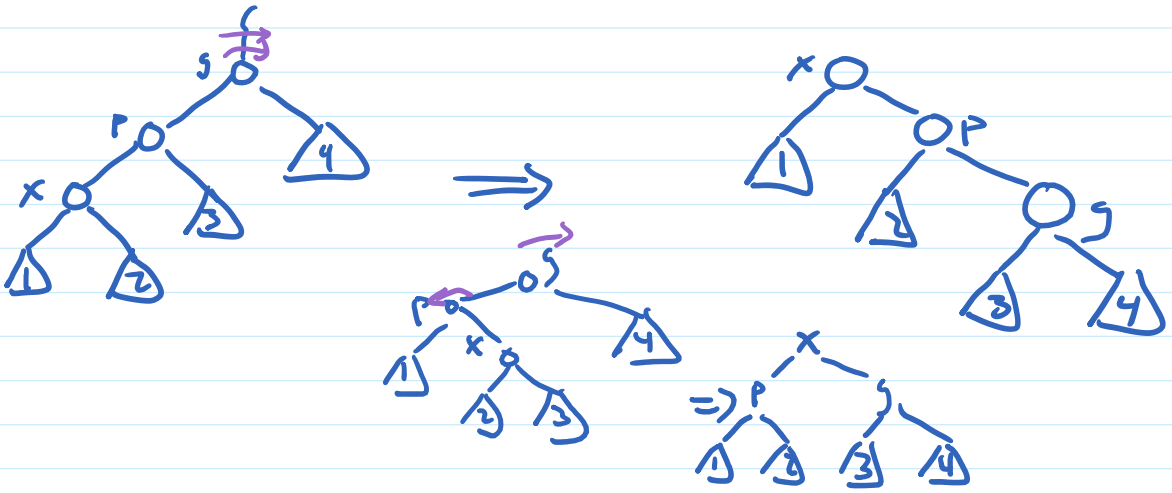
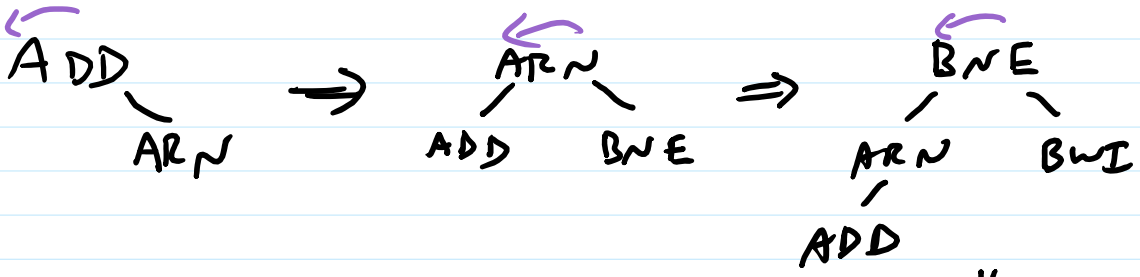
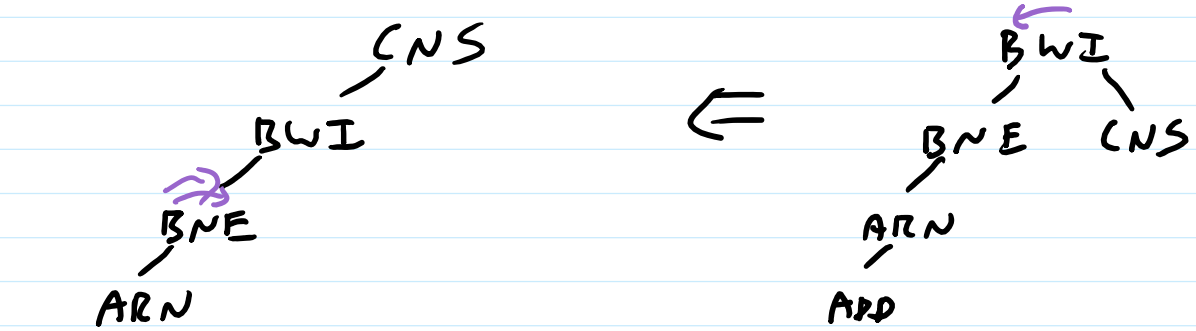
After any operation, rotate node to top 2 levels at a time

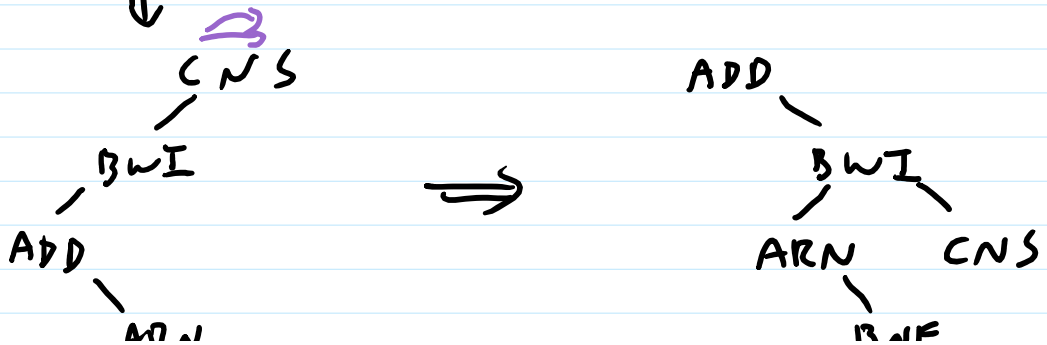


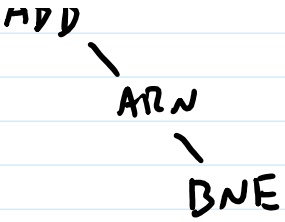
ADD ARN BNE BWI CNS CPT EZE LOS MEX PVG

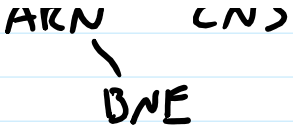




find(ADD)







amortized $O(\log n)$ time for find/add/remove

BONUS: Frequently accessed nodes usually near top

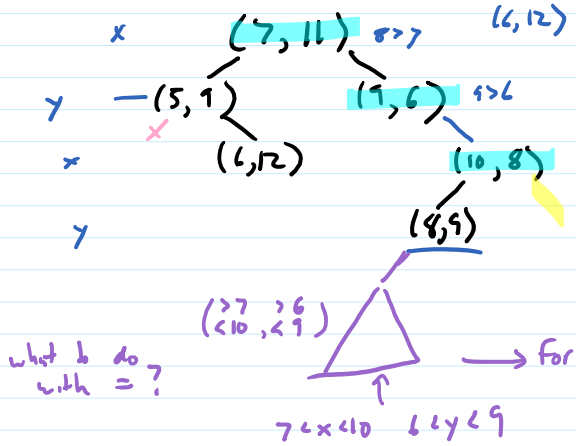
kd-Trees

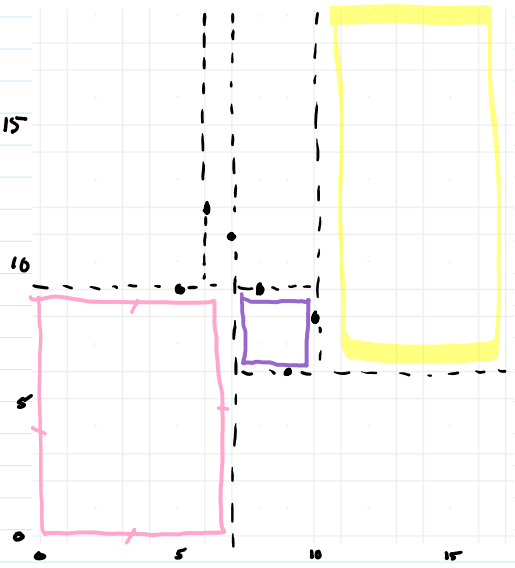
k = 2 for 2-D

Binary tree with BST order property rotating through dimensions

levels 0, k, 2k, ... 1st dimension

levels 1, k+1, ... 2nd dimension





what to do with =?

→ for range queries, keep track of bounds on coords of current node stop when no overlap w/ desired range

7 < x < 10 6 < y < 9

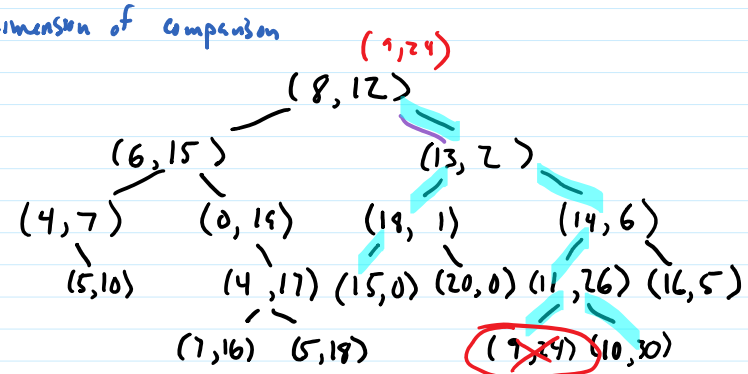
add / find

normalish BST add, but rotate through dimension of comparison

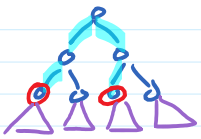
remove

find point to remove

remove (8,12)



recursively remove replacement until removing leaf



to find min x value
if current splits on y
recurse on both children
else
recurse on left child

every 2 levels we reduce to 2 subproblems of size $\frac{n}{4}$

$T(n) = 2T(\frac{n}{4}) + O(1)$ → Master Thm → $T(n)$ is $O(n^{\log_4 2})$

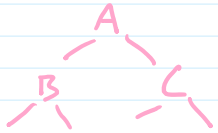
if tree balanced $O(\sqrt{n})$

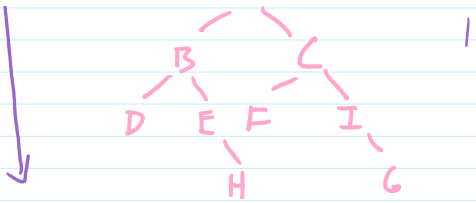
build

sort on x: E H D B A I C G F

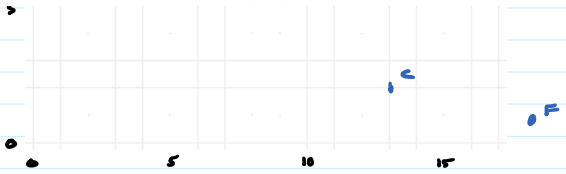
build right subtree I C G F
build left subtree E H D B

find median y by sorting: F C G I





sort on y : D B H E



build
sort on curr dim $O(n \log n)$
root ← median
left ← build (pts before)
right ← build (pts after)

$O(n \log^2 n)$ can do better

really just want median
can find median in expected $O(n)$ [easy]
worst-case $O(n)$ [hard]

1) sort on x
2) sort on y

build (x, y) ⟶ build (cut, other)

same points, sorted differently

cut E H D B A I C G F
other F C G D A B H I E

root ← median (cut)

cut_L ← 1st half of cut
cut_R ← 2nd half of cut

can do in $O(n)$ (no sort)

$other_L$ ← pts from other in L
$other_R$ ← pts from other in R

D B H E
F C G I

left ← build ($other_L$, cut_L)
right ← build ($other_R$, cut_R)

$T(n) = 2T(\frac{n}{2}) + O(n)$

$O(n \log n)$