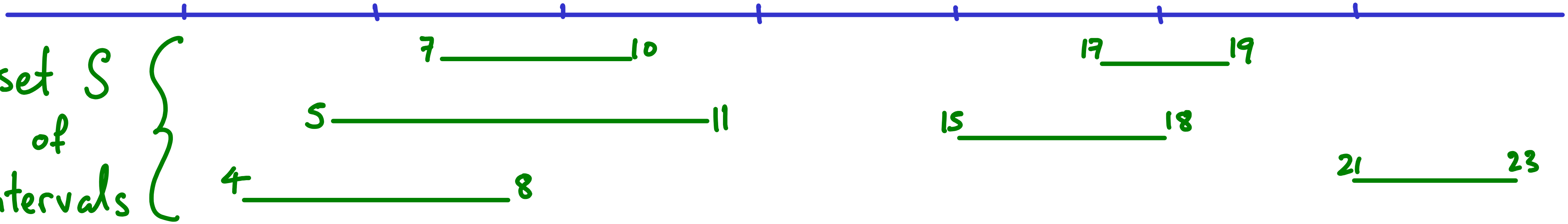
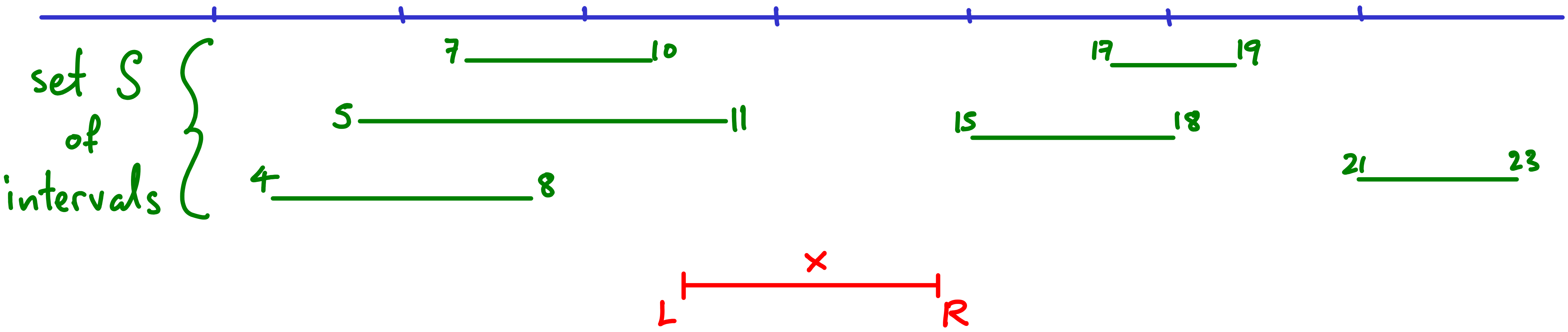
INTERVAL TREES

set S of intervals



INTERVAL TREES



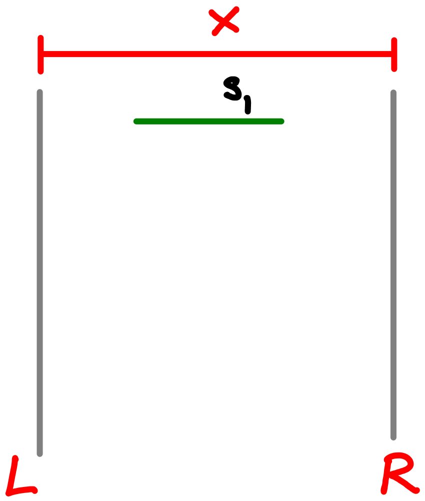
Query : given an interval x,

return any interval in the set S that partially overlaps x

(if one exists)

types of overlap:

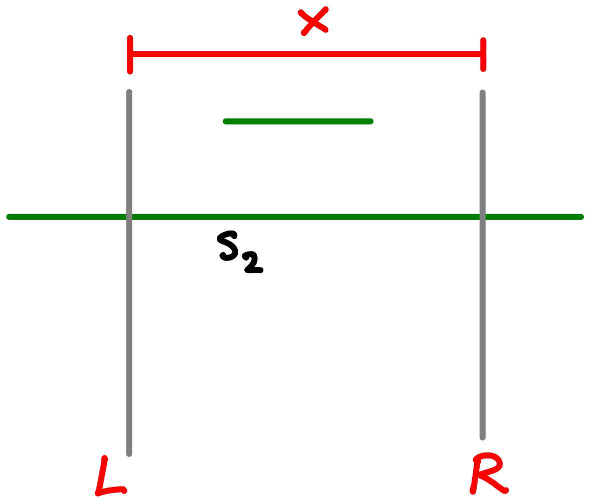
i) "smaller"



types of overlap:

1) "smaller"

2) "bigger"

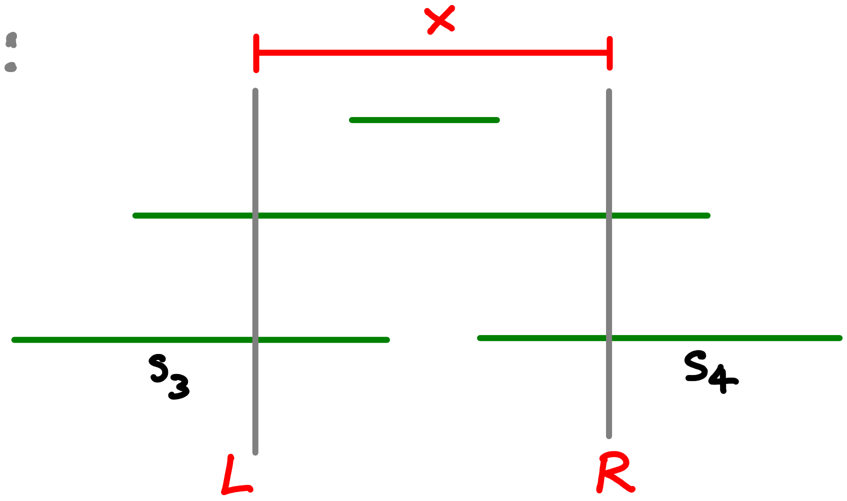


types of overlap:

1) "smaller"

2) "bigger"

3) "left" & "right"

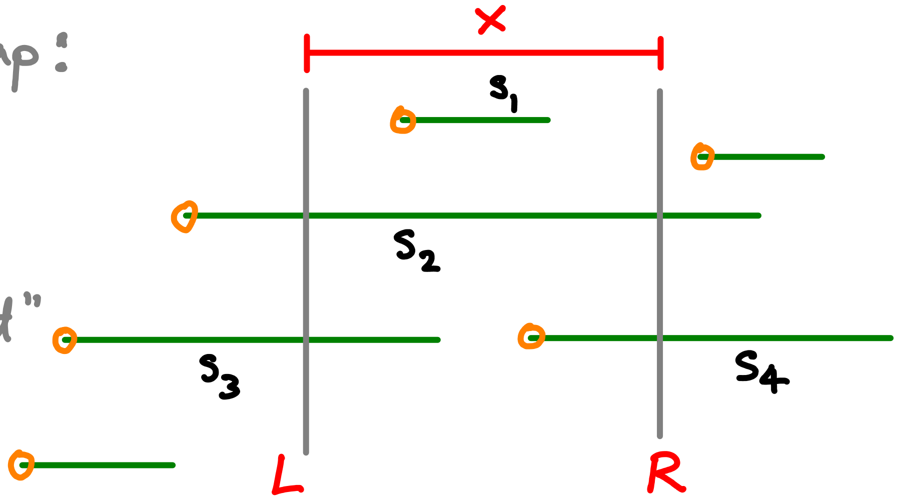


types of overlap:

1) "smaller"

2) "bigger"

3) "left" & "right"



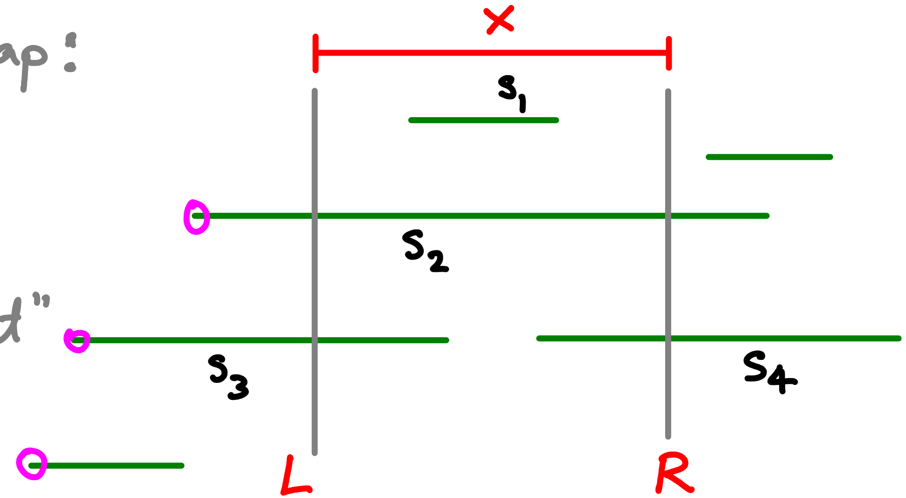
First comparison: $lo[s_i]$ vs L

types of overlap:

1) "smaller"

2) "bigger"

3) "left" & "right"



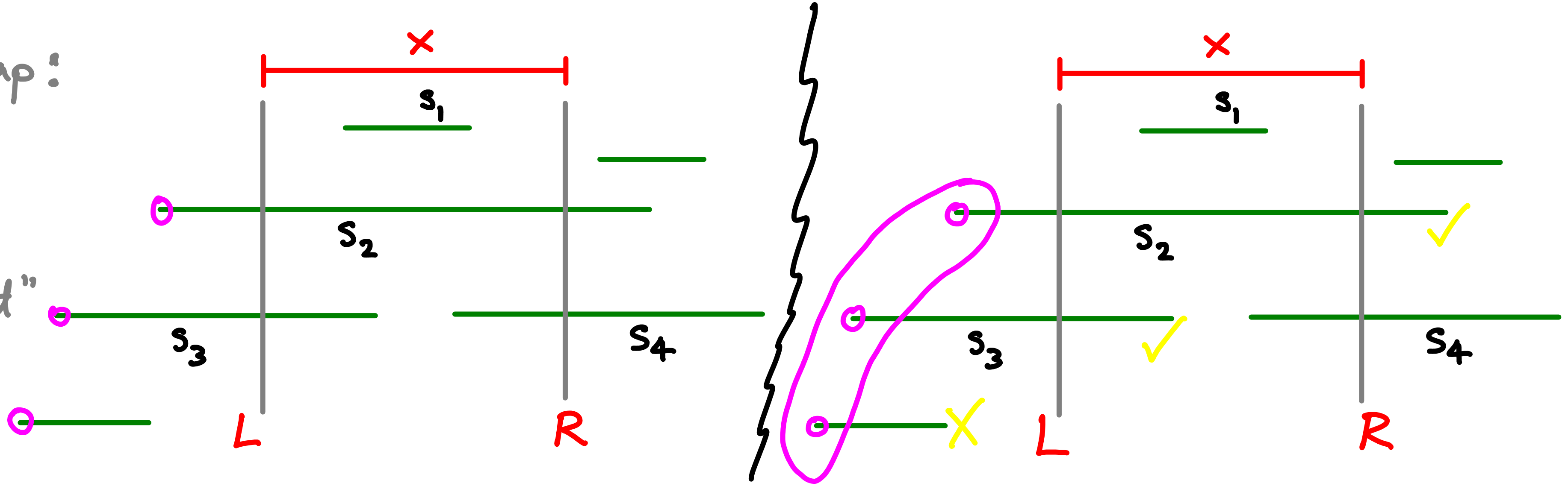
First comparison: $lo[s_i]$ vs L

$<$

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

types of overlap:

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



First comparison: $lo[s_i]$ vs L

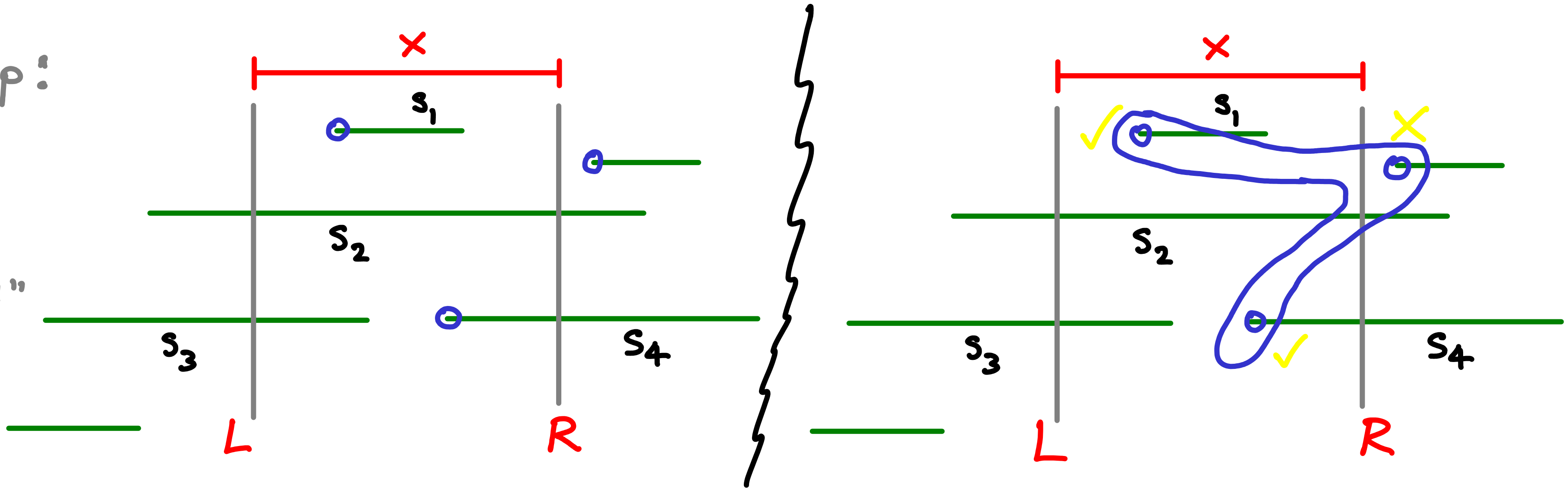
If $lo[s_i] \leq L$
AND
$hi[s_i] \geq L$
then overlap

<

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

types of overlap:

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



First comparison: $lo[s_i]$ vs L

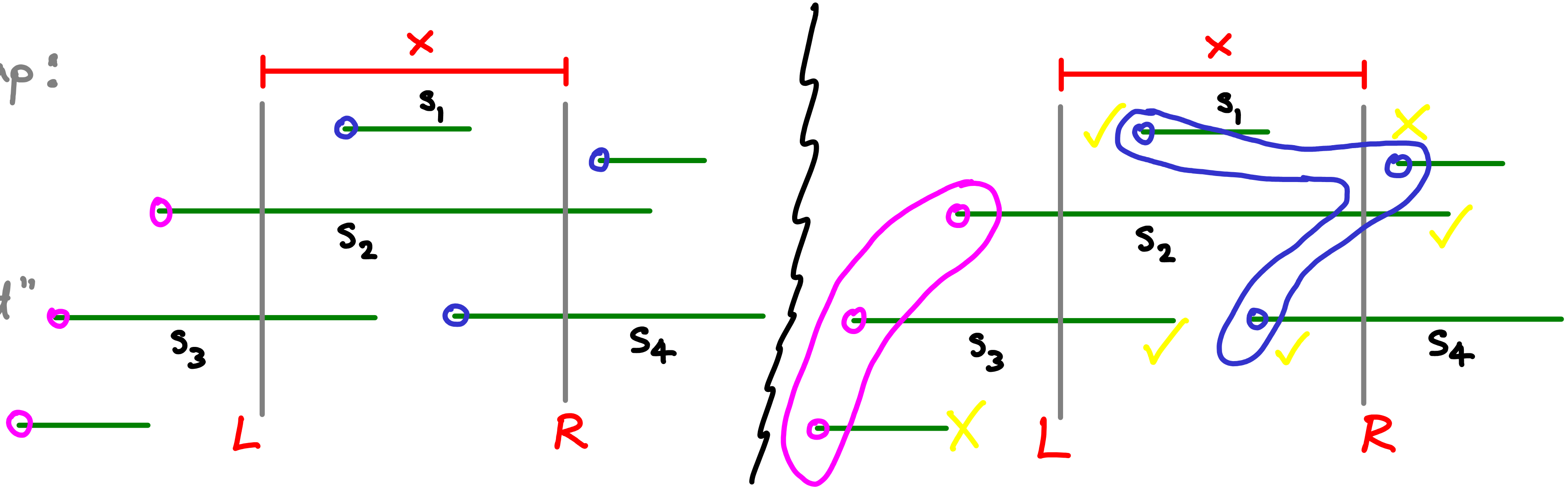
$>$

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

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

types of overlap:

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



First comparison: $lo[s_i]$ vs L ⟶

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

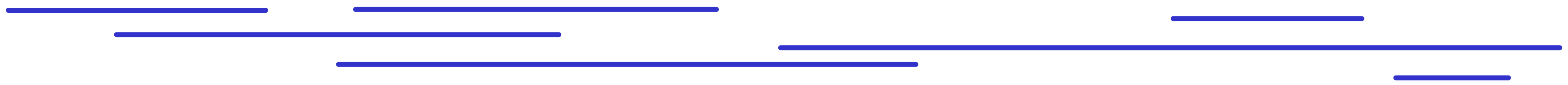
$>$: is there some small enough $lo[s_i]$?

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

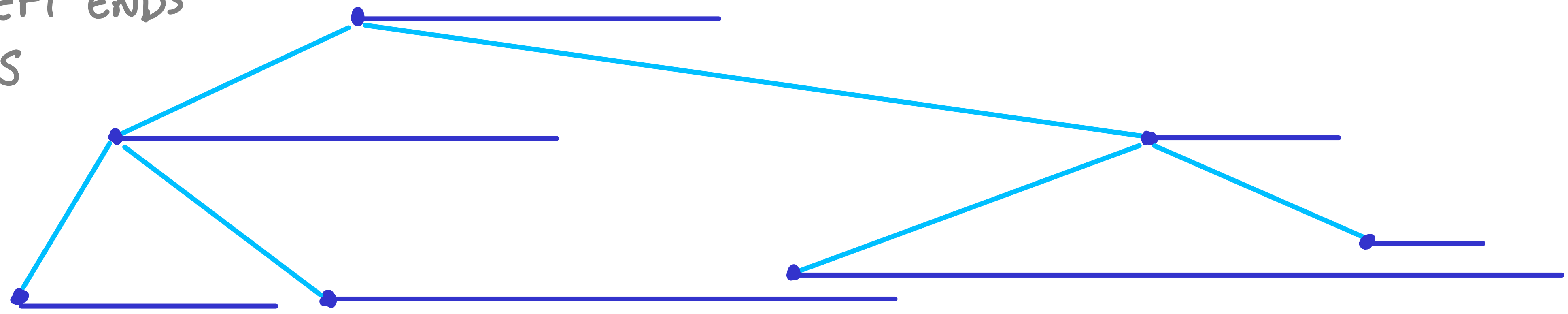
If $lo[s_i] \geq L$ AND $\leq R$ then overlap

SEARCHING FOR OVERLAPPING INTERVALS

1D :



BST w/ LEFT ENDS
as KEYS

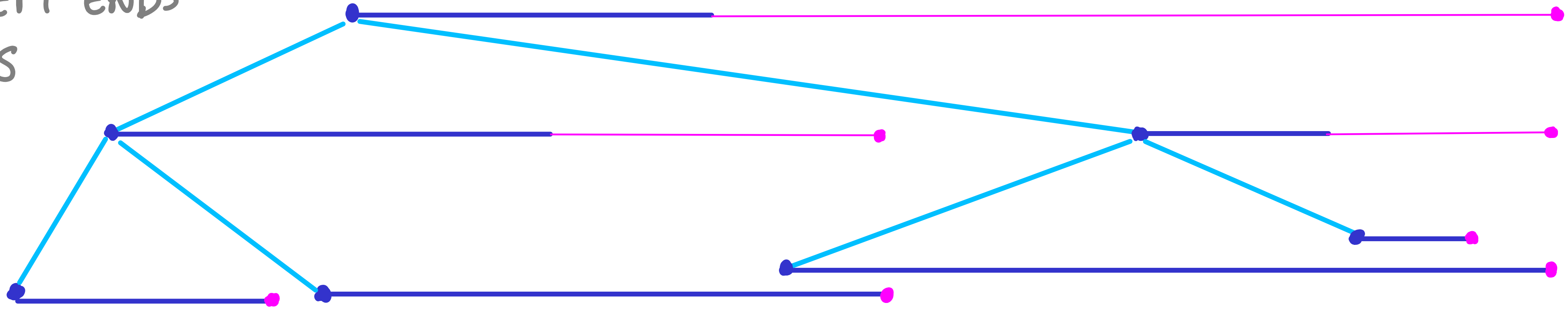


SEARCHING FOR OVERLAPPING INTERVALS

1D:

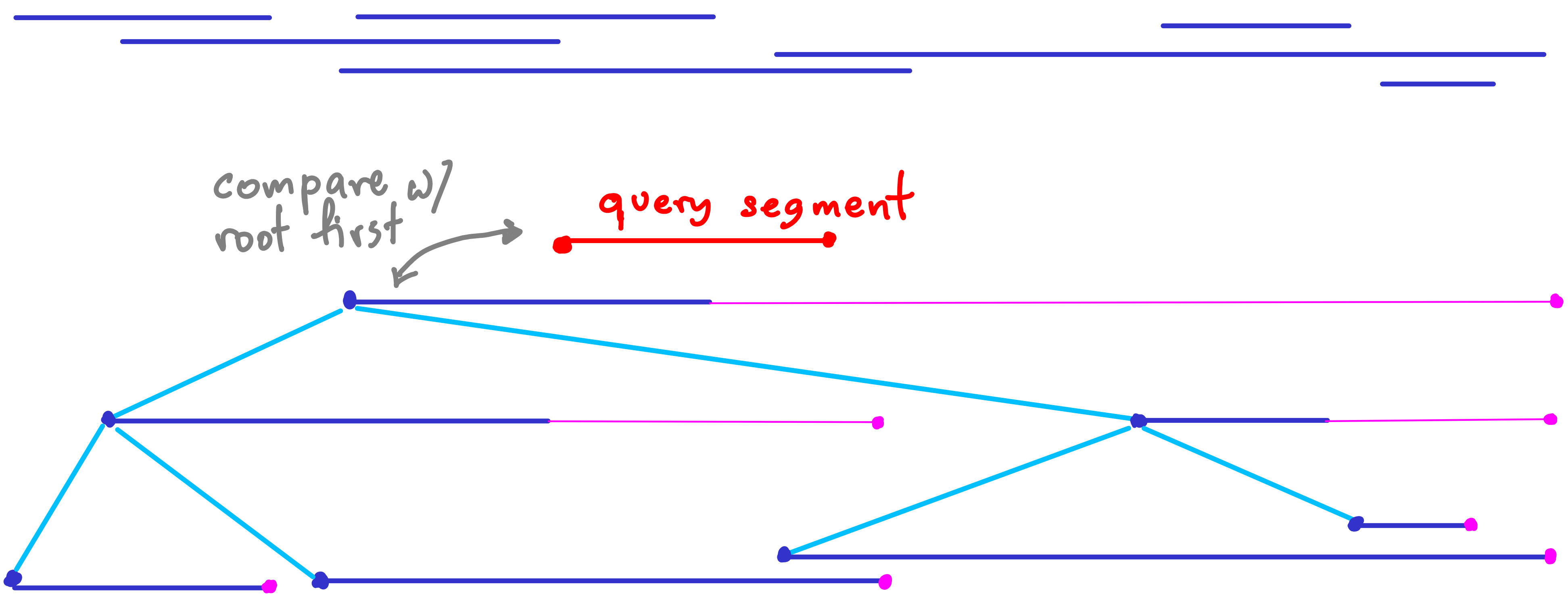
Augment: MAX RIGHT END of SUBTREE

BST w/ LEFT ENDS as KEYS



SEARCHING FOR OVERLAPPING INTERVALS

1D :

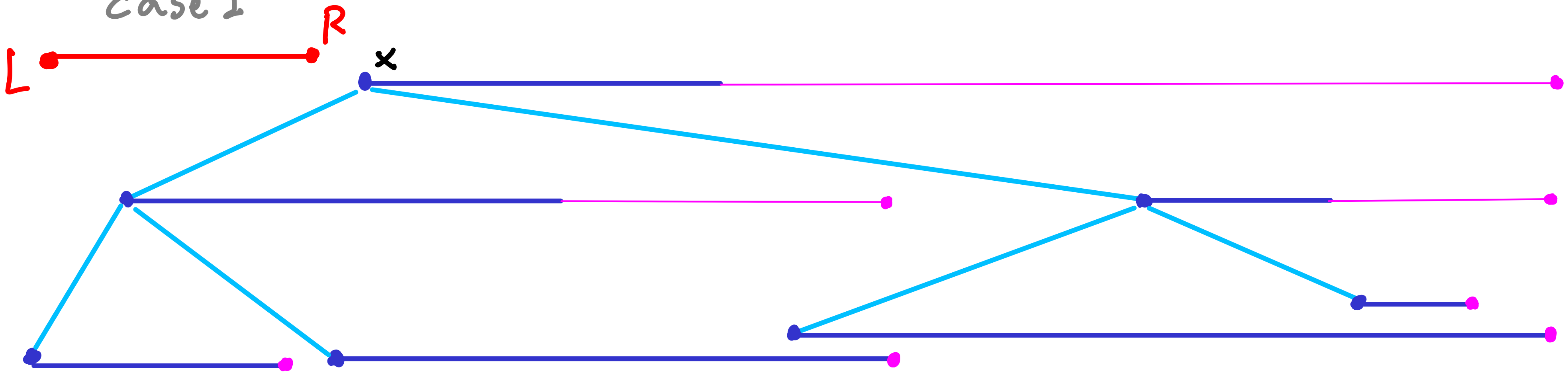


SEARCHING FOR OVERLAPPING INTERVALS

1D :

IF NO OVERLAP

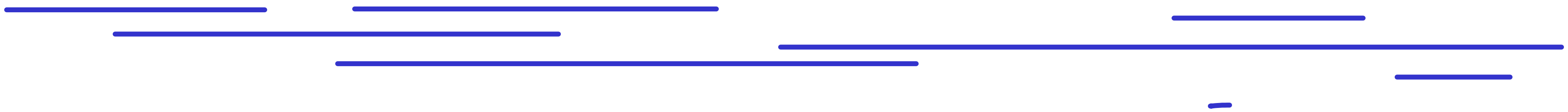
case 1

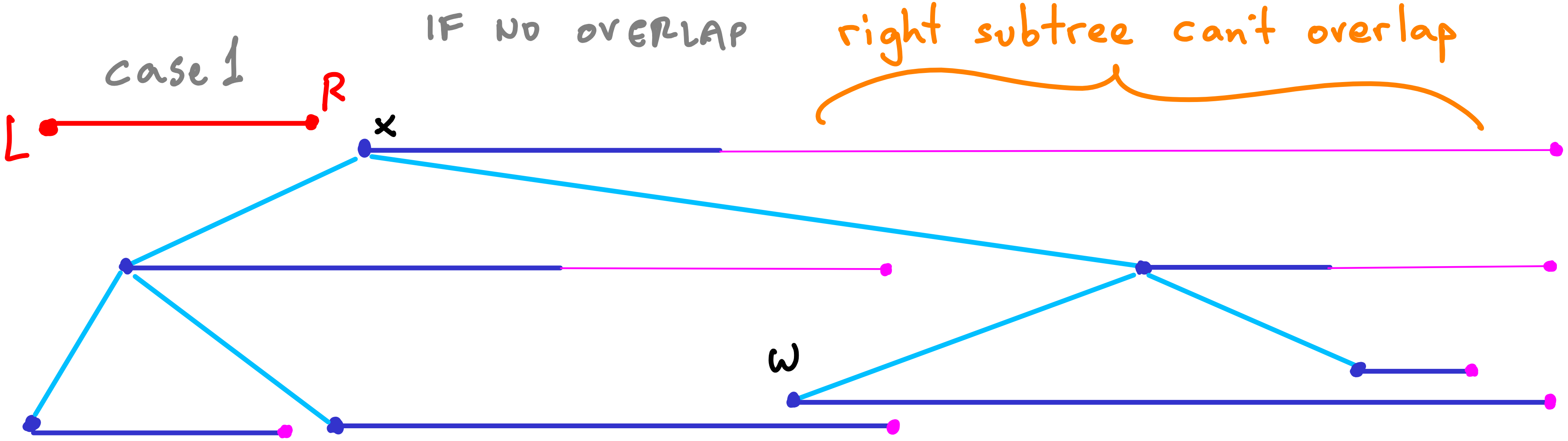


$R < x$

SEARCHING FOR OVERLAPPING INTERVALS

1D :





$$R < x < w$$

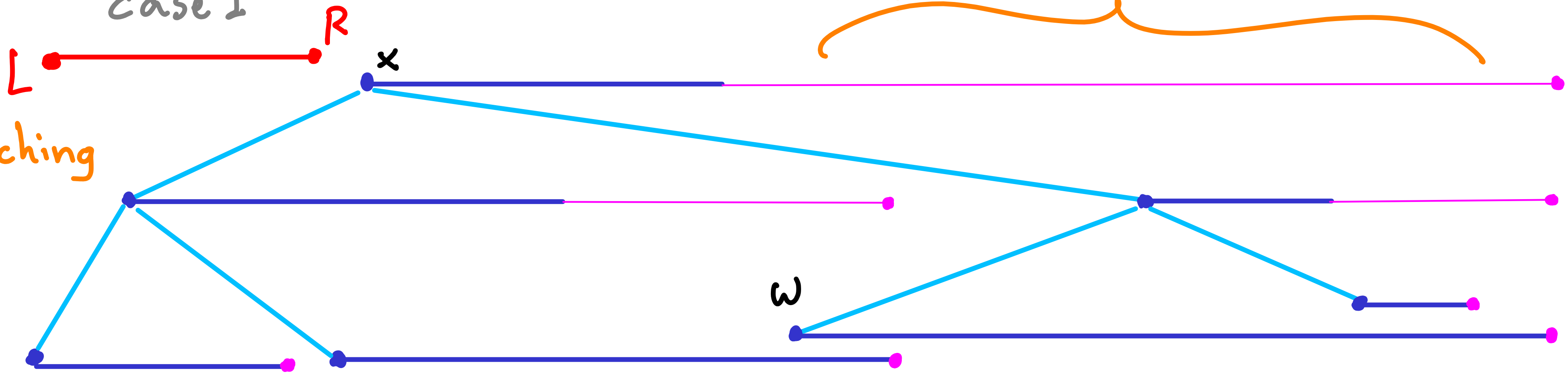
SEARCHING FOR OVERLAPPING INTERVALS

1D :

IF NO OVERLAP

right subtree can't overlap

case 1

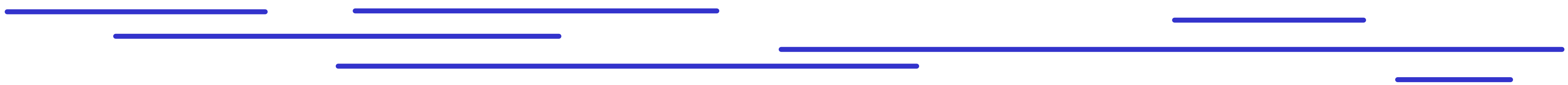


keep searching LEFT

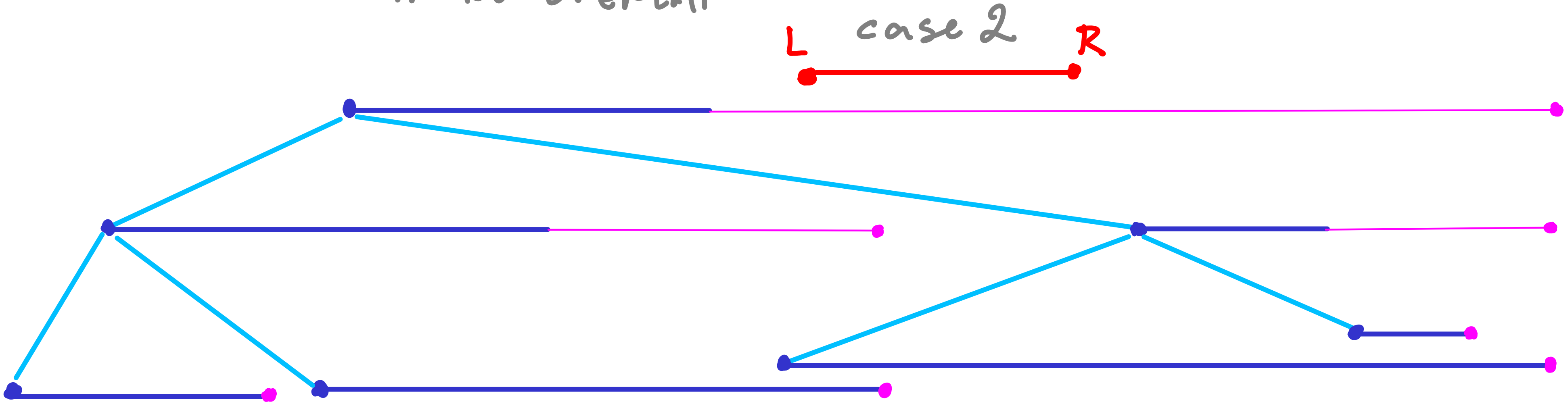
$R < x < w$

SEARCHING FOR OVERLAPPING INTERVALS

1D:



IF NO OVERLAP



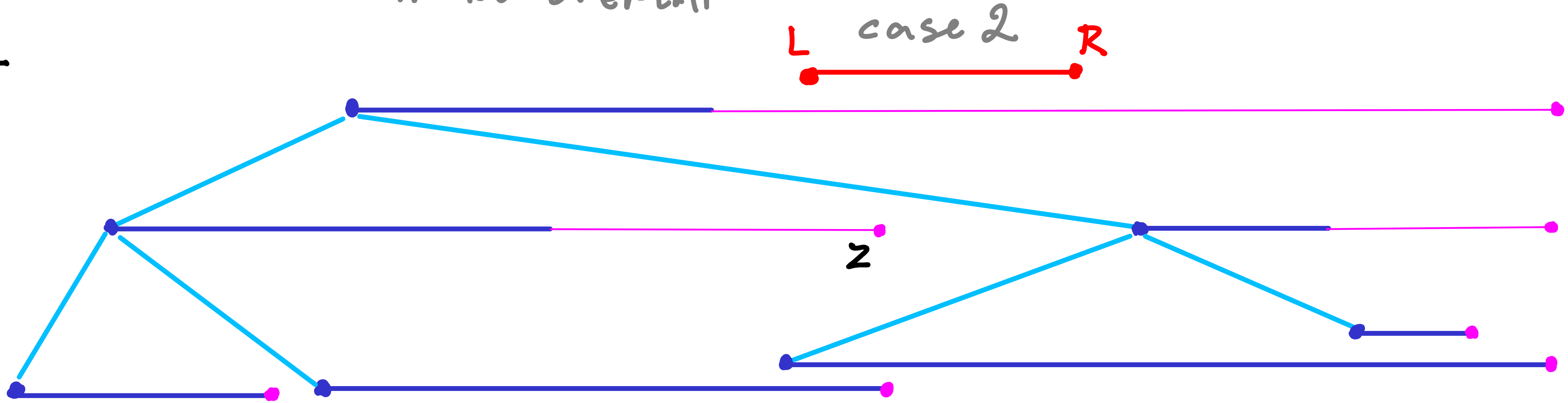
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF NO OVERLAP

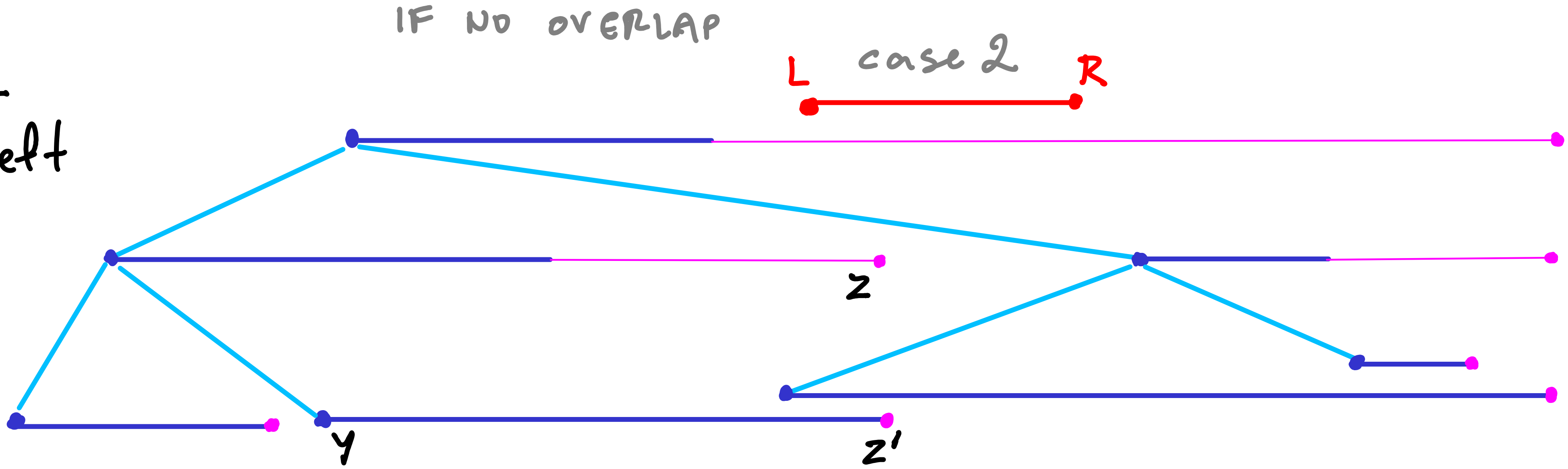
IF $z \geq L$

?



SEARCHING FOR OVERLAPPING INTERVALS

1D:



IF $z \geq L$

search left

$\exists\ \overline{yz'}$

s.t.

$y < L < z'$

guaranteed overlap

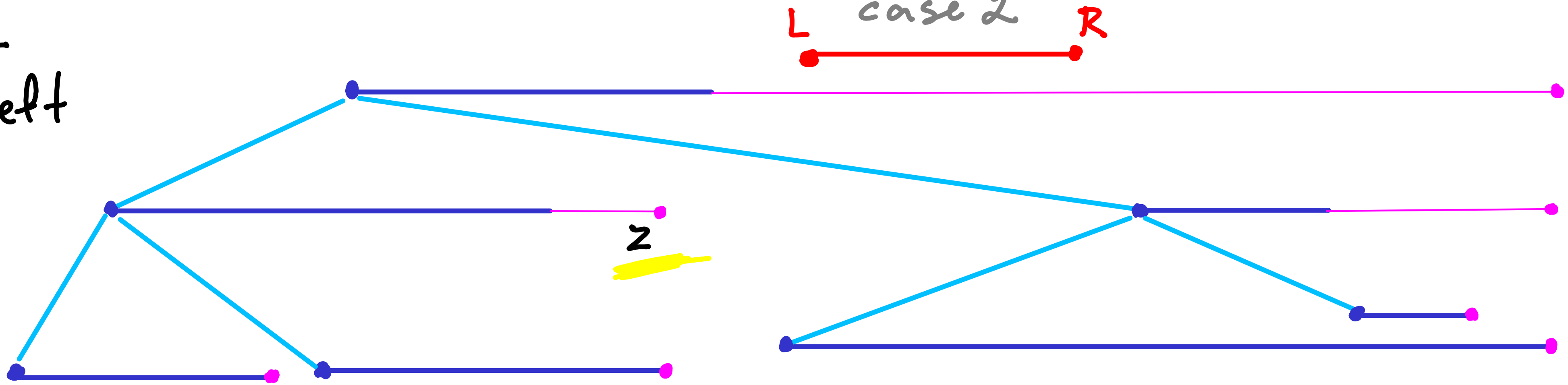
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF NO OVERLAP

IF $z \geq L$

search left



$\exists\ \overline{yz'}$

s.t.

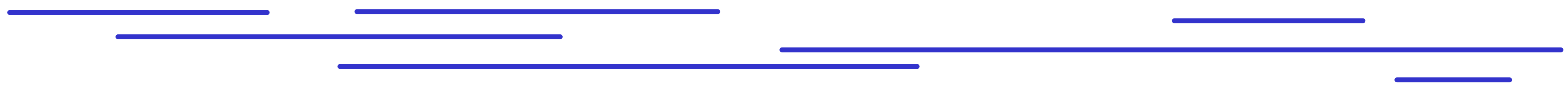
$y < L < z'$

guaranteed overlap

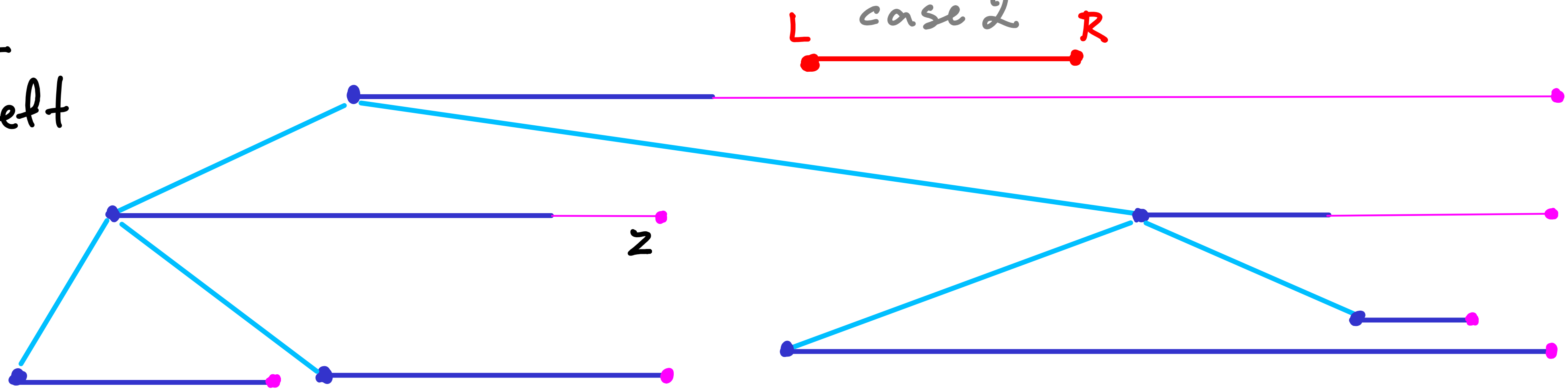
else $(z < L)$

SEARCHING FOR OVERLAPPING INTERVALS

1D :



IF NO OVERLAP



IF $z \geq L$

search left

$\exists$ $\overline{yz'}$

s.t.

$y < L < z'$

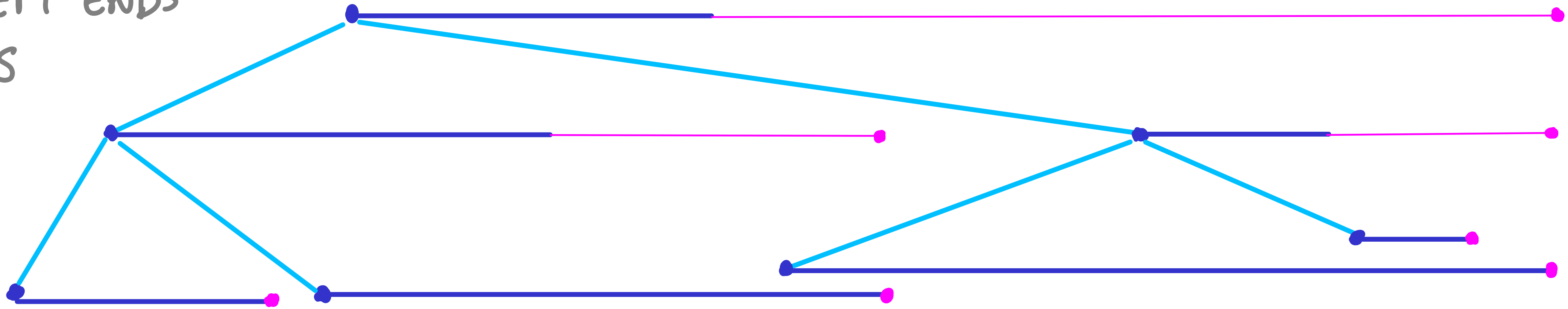
guaranteed overlap

else ($z < L$)

- No overlap to left
- search right

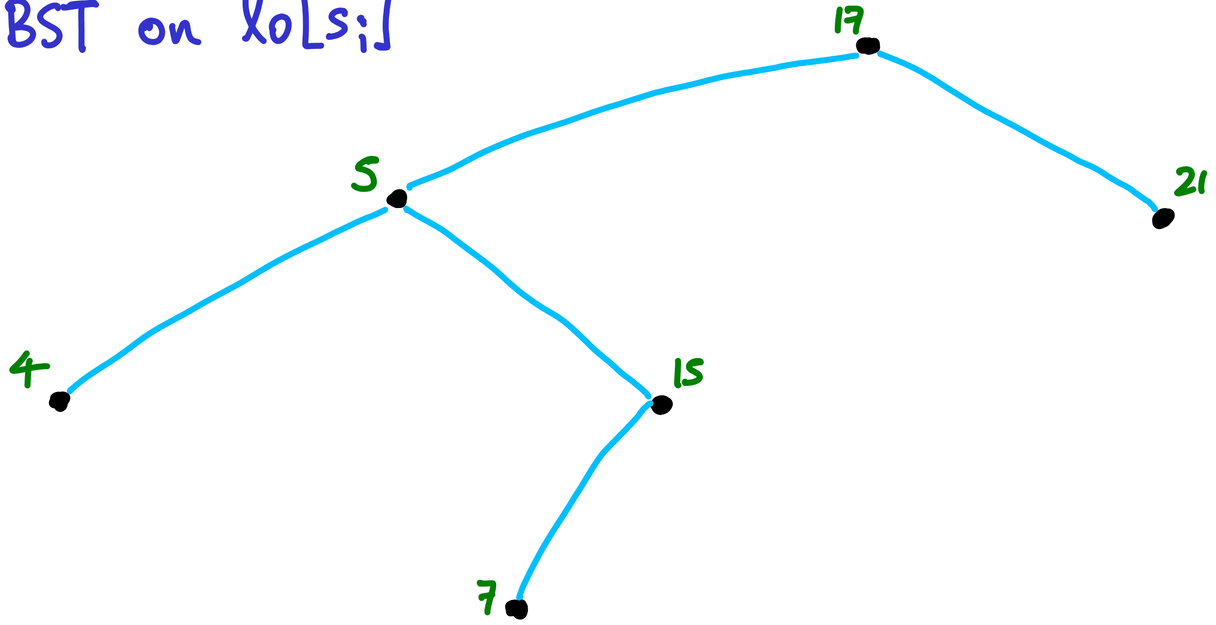
How can we update MAX RIGHT END of SUBTREE ?

BST w/ LEFT ENDS as KEYS

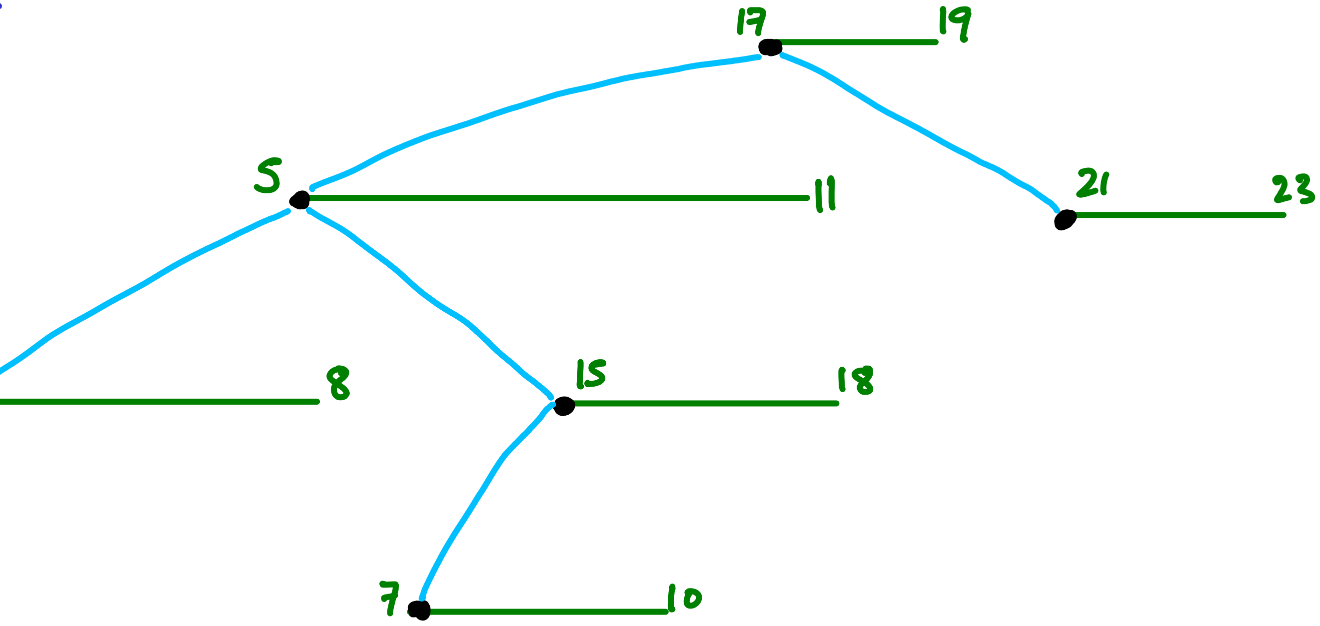



7 10
5 11
4 8
17 19
15 18
21 23

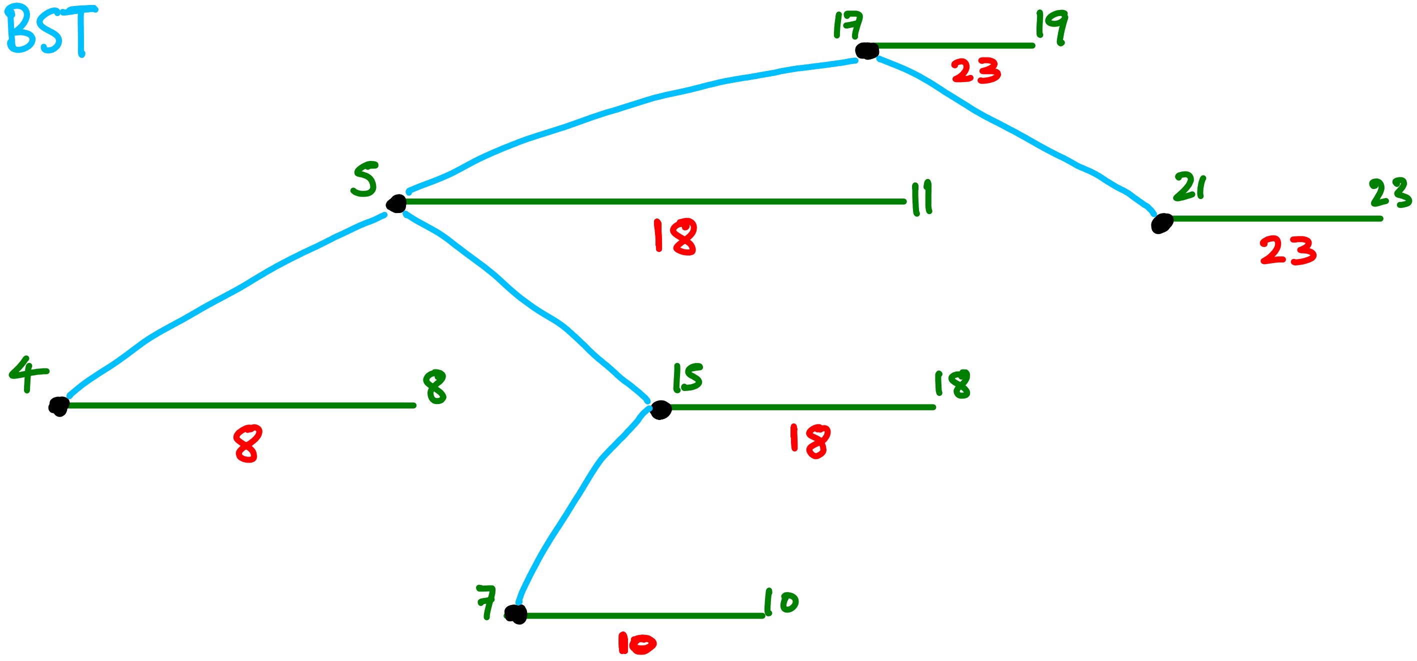
BST on $lo[s_i]$

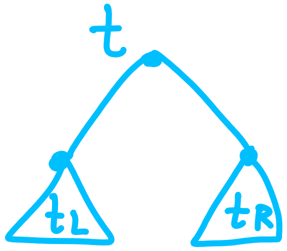


BST

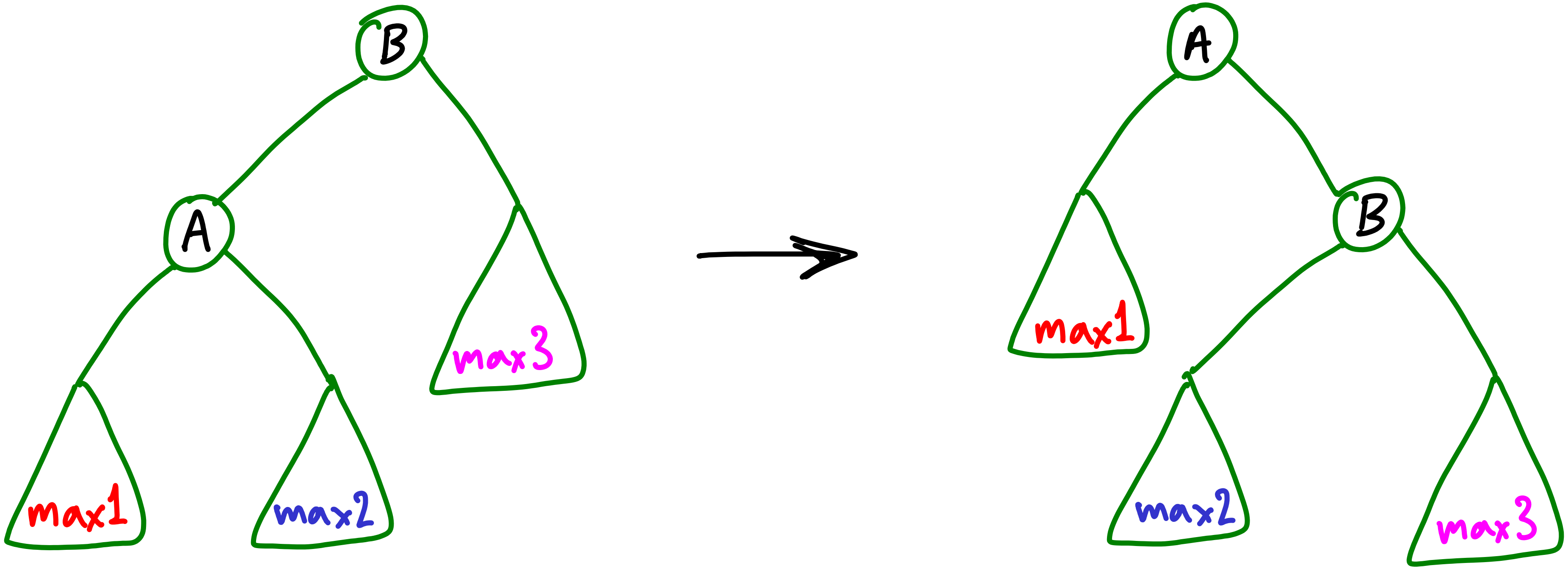


augmented
BST





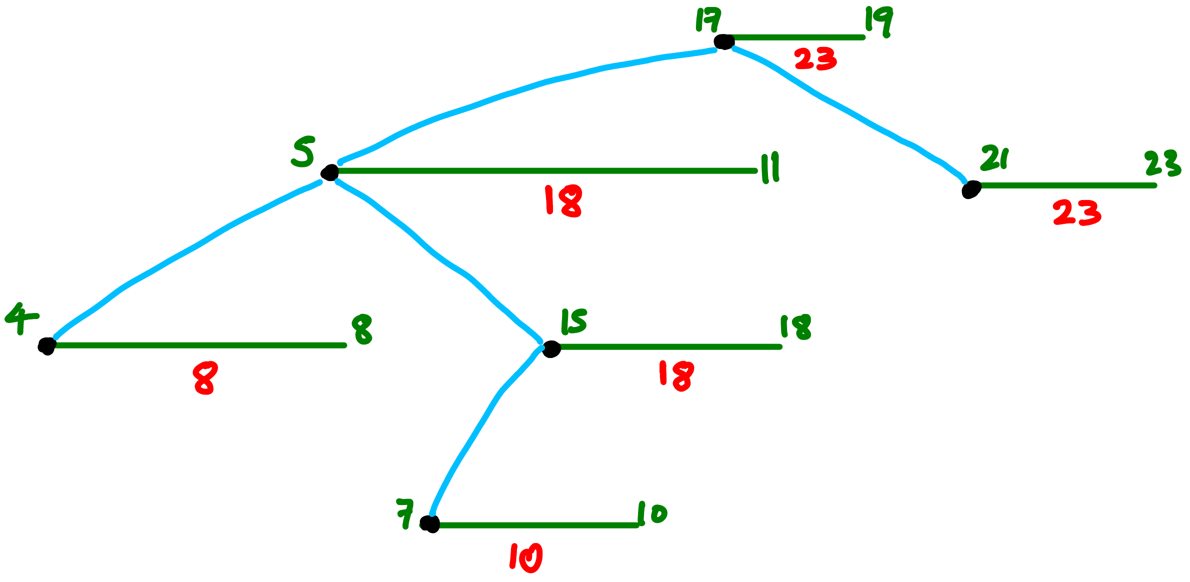
$$\max(t) = \max\begin{cases} hi(t) \\ \max(t_L) \\ \max(t_R) \end{cases}$$



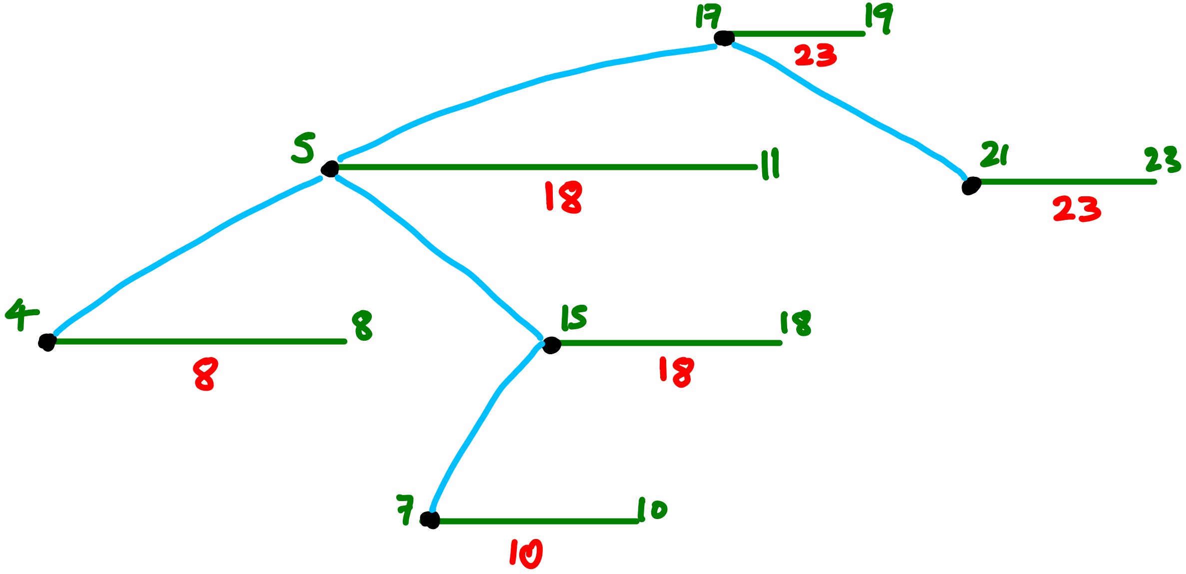
max1 , max2 , max3 : unchanged by rotation

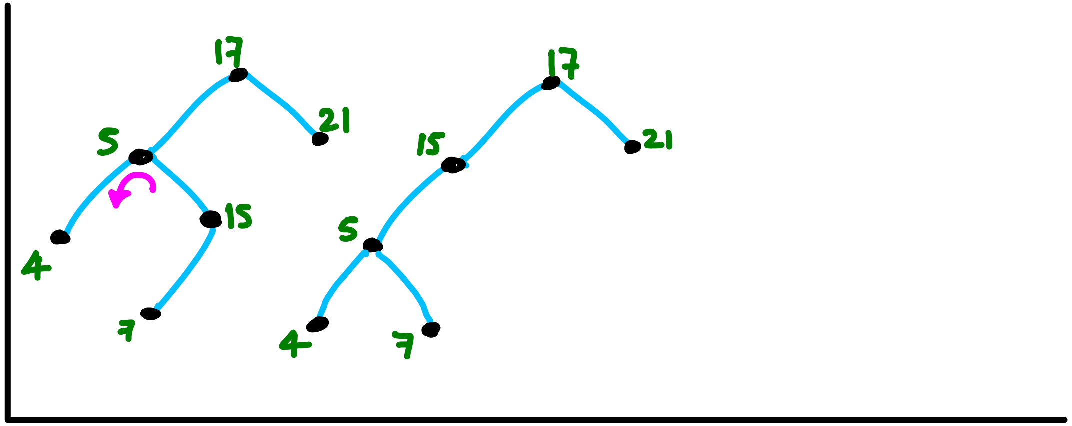
max(A) & max(B) : trivial to update

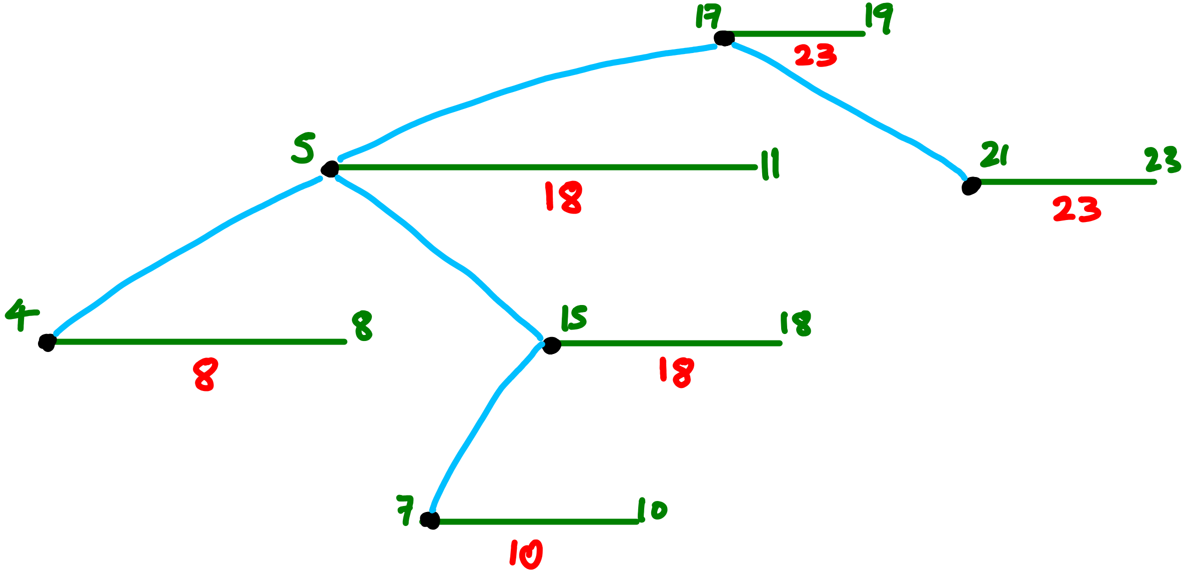
we can maintain a __balanced__ BST augmented w/ max value of subtrees

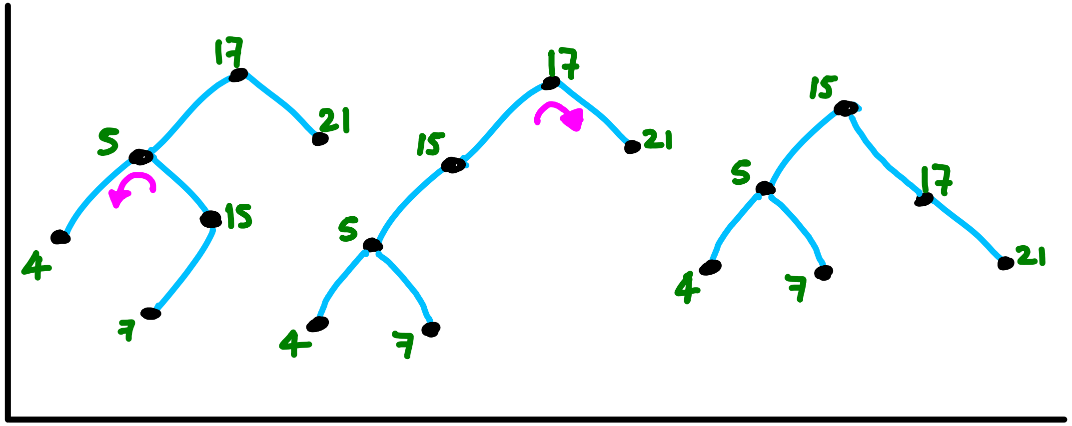


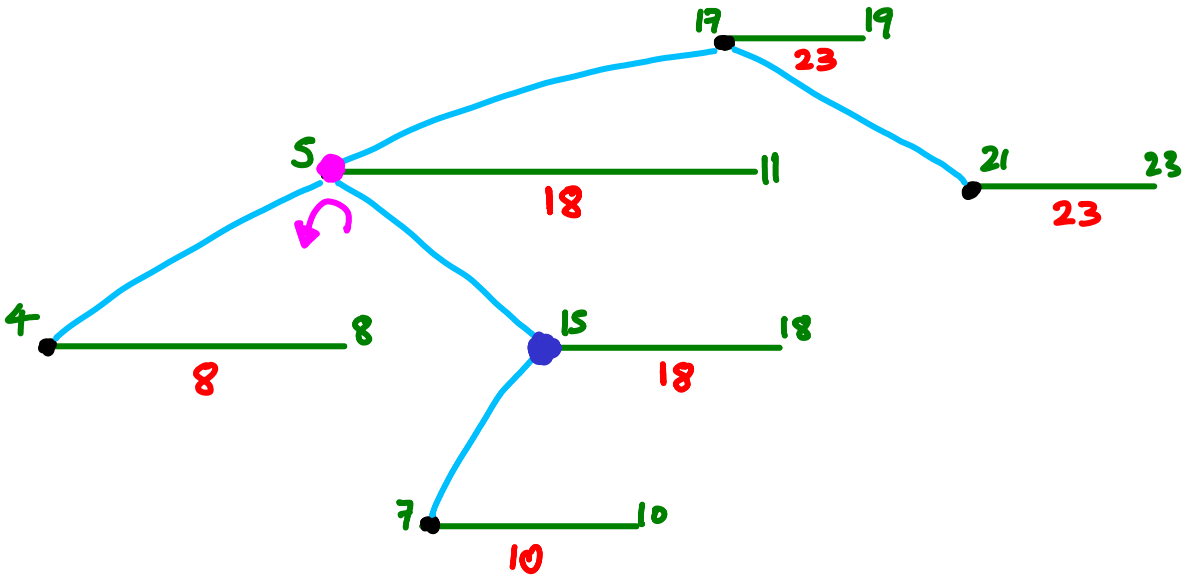
17
19
23
5
11
18
21
23
23
4
8
8
15
18
18
7
10
10

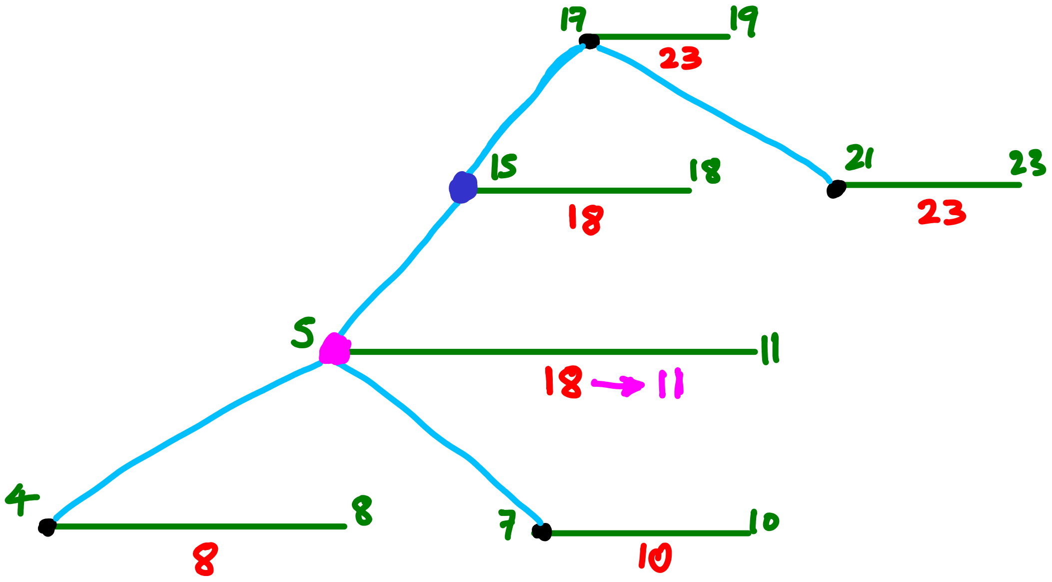

17
19
23
5
11
18
21
23
23
4
8
8
15
18
18
7
10
10

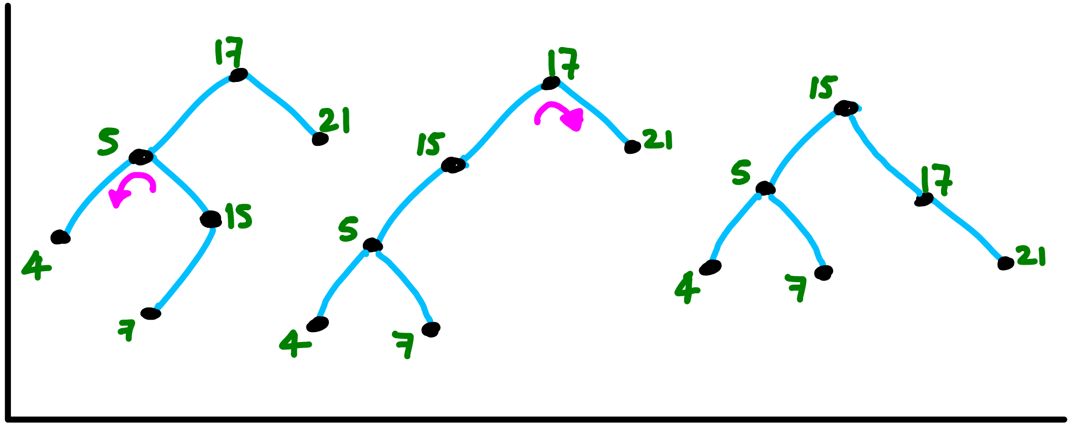

17
5
21
4
15
7
17
15
21
5
4
7

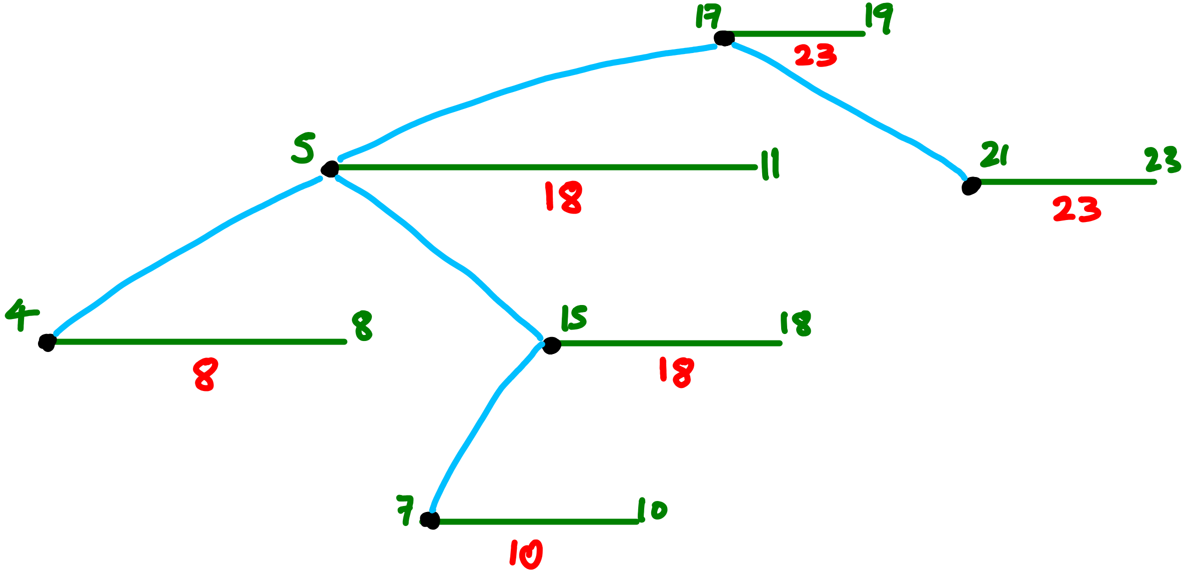

17
19
23
5
11
18
21
23
23
4
8
8
15
18
18
7
10
10

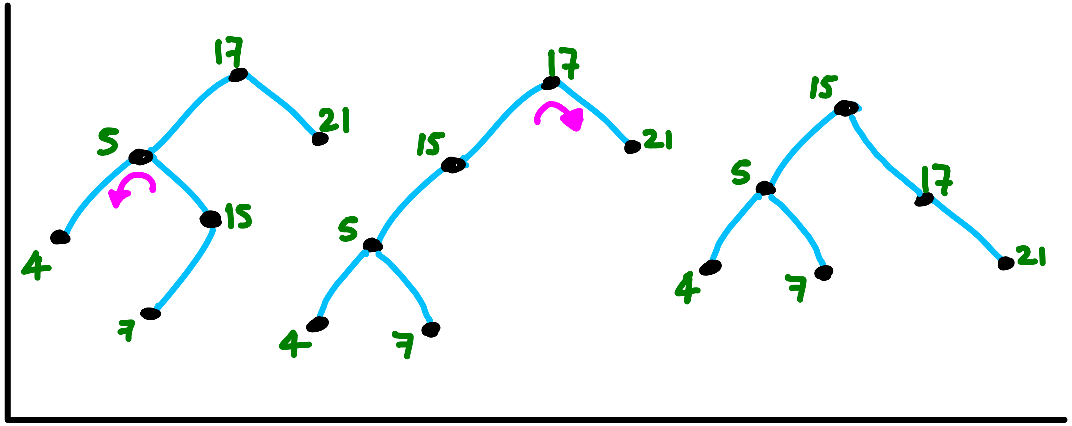

17
5
21
4
15
7
17
15
21
5
4
7
15
5
17
4
7
21

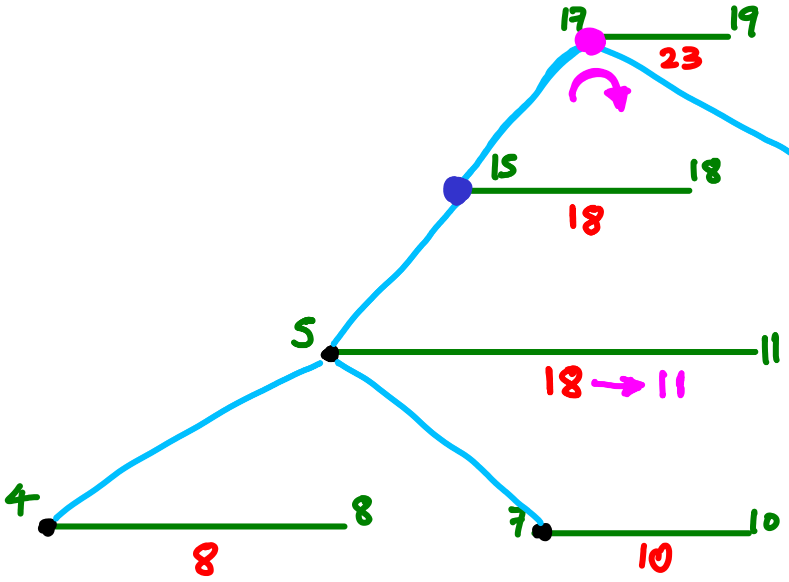

17
19
23
5
11
18
21
23
23
4
8
8
15
18
18
7
10
10

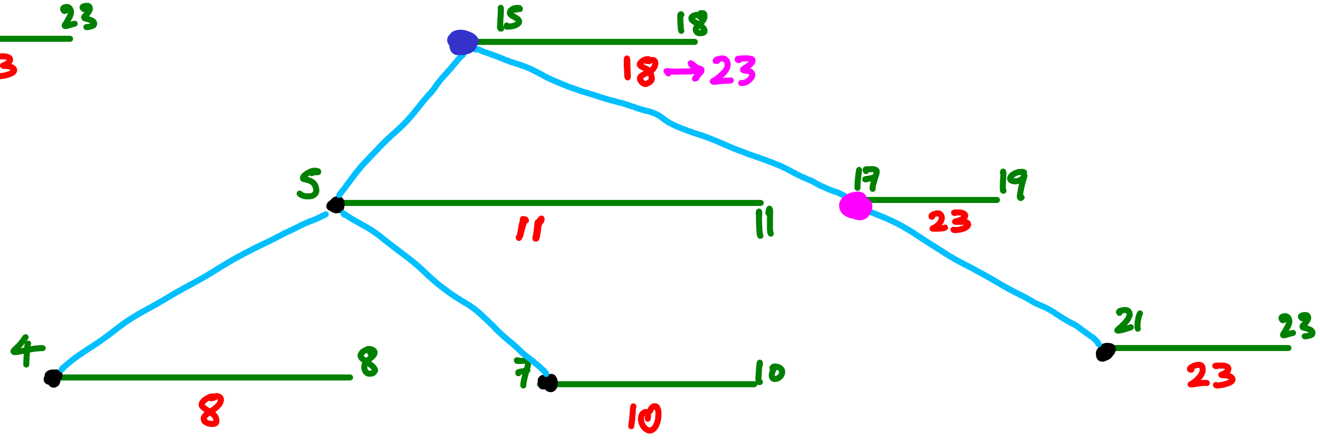

17
19
23
15
18
18
21
23
23
5
11
18 → 11
4
8
8
7
10
10


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


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


17
5
21
4
15
7
17
15
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