FINDING THE RANK OF AN ELEMENT IN A SET

Use array:

P	F	C	H	Q	A	N	D	M
1	2	3	4	5	6	7	8	9

rank(F) = ?

→ partition

C	A	D	F	P	H	Q	N	M
1	2	3	4	5	6	7	8	9

$\Theta(n)$

OK if done once.

Not for multiple queries

Preprocess (sort)

A	C	D	F	H	M	N	P	Q
1	2	3	4	5	6	7	8	9

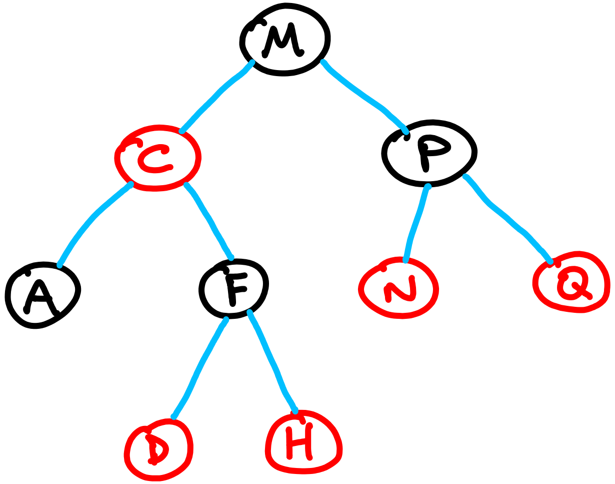
$O(n\log n)$

Now all queries: $O(1)$

What if we want to insert/delete? → bad $O(n)$

FINDING THE RANK OF AN ELEMENT in a DYNAMIC SET with PREPROCESSING

(allow insertions & deletions "quickly")



RB-tree contains sorted letters

Now we can quickly restore sorted order

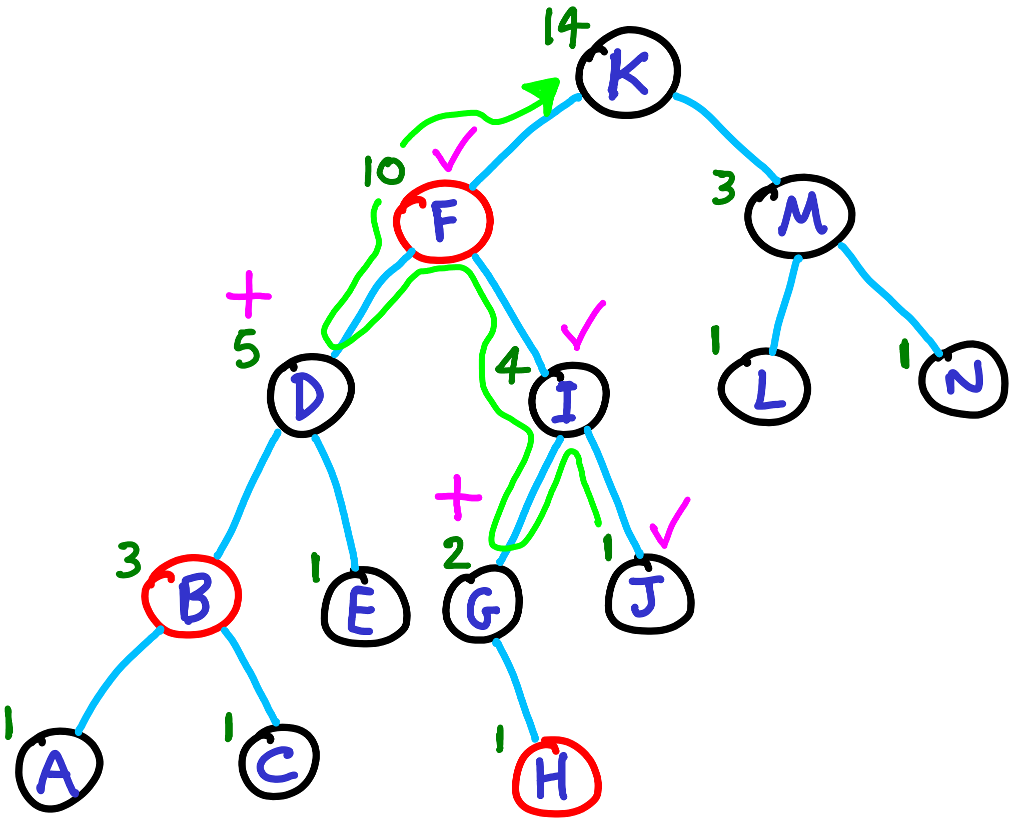
Store ranks... → bad

(too many ranks change w/ insert)

Dynamic X

USING AN AUGMENTED R-B TREE TO FIND RANKS

(with subtree sizes)

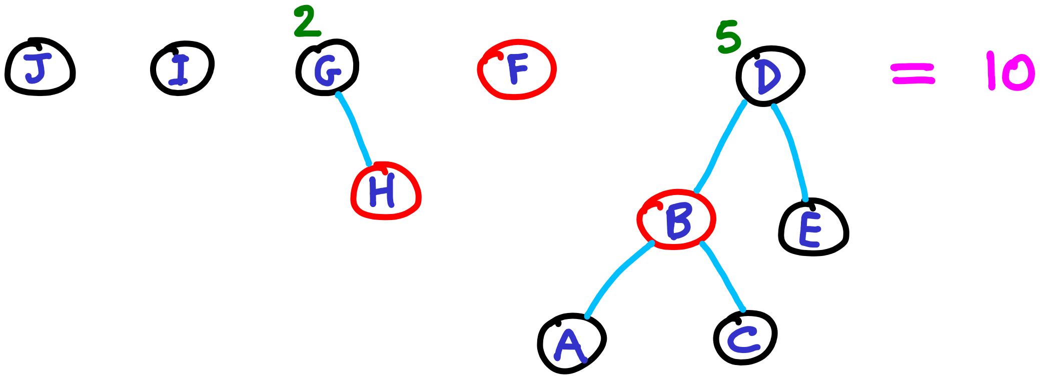


Rank(J) : Walk up,

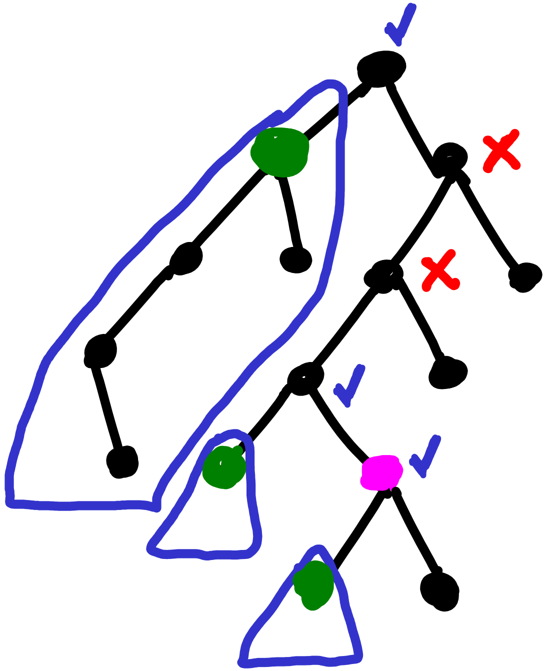
✓ count smaller ancestors,

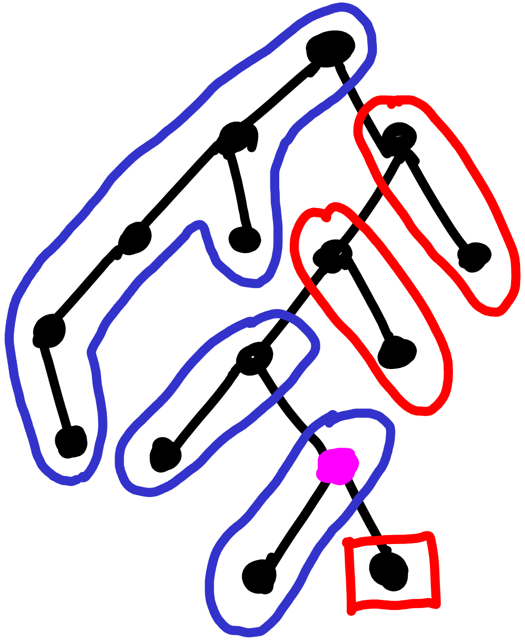
but also

+ add sizes of subtrees containing smaller numbers.



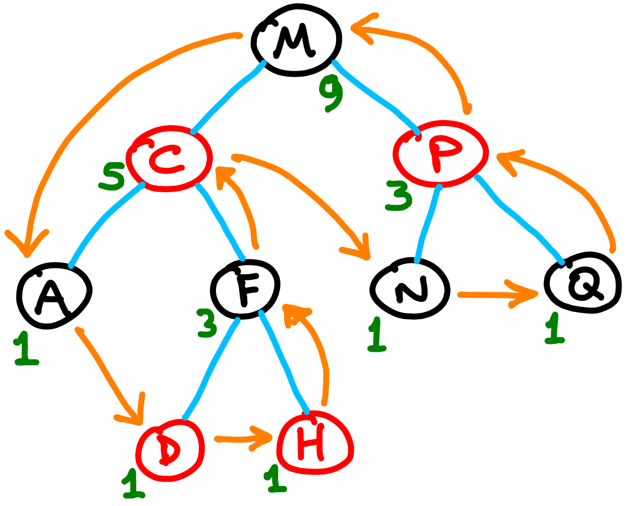
Must walk all the way up





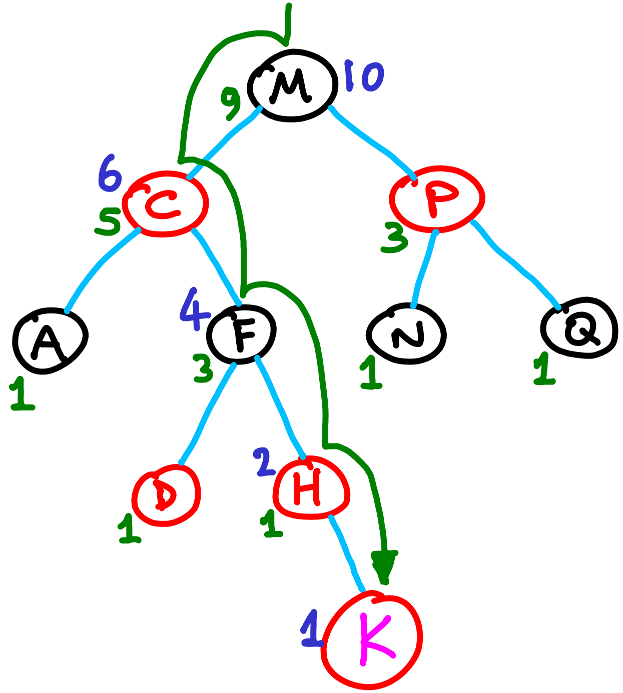
$O(\log n)$ time

The balanced BST can be built in $\Theta(n \log n)$ time



Compute subtree sizes after building by postorder walk...

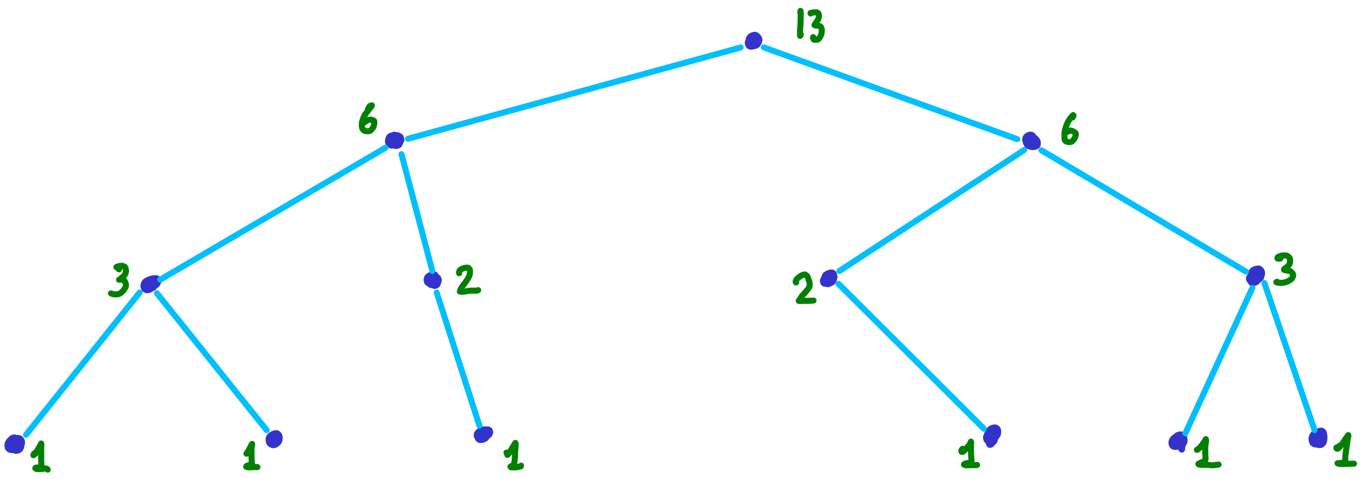
... or update path when inserting



BUT...

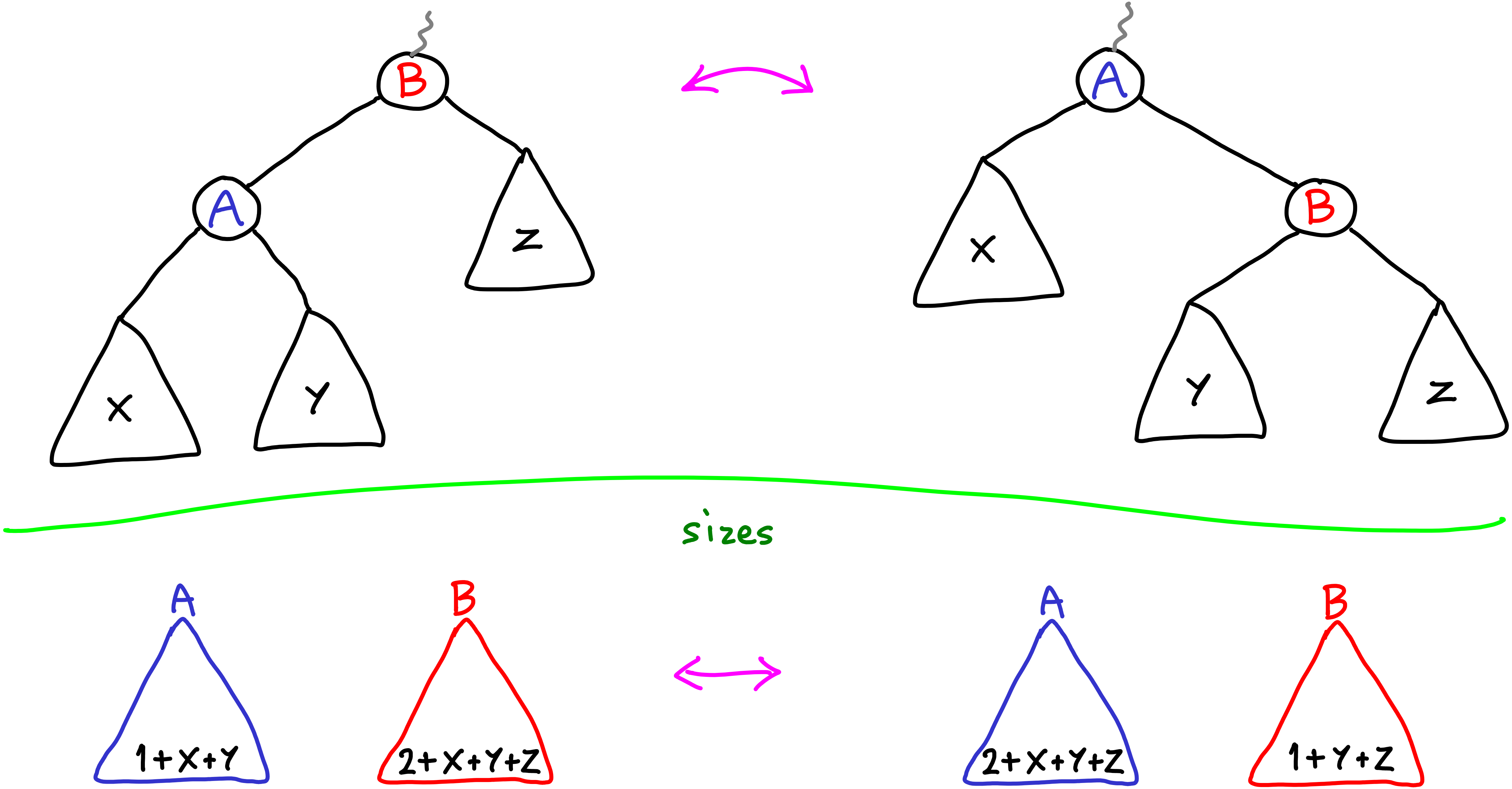
we will need to rebalance

Can we update subtree sizes when inserting/deleting data?



Use a RB tree

↳ when are subtree sizes affected? Rotations


B
A
z
x
y
A
B
x
y
z
sizes
A
1+x+y
B
2+x+y+z
A
2+x+y+z
B
1+y+z

AUGMENTED TREE TO FIND RANKS

- easy to find rank:
  - look at ancestor path & some adjacent subtree sizes

- subtree sizes can be updated when inserting and rebalancing

$O(\log n)$ per search/insertion/deletion

DYNAMIC SELECTION

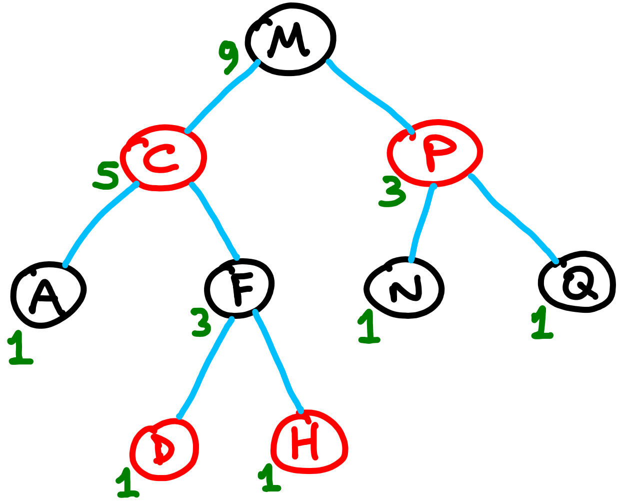
find the i-th smallest element in a set

Static: $\Theta(n)$

Dynamic: $O(n\log n)$ preprocessing → balanced BST w/ subtree sizes

$O(\log n)$ query / insert / delete

(similar … just need to see how to query)



```
Select(x, i)   \\ get i-th element in subtree rooted at x.
    $k \leftarrow 1 + \text{size}(l_x)$   \\ $l_x$ : left child of x
    if $i = k$ , return x.
    else if $i < k$, return Select($l_x$, i)
        else ($i > k$) return Select($r_x$, i-k)
```

example: $i=5$

$k=6$

$i=5$, $k=2$

$i=3$, $k=2$

$i=1$, $k=1$

```
Select(root, 5)
    $k \leftarrow 1 + 5$
    $i < k \Rightarrow$ Select(C, 5)
        $k = 1 + 1$
        $i > k \Rightarrow$ Select(F, 3)
            $k = 1 + 1$
            $i > k \Rightarrow$ Select(H, 1)
                $k = 1 + 0$
                $i = k \Rightarrow$ return H
```