

09-XB

BST Application

Interval Tree

Your instinct, rather than precision stabbing, is more
about just random bludgeoning.

邓俊辉

deng@tsinghua.edu.cn

Stabbing Query

- ❖ Given a set of intervals in general position on the **x**-axis:

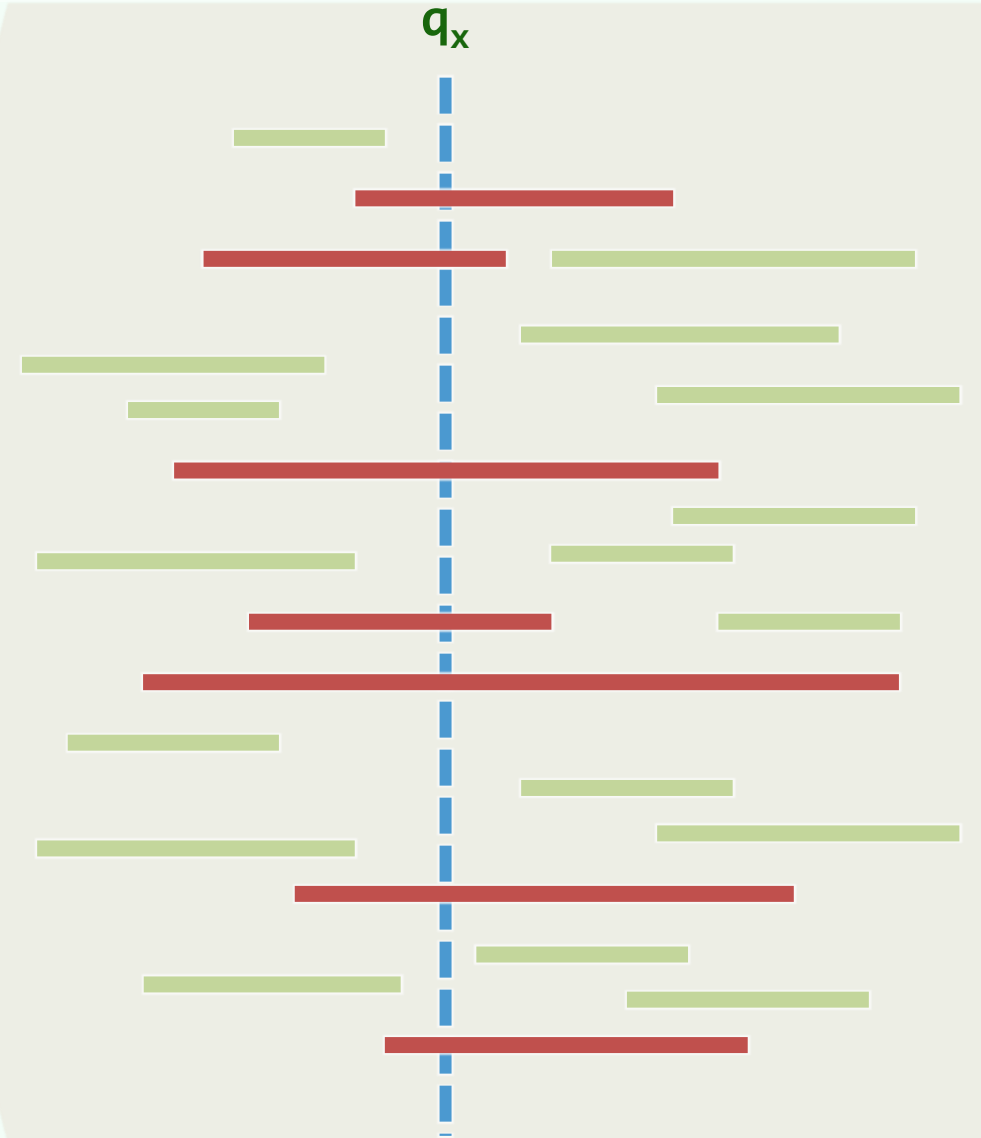
$$S = \{ s_i = [x_i, x_i'] \mid 1 \leq i \leq n \}$$

and a query point q_x

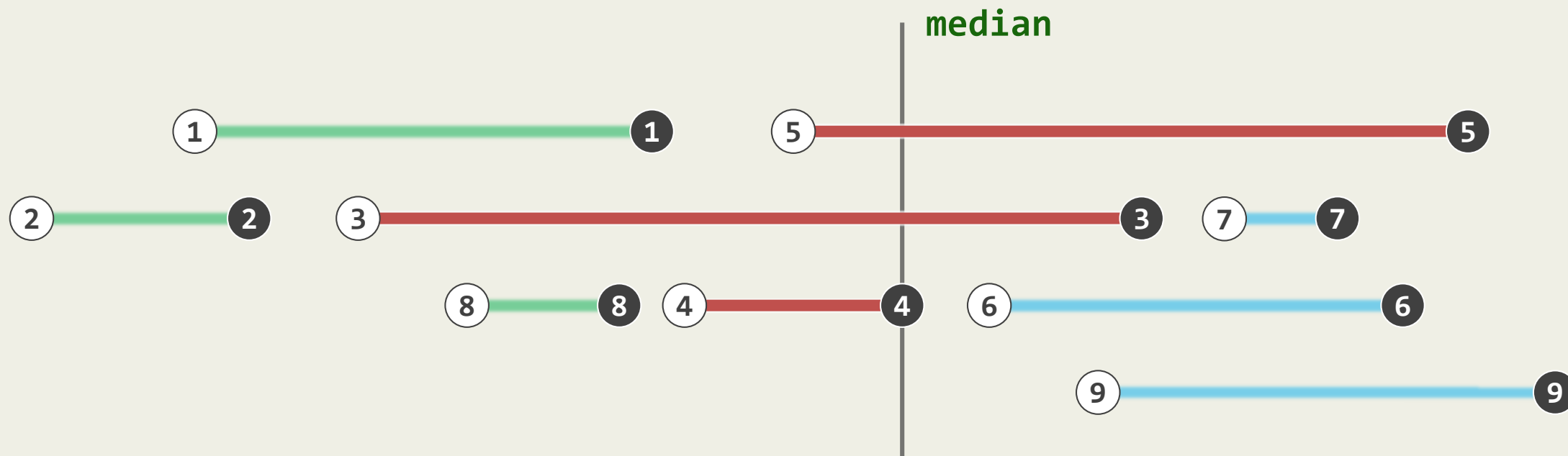
- ❖ Find all intervals that contain q_x

$$\{ s_i = [x_i, x_i'] \mid x_i \leq q_x \leq x_i' \}$$

- ❖ To solve this query,
we will use the so-called interval tree ...



Median

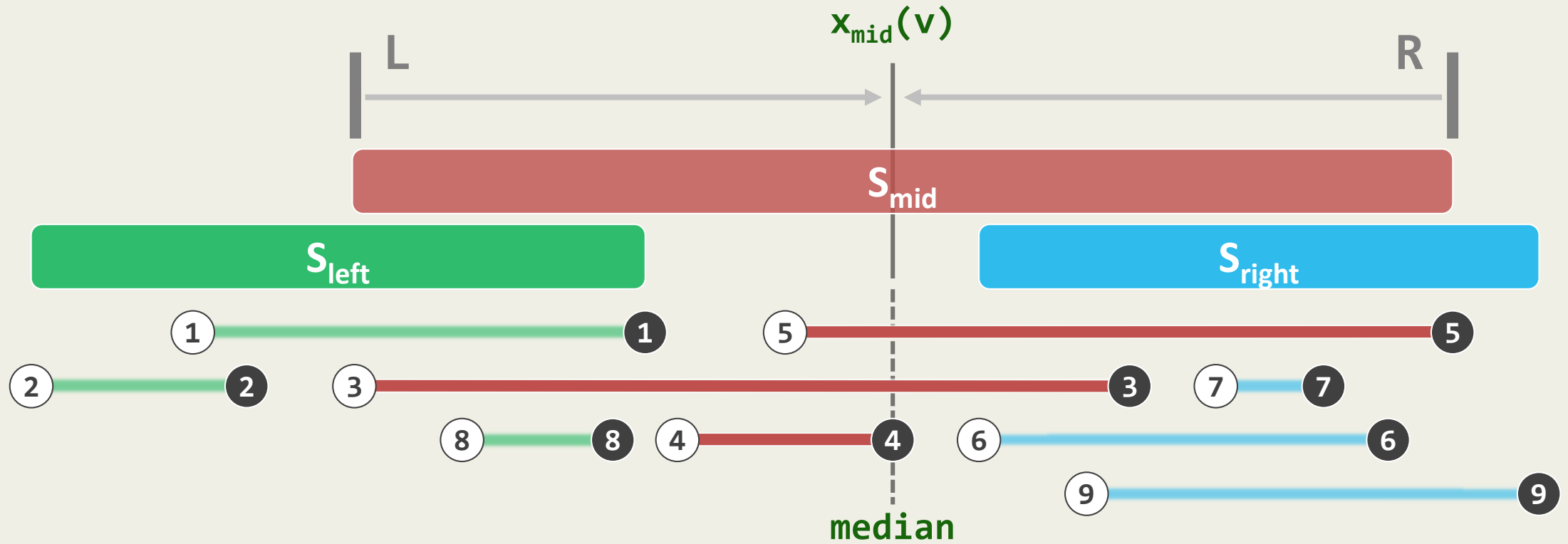


❖ Let $S = \{ s_1, \dots, s_n \}$ be the set of intervals

❖ Let $P = \partial S$ be the set of all endpoints //by general position assumption, $|P| = 2n$

❖ Let $\text{median}(P) = x_{\text{mid}}$ be the median of P

Partitioning



❖ All intervals can be then categorized into 3 subsets :

$$S_{left} = \{ S_i \mid x'_i < x_{mid} \} \quad S_{mid} = \{ S_i \mid x_i \leq x_{mid} \leq x'_i \} \quad S_{right} = \{ S_i \mid x_{mid} < x_i \}$$

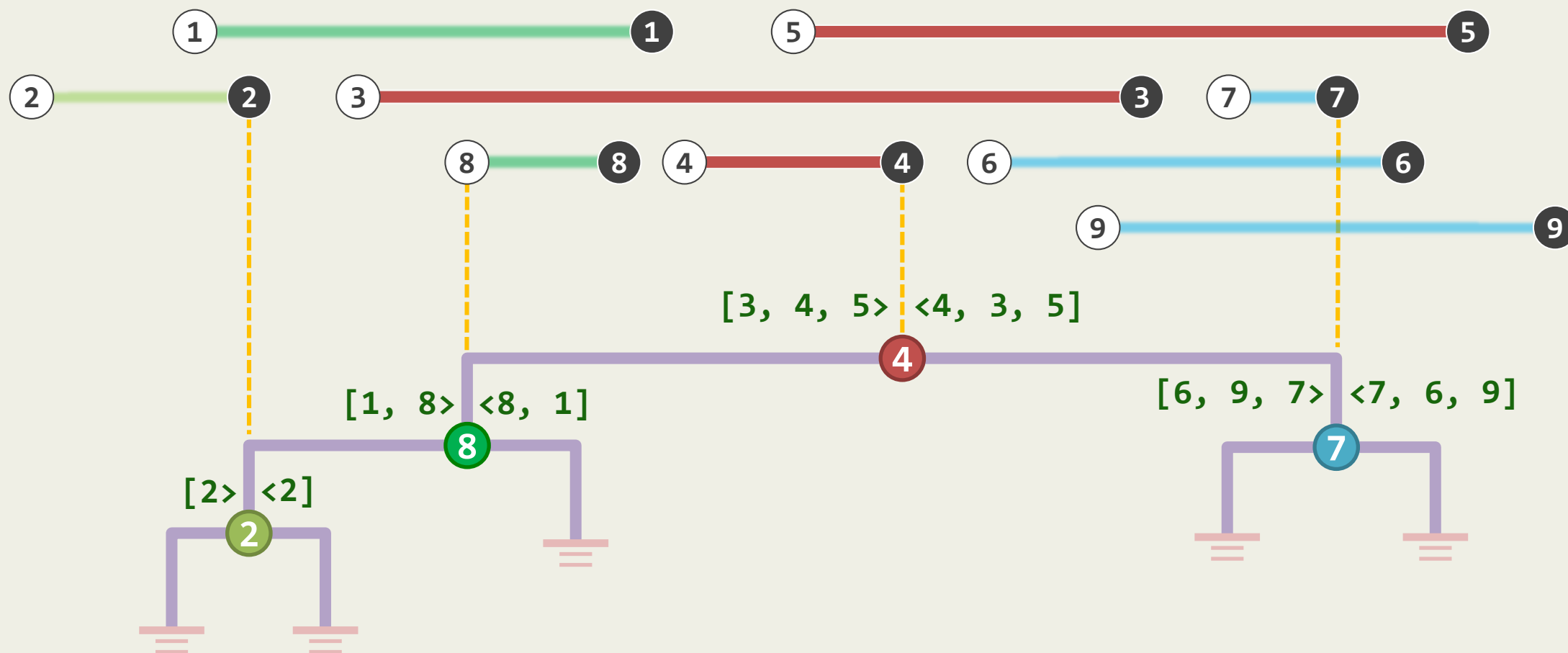
❖ $S_{left/right}$ will be **RECURSIVELY** partitioned until they are empty (leaves)

Balance & $O(\log n)$ Depth

$$\max\{ |S_{left}|, |S_{right}| \} \leq n/2$$

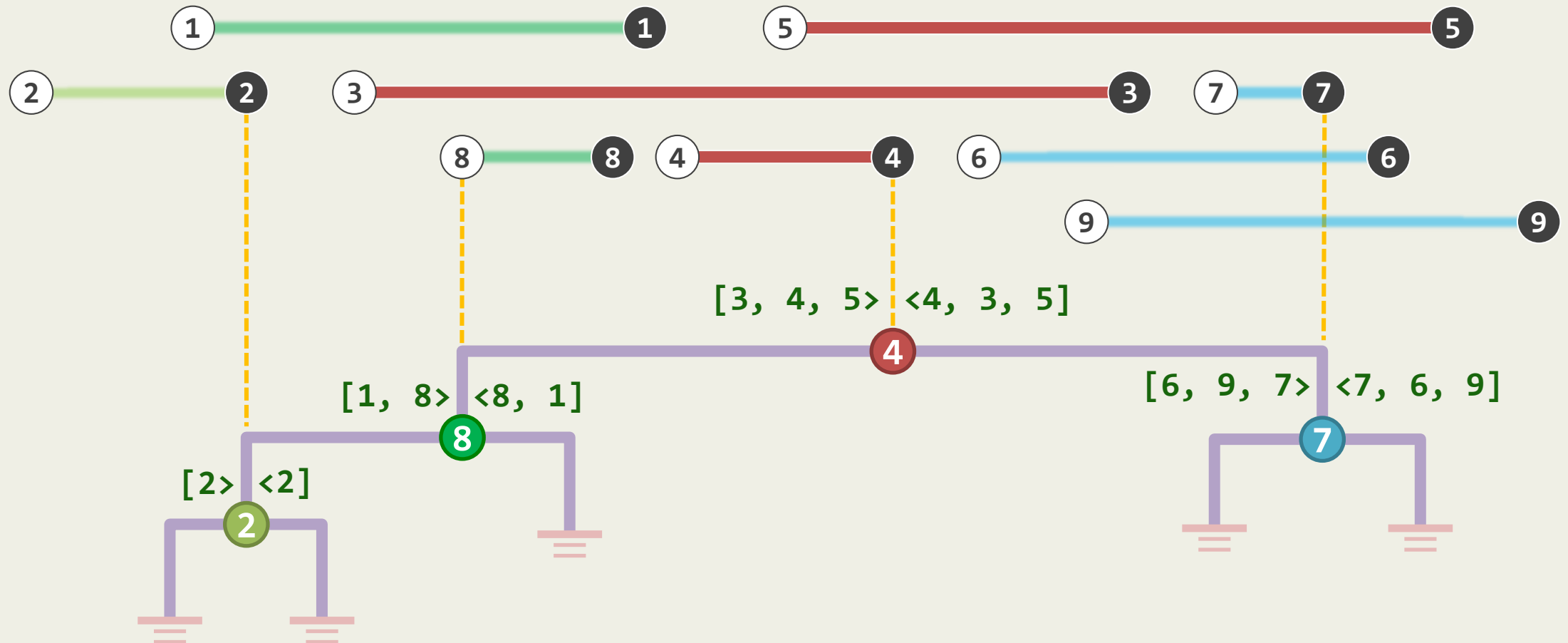
Best case: $|S_{mid}| = n$

Worst case: $|S_{mid}| = 1$



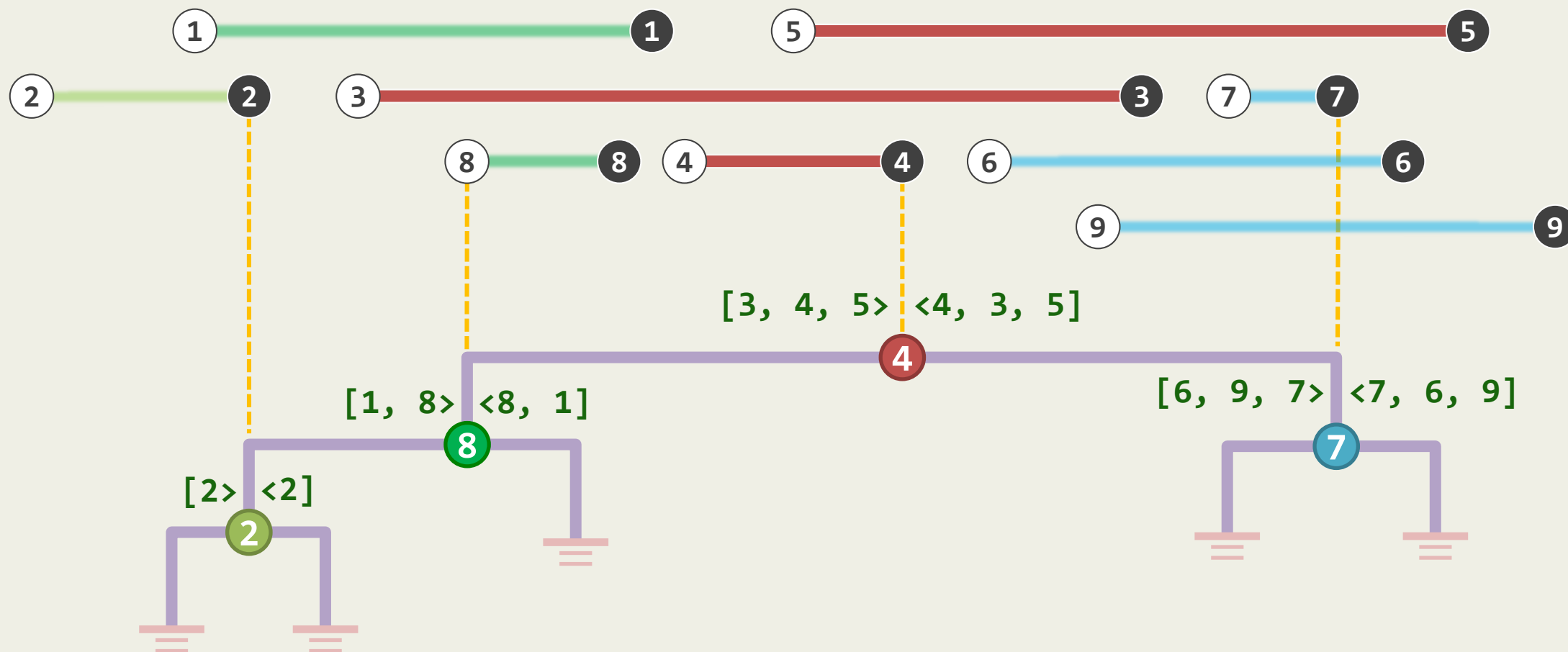
Associative Lists

❖ $L_{\text{left/right}}$ = all intervals of S_{mid} sorted by the **left/right** endpoints



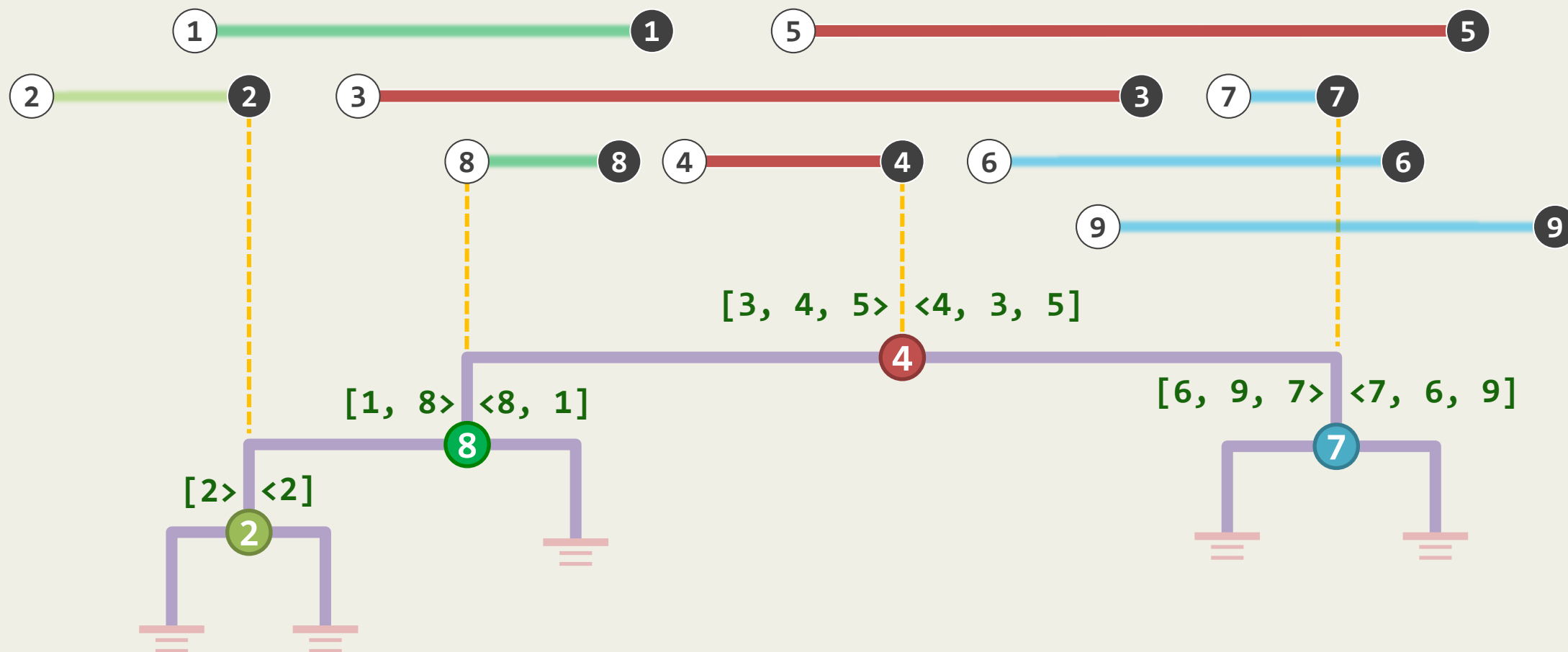
$O(n)$ Size

❖ Each segment appears twice (one in each list)



$O(n \log n)$ Construction Time

❖ Hint: avoid repeatedly sorting



queryIntervalTree(v , q_x)

```
if ( ! v ) return; //base
```

```
if (  $q_x < x_{mid}(v)$  )
```

```
    report all segments of  $S_{mid}(v)$  containing  $q_x$ ;
```

```
    queryIntervalTree(  $lc(v)$ ,  $q_x$  );
```

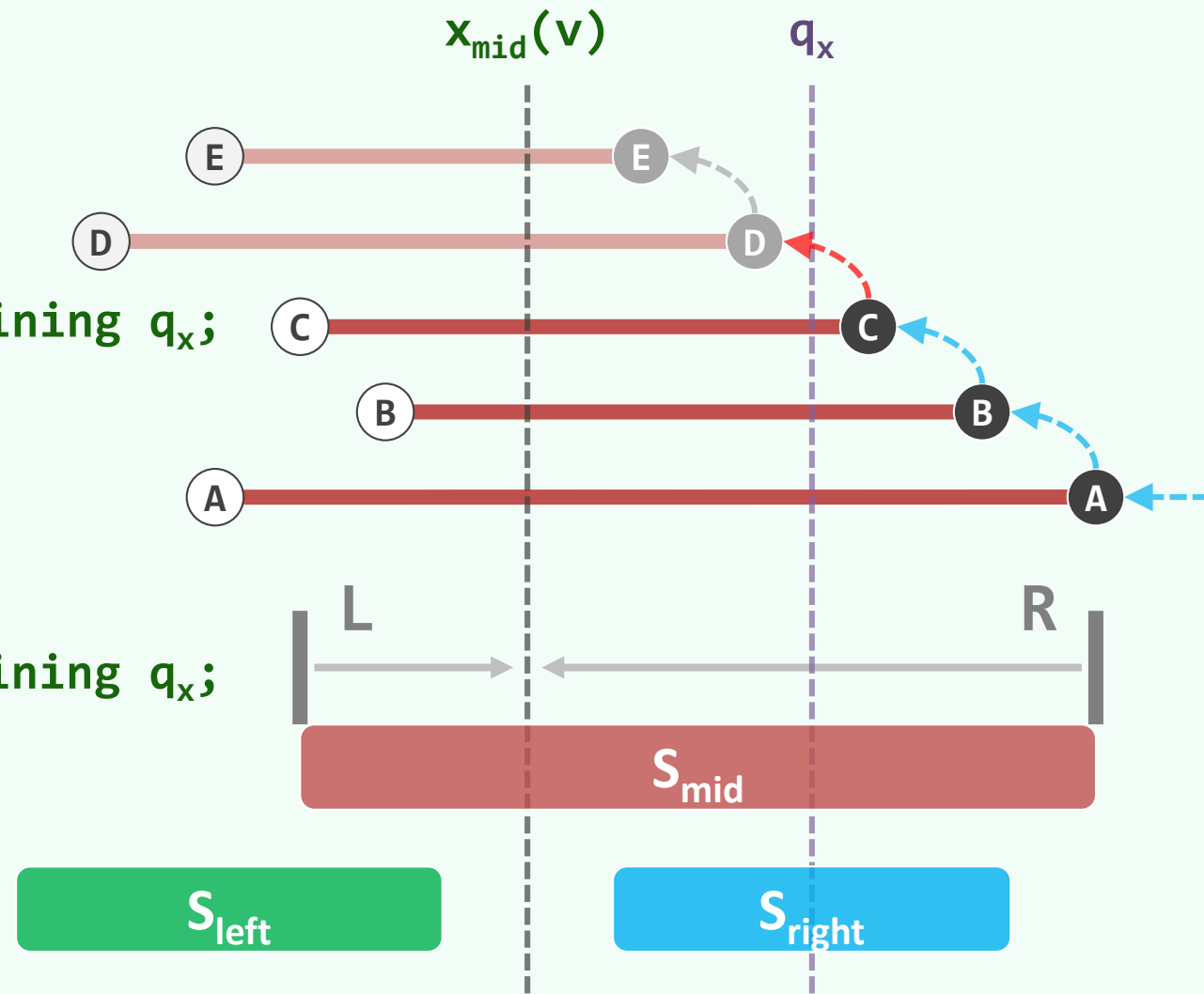
```
else if (  $x_{mid}(v) < q_x$  )
```

```
    report all segments of  $S_{mid}(v)$  containing  $q_x$ ;
```

```
    queryIntervalTree(  $rc(v)$ ,  $q_x$  );
```

```
else //with a probability  $\approx 0$ 
```

```
    report all segments of  $S_{mid}(v)$ ; //both  $rc(v)$  &  $lc(v)$  can be ignored
```



$O(r + \log n)$ Query Time

- ❖ Each query visits $\mathcal{O}(\log n)$ nodes

//LINEAR recursion

