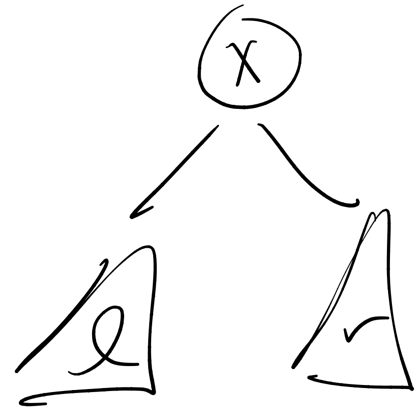
Lecture 10: Parallel Sorting

Insertion sort	$O(n^2)$	work
Mergesort on lists	$O(n \log n)$	work
	$O(n)$	span
Mergesort on trees	$O(n \log n)$	work
	$O((\log n)^3)$	span

Empty   Node(l, x, r)

o

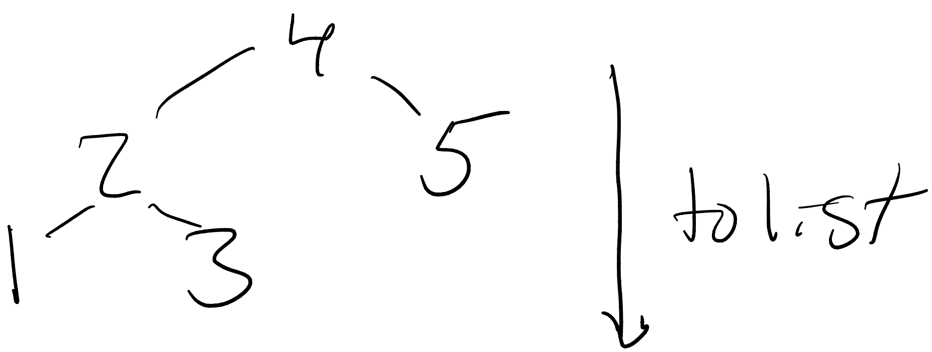


A tree t is sorted - - - -

1. to list(t) is a sorted tree

2. Inductive definitin

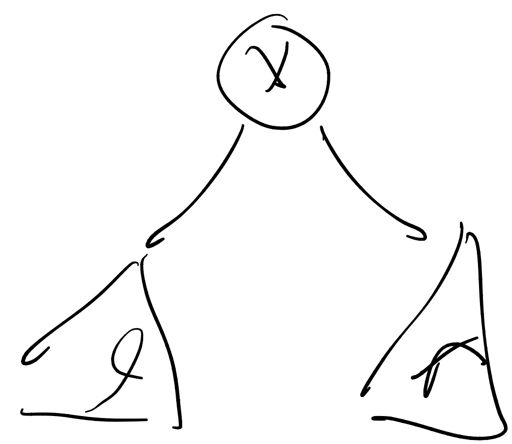
```
fun tolist(t) =
  case t of
     Empty => []
   | Node(l, x, r) => tolist(l) @ ([x]
                        @ tolist(r))
```



[1, 2, 3, 4, 5]

A tree is sorted iff

- It's Empty
- It's Node(l, x, r)
  - l is sorted
  - r is sorted
  - everything in l ≤ x
  - everything in r ≥ x



$l \le x$

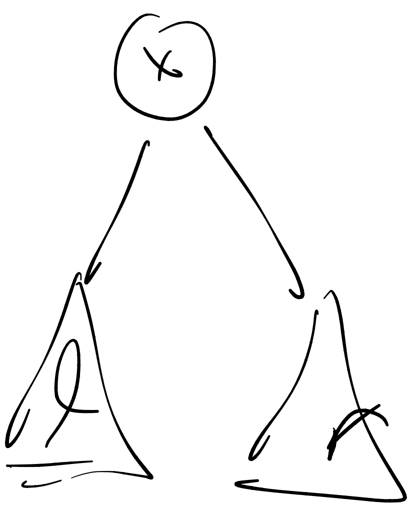
$x \le r$

Mergesort:

1. split into two subproblems, each of half the size
2. recur to sort
3. merge resulting sorted <u>trees</u> together into one.

```
(* Spec: mergesort(t) is a sorted tree
         with the same numbers
         as t, assume t is balanced *)

fun mergesort (t: tree) : tree =
   case t of
      Empty => Empty
    | Node(l, x, r) =>
   merge
   (merge( mergesort l, mergesort r),
     Node(Empty, x, Empty))
```

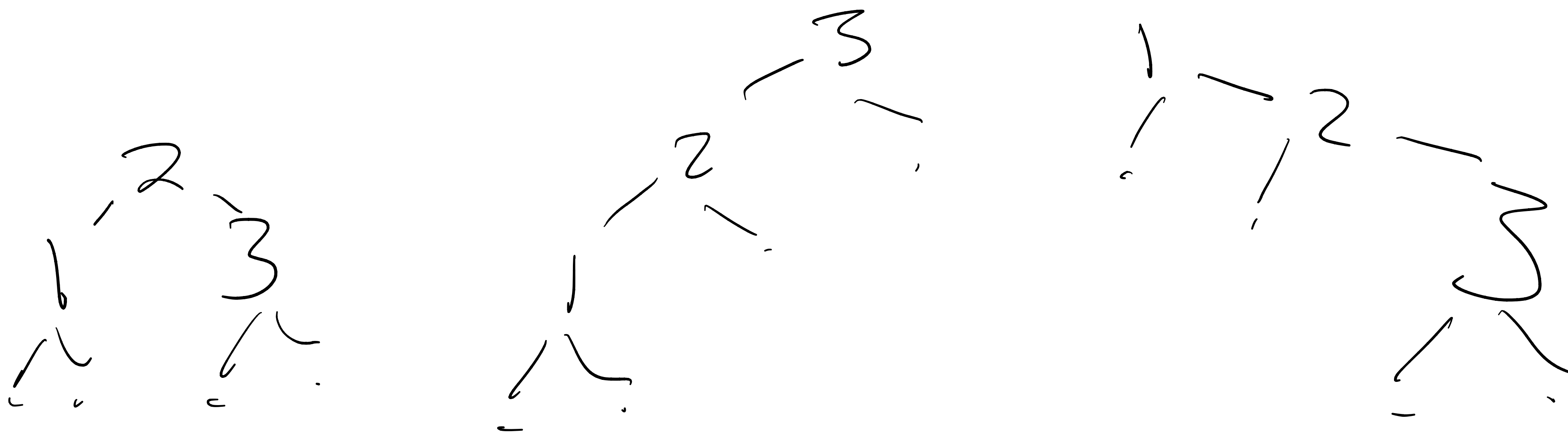


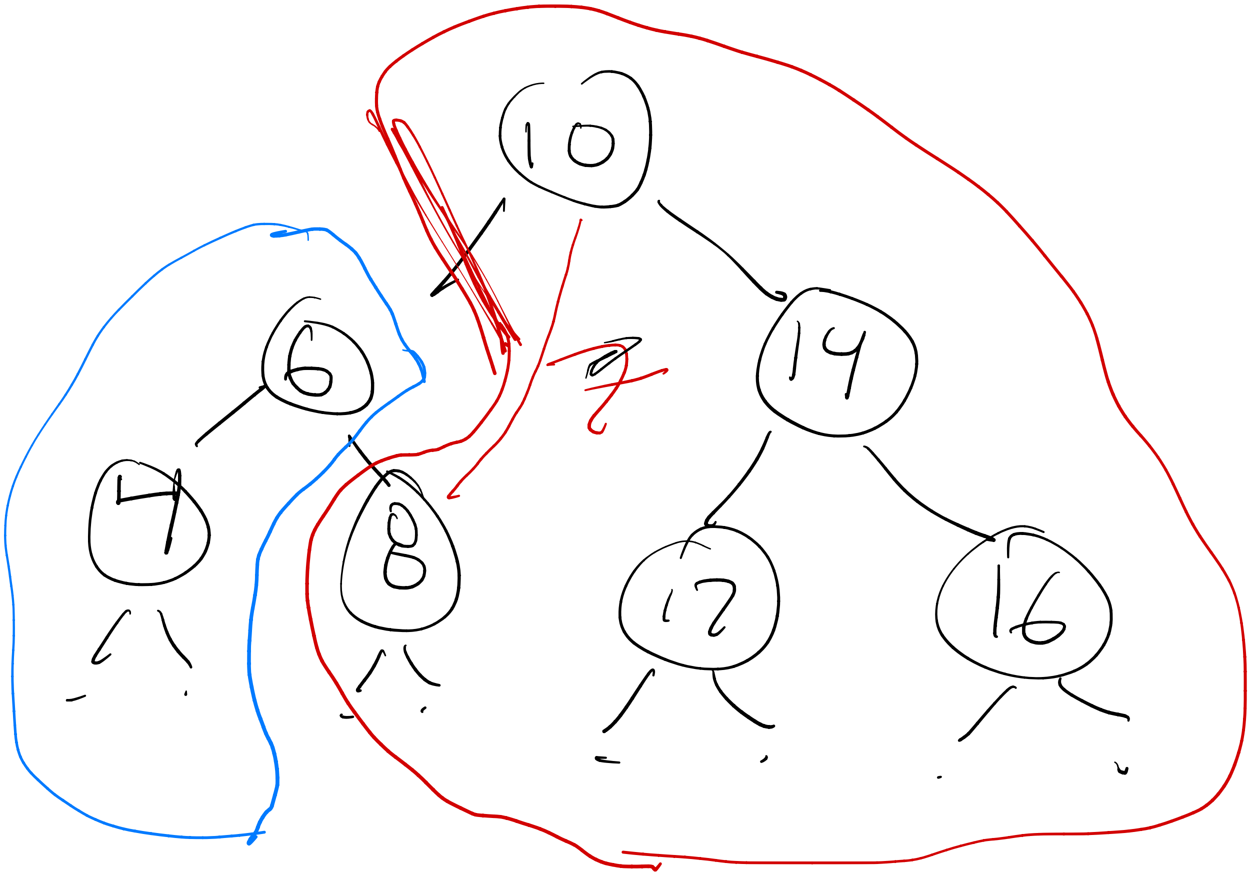
```
(* Spec: given two sorted trees,
   output a sorted tree with
   the same elts as both *)

fun merge (t1 : tree, t2 : tree) : tree =
  case t1 of
    Empty => t2
  | Node(l1, x, r1) =>
      let val (l2, r2) =
            split(t2, x)
      Node( merge(l1, l2),
            x,
            merge(r1, r2))
  end
```

everything in t2 that is ≤ X

everything in t2 that is ≥ X





want

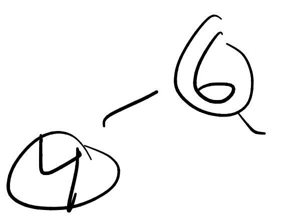
- everything $\le x$
- everything $> x$

$\downarrow$

output is sorted still

bound = 7

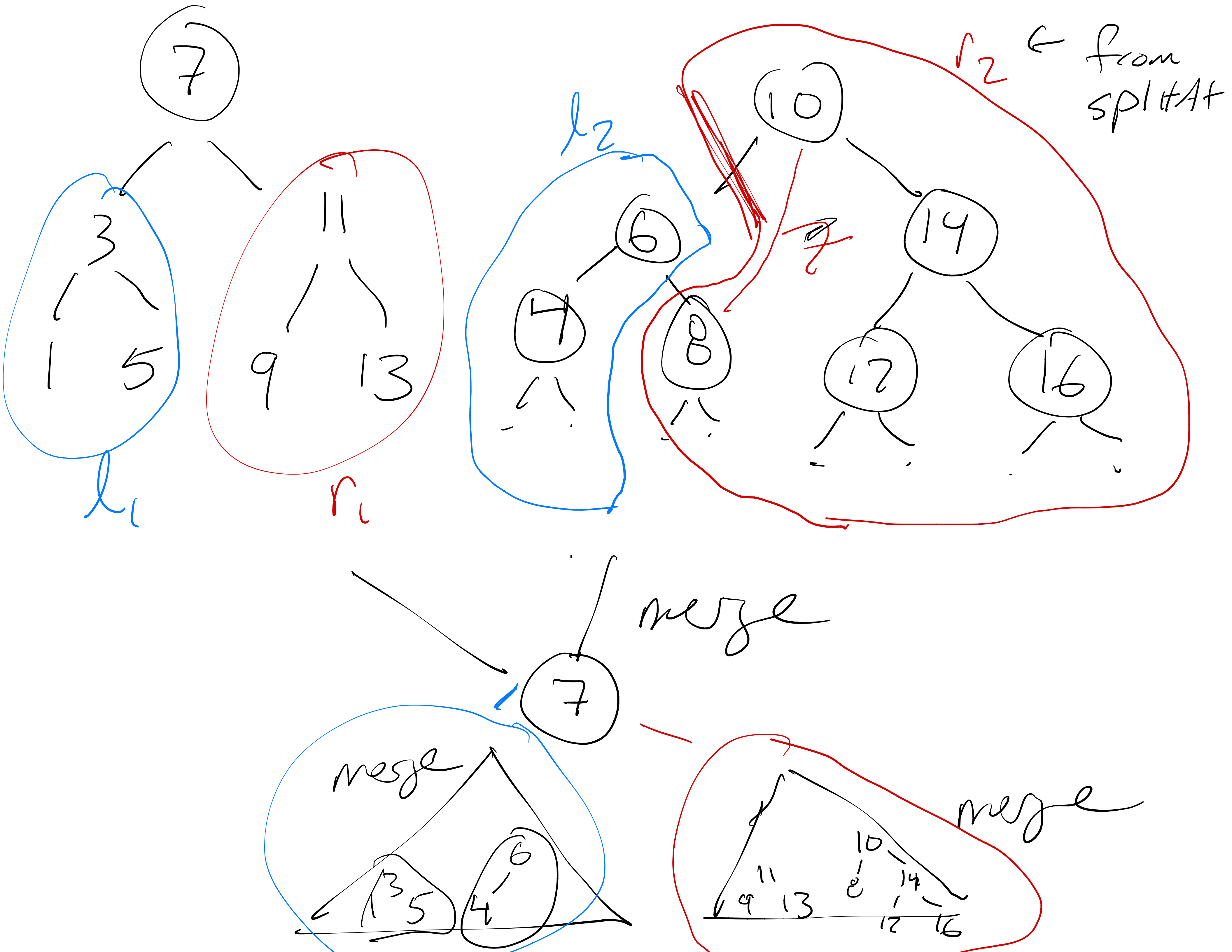
$x = 10$

$ll =$ 

$lr = 8$

```
(* Spec: given a sorted tree t, make (l, r)
     where l is        ≤ bound, sorted
           r is        > bound *)

fun splitAt(t: tree, bound: int) : tree * tree =
    case t of
        Empty => (Empty, Empty)
      | Node(l, x, r) => case bound < x of
            true => let val (ll, lr) = splitAt(l)
                    in (ll, Node(lr, x, r))
                    end
          | false => let val (rl, rr) = splitAt(r)
                     (Node(l, x, rl), rr)
                     end
```



← from splitAt
r_2
l_2
l_1
r_1
merge
merge
merge

```
(* Spec: mergesort(t) is a sorted tree
         with the same numbers
         as t, assuming t is balanced *)

fun mergesort (t: tree) : tree =
    case t of
       Empty => Empty
     | Node(l, x, r) => rebalance
       merge
       (merge( mergesort l, mergesort r),
        Node(Empty, x, Empty)))
```

Case for Empty: TS: ms(Empty) is a sorted tree w/ elts as Empty

mergesort(Empty)

$\mapsto$ Empty

Case for Node(l, x, r):

IH for l: mergesort(l) is a sorted tree with the same elts as l

IH for r: mergesort(r) is a sorted tree with the same elts as r

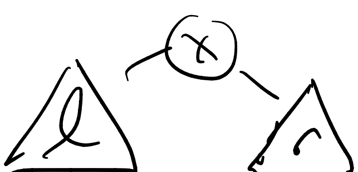
TS: mergesort(Node(l, x, r)) is a sorted tree w/ same elts as Node(l, x, r) mergesort(Node(l, x, r))

$\mapsto$ merge(merge(mergesort l,
mergesort r),
Node(Empty, x, Empty))

sorted and elts as l

sorted and elts as r

Sorted? by spec for merge,
merge(ms l, ms r) is sorted
by spec for merge again,
merge(that, x) is sorted

elts  spec for merge

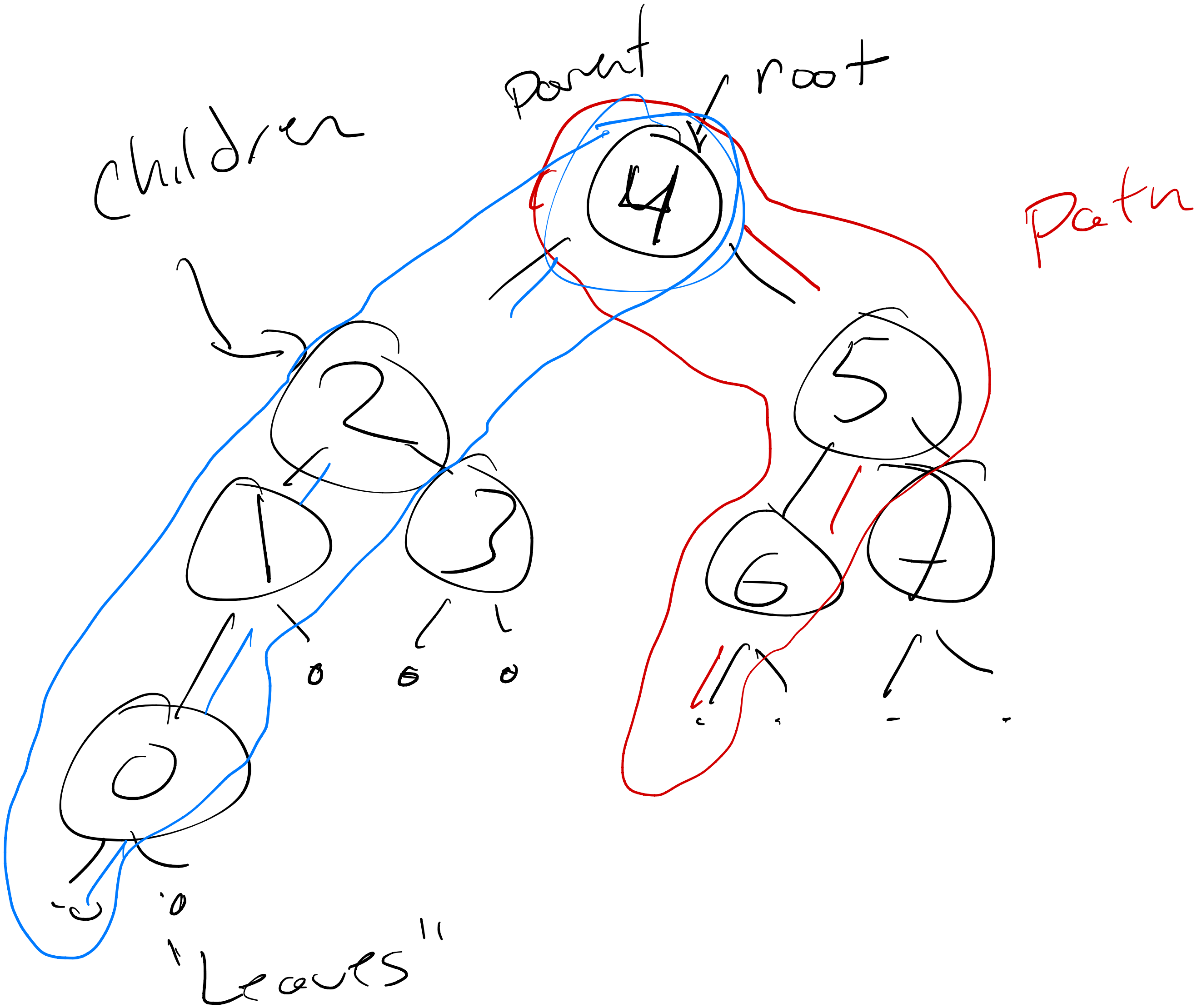
Work and span

$$W_{ms}(n) = \underbrace{\text{“}W_{merging}\text{”}}_{O(n)} + 2\,W_{ms}\left(\frac{n}{2}\right)$$

size of the tree

$$= n + 2\,W_{ms}\left(\frac{n}{2}\right)$$

$$O(n \log n)$$



depth = length (nodes/elts) of longest path from root to a leaf

$$S_{\mathit{splitAt}}(\underset{\substack{\uparrow \\ \text{depth of} \\ \text{tree}}}{d}) \le 1 + S_{\mathit{split}}(d-1)$$

is $O(d)$

$$S_{\mathit{merge}}(\underset{\substack{\uparrow \\ \text{depth of} \\ t_1}}{d_1}, \underset{\substack{\uparrow \\ \text{depth} \\ t_2}}{d_2}) = S_{\mathit{split}}(d_2) + S_{\mathit{merge}}(d_1 - 1, d_2)$$

$$= \underbrace{d_2 + S_{\mathit{merge}}(d_1 - 1, d_2)}_{O(d_1 \cdot d_2)}$$

$$S_{\mathit{mergesort}}(\quad) =$$

$$S_{mergesort}(n) = S_{mergesort}\left(\frac{n}{2}\right) + S_{merge}\left(\frac{\log n}{2}, \frac{\log n}{2}\right)$$

size of the balanced tree

d is proportional to $\log_2 n$

depth of msl

msr

$$+ \; S_{merge}\left(\frac{2\log n}{2}, \frac{1}{1}\right)$$

depth of merge(ms l, ms r)

[depth(merge(l, r)) ≤ depth l + depth r]

[mergesort outputs a balanced tree?]

$Sms(n) = Sms\left(\frac{n}{2}\right) + (\log n)^2$

$\cancel{+2\log n}$

$Sms(n) \approx Sms\left(\frac{n}{2}\right) + (\log n)^2$

$T(n) = T\left(\frac{n}{2}\right) + (\log n)^2$

idea: loop runs $(\log n)$ times

$$\leq \underbrace{\log(n)^2 + \log(n)^2 + \cdots + \log(n)^2}_{\log(n) \text{ times}}$$

Smergesort(n) is $O((\log n)^3)$

mergesort for trees

is <u>more</u> parallelizable

than ms for lists