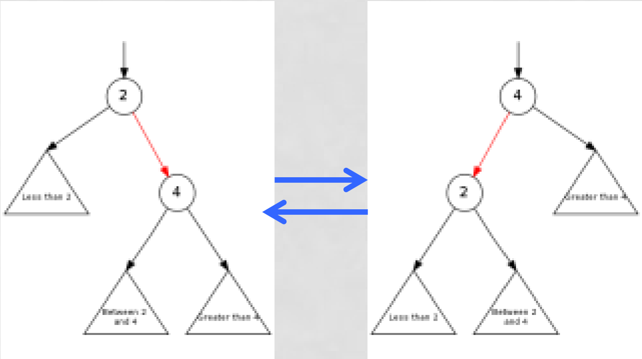
LEFT-LEANING RED BLACK TREES

TONY LIU AND MICHAEL SHAW

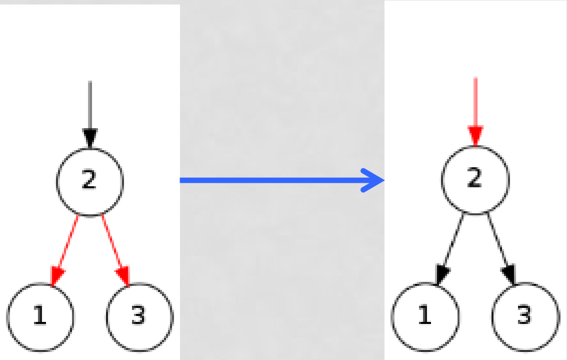
LLRB OPERATIONS

Rotate Left



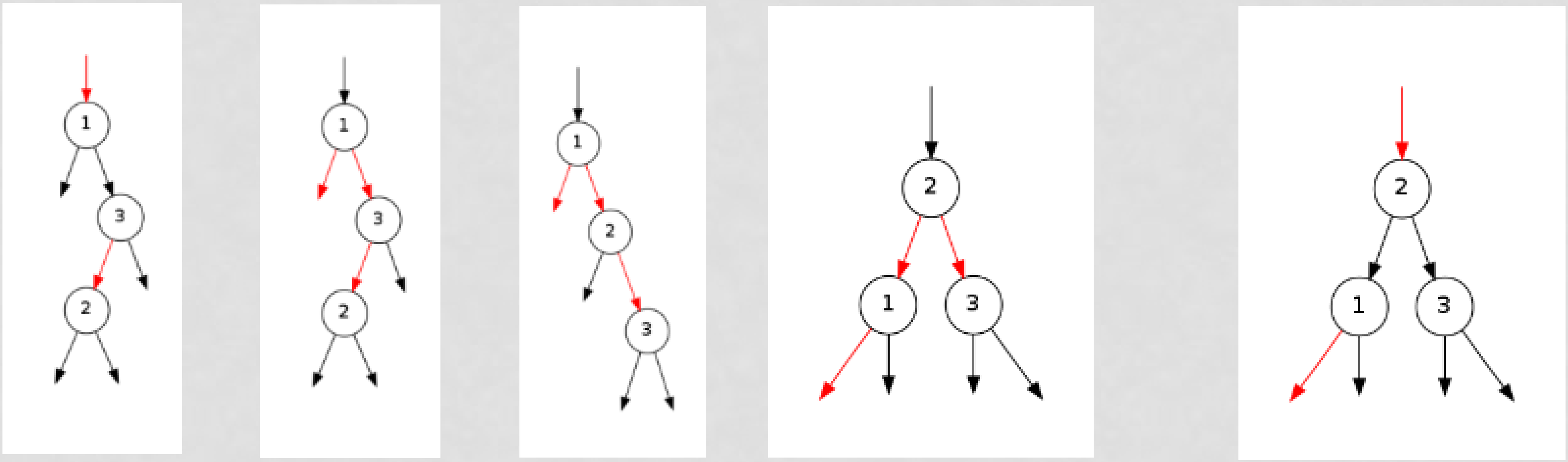
Rotate Right

Flip Colors

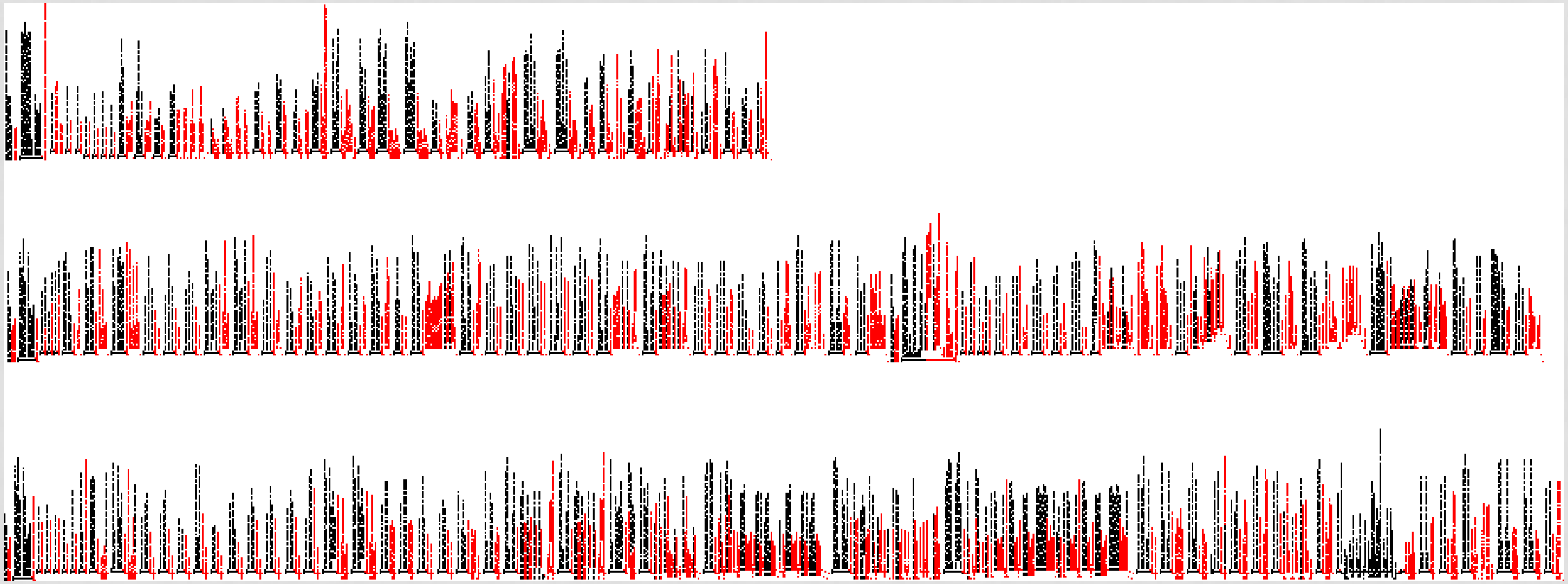


LLRB OPERATIONS CONT.

Move Red Left (Remove Operation)



CODE SNAPSHOTS

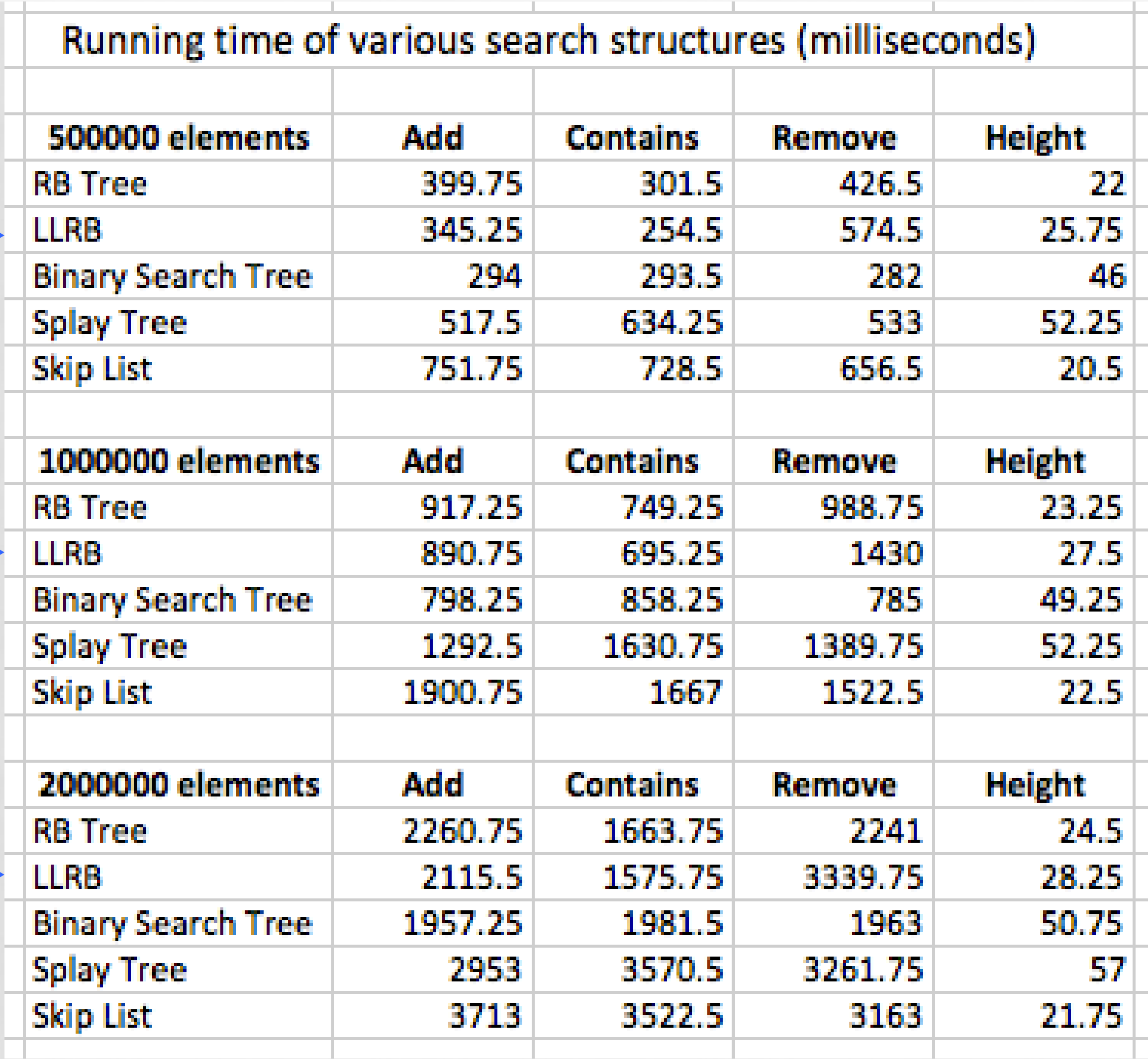


LLRB

BST

RB Tree

PERFORMANCE



Running time of various search structures (milliseconds)

500000 elements	Add	Contains	Remove	Height
RB Tree	399.75	301.5	426.5	22
LLRB	345.25	254.5	574.5	25.75
Binary Search Tree	294	293.5	282	46
Splay Tree	517.5	634.25	533	52.25
Skip List	751.75	728.5	656.5	20.5

1000000 elements	Add	Contains	Remove	Height
RB Tree	917.25	749.25	988.75	23.25
LLRB	890.75	695.25	1430	27.5
Binary Search Tree	798.25	858.25	785	49.25
Splay Tree	1292.5	1630.75	1389.75	52.25
Skip List	1900.75	1667	1522.5	22.5

2000000 elements	Add	Contains	Remove	Height
RB Tree	2260.75	1663.75	2241	24.5
LLRB	2115.5	1575.75	3339.75	28.25
Binary Search Tree	1957.25	1981.5	1963	50.75
Splay Tree	2953	3570.5	3261.75	57
Skip List	3713	3522.5	3163	21.75

CONCLUSIONS

- LLRB is competitive with other data structures on most functions
- Can be implemented in about 300 lines of code
- Left-leaning aspect simplifies the tree operations