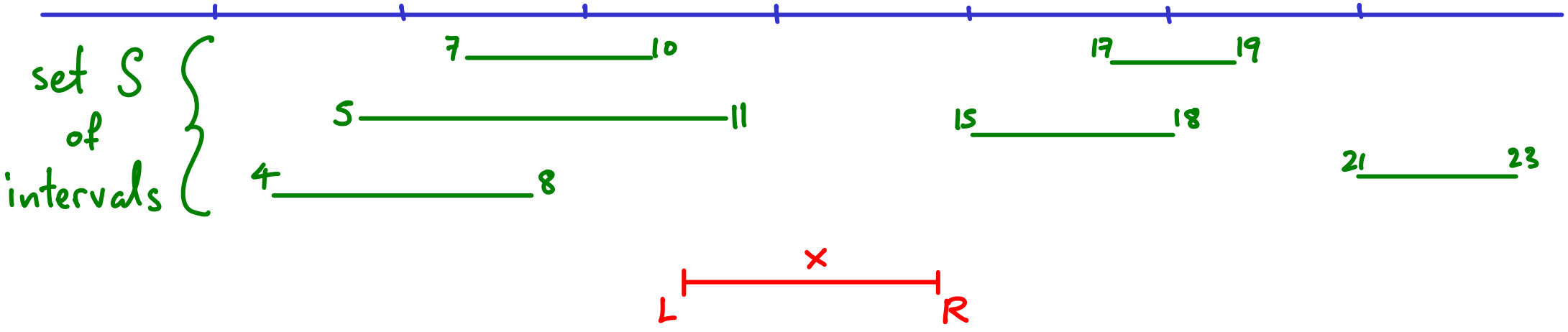
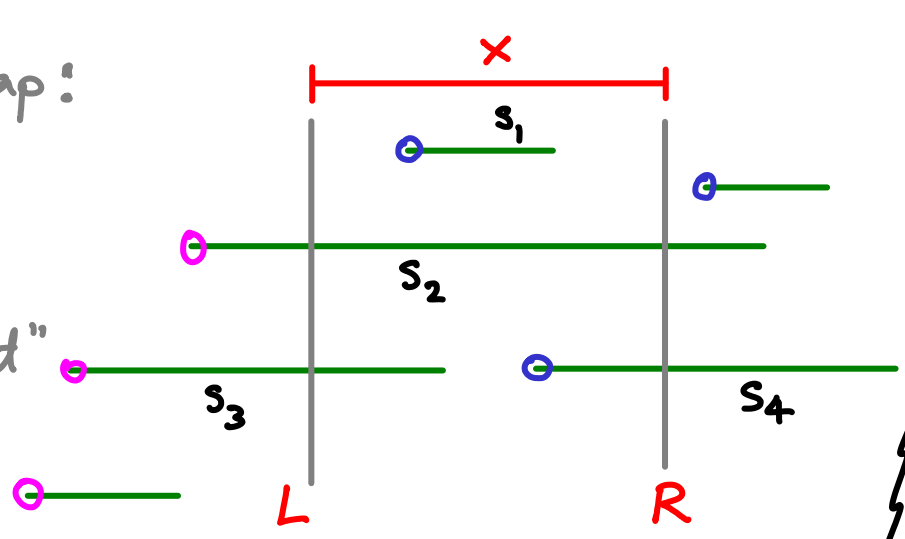
INTERVAL TREES

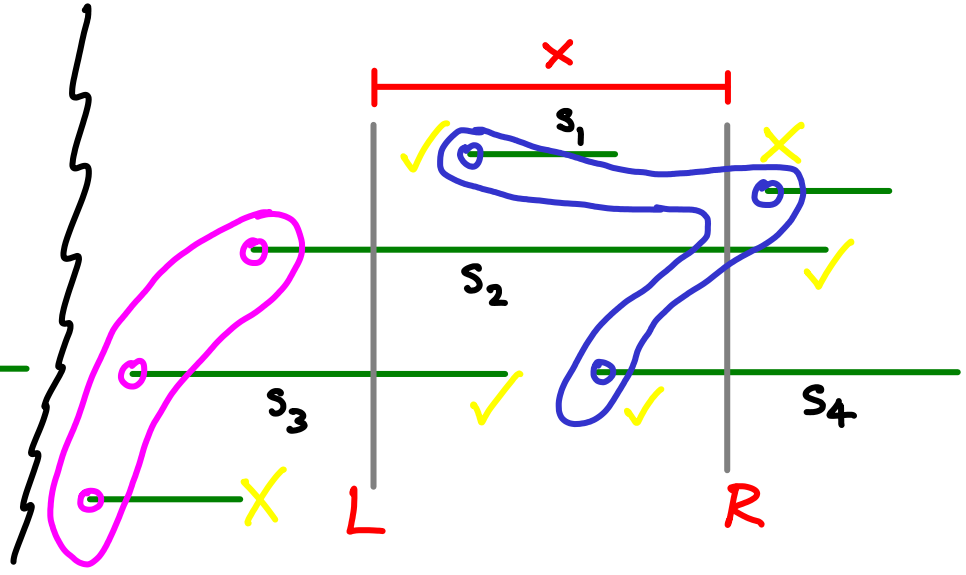


Query : given an interval x,
return any interval in the set S that partially overlaps x
(if one exists)

types of overlap:

1) "smaller"
2) "bigger"
3) "left" & "right"





First comparison: $lo[s_i]$ vs L

<

is there some large enough $hi[s_i]$?

>

is there some small enough $lo[s_i]$?

If $lo[s_i] \leq L$ AND $hi[s_i] \geq L$ then overlap

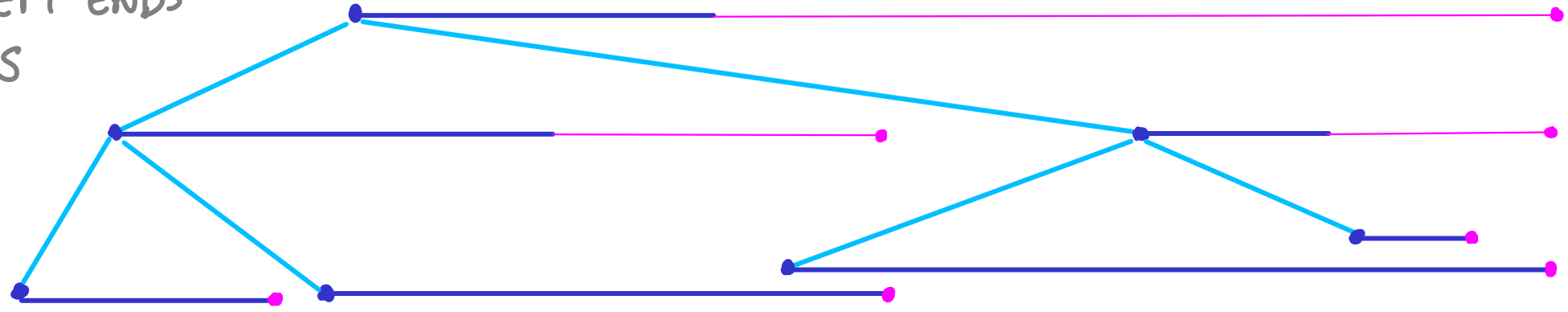
If $lo[s_i] \geq L$ AND $\leq R$ then overlap

SEARCHING FOR OVERLAPPING INTERVALS

1D:

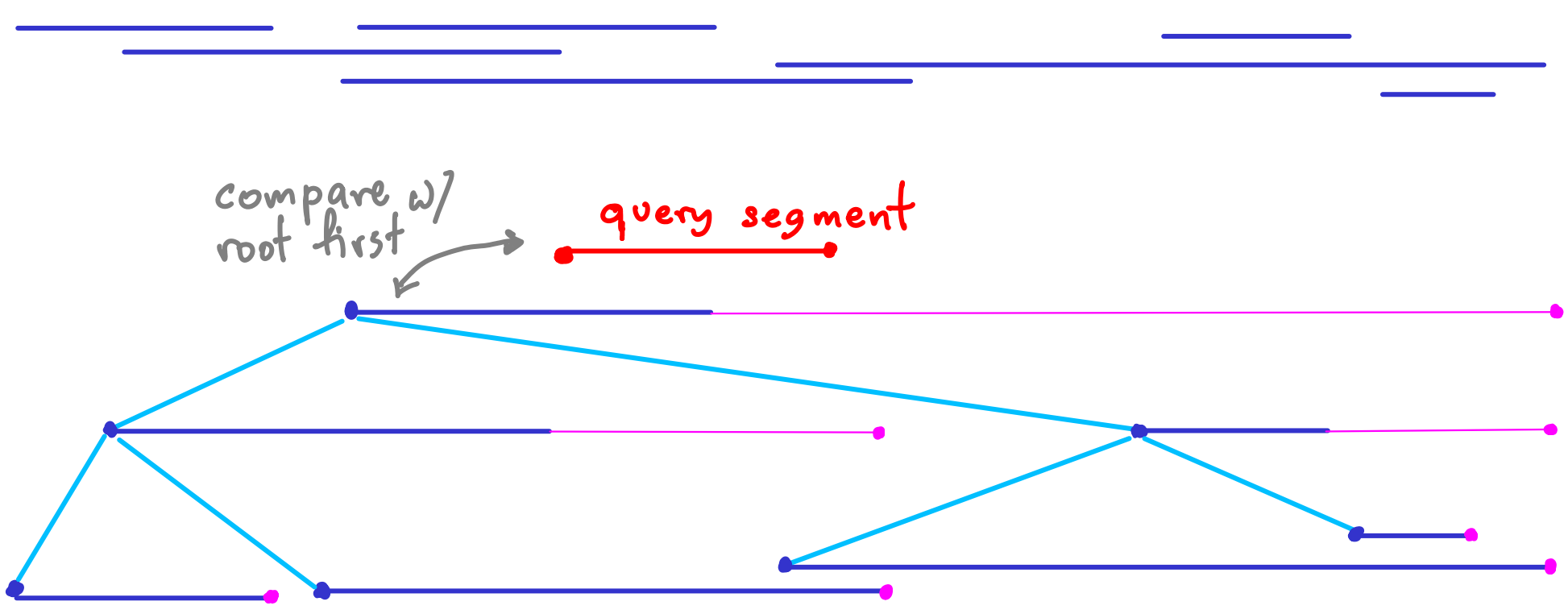
BST w/ LEFT ENDS as KEYS

Augment: MAX RIGHT END of SUBTREE



SEARCHING FOR OVERLAPPING INTERVALS

1D :



SEARCHING FOR OVERLAPPING INTERVALS

1D :

IF NO OVERLAP

right subtree can't overlap

case 1

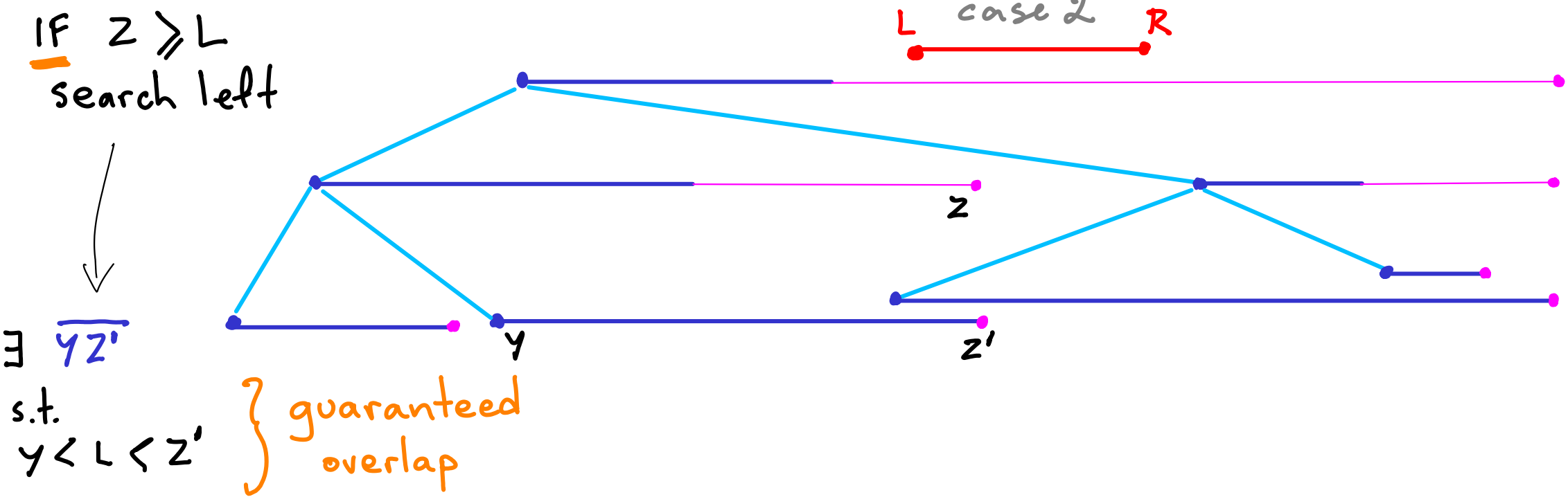
$R < x < w$

SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF NO OVERLAP

IF $z \geq L$
search left

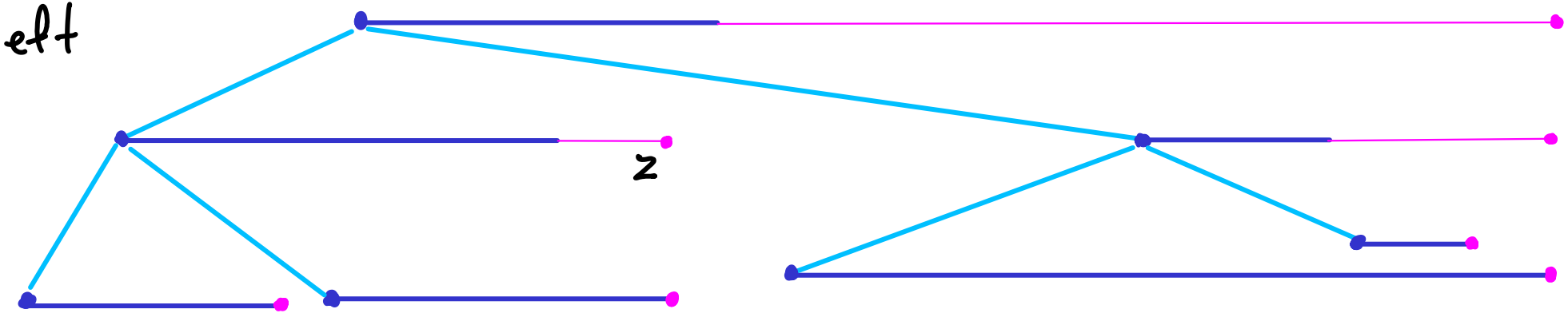


SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF NO OVERLAP

IF $Z \geq L$
search left



$\exists \overline{YZ'}$
s.t.
$Y < L < Z'$

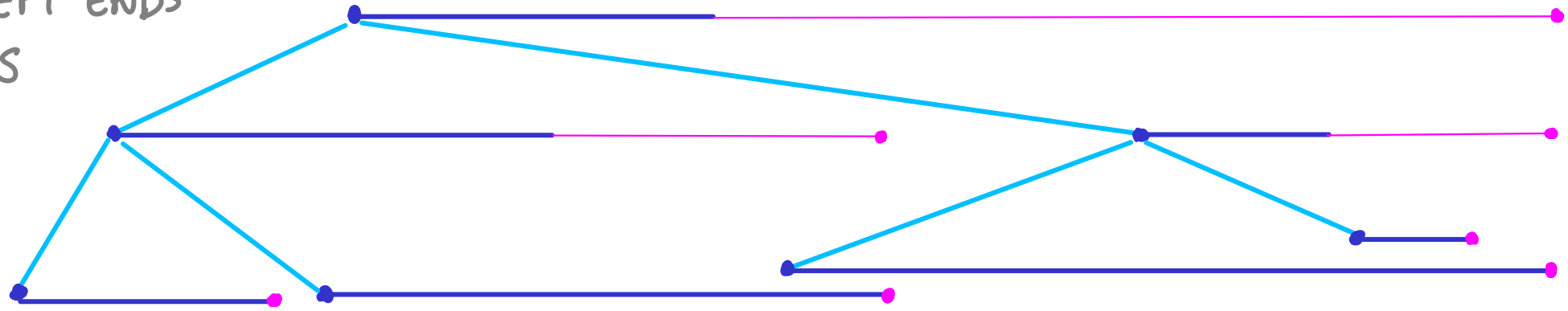
} guaranteed overlap

else $(Z < L)$

- NO overlap to left
- search right

How can we update MAX RIGHT END of SUBTREE ?

BST w/ LEFT ENDS as KEYS



7 — 10

5 — 11

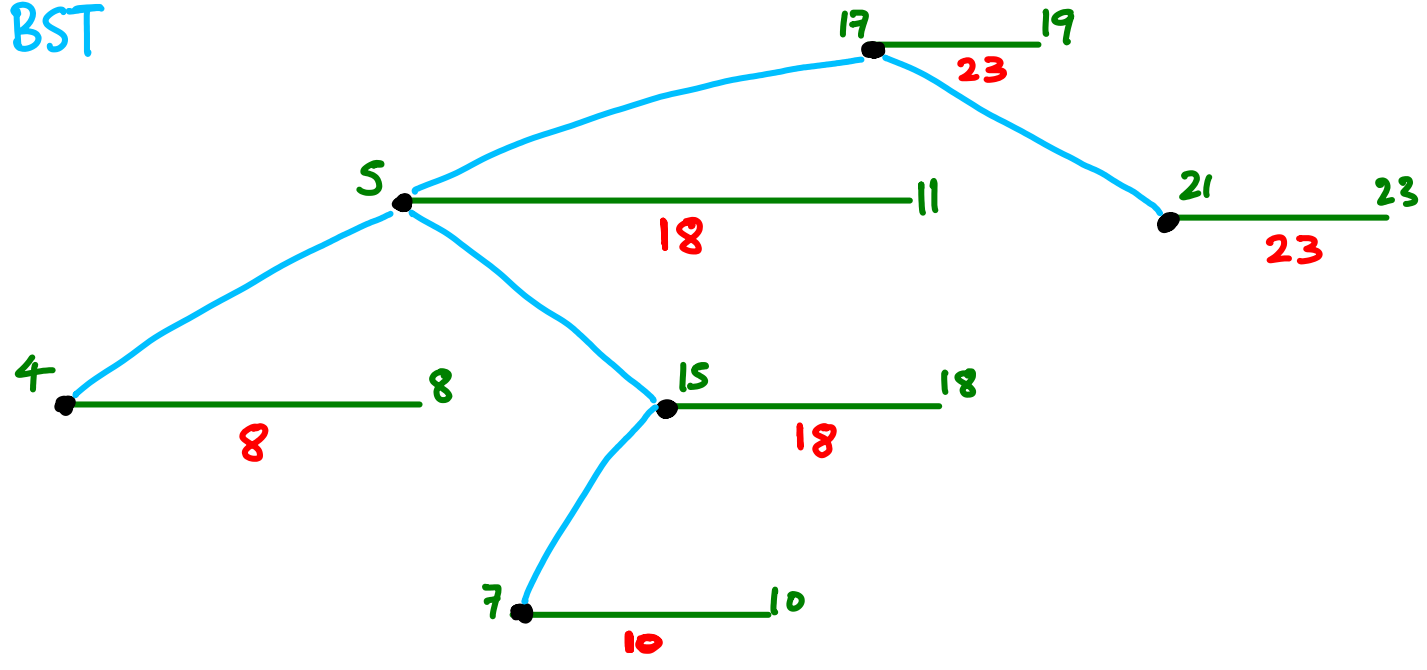
4 — 8

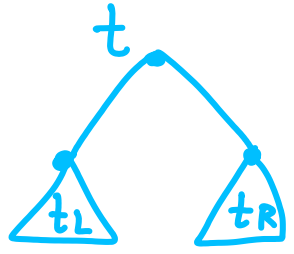
17 — 19

15 — 18

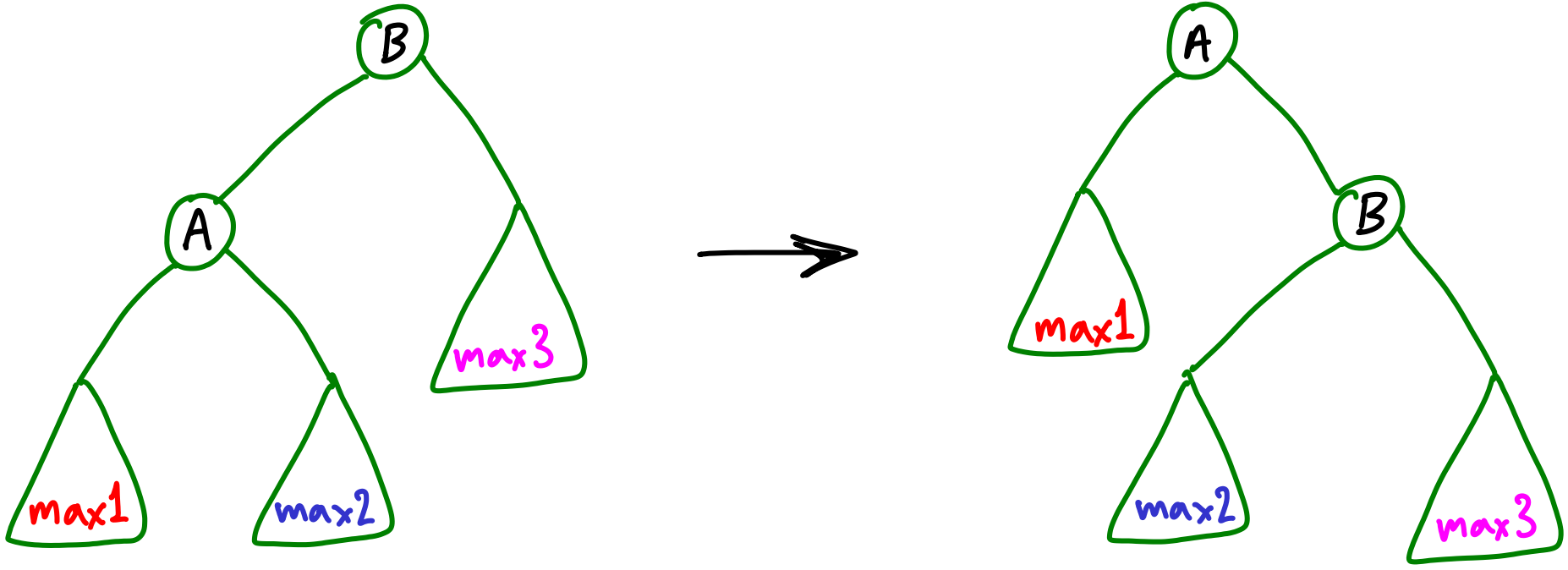
21 — 23

augmented BST





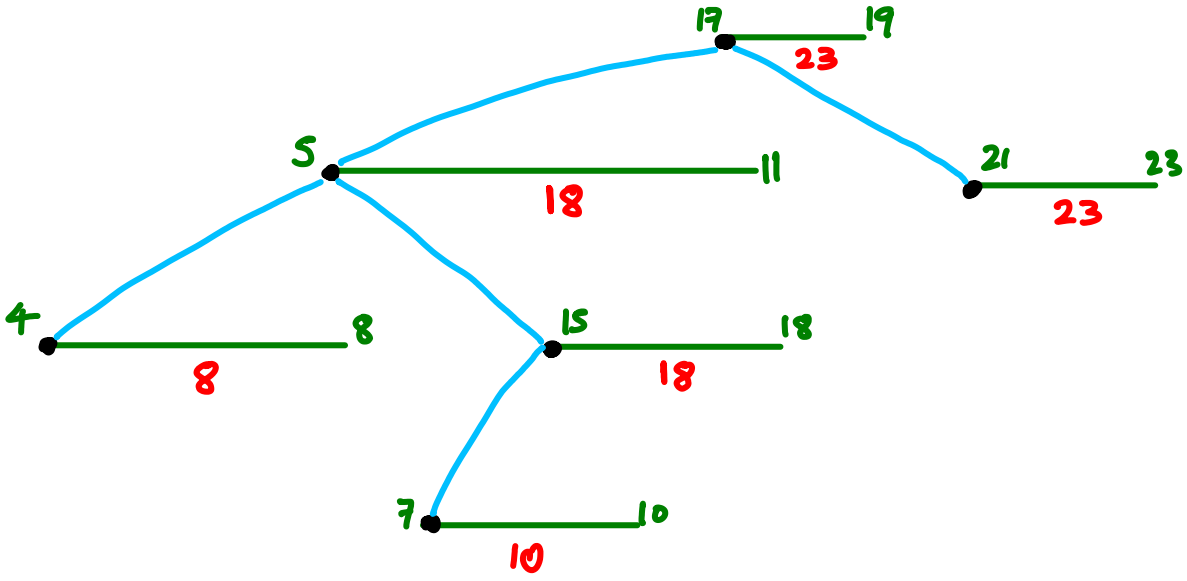
$$max(t) = max\begin{cases} hi(t) \\ max(t_L) \\ max(t_R) \end{cases}$$

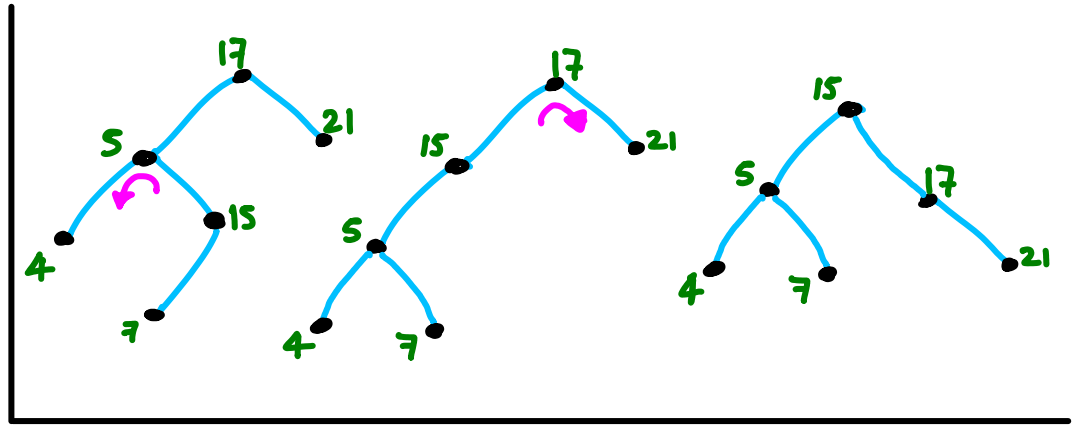


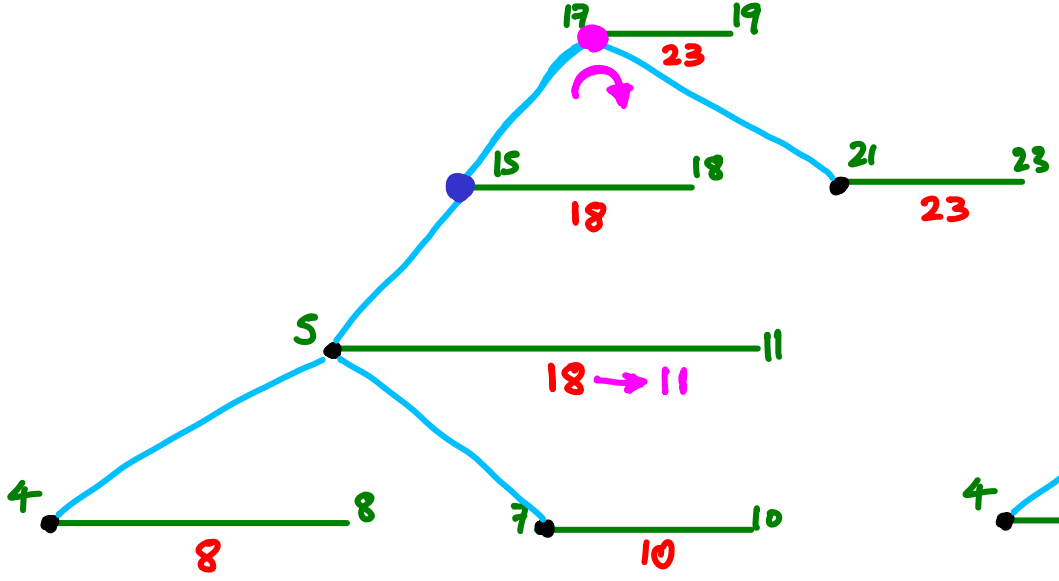
max1 , max2 , max3 : unchanged by rotation

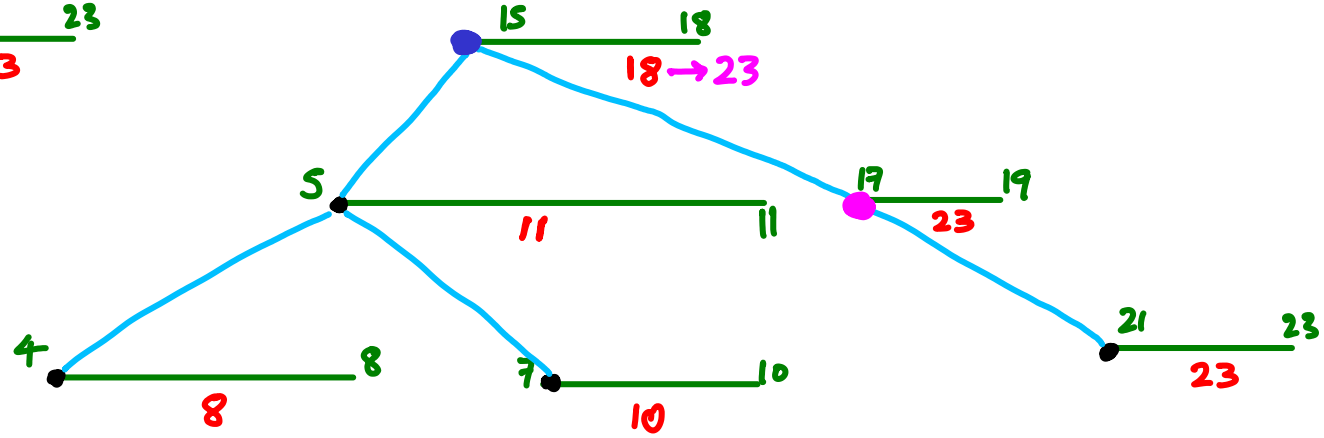
max(A) & max(B) : trivial to update

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


17
19
23
5
11
18
21
23
23
4
8
8
15
18
18
7
10
10


17
21
5
15
4
7
15
17
21
5
4
7
15
5
17
4
7
21


17
19
23
15
18
18
21
23
23
5
11
18 → 11
4
8
8
7
10
10


15
18
18 → 23
5
11
11
17
19
23
4
8
8
7
10
10
21
23
23