Lecture 10:

Parallel
Running
Time
Analysis

$n = 1$ billion $\quad 1,000,000,000$

$n^2 = 1$ quadrillion $\quad 1 \times 10^{18}$

$\log_2 n \approx 30$

$2^{30} \approx 1$ billion

$n \log_2 n = 30$ billion

n elemts [8, 1, 6, 4, 2, 3, 5, 7]

$\frac{n}{2}$ elements in 2 subproblems [8, 6, 2, 5] [1, 4, 3, 7]

$\frac{n}{4}$ elts in 4 subproblems [8, 2] [6, 5] [1, 3] [4, 7]

Idea:

1. divide into 2 subproblems
2. solve them recursively
3. merge results

(* spec: ms(l) computes a sorted perm. of l *)

```
fun ms(l) =
    case l of
        [] => []
      | [x] => [x]
      | _ => let val (p1, p2) = split(l)
             in
                 merge(ms p1, ms p2)
             end
```

(* Spec: split(l) = (p1, p2) where p1 @ p2 ↳ built-in append
is a perm of l, and
length(p1) = length(p2) [±1] *)

```
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
```

```
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
```

$$W_{split}(\underbrace{n}_{\text{size of } l\text{? length}}) = \overbrace{k}^{\text{steps that split takes}} + W_{split}(\underbrace{n-2}_{\text{size of recursive input}})$$

$$\approx 1 + W_{split}(n-2)$$

$$\approx \frac{n}{2}$$

$$O(n)$$

$$W(n) = 1 + W(n-2)$$

$$= 1 + (1 + W(n-4))$$

$$= 1 + (1 + (1 + W(n-6)))$$

$$= \underbrace{1 + 1 + 1 + 1 + \cdots + 1}_{\frac{n}{2} \text{ times}}$$

Span: # steps it takes to run the program with "enough" processors

parallel

Work: # steps it takes to run a program on 1 processor

sequential

```
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
```

steps that split takes

$$S_{split}(n) = k + S_{split}(n-2)$$

span

size of l? length

size of recursive input

$$\cancel{=}\ k\frac{n}{2}$$

is $O(n)$

```
(* Purpose: Given sorted lists l1 and l2
            make a sorted perm. of l1@l2 *)

fun merge(l1: int list, l2: int list) : int list =
    case (l1, l2) of

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

E.g.

merge([1,4,8],
      [2,3,6]) = [1,2,3,4,6,8]

P

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

$$W_{merge}(s) \doteq k + W_{merge}(s-1)$$

size of merge's input $\approx 1 + W_{merge}(s-1)$ $\boxed{O(s)}$

i.e. length(l_1) + length(l_2)

$$S_{merge}(s) = k + S_{merge}(s-1) \quad \boxed{O(s)}$$

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

$$\cancel{W_{merge}\left(W_{ms}\left(\tfrac{n}{2}\right), W_{ms}\left(\tfrac{n}{2}\right)\right)}$$

$$W_{ms}(n) = k$$

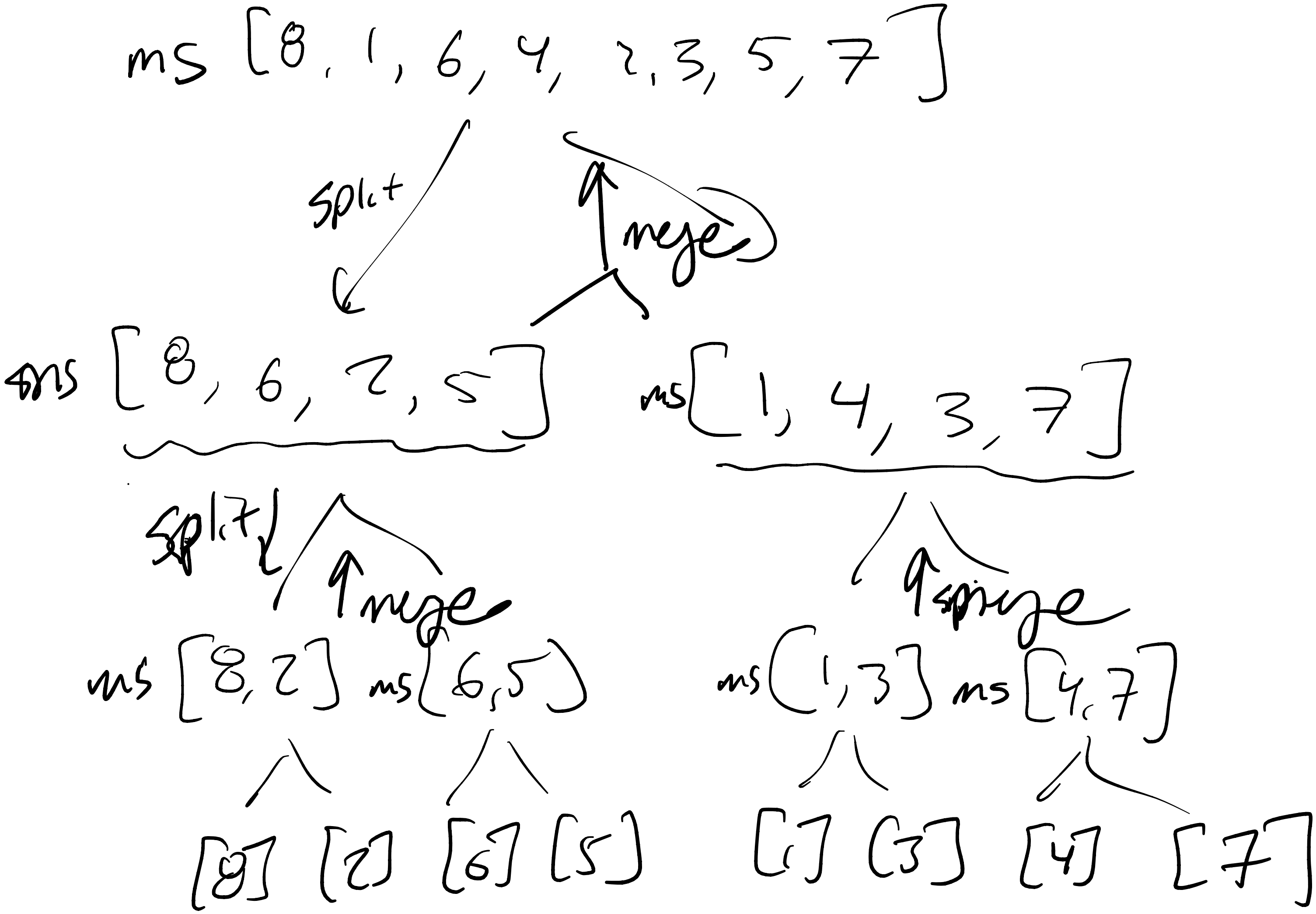
length of l

$$+ W_{split}(\underline{\ n\ })$$

$$+ W_{ms}(\underline{\ \tfrac{n}{2}\ })$$ → size of the output of split

$$+ W_{ms}(\underline{\ \tfrac{n}{2}\ })$$

$$+ W_{merge}(\underline{\ n\ })$$


ms [8, 1, 6, 4, 2, 3, 5, 7]
split
merge
ms [8, 6, 2, 5]
ms [1, 4, 3, 7]
split
merge
merge
ms [8, 2]
ms [6, 5]
ms [1, 3]
ms [4, 7]
[8]
[2]
[6]
[5]
[1]
[3]
[4]
[7]

$$W_{ms}(n) = k$$

length of ℓ

$$+ \; W_{split}(\underline{\quad n \quad}) \longrightarrow O(n)$$

$$+ \; W_{ms}(\underline{\quad \tfrac{n}{2} \quad})$$

$$+ \; W_{ms}(\underline{\quad \tfrac{n}{2} \quad})$$

$$+ \; W_{merge}(\underline{\quad n \quad}) \longrightarrow O(n)$$

$$\leq kn + 2W_{ms}\left(\tfrac{n}{2}\right)$$

$$\sim \underbrace{n}_{\text{time for splitting + merging}} + \underbrace{2W_{ms}\left(\tfrac{n}{2}\right)}_{\text{2 recursive calls}}$$

Tree method

$O(n \log_2 n)$

levels?

$\log_2 n$

$$n \longrightarrow n$$

$$\frac{n}{2} + \frac{n}{2} \longrightarrow \frac{2n}{2} = n$$

$$\frac{n}{4} + \frac{n}{4} + \frac{n}{4} + \frac{n}{4} \longrightarrow \frac{4n}{4} = n$$

$$\frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} + \frac{n}{8} \longrightarrow \frac{8n}{8} = n$$

$$\vdots \longrightarrow \frac{16n}{16} = n$$

$$\vdots$$

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

$$S_{ms}(n) = k$$

length of l

$$+ S_{split}(\underline{\ \ n\ \ }) \longrightarrow O(n)$$

$$+ \text{maximum}\left(S_{ms}\left(\underline{\tfrac{n}{2}}\right), S_{ms}\left(\underline{\tfrac{n}{2}}\right)\right) \longrightarrow 1\, S_{ms}\left(\tfrac{n}{2}\right)$$

$$+ S_{merge}(\underline{\ \ n\ \ }) \longrightarrow O(n)$$

$$= n + S_{ms}\left(\tfrac{n}{2}\right)$$

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

$$= n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \frac{n}{16} + \cdots$$

$$= n\left(\underline{1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \cdots}\right)$$

$$\le \underline{2n}$$

i.e. $\boxed{\underline{O(n)}}$

1 billion

work $n \log_2 n \approx 30$ billion

span $n \approx 1$ billion

time on p procs. $\max\left(\frac{\text{work}}{\#\text{proc}}, \text{span}\right)$

$\max\left(\frac{30 \text{ bill.}}{p}, 1 \text{ billion}\right)$

means $\longrightarrow$ can only use 30 procs!

Sorting

Trees

instead of lists