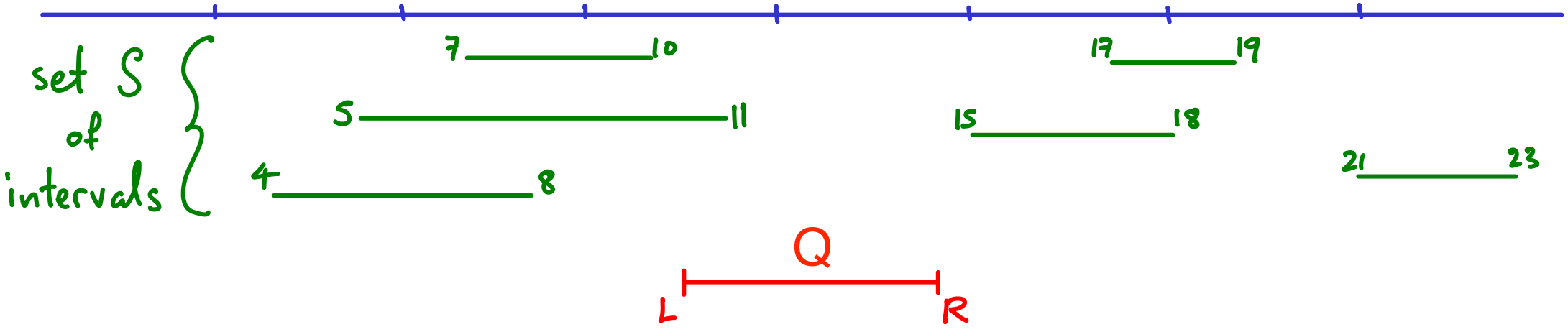
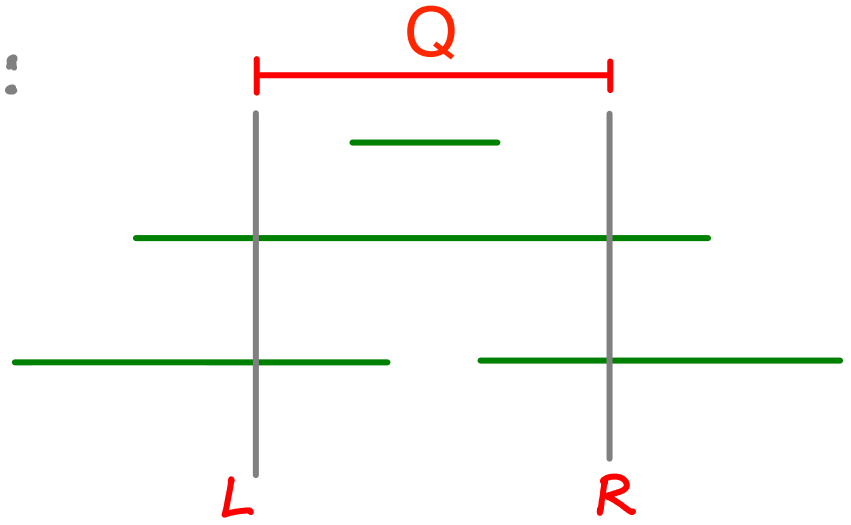
INTERVAL TREES



Query : given an interval Q,
return any interval in the set S that partially overlaps Q
(if one exists)

types of overlap:



For Q to not overlap an input interval, Q must be:

entirely to the right OR entirely to the left

high < L

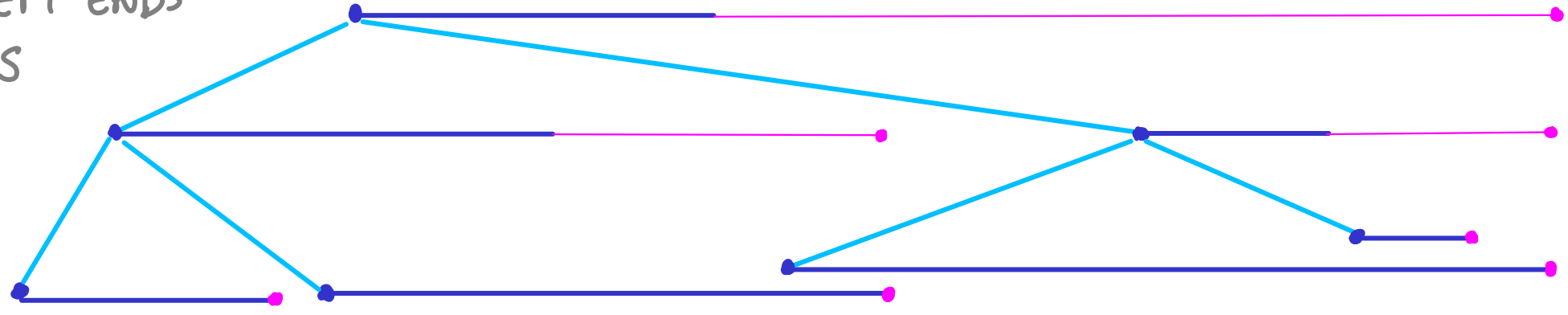
R < low

O(1) time

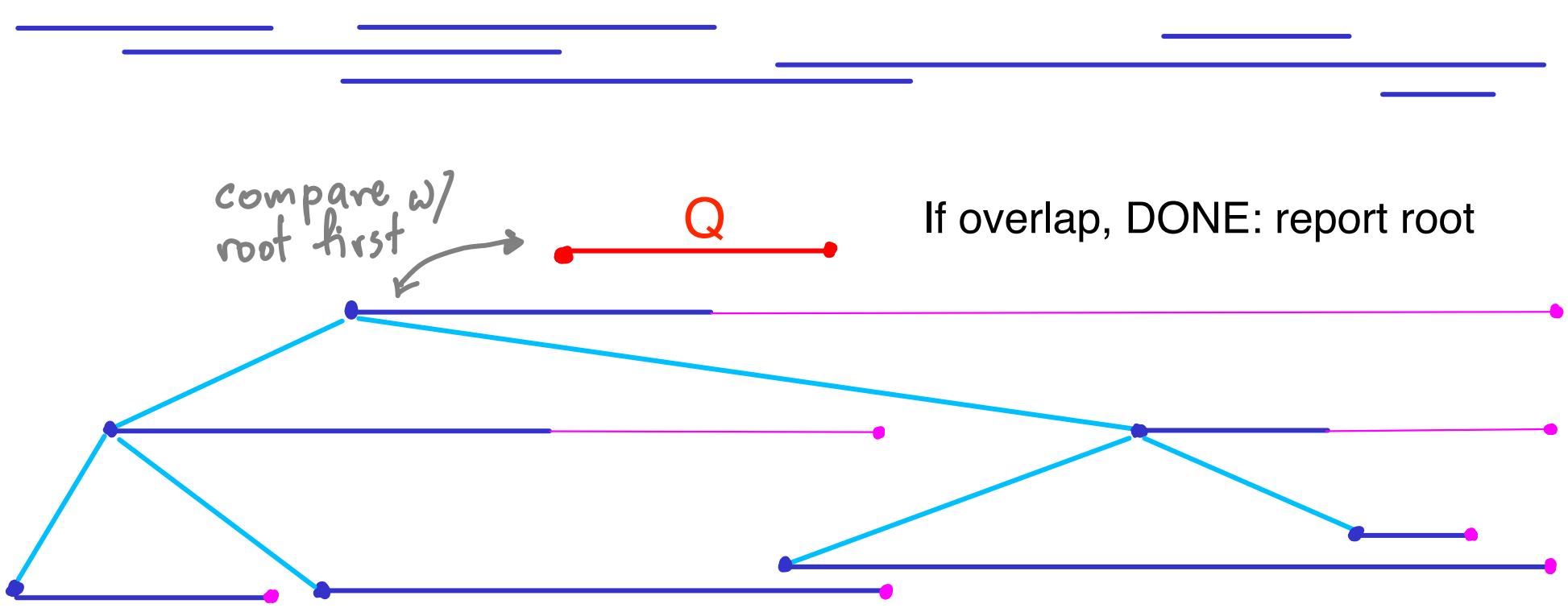
1D :

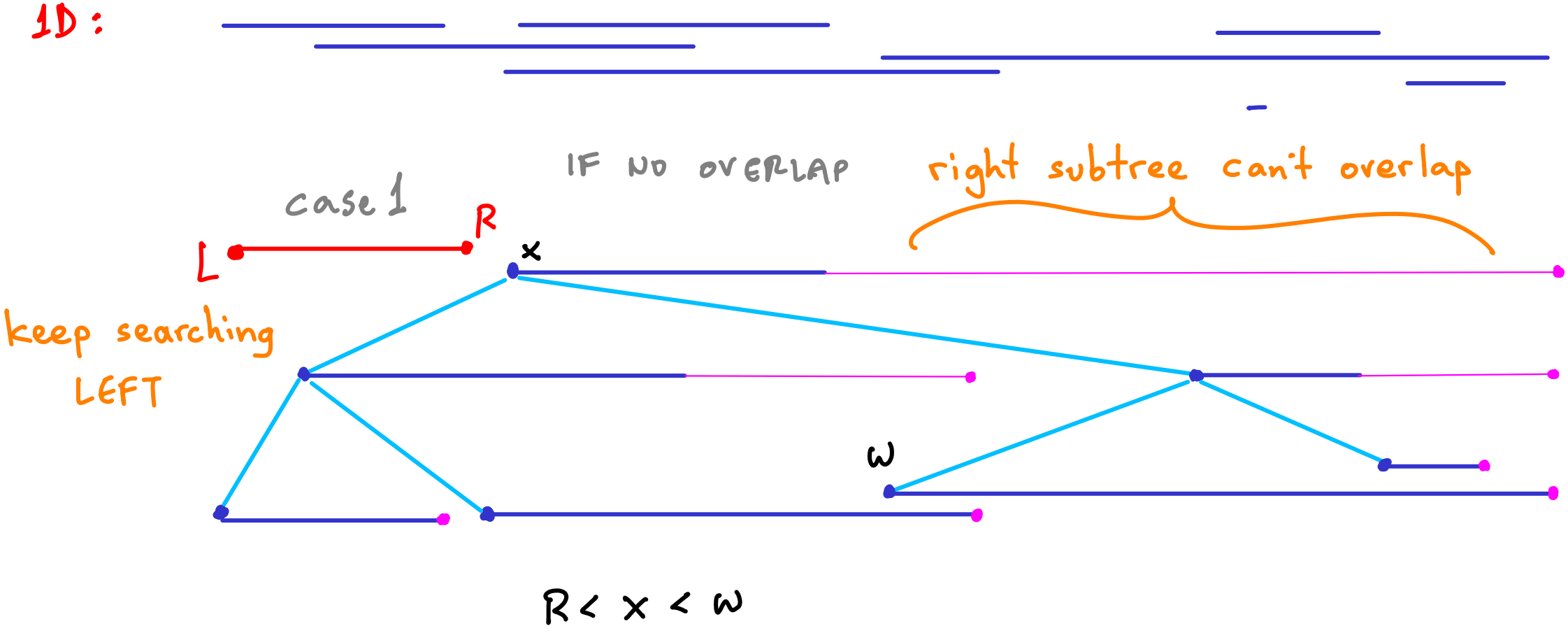
Augment: MAX RIGHT END of SUBTREE

BST w/ LEFT ENDS as KEYS



1D :

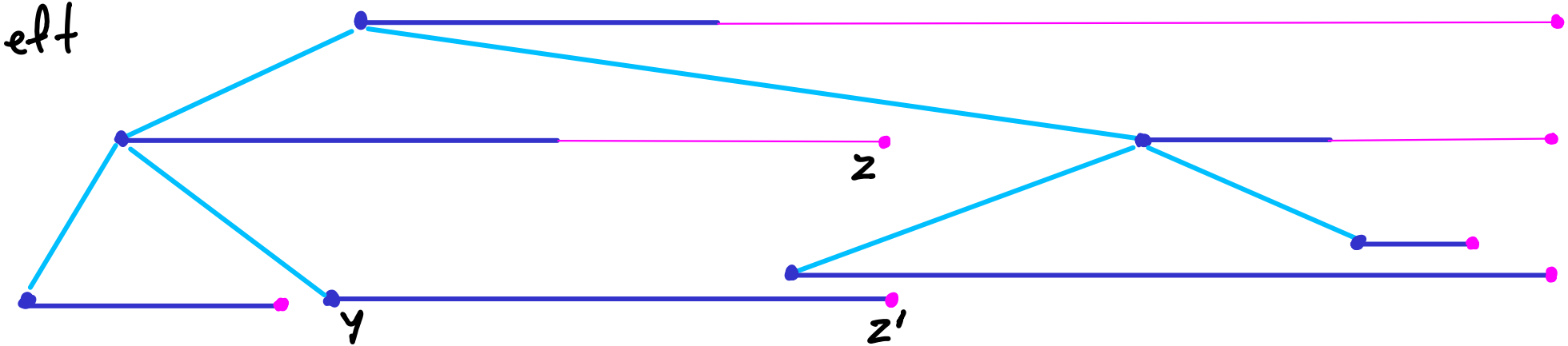



1D :
IF NO OVERLAP
right subtree can't overlap
case 1
L
R
x
keep searching
LEFT
w
R < x < w

1D :

IF NO OVERLAP

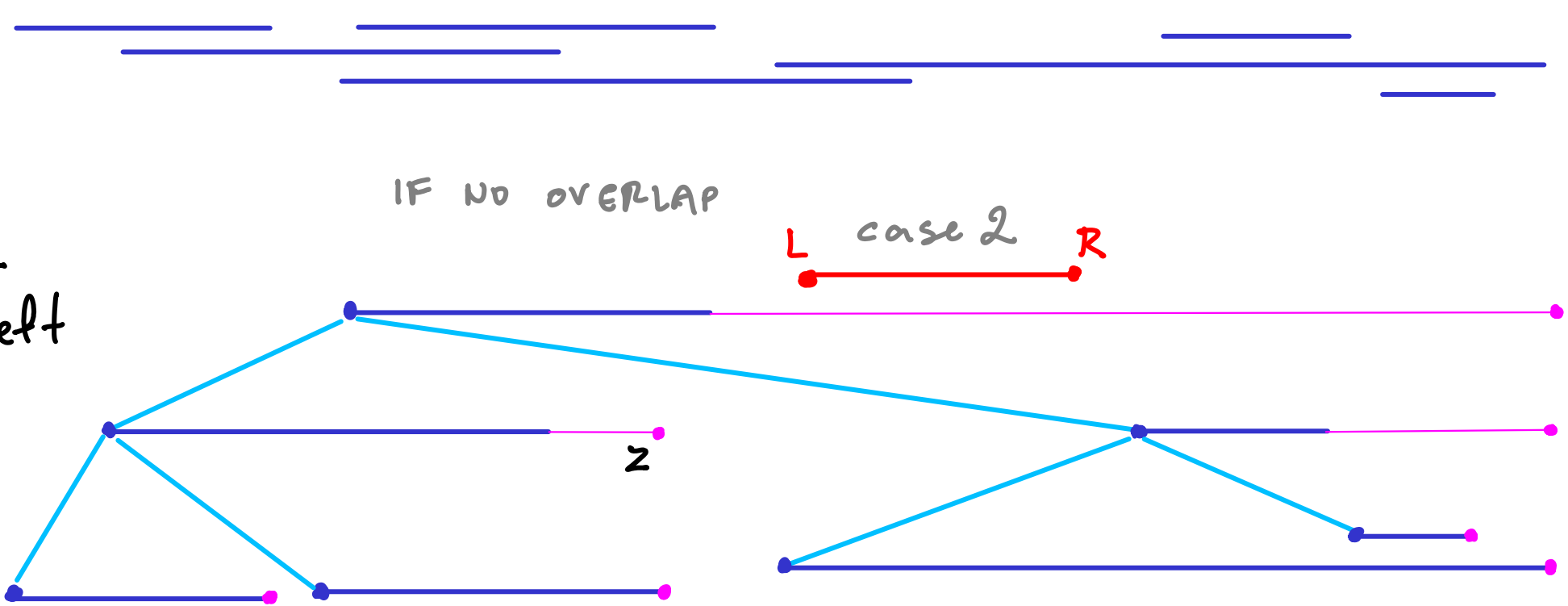
IF $z \geq L$
search left



$\exists \overline{yz'}$
s.t.
$y < L < z'$

} guaranteed overlap

1D :



IF $z \geq L$
search left

$\exists$ $\overline{yz'}$
s.t.
$y < L < z'$

} guaranteed overlap

else $(z < L)$

- NO overlap to left
- search right

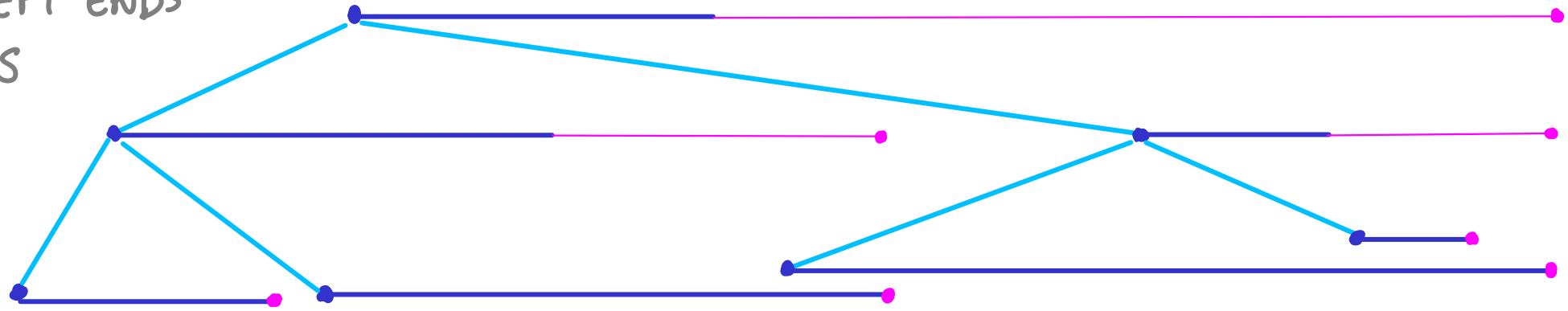
Spend O(1) time at a node,
the recurse on only one side.

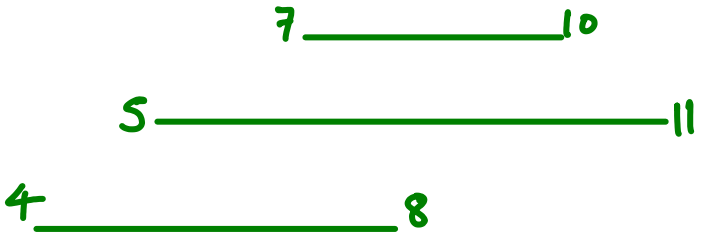
Given balanced tree, total time is O(logn)

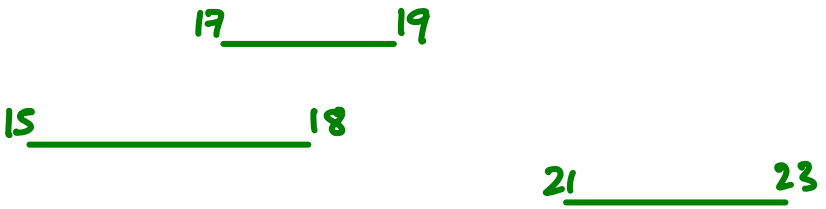
But can we build and maintain the augmented info?

How can we update MAX RIGHT END of SUBTREE ?

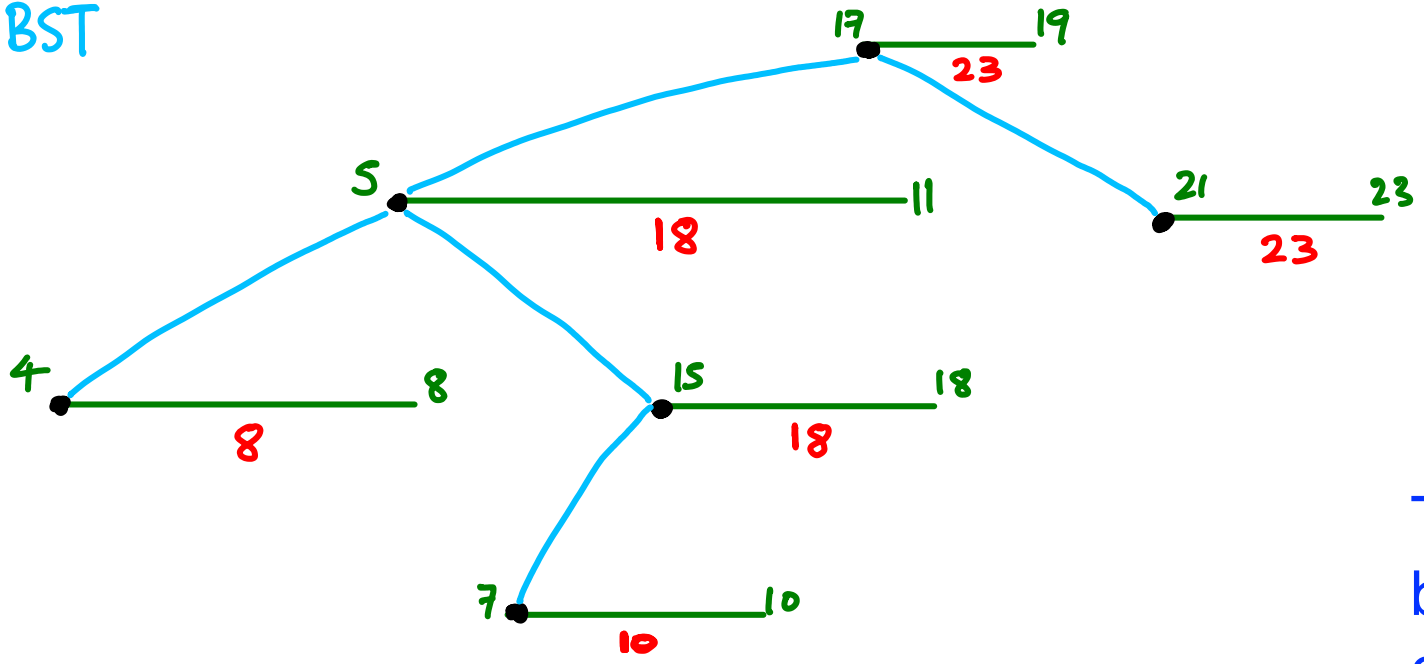
BST w/ LEFT ENDS as KEYS

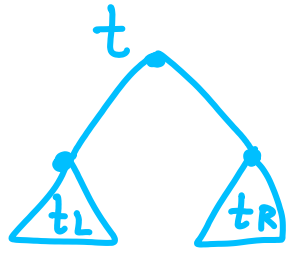






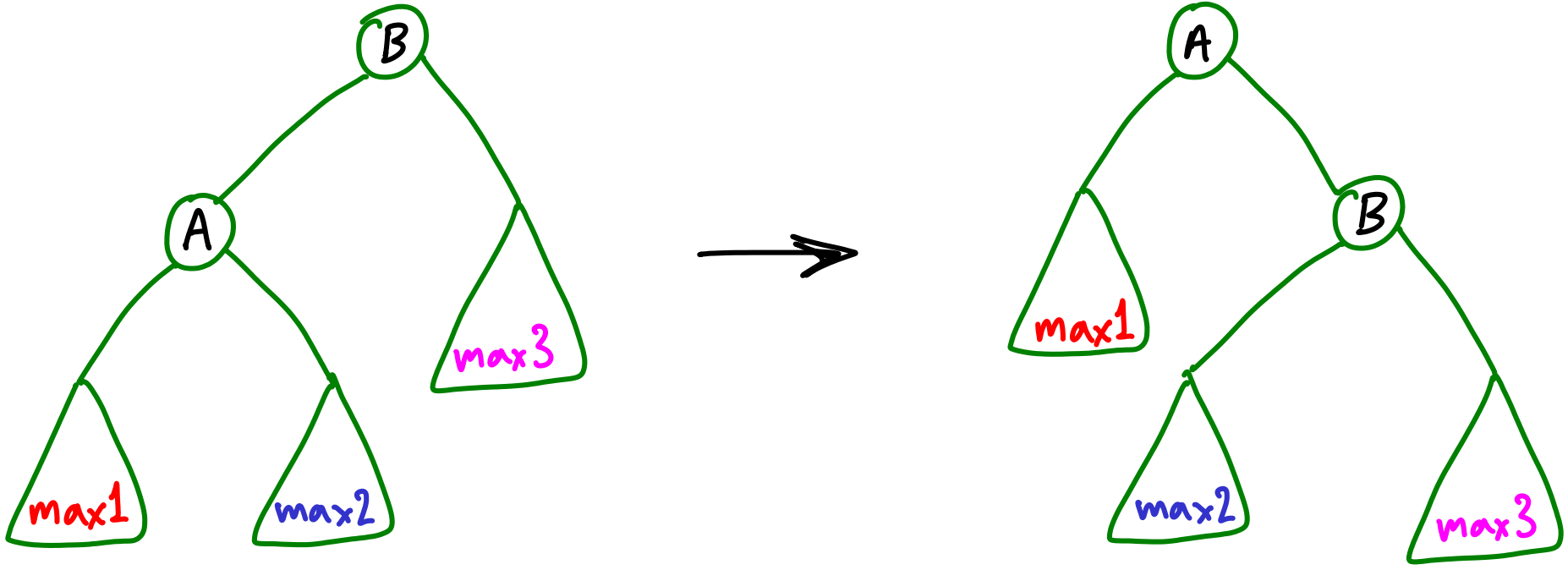
augmented BST





$$max(t) = max\begin{cases} hi(t) \\ max(t_L) \\ max(t_R) \end{cases}$$

Trivial to build:
bottom up after tree is made,
or during insertion



max1 , max2 , max3 : unchanged by rotation

max(A) & max(B) : trivial to update

we can maintain a balanced BST augmented w/ max value of subtrees