

Binary Search Trees: Splay Trees: Operations

Daniel Kane

Department of Computer Science and Engineering
University of California, San Diego

Data Structures Fundamentals
Algorithms and Data Structures

Learning Objectives

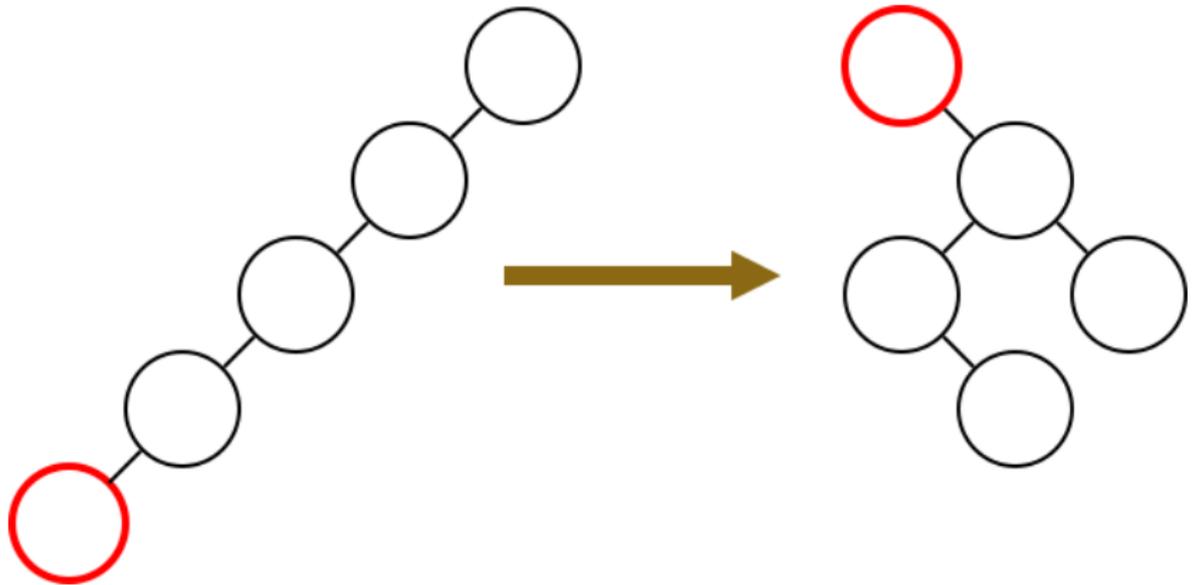
- Implement a splay tree.

Last Time

- Idea: bring each query node to the root.
- Introduced splay operation to bring node to root.

Sometimes Slow

Splay operation is sometimes slow:



Amortized Analysis

Need to amortize.

Theorem

The **amortized** cost of doing $O(D)$ work and then splaying a node of depth D is $O(\log(n))$ where n is the total number of nodes.

Amortized Analysis

Need to amortize.

Theorem

The **amortized** cost of doing $O(D)$ work and then splaying a node of depth D is $O(\log(n))$ where n is the total number of nodes.

We will prove this later, but using it we can implement all our operations.

Find

$\text{STFind}(k, R)$

$N \leftarrow \text{Find}(k, R)$

$\text{Splay}(N)$

return N

Runtime

Suppose node at depth D .

- $O(D)$ time to find N .
- Splay N .
- Amortized cost $O(\log(n))$.

Runtime

Suppose node at depth D .

- $O(D)$ time to find N .
- Splay N .
- Amortized cost $O(\log(n))$.

You pay for the work of finding N by splaying to rebalance the tree.

Important Point

Even if you fail to find k , you must still splay the closest node you found. Otherwise your operation did $O(D)$ work with nothing to amortize against.

Insert

Insert, then splay

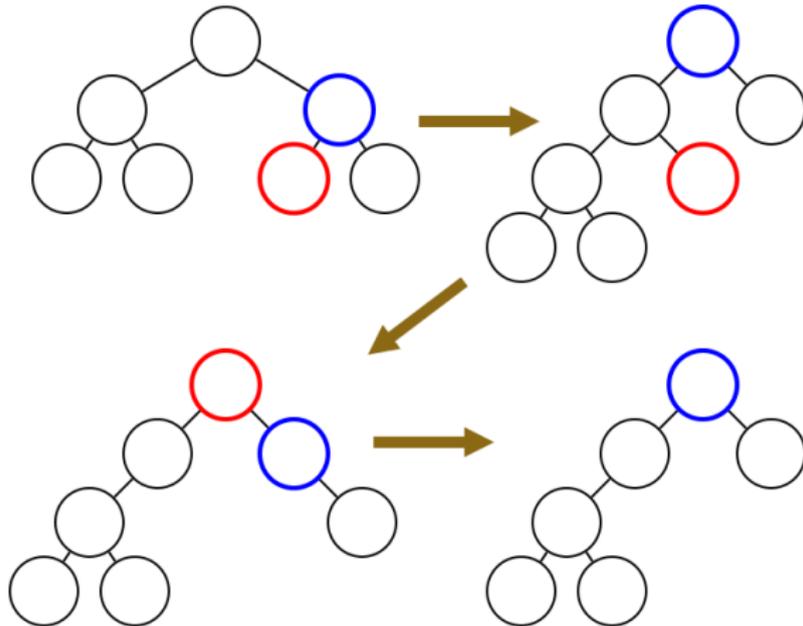
$\text{STInsert}(k, R)$

$\text{Insert}(k, R)$

$\text{STFind}(k, R)$

Delete

Bring N and successor to top. Deletes easily.



Delete

STDelete(N)

Splay(Next(N))

Splay(N)

$L \leftarrow N.\text{Left}$

$R \leftarrow N.\text{Right}$

$R.\text{Left} \leftarrow L$

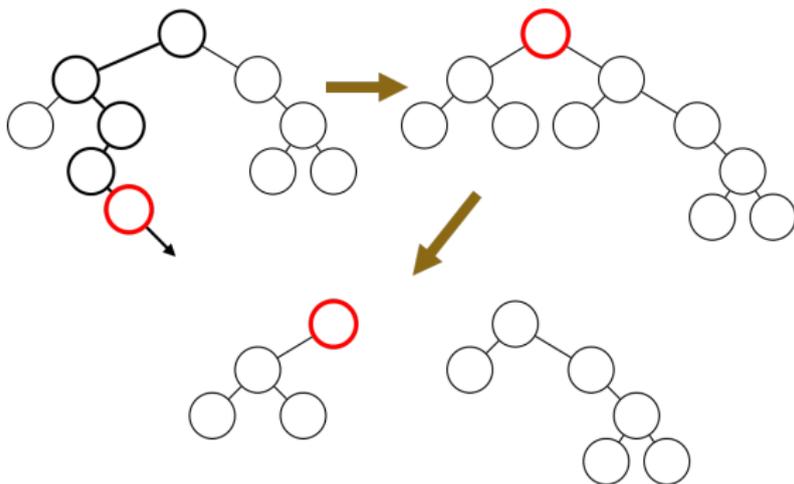
$L.\text{Parent} \leftarrow R$

Root $\leftarrow R$

$R.\text{Parent} \leftarrow \text{null}$

Split

Splay and Cut.



Split

STSplit(R, x)

$N \leftarrow \text{Find}(x, R)$

Splay(N)

if $N.\text{Key} > x$:

 return CutLeft(R)

else if $N.\text{Key} < x$:

 return CutRight(R)

else

 return $N.\text{Left}, N.\text{Right}$

CutLeft

CutLeft(N)

$L \leftarrow N.\text{Left}$

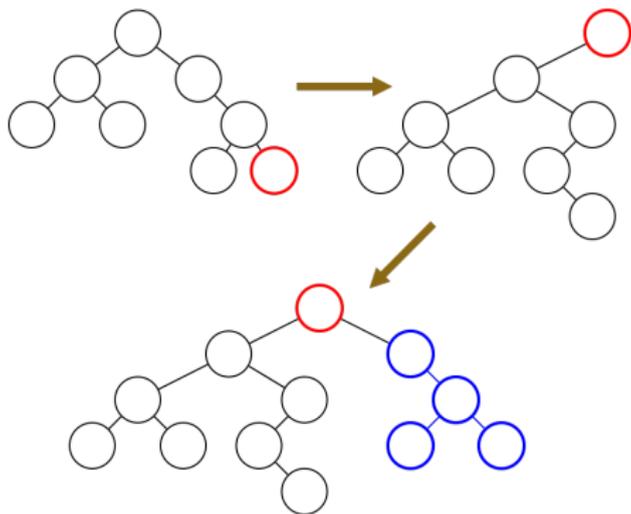
$N.\text{Left} \leftarrow \text{null}$

$L.\text{Parent} \leftarrow \text{null}$

return L, N .

Merge

Splay largest element of first tree, and stick second tree as child of the root.



Note after splay, root has no right child.

Merge

STMerge(R_1, R_2)

$N \leftarrow \text{Find}(\infty, R_1)$

Splay(N)

$N.\text{Right} \leftarrow R_2$

$R_2.\text{Parent} \leftarrow N$

Summary

Splay trees perform all operations simply in $O(\log(n))$ amortized time.