Lecture 8: Sorting

→ Divide + combine

→ Mergesort

→ insertion sort

$O(n^2)$ work → $O(n \cdot \log_2 n)$ work

$n \times n$

$n \times \log_2 n$

atoms in the universe 10^{80}

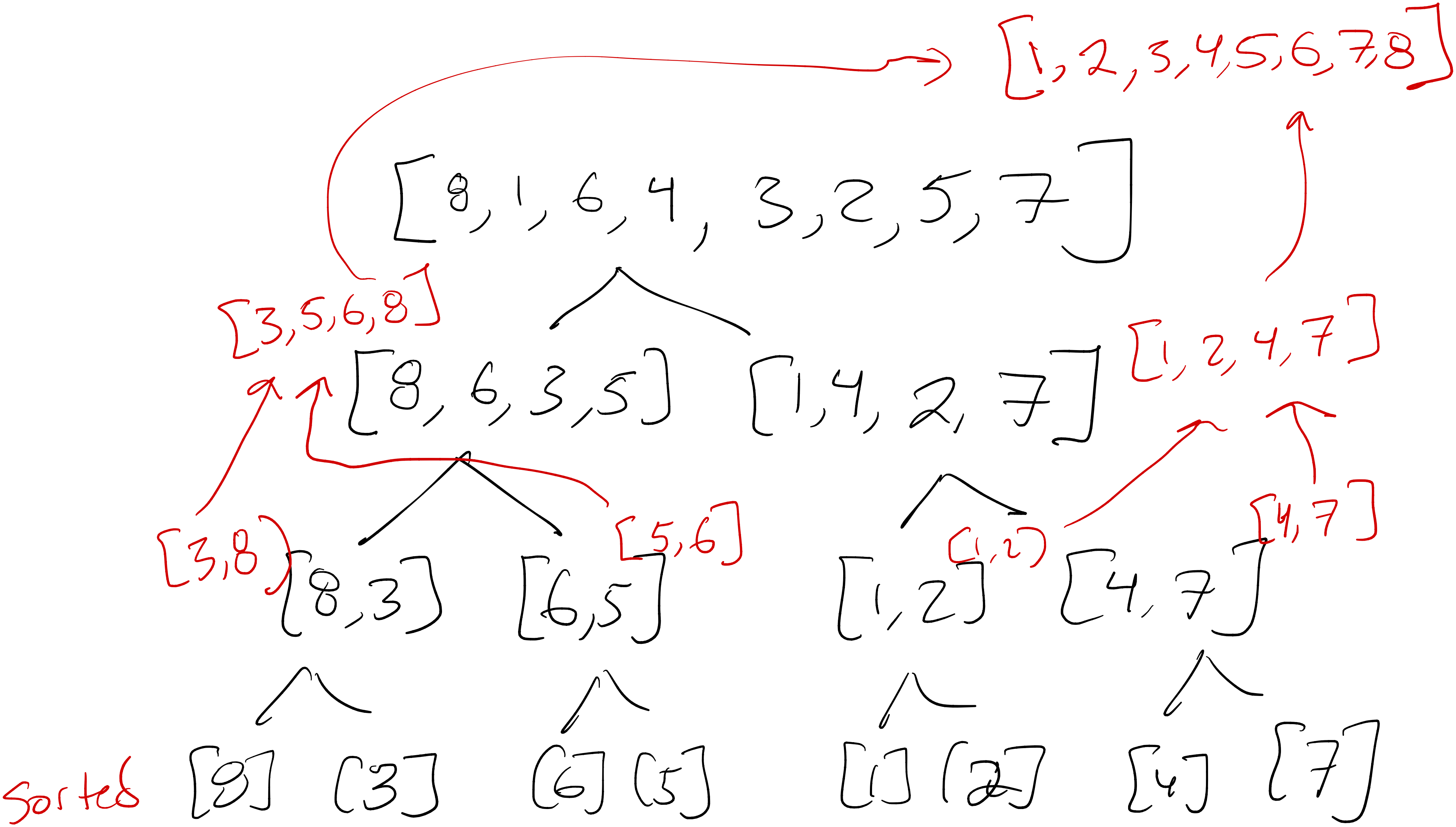
$$\log_2 10^{80} \approx 265$$

Divide + combine

1. divide original problem into two (or more) subproblems, each of roughly half the size — helper

2. solve the subprobs. recursively

3. combine results into a solution to the original problem — helper


[1, 2, 3, 4, 5, 6, 7, 8]
[8, 1, 6, 4, 3, 2, 5, 7]
[3, 5, 6, 8]
[1, 2, 4, 7]
[8, 6, 3, 5]
[1, 4, 2, 7]
[3, 8]
[5, 6]
[1, 2]
[4, 7]
[8, 3]
[6, 5]
[1, 2]
[4, 7]
Sorted
[8]
[3]
[6]
[5]
[1]
[2]
[4]
[7]

(* Spec: output a sorted permutation
of l *)

```
fun mergesort(l: int list): int list =
    let val (p1, p2) = split l          (1) divide
    in
        merge                           (3) combine
          (mergesort p1,
           mergesort p2)                (2) recur
    end
```

```
(* Spec  split(l) = (p1, p2)  where
         p1 @ p2  is a perm of l
   and   length(p1) = length(p2) [+1] *)
                                maybe

fun split(l : int list) : int list * int list =
    case l of
        [] => ([], [])
      | [x] => ([x], [])
      | x::y::xs => let val (p1, p2) = split xs
                    in (x::p1, y::p2)
                    end
```

(* Spec: If l_1 and l_2 are sorted, then
merge(l_1, l_2) is a sorted perm.
of l_1 @ l_2 *)

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

x :: y :: merge(xs, ys)

l_2

l_1

simultaneous induction / recursion

Proof by sim. induction on l, l_2

case for ([], [])

case for ([], y::ys)

case for (x::xs, [])

case for (x::xs, y::ys) IH for (xs, y::ys)
for (x::xs, ys)

1. $x \leq y$ is true
2. $y < x$ is false

$\text{merge}(x::xs,\ y::ys) \cong x::\text{merge}(xs,\ y::ys)$

To show: $x::\text{merge}(xs,\ y::ys)$ is a sorted perm of $(x::xs) @ (y::ys)$

IH: $\text{merge}(xs,\ y::ys)$ is a sorted perm of $xs @ (y::ys)$
[because xs is sorted ✓, $y::ys$ is sorted ✓]

So
- $x::\text{merge}(xs, y::ys)$ is a perm of $x::(xs @ y::ys) \cong (x::xs) @ y::ys$ ✓
- $x::\text{merge}(xs, y::ys)$ is sorted
  - $x \le xs$ b/c l_1 sorted
  - $x \le y$ test
  - $x \le ys$ l_2 sorted

```
(* Spec: output a sorted permutation
         of l *)

fun mergesort(l : int list) : int list =
  case l of
    [] => []
  | [x] => [x]
  | _ => let val (p1, p2) = split l      (1) divide
  in
    merge                                (3) combine
      (mergesort p1,
       mergesort p2)                     (2) recur
  end
```

sublist

Proof induction on l

base cases $[\,]$ ✓

$[x]$ ✓

---

① by spec for split,

otherwise $p_1 @ p_2$ is a perm of l

and $\text{length}(p_1) = \text{length}(p_2)\ [+1]$

$l = [\,]$	$p_1 = [\,]$	$p_2\ [\,]$
$l = [2]$	$p_1 = [2]$	$p_2 = [\,]$
$l = [a, b]$	$p_1 = [a]$	$p_2 = [b]$

If length $l \geq 2$

then length p_1 < length l
length p_2

(2) IH: mergesort(p_1) is sorted perm p_1
mergesort(p_2) is sorted perm p_2

(3) spec for merge: merge(ms p_1, ms p_2) is a sorted perm of ms p_1 @ ms p_2

merge(ms p_1, ms p_2) is a perm of l is$_{\text{perm}}$ p_1 @ p_2

$$ms([\,])$$

$$\mapsto merge(ms[\,],\ ms[\,])$$

$$\mapsto merge(merge(ms[\,],\ ms[\,]),\ ms[\,])$$

$$\vdots$$

Work for mergesort

① divide

$$W_{split}(n) = \cancel{k}^{\,I} + W_{split}(n-2)$$

length of l

closed form $\approx \frac{n}{2}$ $\boxed{O(n)}$

② $W_{merge}(n, m) \le$ bigger of $\begin{cases} k + W_{merge}(n-1, m) \\ \text{or} \\ k + W_{merge}(n, m-1) \end{cases}$

n: length l_1 m: length l_2

"change of base"

$$W(0) = \cancel{k_0}\ 1$$

$$W(s) = \cancel{k_1}\ 1 + W(s-1)$$

merge ↑ $\qquad\qquad$ merge

length l_1 + length l_2

$$O(s)$$

$$
\begin{aligned}
W_{ms}(\underset{\substack{\uparrow \\ \text{length of } \ell}}{n}) &= W_{split}(n) \\
&\quad + W_{ms}\Big(\underbrace{\tfrac{n}{2}}_{\text{length of } p_1}\Big) + W_{ms}\Big(\underbrace{\tfrac{n}{2}}_{\text{length of } p_2}\Big) \\
&\quad + W_{merge}\Big(\underbrace{\tfrac{n}{2}}_{\text{length of ms } p_1} + \underbrace{\tfrac{n}{2}}_{\text{length of ms } p_2}\Big) \\
&= \underbrace{W_{split}(n)}_{O(n)} + \underbrace{W_{merge}(n)}_{O(n)} + 2\,W_{ms}\Big(\frac{n}{2}\Big) \\
&\approx n + 2\,W_{ms}\Big(\frac{n}{2}\Big)
\end{aligned}
$$

$$W_{ms}(n) = n + 2\,W_{ms}\left(\tfrac{n}{2}\right)$$
$$= n + 2\left(\tfrac{n}{2} + 2\,W_{ms}\left(\tfrac{n}{4}\right)\right)$$
$$= n + 2\left(\tfrac{n}{2} + 2\left(\tfrac{n}{4} + 2\,W_{ms}\left(\tfrac{n}{8}\right) + \dots\right)\right)$$

tree method

$$n \longrightarrow n$$
$$+ \quad +$$
$$\tfrac{n}{2} \quad \tfrac{n}{2} \longrightarrow \tfrac{2n}{2} = n$$
$$+ \; + \quad + \; +$$
$$\tfrac{n}{4} \quad \tfrac{n}{4} \quad \tfrac{n}{4} \quad \tfrac{n}{4} \longrightarrow \tfrac{4n}{4} = n$$
$$++ \; ++ \; ++ \; ++$$
$$\tfrac{n}{8} \; \tfrac{n}{8} \; \tfrac{n}{8} \; \tfrac{n}{8} \; \tfrac{n}{8} \; \tfrac{n}{8} \; \tfrac{n}{8} \; \tfrac{n}{8} \longrightarrow \tfrac{8n}{8} = n$$
$$\vdots$$

times divide n by 2 $= \log_2 n$

$W_{ms}(n)$ is $O(\underbrace{n}_{\text{work at each level}} \cdot \underbrace{\log_2 n}_{\#\text{ levels}})$