Lecture 12:

- finish mergesort on trees

(- Polymorphism?)

Divide + Combine

1) Divide problem into subproblems

2) Recur

3) Combine results

<u>mergesort</u>	vs <u>QS</u>
<u>split</u>	filter less greater
<u>merge:</u> based on sort	<u>append</u>

Time

$$W(n) = \underline{W_{divide}} + \underbrace{2W\left(\frac{n}{2}\right)}_{divide} + \underline{W_{combine}}$$

$$S(n) = \underline{S_{divide} + S_{combine}} + S\left(\frac{n}{2}\right)$$

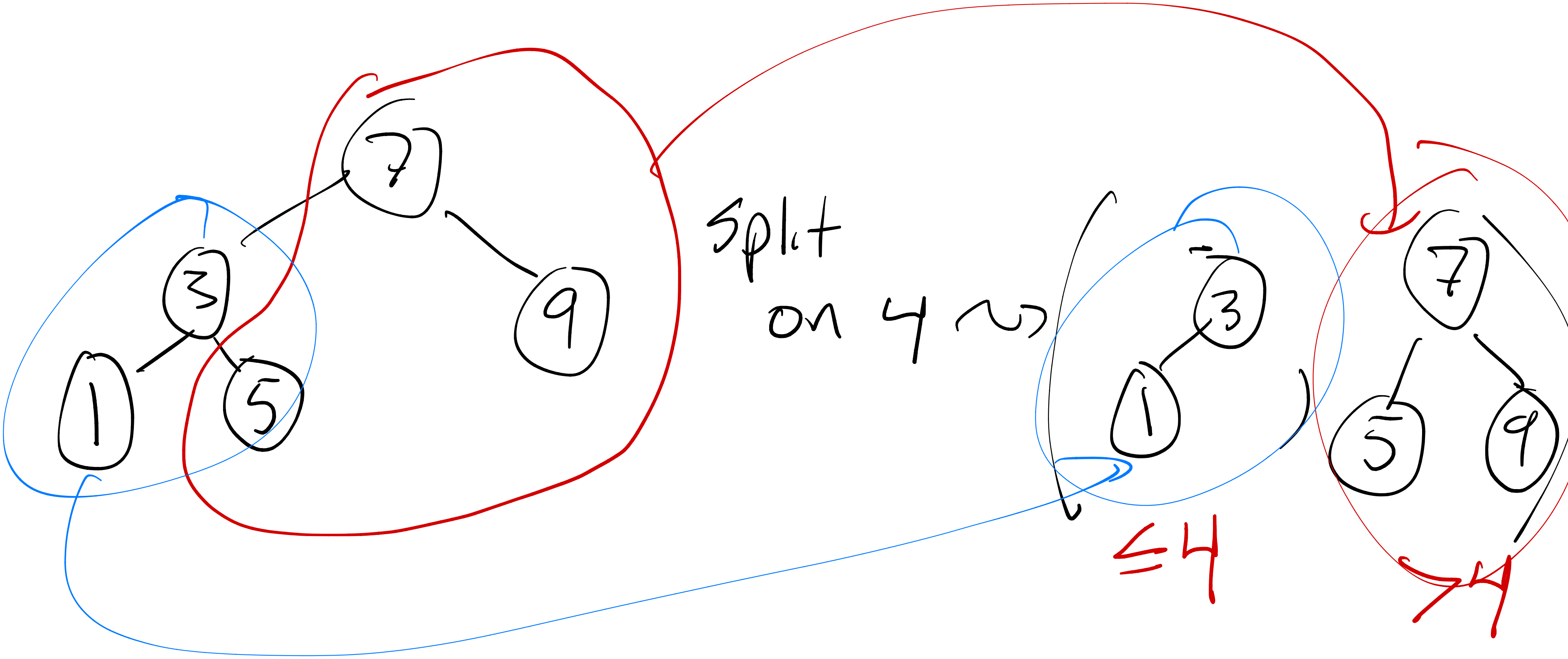
```
fun ms (t: tree) : tree =
  case t of
    Empty => Empty
  | Node(l, x, r) =>
```

split divide

```
      merge( merge(ms(l), ms(r)),
             Node(Empty, x, Empty))
```

Combine

```
fun merge (l1:int list, l2:int list) =
    case (l1, l2) of
         ⋮
       (x::xs, y::ys) =>
           case x<y
              true => x::merge(xs, l2)
            | false => y::merge(l1, ys)
```



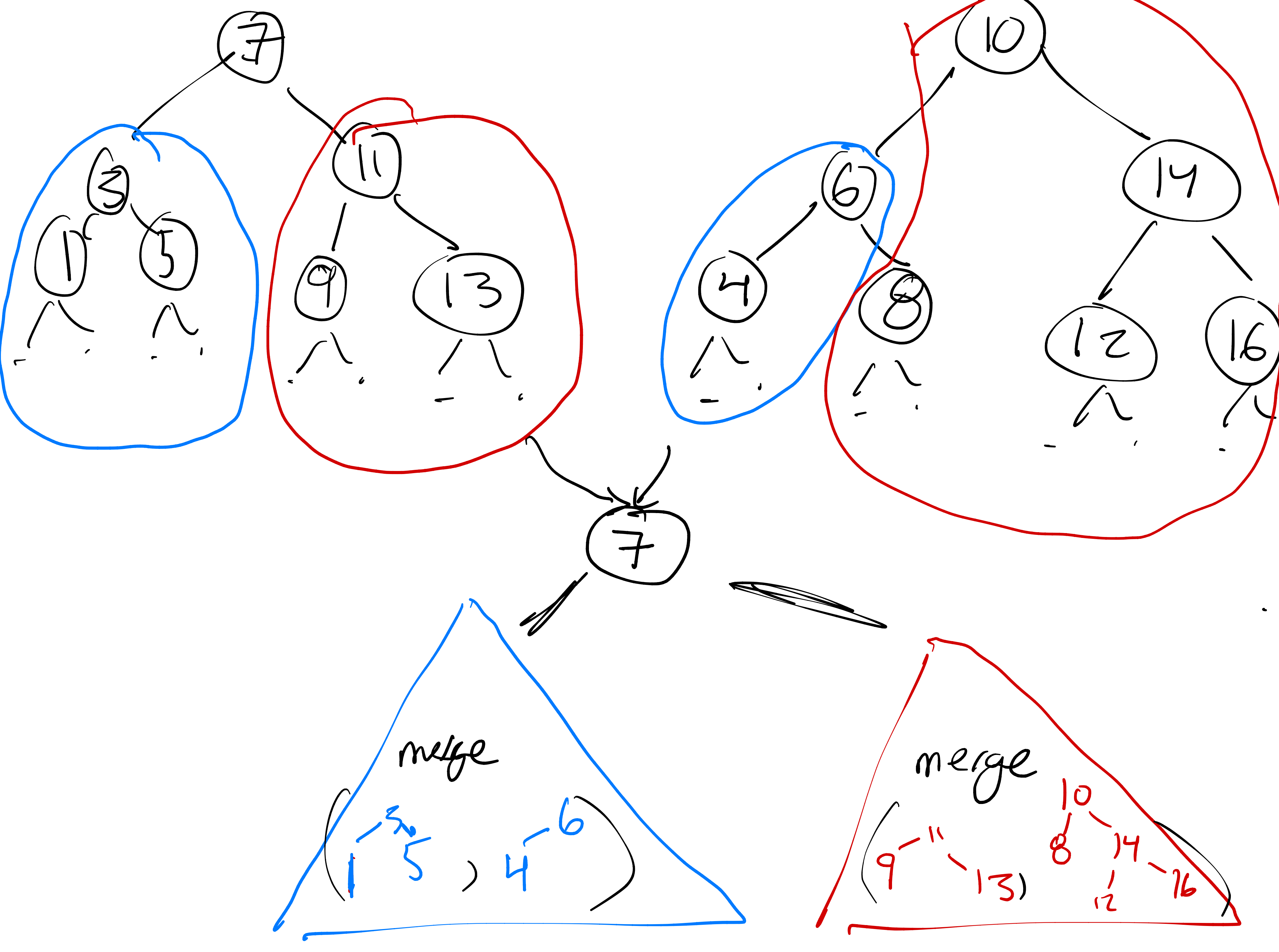
SplitAt: value-based split

takeAndDrop: size-based split

```
(* If t is sorted, then split(t,b) = (l, r)
                          where l ≤ b  r > b *)
fun splitAt(t: tree, b: int): tree * tree =
    case t of
      Empty => (Empty, Empty)
    | Node(l, x, r) => case b < x of
        true => let val (ll, lr) = splitAt(l, b)
                in
                   (ll ≤b , Node(lr, x, r) ≥b )
                end
      | false => let val (rl, rr) = splitAt(r, b)
                 in
                 end (Node(l, x, rl) ≤b , rr ≥b )
```

(* Spec: merge(t_1, t_2) is a tree with all elts of t_1 and t_2 and if t_1, t_2 are sorted and only then merge(t_1, t_2) is sorted *)

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



7
3
1
5
11
9
13
10
6
4
8
14
12
16
7
merge
3
1
5
4
6
merge
10
11
9
13
8
14
12
16

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

split

divide

Combine

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

size

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

$$= O(n \log n)$$

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

split / divide

2 calls

Combine

$$S_{ms}(n) = S_{merging} + 1 \cdot S_{ms}\left(\frac{n}{2}\right)$$

size

balanced

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

$S_{merge}(d_1, d_2)$ is $O(\underline{d_1 \cdot d_2})$

$$S_{merge}(\underline{d_1}, \underline{d_2}) = S_{split}(d_2) + \max(S_{merge}(d_1 - 1, d_2), S_{merge}(d_1 - 1, d_2))$$

d_1 – depth of t_1

d_2 – depth of t_2

$$= \boxed{d_2} + S_{merge}(d_1 - 1, d_2)$$

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

depth of these < depth t

$$S_{split}(d) = k + S_{split}(d-1)$$

d – depth of t

is $O(d)$

```
fun ms (t: tree) : tree =
  case t of
    Empty => Empty
  | Node(l, x, r) =>
rebalance(merge (①merge (ms(l), ms(r)),
          Node(Empty, x, Empty)))
```

split divide

2 calls

combine

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

$$\left.\begin{aligned} &\le (\log n)^2 \\ &+ (\log n)^2 \\ &+ (\log n)^2 \\ &+ \ldots \\ &+ (\log n)^2 \end{aligned}\right\} \log n \text{ times}$$

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

depth of ms l

depth of ms r

$$S_{ms}(n) = S_{merge}(\log_2 n, \log_2 n) + S_{ms}\left(\frac{n}{2}\right)$$

n: size

balanced

depth of merge①

$$+ S_{merge}(2\log_2 n, 1)$$

$$+ S_{reb}(\ldots)$$

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

mergesort

	Lists	Trees
Work	$O(n \log n)$	$O(n \log n)$ "work-efficient"
Span	$O(n)$	$O((\log n)^3)$
	linear	sublinear

time on P procs is $\approx \boxed{\max\left(\frac{W}{P}, S\right)}$

100 things

2 procs

	1	2	3	4		50
P_1	✓	✓	✓	✓	...	✓
P_2	✓	✓	✓	✓		✓

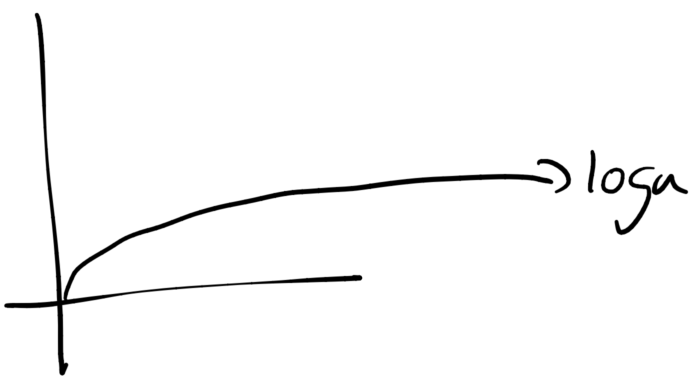
number of procs usefully use $\approx \frac{W}{S}$

$n = 1$ billion

lists

$$\frac{O(n \log n)}{O(n)} \quad \boxed{\frac{w}{S}}$$

$$P \approx \log n \quad 30$$



trees

$$\frac{O(n \cancel{\log n})}{O((\log n)^{\cancel{3}2})} \quad \begin{matrix} 30 \text{ billion} \\ 27{,}000 \end{matrix}$$

$$P \approx \frac{n}{(\log n)^2} \quad 1{,}000{,}000$$