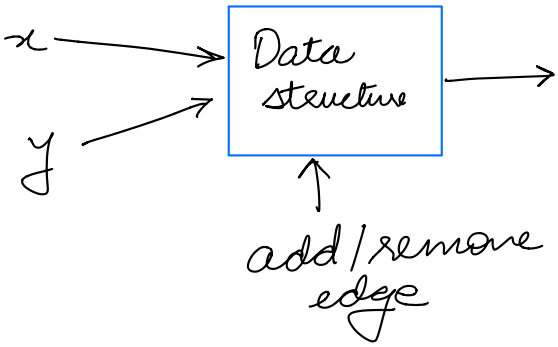
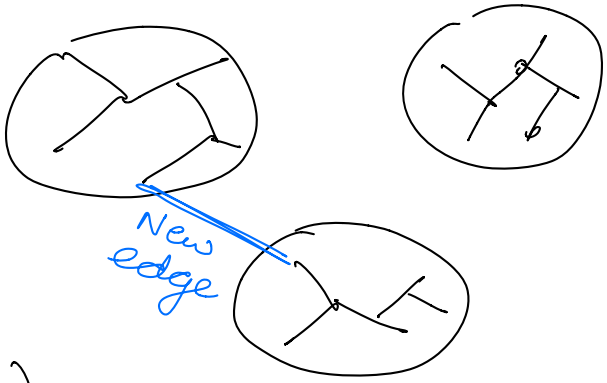
Problem:



Output in $O(\log n)$ time if x, y are connected

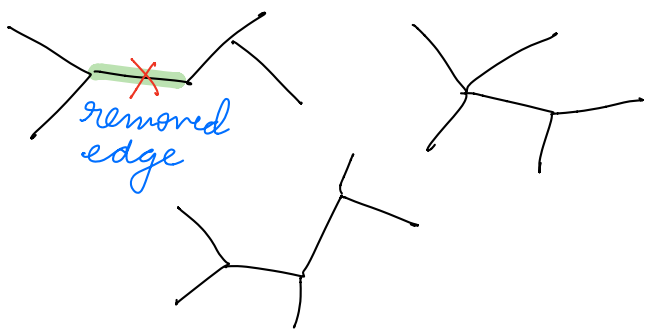
Edge Insertion

→ Union find data-structure



Update time = $O(\log^* n)$

Edge deletion



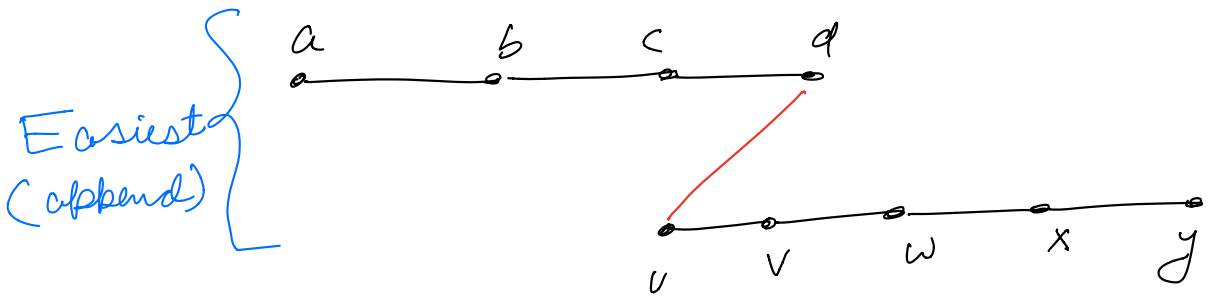
This lecture:

- Fully dynamic connectivity for
  - G is collection of paths
  - G is collection of trees

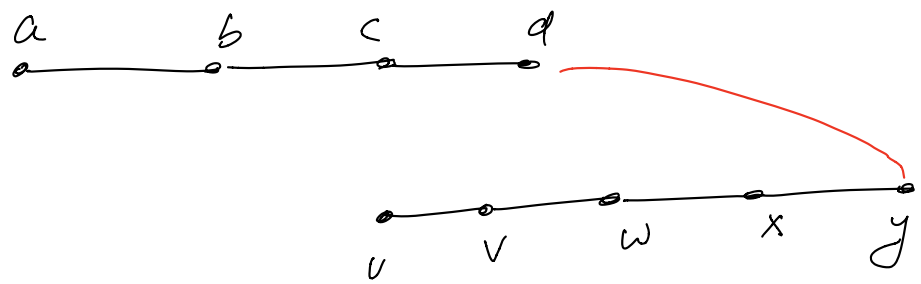
Simpler Problem: Dynamic connectivity on Paths

It is gauranteed that G is alway collection of paths

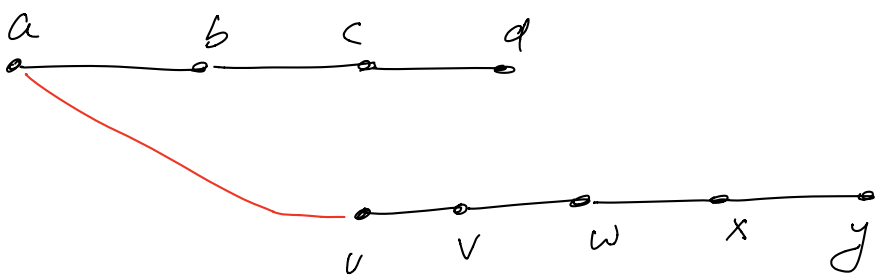
(1) Add an edge b/w 2 paths



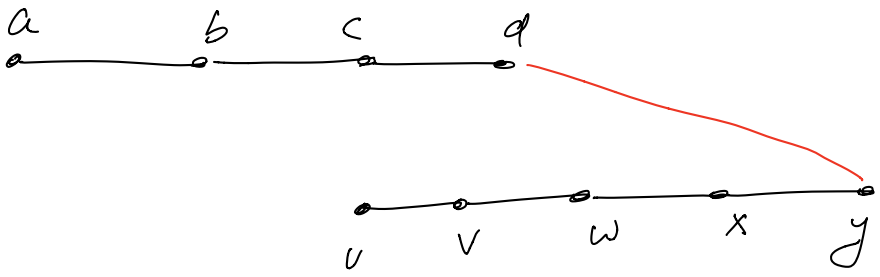
Or



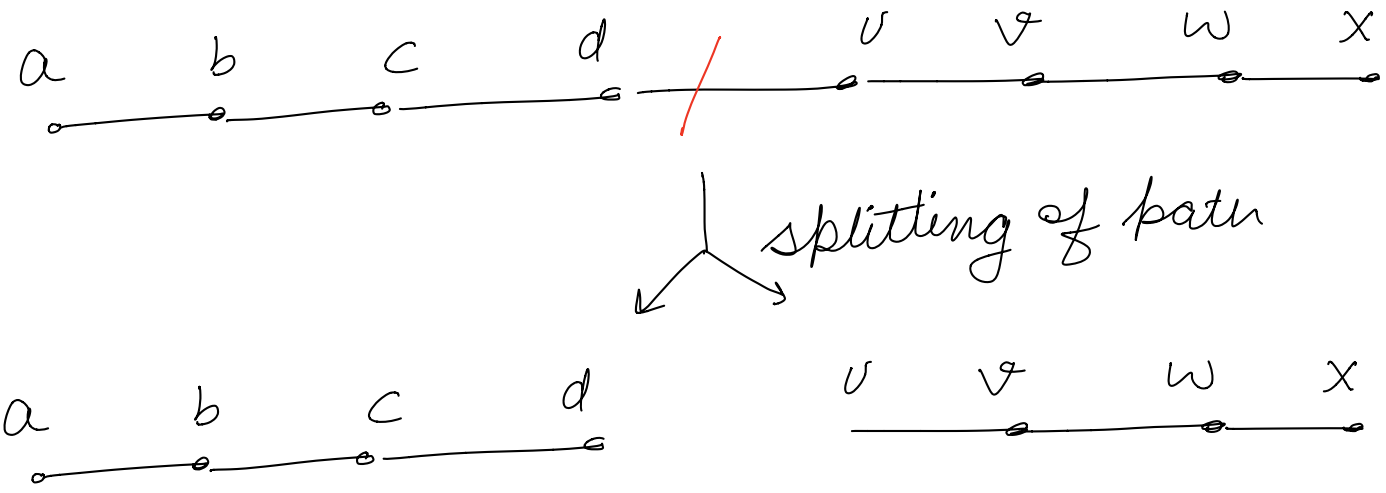
Or



Or



(2) Remove an edge from a path



How to achieve this in $O(\log n)$ time?

We maintain paths as binary search trees using AVL tree data structure.

We need data structure to

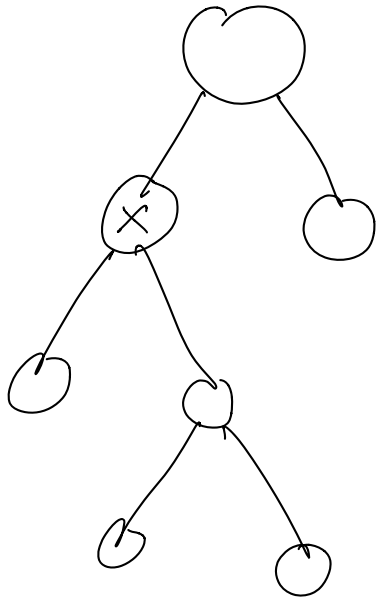
- Merge 2 AVL trees
- Split a AVL tree
- Spin an AVL tree

} $O(\log n)$ time

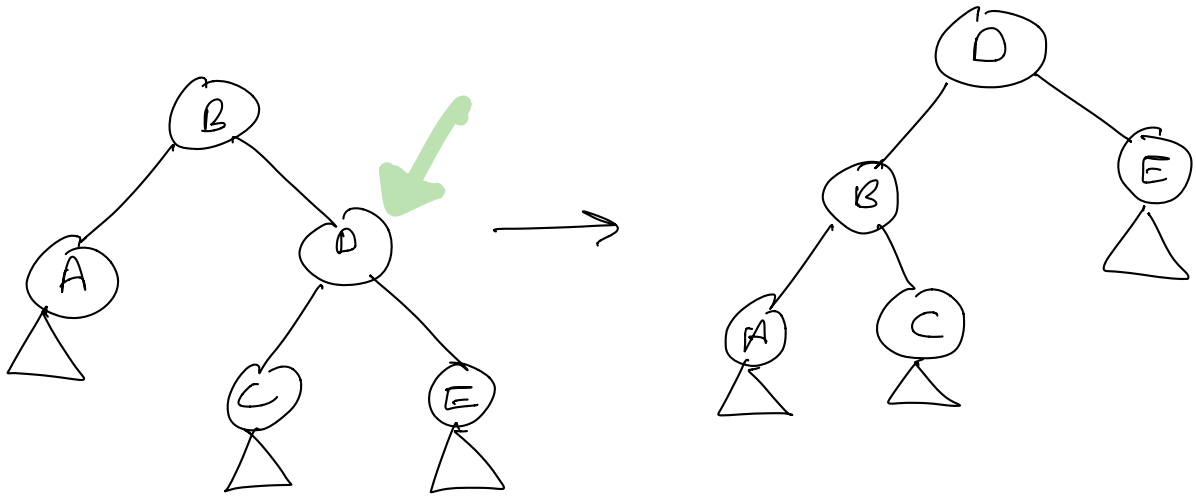
What is AVL tree?

$BF(x) = height(Right\ subtree) - height(left\ subtree)$

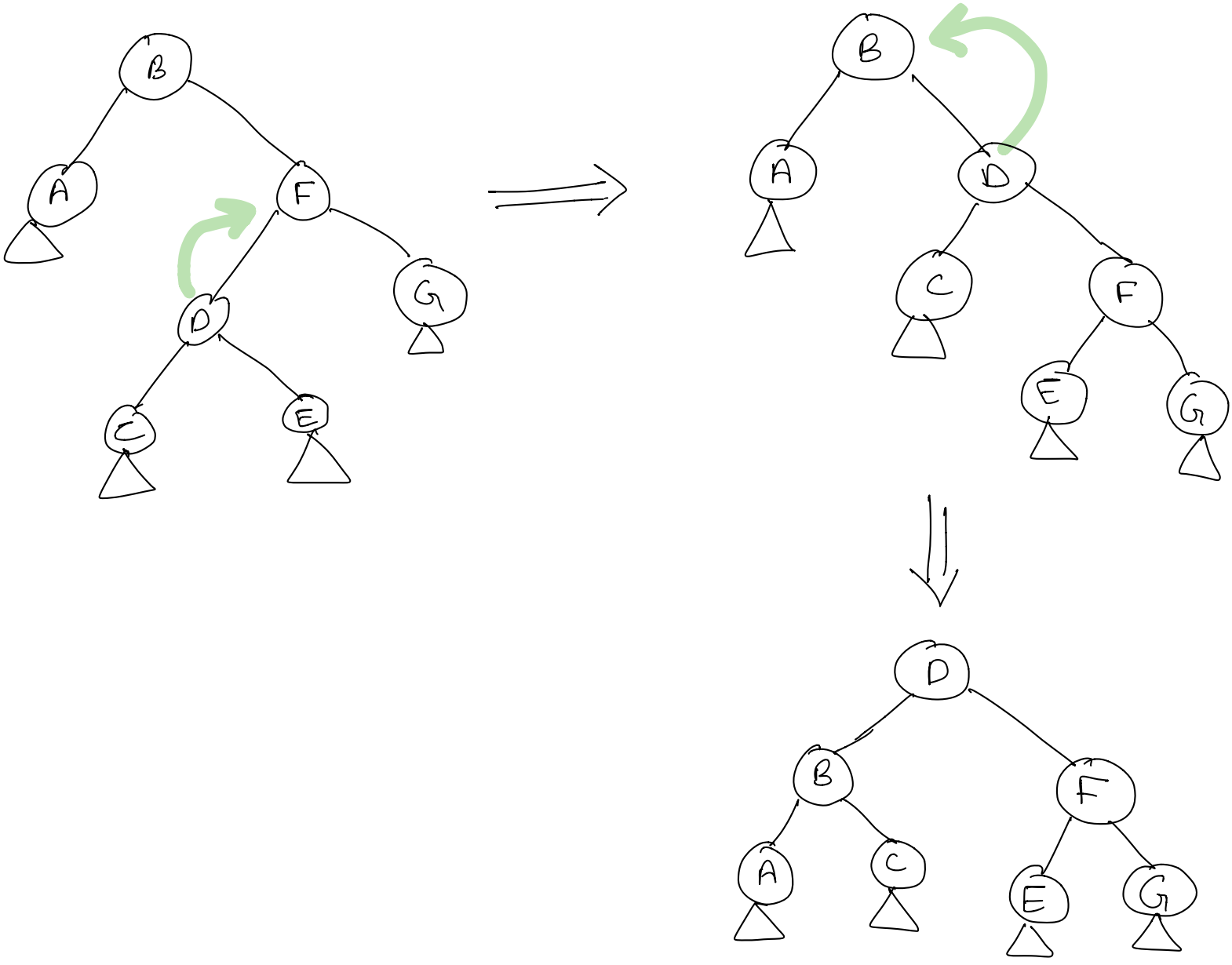
$\in \{-1, 0, 1\}$



Single Rotation:



Double Rotation:

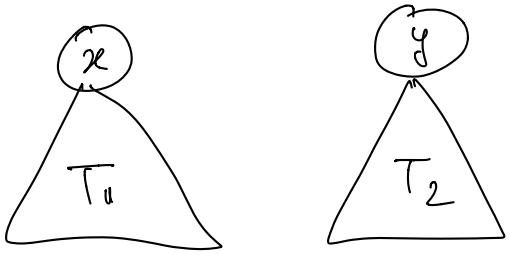


Lemma:

Using singe / double rotation we can rebalance tree after insertion / deletion

Proof: H.W.

How to merge 2 AVL trees:



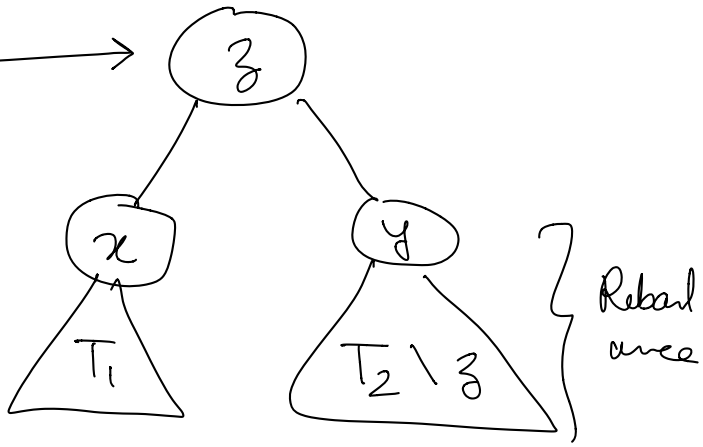
(assume entries of $T_1 \leq T_2$)

Step 1:

Take out $z \leftarrow$ smallest element of T_2

Step 2: output

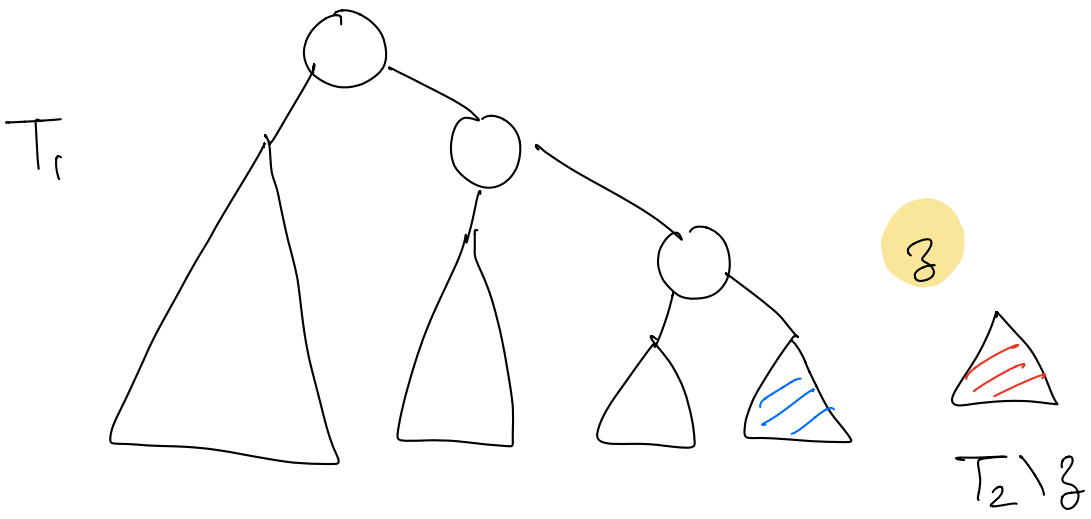
This node may not be balanced. →



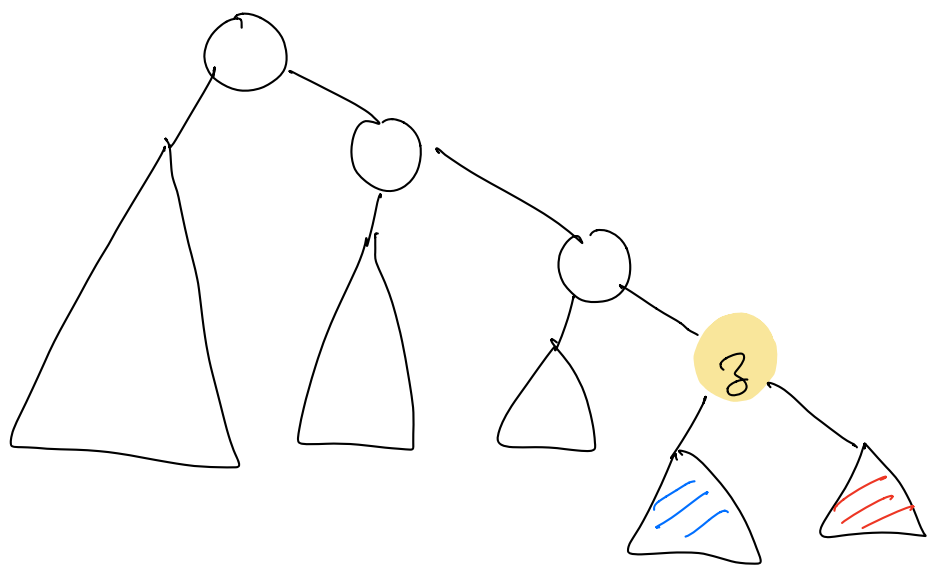
Time complexity = O(height of T_2)

$= O(\log n)$

Assume height $(T_2 \setminus 3) <$ height (T_1)



$\Rightarrow$



Time to find $3 = O(\log n)$

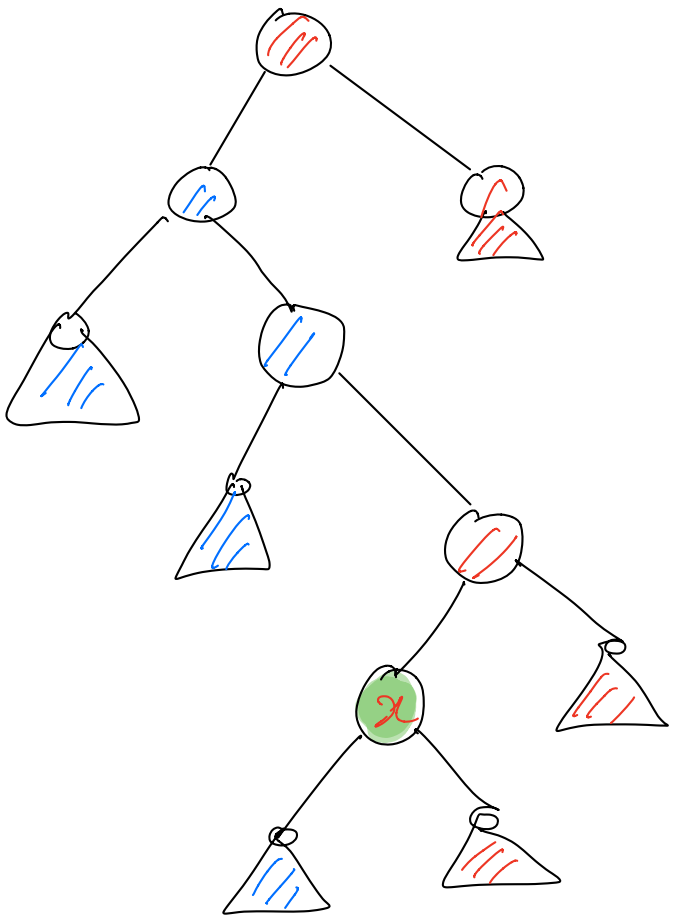
Time to merge $= O(\log n)$

} H.W.

How to perform split?

Suppose $x \in T$ and split at $\leq x$, $> x$

2n tree T_1 | 2n tree T_2



$\log n$ nodes & trees — just simply merge them.

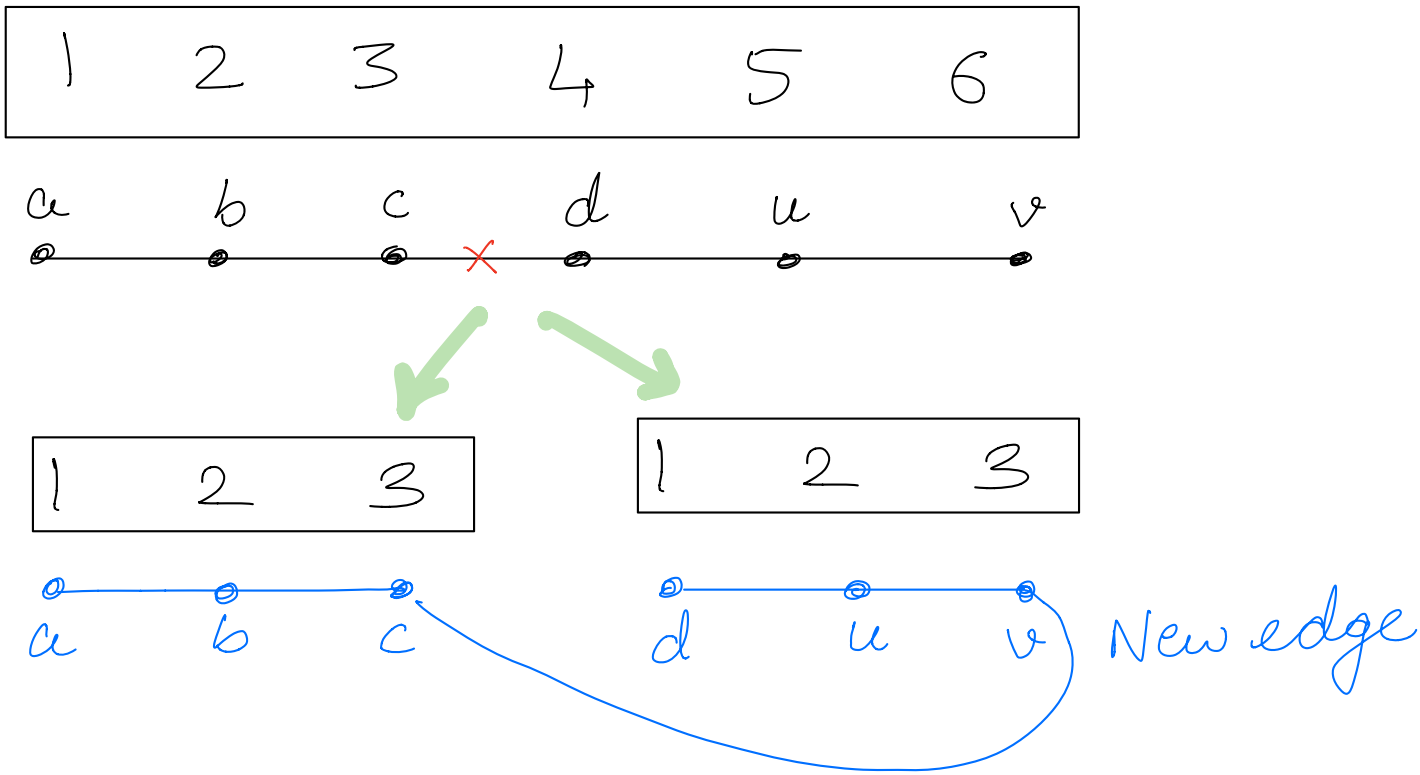
Time $= O(\log^2 n)$

↓

Can be improved to $O(\log n)$ as well.

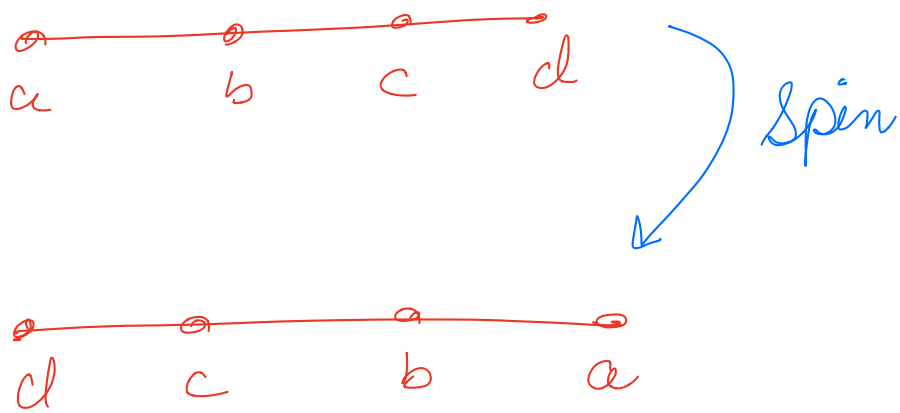
Example

Split $x = 3$

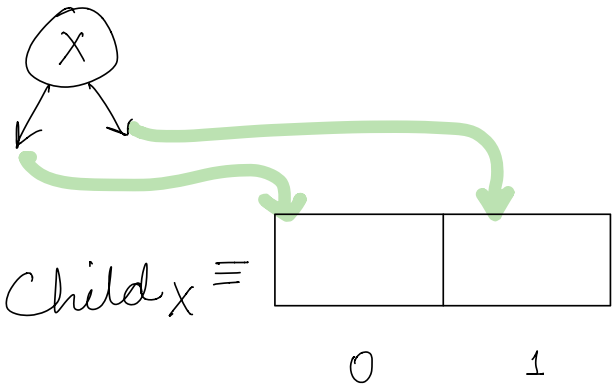


We need 3 operations:

- Merge
- Split
- Spin



How to efficiently perform Spin Operation ?

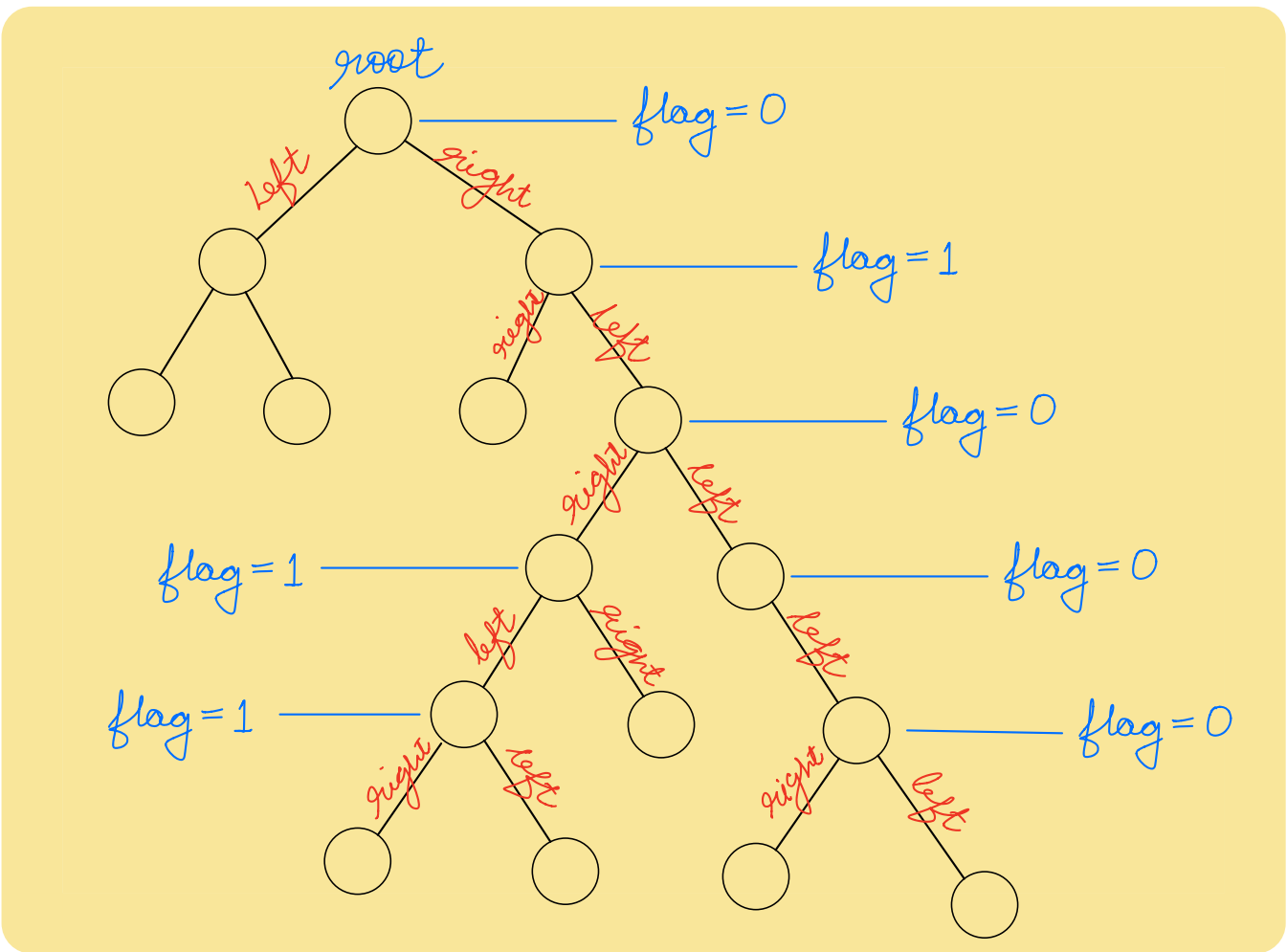


$flag \in \{0, 1\}$

For root:

$$Left(x) = Child_x\left[(0 + flag) \bmod 2\right]$$

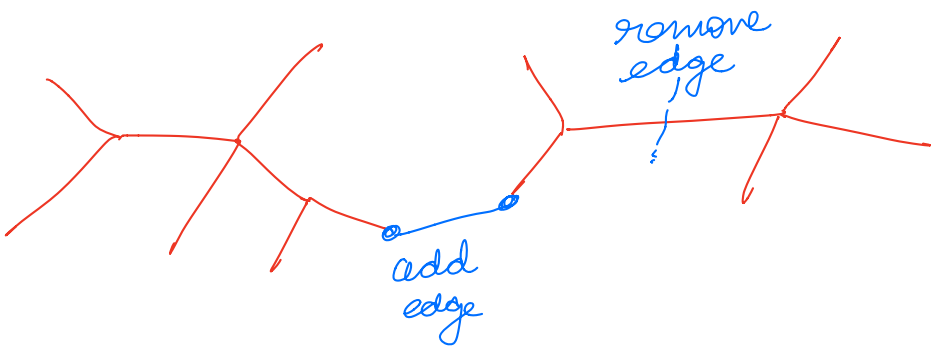
$$Right(x) = Child_x\left[(1 + flag) \bmod 2\right]$$



Two vertices are connected if their roots are same. — can be verified in $O(\log n)$ time.

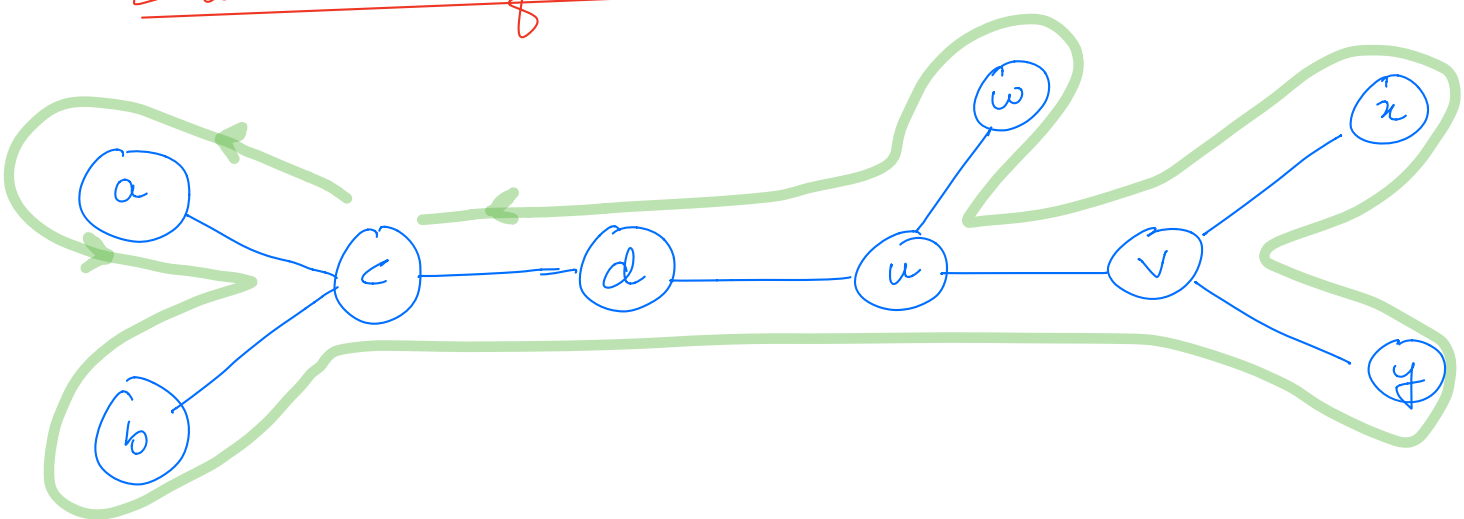
↖ height of AVL tree

Problem: Dynamic connectivity on Trees



Ques: How to represent trees as linear structure

Euler tour of a tree



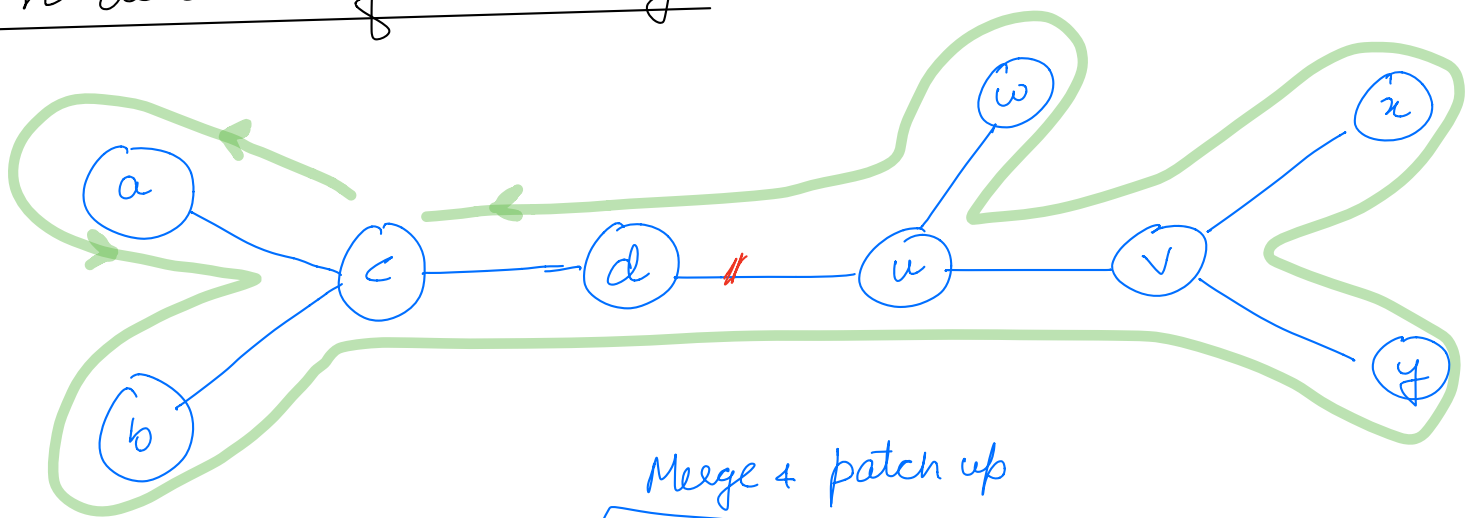
c a c b c d u v y v x v u w u d c

→ A leaf node appears only once.

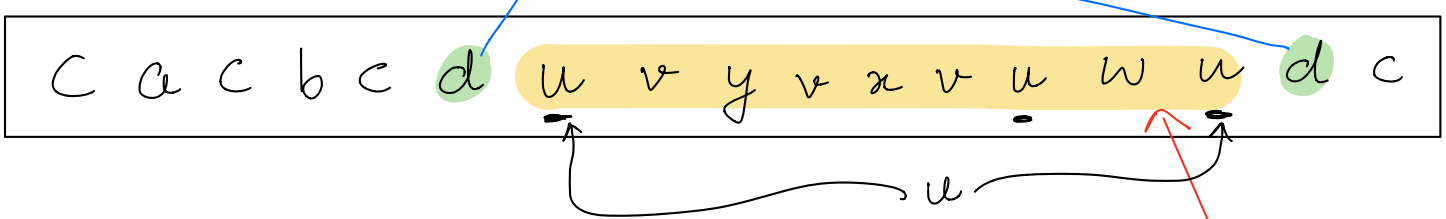
→ Non-leaf node appears degree times

→ Root appears (deg+1) times

On deletion of (u,d) edge:



Merge & patch up

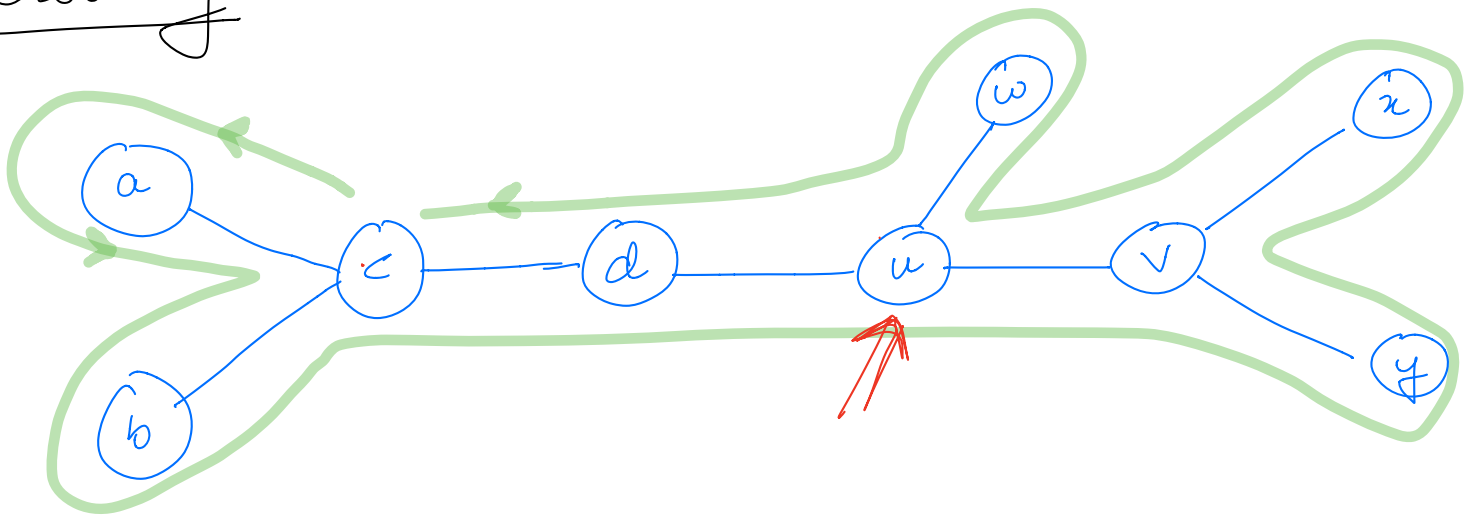


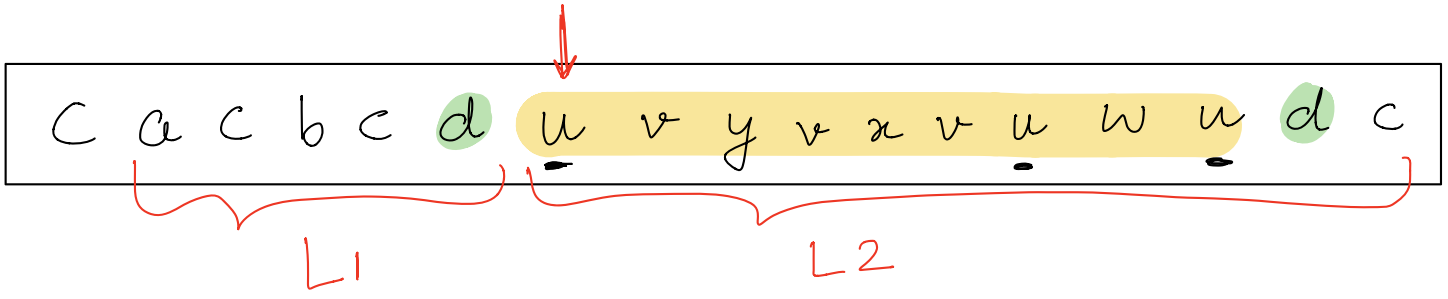
We split the euler tour and lists

For each vertex z maintain a pointer to first & last occurence of z in list

How to retrieve this sublist.

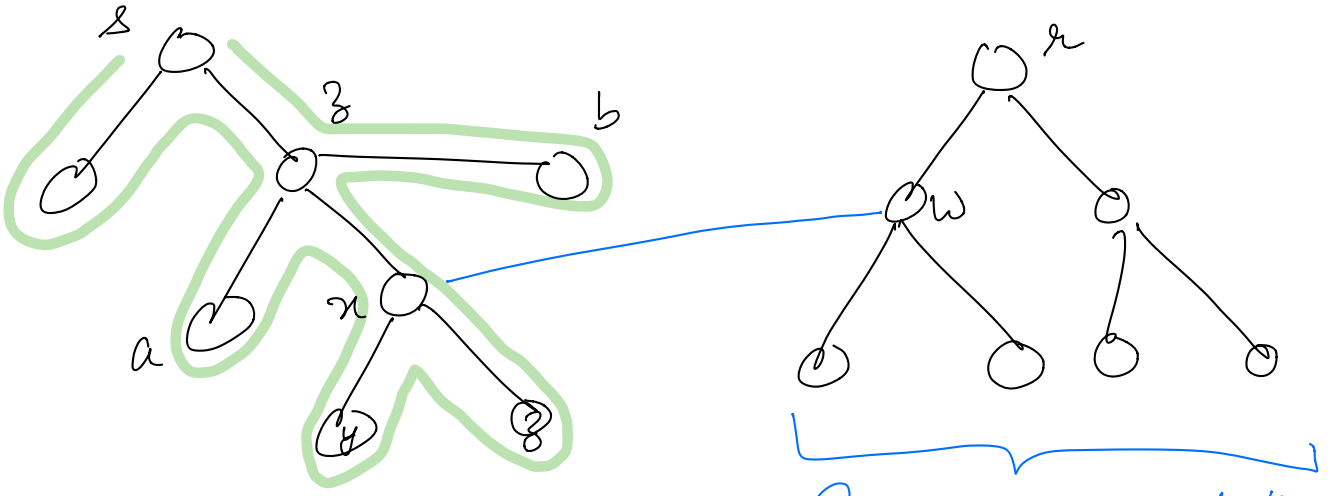
Rerooting :

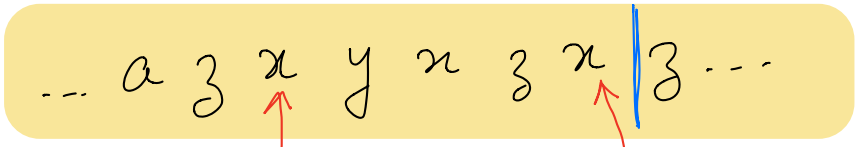


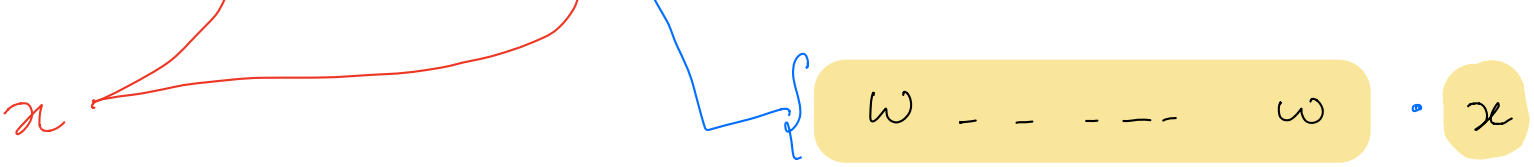


We need to create L2 · L1 · u

Addition of edge :







Time for insertion / deletion : $O(\log n)$ worst case

Query : $O(\log n)$ — just need to compare roots of BST.