

07-D

BST Application

Range Tree

顺藤摸瓜

邓俊辉

deng@tsinghua.edu.cn

Coherence

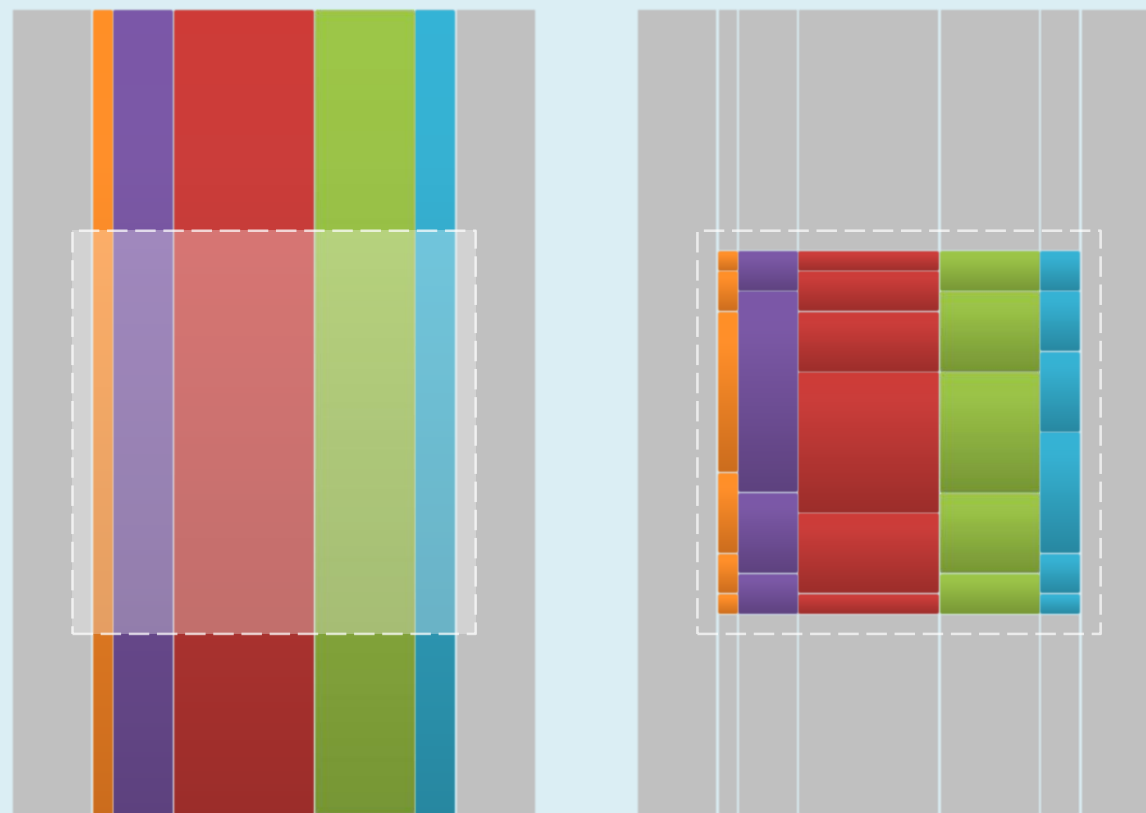
❖ For each query, we

- need to repeatedly search

DIFFERENT *y*-lists,

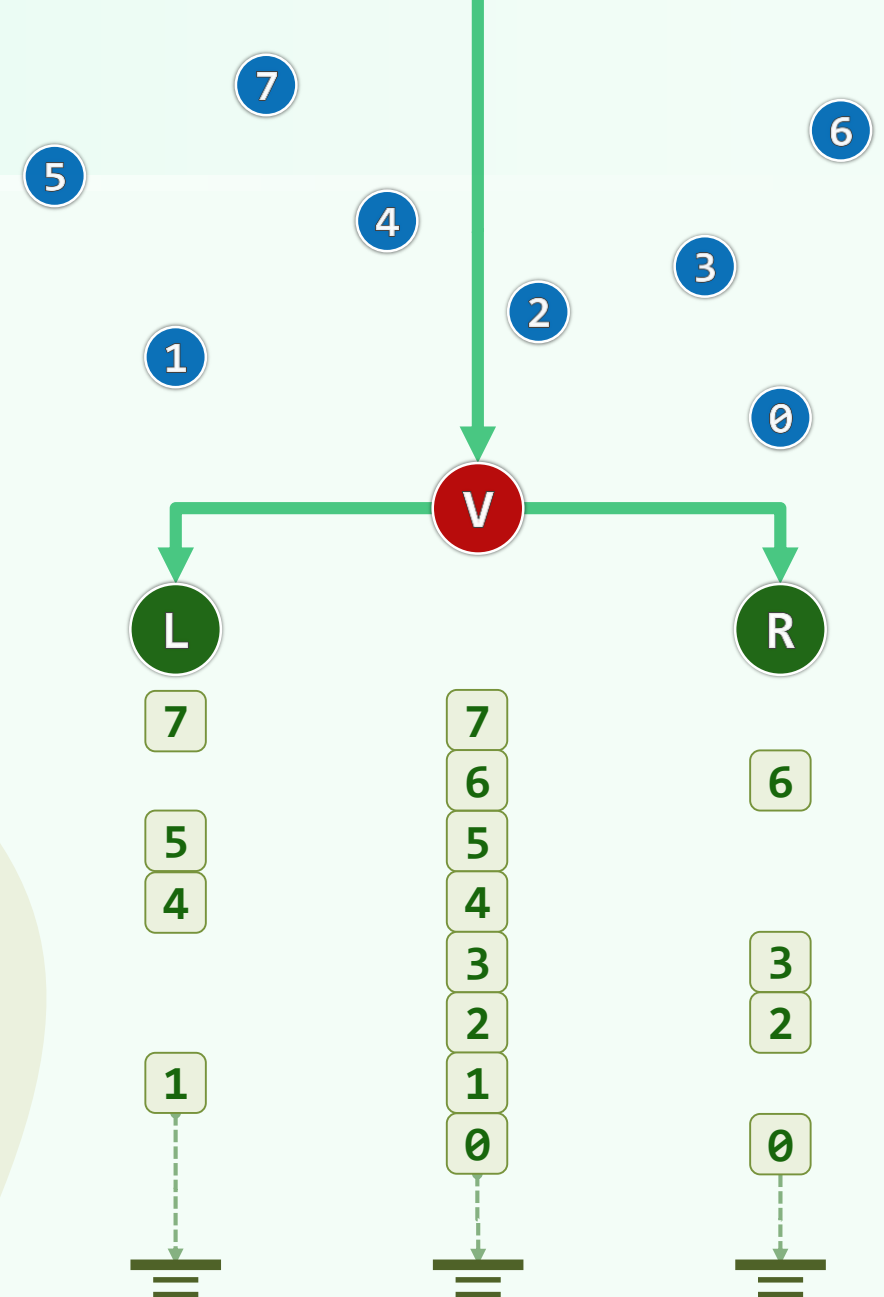
- but always with

the **SAME** key



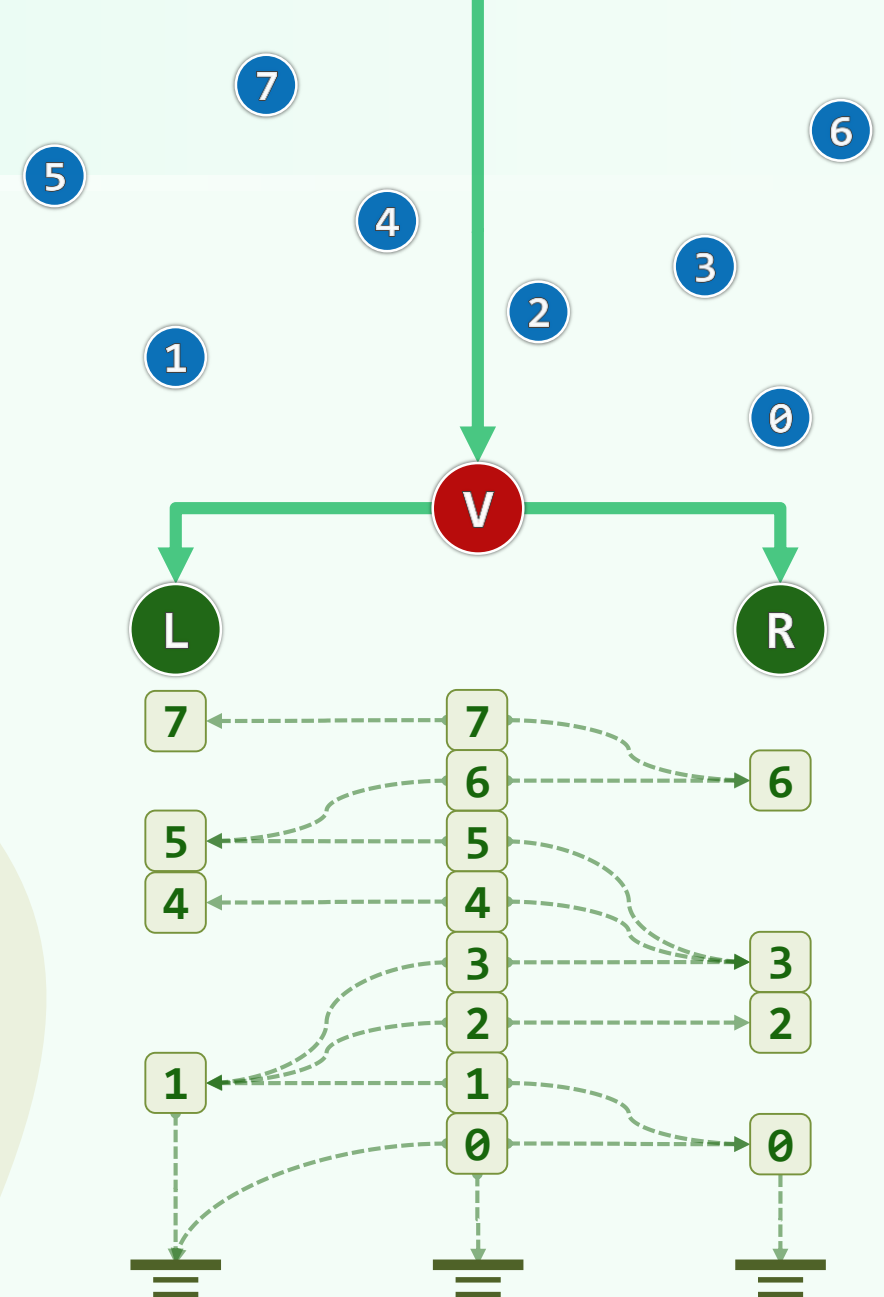
BBST<BBST<T>> --> BBST<List<T>>

- ❖ Each y-search is just
a **1D** query without further recursions
- ❖ So it not necessary
to store each canonical subset
as a BBST
- ❖ Instead, a sorted y-**list** simply works



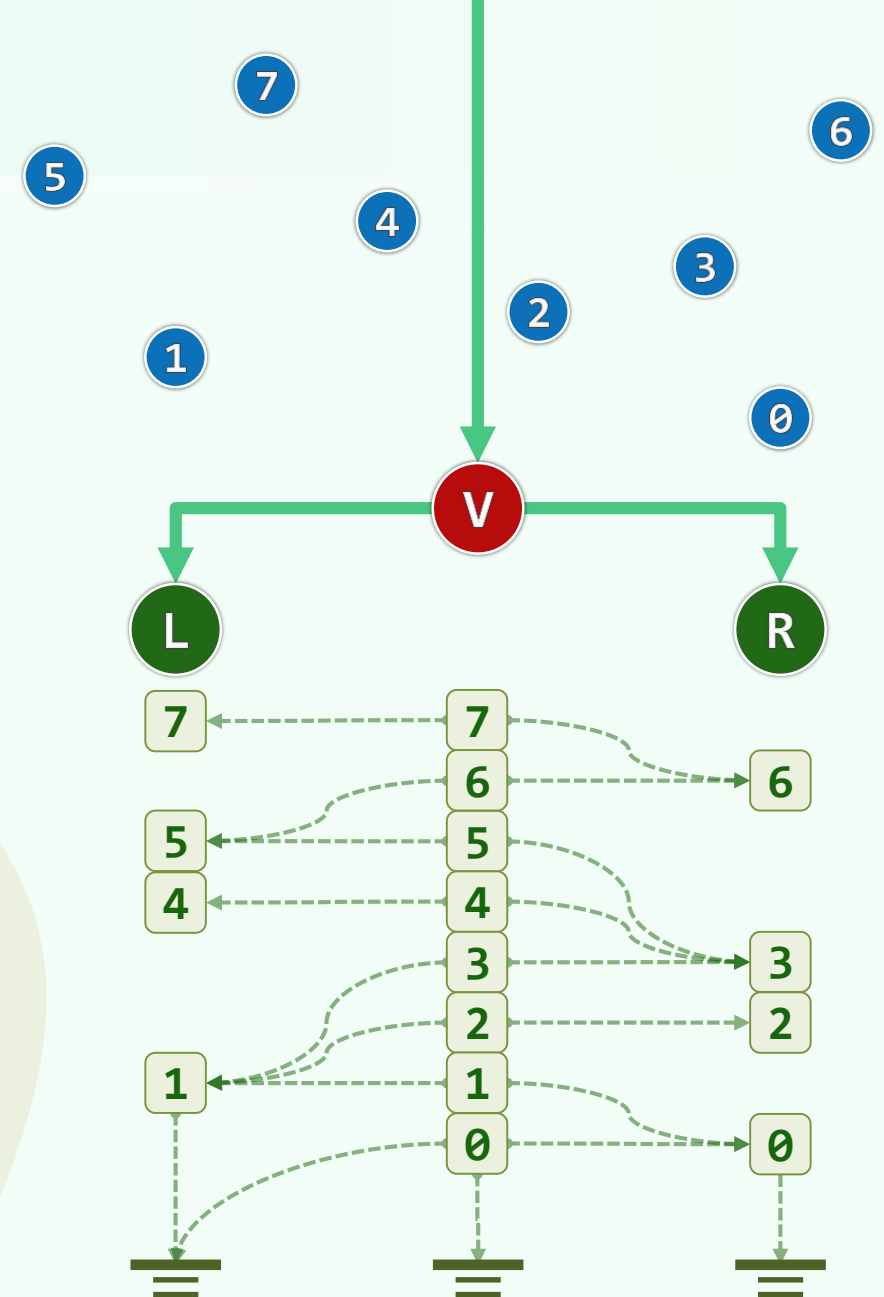
Links Between Lists

- ❖ We can **CONNECT** all the different lists into a **SINGLE** massive list
- ❖ Thus, once a parent y-list is searched, we can get, in $O(1)$ time, the entry for child y-list by following the link between them



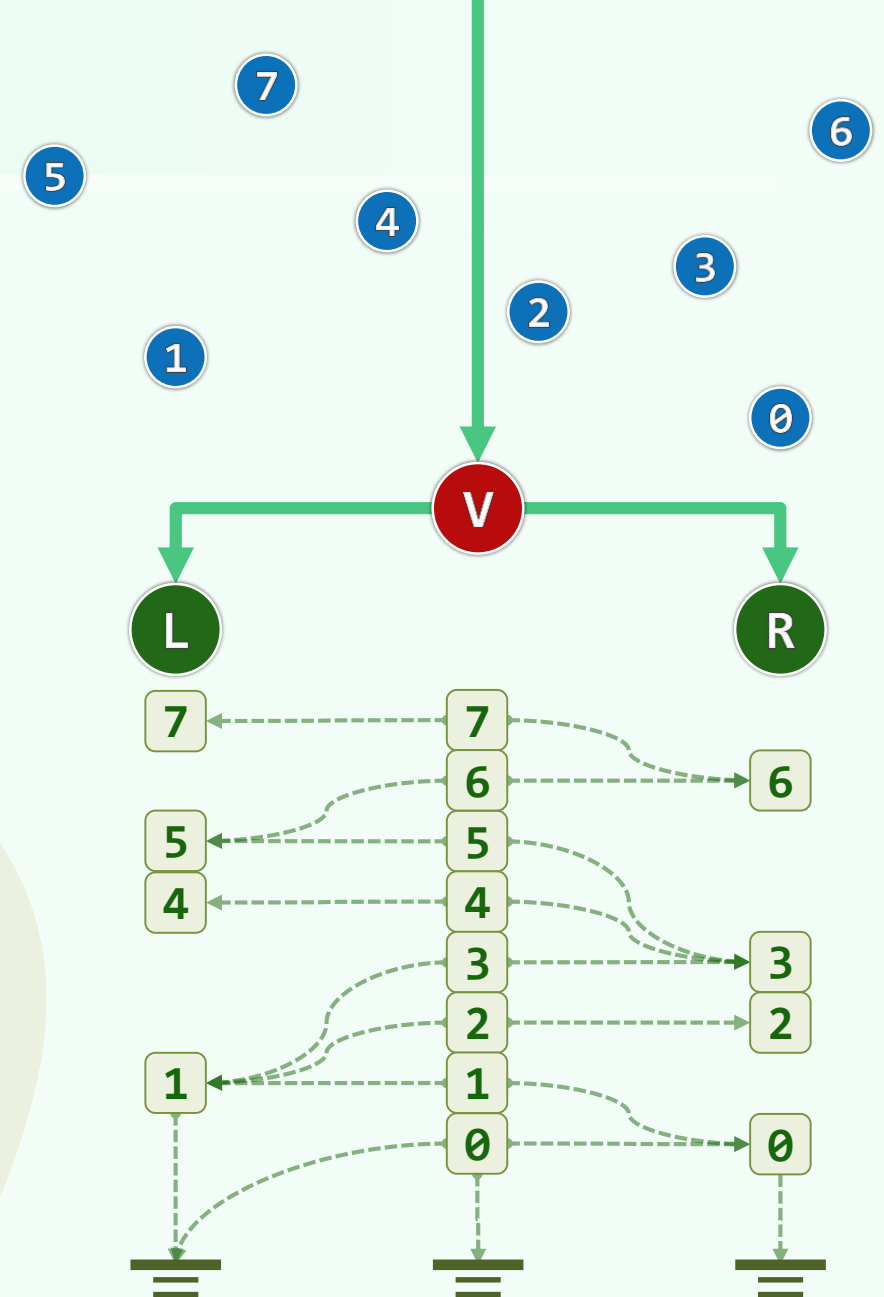
Using Coherence

- ❖ To answer a 2D range query, we will
do an $\mathcal{O}(\log n)$ search
on the **y**-list for the **LCA**
- ❖ Thereafter, while descending the **x**-tree, we can
keep track of the position of **y**-range
in each successive list in $\mathcal{O}(1)$ time
- ❖ This technique is called



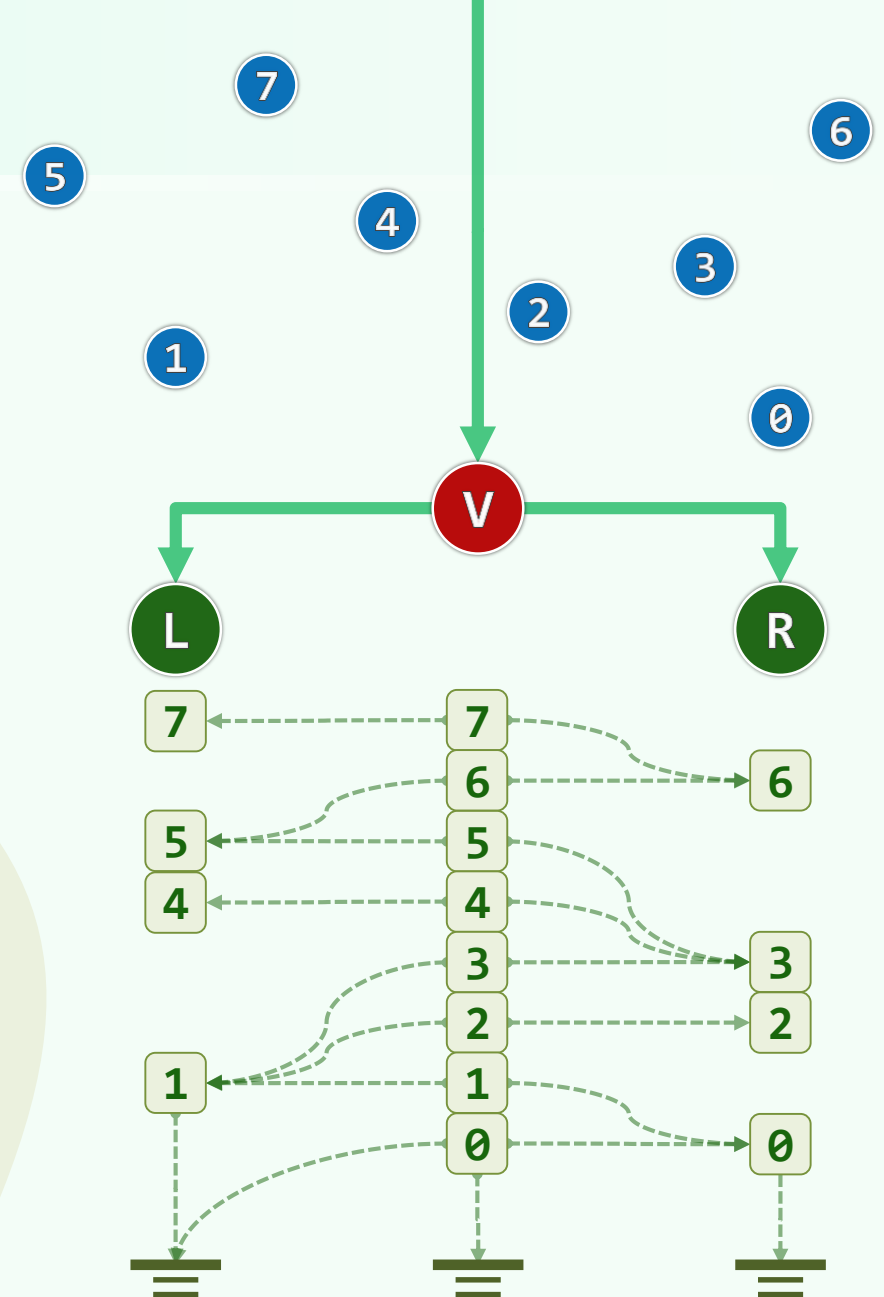
Fractional Cascading

- ❖ For each item in A_v ,
we store two pointers to
the item of **NLT** value
in A_L and A_R resp.
- ❖ Hence for any y -query with q_y ,
once we know its entry in A_v , we can
determine its entry in either A_L or A_R
in $\mathcal{O}(1)$ additional time



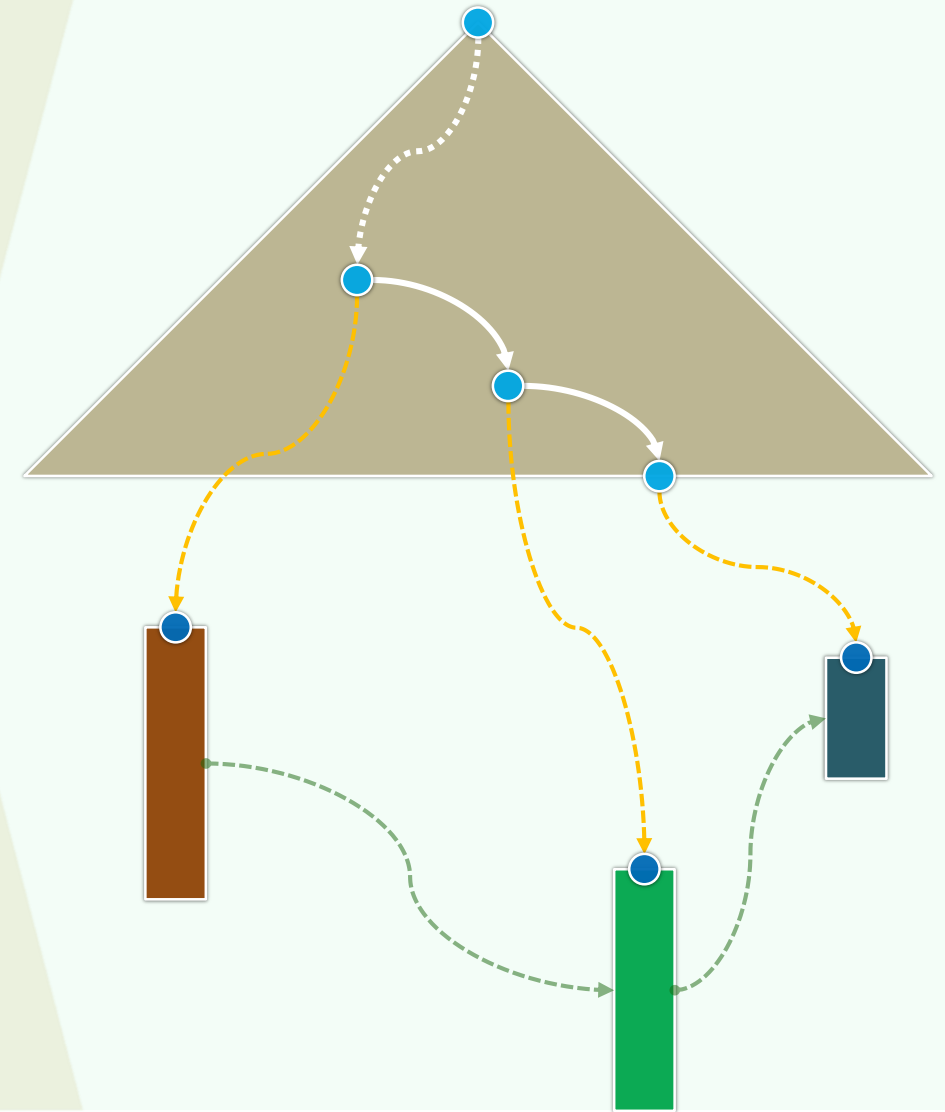
Construction By 2-Way Merging

- ❖ Let V be an internal node in the x -tree with L/R its left/right child resp.
- ❖ Let A_V be the y -list for v and A_L/A_R be the y -lists for its children
- ❖ Assuming no duplicate y -coordinates, we have
 - A_V is the disjoint union of A_L and A_R , and hence
 - A_V can be obtained by merging A_L and A_R (in **linear** time)



Complexity

- ❖ An MLST with fractional cascading is called a **range tree**
- ❖ A **y**-search for root is done in $\mathcal{O}(\log n)$ time
- ❖ To drop down to each next level, we can get, in $\mathcal{O}(1)$ time, the current **y**-interval from that of the **prior** level
- ❖ Hence, each 2D orthogonal range query
 - can be answered in $\mathcal{O}(r + \log n)$ time
 - from a data structure of size $\mathcal{O}(n \cdot \log n)$,
 - which can be constructed in $\mathcal{O}(n \cdot \log n)$ time



Beyond 2D

- ❖ Unfortunately, the trick of fractional cascading can **ONLY** be applied to the **LAST** level the search structure
- ❖ Given a set of n points in \mathcal{E}^d , an orthogonal range query
 - can be answered in $\mathcal{O}(r + \log^{d-1} n)$ time
 - from a data structure of size $\mathcal{O}(n \cdot \log^{d-1} n)$,
 - which can be constructed in $\mathcal{O}(n \cdot \log^{d-1} n)$ time

