sors suggestions

1. aurora vs trap remix

disc. 13
cs61b sp22

nicholas sparks
97 avalon

graphs and LLRBs

slides
bit.ly/abhi-disc

attendance
bit.ly/abhi-attendance

announcements

announcements

1. Homework 8 due Tuesday 4/19

announcements

1. Homework 8 due Tuesday 4/19
2. Week 13 Survey due Tuesday 4/19

announcements

1. Homework 8 due Tuesday 4/19
2. Week 13 Survey due Tuesday 4/19
3. Gitlet checkpoint due Friday 4/22

announcements

1. Homework 8 due Tuesday 4/19
2. Week 13 Survey due Tuesday 4/19
3. Gitlet checkpoint due Friday 4/22
4. Lab 14 due Friday 4/22

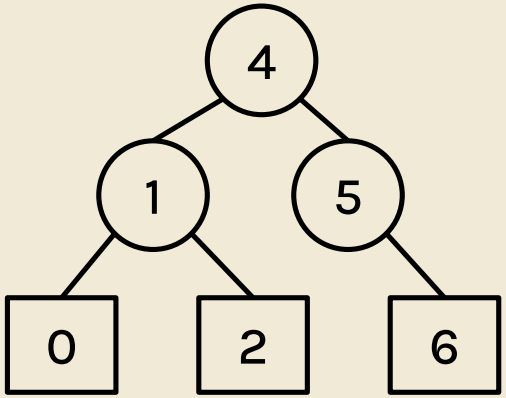
general questions, lecture, etc.

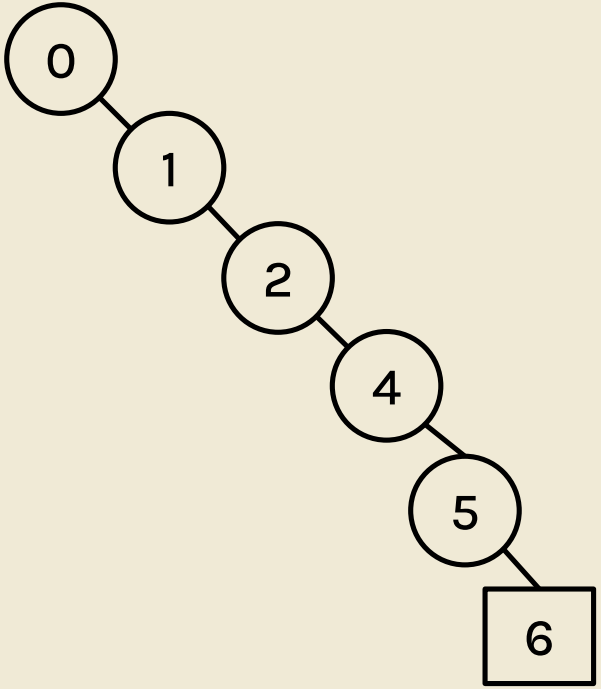
	Best Case Runtime	Worst Case Runtime	Space	Demo	Notes
Selection Sort	$\Theta(N^2)$	$\Theta(N^2)$	$\Theta(1)$	Link	
Heapsort (in place)	$\Theta(N)$*	$\Theta(N \log N)$	$\Theta(1)$	Link	Bad cache (61C) performance.
Mergesort	$\Theta(N \log N)$	$\Theta(N \log N)$	$\Theta(N)$	Link	Fastest of these ^
Insertion Sort (in place)	$\Theta(N)$	$\Theta(N^2)$	$\Theta(1)$	Link	Best for small N or almost sorted.
QuickSort LTHS (left pivote tony hoare, shuffled)	$\Theta(N \log N)$	$\Theta(N^2)$	$\Theta(\log N)$ (call stack)	Link	Empirically the fastest sort, rare worst case
LSD Radix Sort	$\Theta(WN+WR)$	$\Theta(WN+WR)$	$\Theta(N+R)$	—	Alphabetical only
MSD Radix Sort	$\Theta(N+R)$	$\Theta(WN+WR)$	$\Theta(N+WR)$	—	Bad caching (61C)

*: An array of all duplicates yields linear runtime for heapsort.
N: Number of keys. R: Size of alphabet. W: Width of longest key.

(Josh Hug 2021)

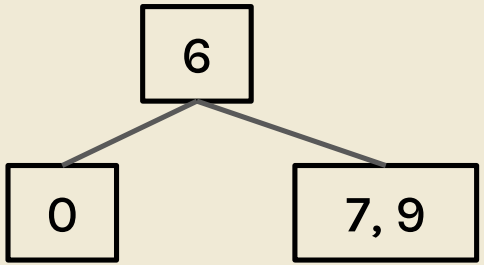
why do we need balance?





how do we get “bushy-ness?”

- guarantee all leaves are same distance from root
- fit multiple items into a node!



B-tree invariants

B-tree invariants

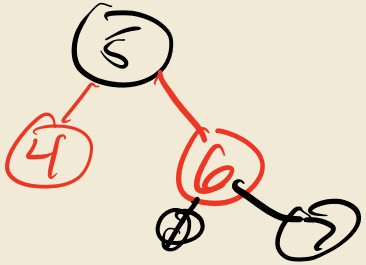
1. need all leaves to be the same distance from the root

B-tree invariants

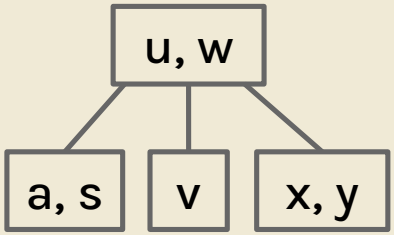
1. need all leaves to be the same distance from the root
2. inner (non-leaf) nodes with *k* items need *k+1* children

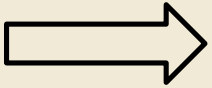
left-leaning red-black trees

- problem with B-tree implementation—complicated
- instead, introduce “glue nodes”—nodes colored red to represent a combination of two nodes



left-leaning red-black trees



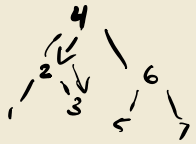


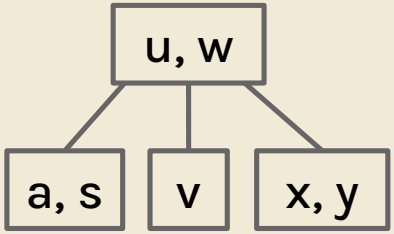
2-3 tree

(B-tree)

LLRB

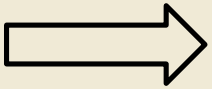
left-leaning red-black trees

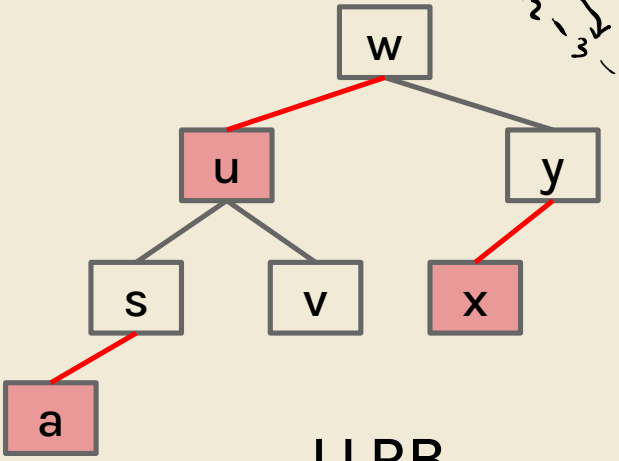




2-3 tree

(B-tree)

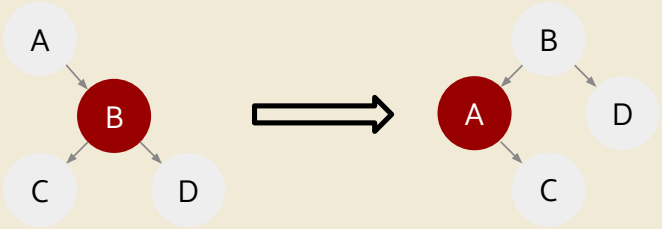




LLRB

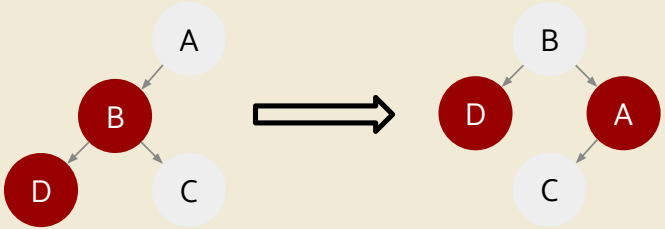
left-leaning red-black trees

rotateLeft(A);



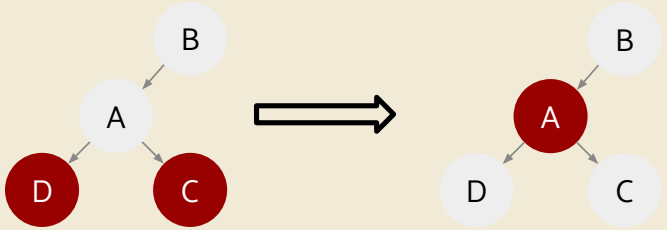
left-leaning red-black trees

rotateRight(A);



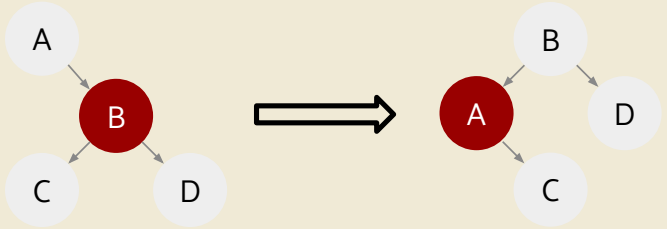
left-leaning red-black trees

colorFlip(A);

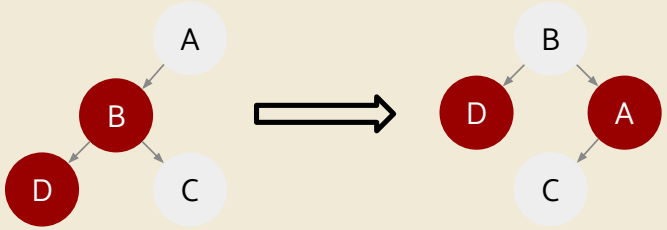


left-leaning red-black trees

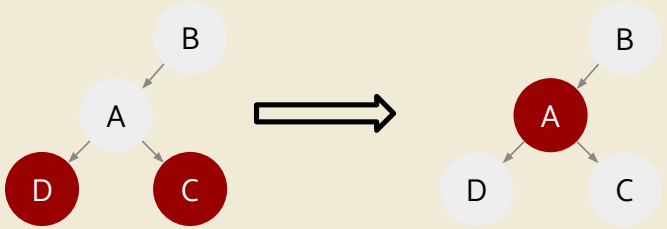
rotateLeft(A);

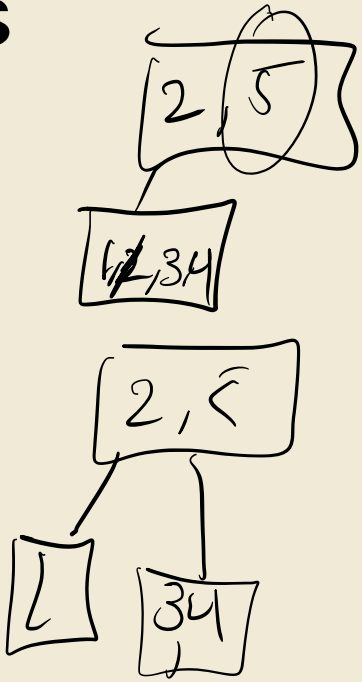


rotateRight(A);



colorFlip(A);



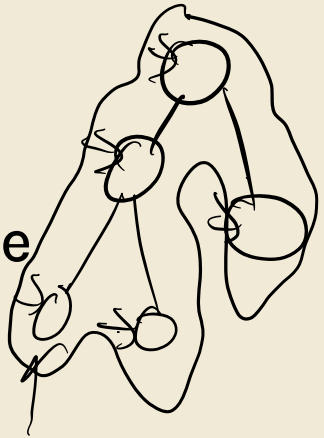


depth-first graph traversals

- **preorder:** visit node, then traverse children
- **inorder:** traverse left child, then node, then traverse right child
- **postorder:** traverse children, then visit node

depth-first graph traversals (informal)

start by tracing the graph counterclockwise



- **preorder:** visit “left sides” of nodes
- **inorder:** visit “bottom sides” of nodes
- **postorder:** visit “right sides” of nodes

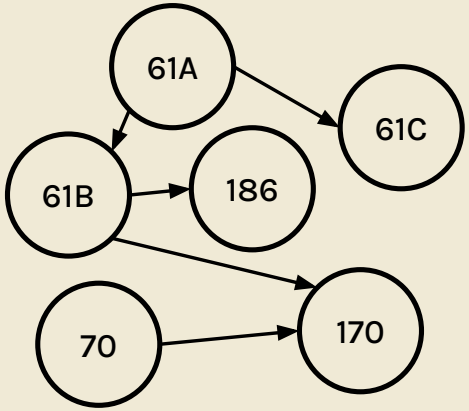
breadth first search (algorithm)

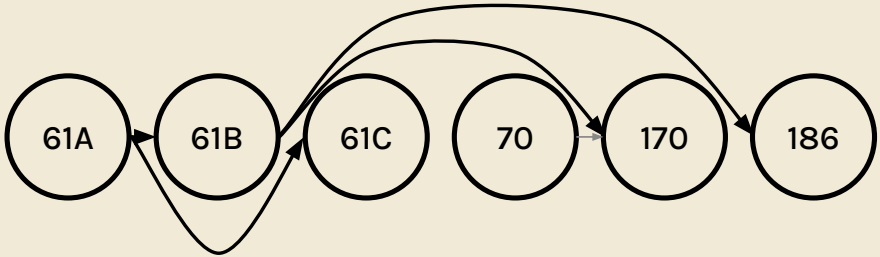
- put the starting node in the queue
- pop the first item from the queue (the starting node) and visit it
  - when you visit a node, add all its children/"neighbors" to the queue
- pop and visit the next item in the queue

continue until there are no more items in the queue!

topological sort

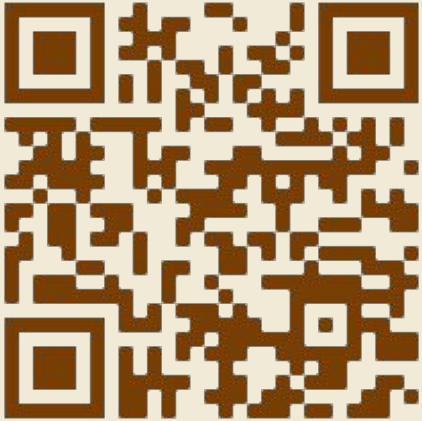
- take a cycle-less graph and “flatten” so all edges point the same way





worksheet
(on 61B website)

attendance
bit.ly/abhi-attendance



feedback
bit.ly/abhi-feedback

slides: **bit.ly/abhi-disc**