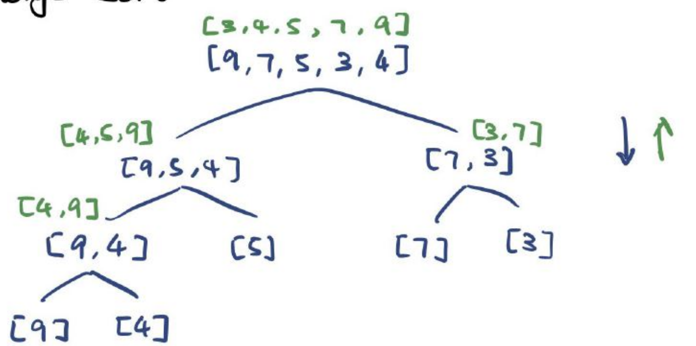
Lec 7 | Divide and Conquer

Merge Sort




```
(* msort int list -> int list
   REQ true
   ENS msort L ↪ L' ; L' is sorted of L.
*)
fun msort ([]: int list) = []
  | msort ([x]) = [x]
  | msort ( L ) =
    let
        val (A: int list , B: int list) = split L
    in
        merge (msort A, msort B)
    end
```

```
(* split : int list -> int list * int list
   REQ true
   ENS split L ↪ (A,B),
       |length A - length B| ≤ 1
       A@B is perm of L
*)
fun split ( [] : int list ) = ([],[])
  | split ([x]) = ([x],[])
  | split (x::y::rest) =
    let
      (A,B) = split rest
    in
      (x::A , y::B)
    end
```

$W_{split}(n)$ when n is len of input list

Match :

- $W_{split}(0) = c_0$
- $W_{split}(1) = c_1$
- $W_{split}(n) = W_{split}(n-2) + c_2$

  $\dots \in O(n)$

```
(* merge : int list * int list -> int list
   REQ input lists sorted
   ENS merge (A, B) ↪ L s.t. L is sorted perm of A@B
*)
fun merge ([], B) = B
  | merge (A, []) = A
  | merge (a::as, b::bs) =
    (case Int.compare (a, b) of
        LESS => a:: merge (as, b::bs)
        EQUAL => a::b:: merge (as, bs)
        GREATER => b:: merge (a::as, bs)
    )
  | merge (A as (a::as), B as (b::bs)) =
    (case Int.compare (a, b) of
        LESS => a:: merge (as, B)
        EQUAL => a::b:: merge (as, bs)
        GREATER => b:: merge (A, bs)
    )
```

with good compiler, num of branches doesn't change cost

slightly more efficient

$W_{merge}(n, m)$ where n, m are len of A, B.

Match:

$W_{merge}(0, m) = c_0$

$W_{merge}(n, 0) = c_1$

$$W_{merge}(n,m) = \begin{cases} c_2 + W_{merge}(n-1, m) & \text{if LESS} \\ c_3 + W_{merge}(n-1, m-1) & \text{if EQUAL} \\ c_4 + W_{merge}(n, m-1) & \text{if GREATER} \end{cases}$$

Hmm hard to unroll...

Try: $s = n+m$

$W_{merge}(s)$

Match

$W_{merge}(0) = k_0$

worse case

$W_{merge}(n) \leq \underline{k_1 + W_{merge}(s-1)}$

$\in O(s)$

Merge sort cost analysis

$W_{msort}(n)$ n is input len

Match:

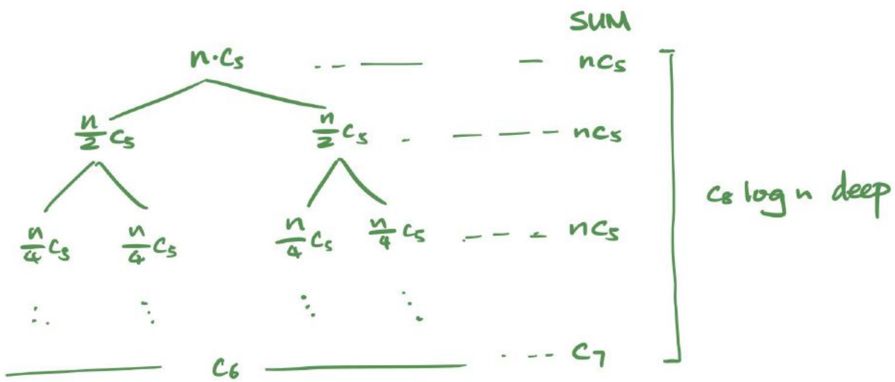
$$
\begin{aligned}
W_{msort}(0) &= c_0 \\
W_{msort}(1) &= c_1 \\
W_{msort}(n) &= c_2 + W_{split}(n) + W_{merge}(n) + W_{msort}(\lfloor \tfrac{n}{2} \rfloor) + W_{msort}(\lceil \tfrac{n}{2} \rceil) \\
&\leq c_2 + c_3 n + c_4 n + 2W_{msort}(\tfrac{n}{2}) \\
&\leq c_5 n + 2W_{msort}(\tfrac{n}{2})
\end{aligned}
$$

(Lemma: splitting doesn't change total length)

linear work to divide

conquer each part

Enters tree method!



TOTAL : $c_8 \log n \cdot nc_5 + c_7 n$

$\in O(n \log n)$

Merge sort span

$S_{msort}(0)$ [omit]

$S_{msort}(1)$

$$S_{msort}(n) = c_2 + S_{split}(n) + S_{merge}(n) + \max\left(S_{msort}(\lfloor \tfrac{n}{2} \rfloor),\ W_{msort}(\lceil \tfrac{n}{2} \rceil)\right)$$

$$\cdots \le nc_4 + S_{msort}(\tfrac{n}{2})$$

(tree method) $= nc_4 + \frac{n}{2}c_4 + \frac{n}{4}c_4 + \cdots + c_1$

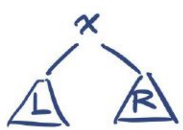
$= 2c_4 n + c_1$

$\in O(n)$

If we do it on tree, we can even get it to $O((\log n)^3)$?

Sorting Tree — definition

Empty ← sorted



← sorted if $\forall_{l \in L},\ l \le x$
$\forall_{r \in R},\ r \ge x$

Insert :=

```
fun insert (x:int, Empty tree) = Node (Empty, x, Empty)
  | insert (x, Node(L,y,R)) =
    ( case Int.compare (x, y) of
        GREATER => Node(L, y, insert (R))
      | _ =>      Node(insert (L), y, R )
    )
```