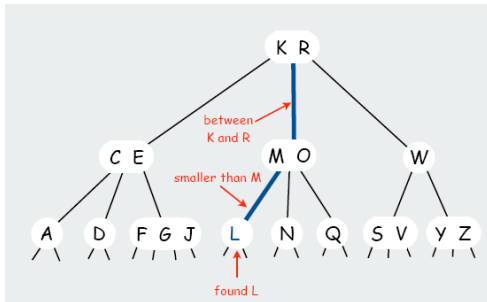
2-3-4 tree

- 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.



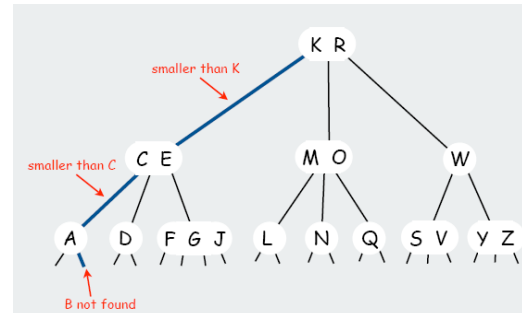
Search

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



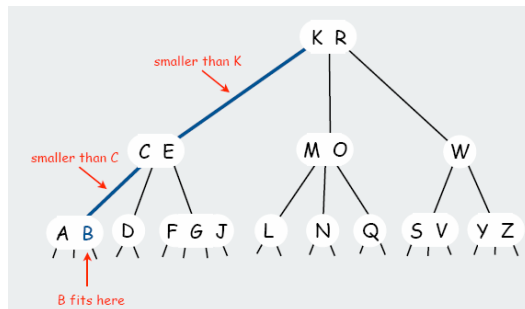
Insert (1)

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



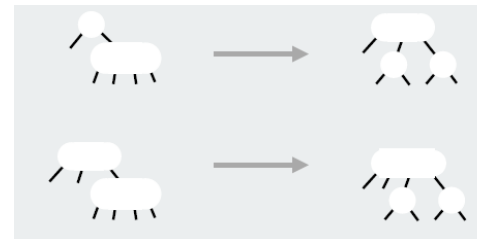
Insert (2)

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

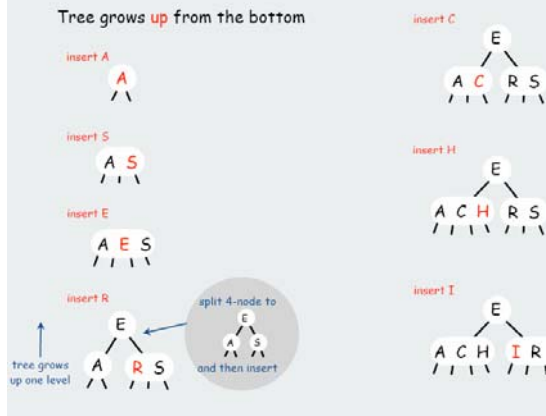


Transformation

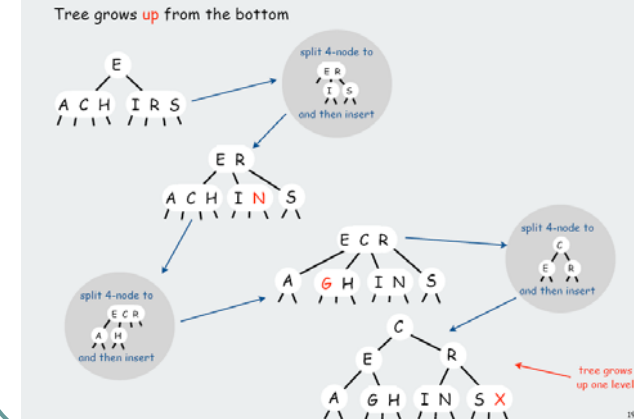
- Local transformations should be applied to keep the tree balanced.
- Ensures that most recently seen node is not a 4-node.
- Transformations to split 4-nodes:



Growth of a tree



Growth of a tree (cont.)



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

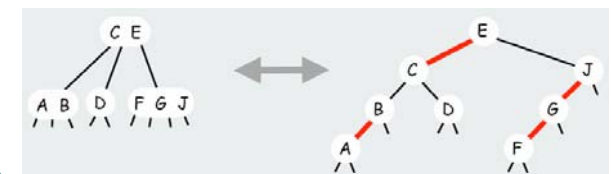
- Worst case: lg N [all 2-nodes]
- Best case: $\log_4 N = 1/2 \lg N$ [all 4-nodes]
- Between 10 and 20 for a million nodes.
- Between 15 and 30 for a billion nodes.

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.



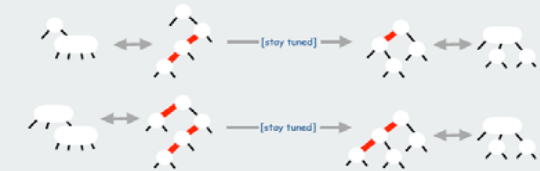
Insert implementation

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.



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 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/>

Jval datatype

- A big union to represent a generic data type
Example: 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);

RB tree routines

- To create a tree
 - JRB make_jrb();
- To insert entries
 - 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));
- To find keys
 - jrb_find_str(), jrb_find_int(), jrb_find_dbl() or jrb_find_gen()

Quiz 1

- Try to compile and run some example programs (using libfdr) given at <http://www.cs.utk.edu/~plank/plank/classes/cs360/360/notes/JRB/>

Quiz 2

- Use libfdr to write the phone book program (add, delete, insert, modify phone numbers). The phone book should be stored in a file.