TLect 8

Divide and Combine

insertion sort → mergesort

$O(n^2)$

$O(n \log_2 n)$

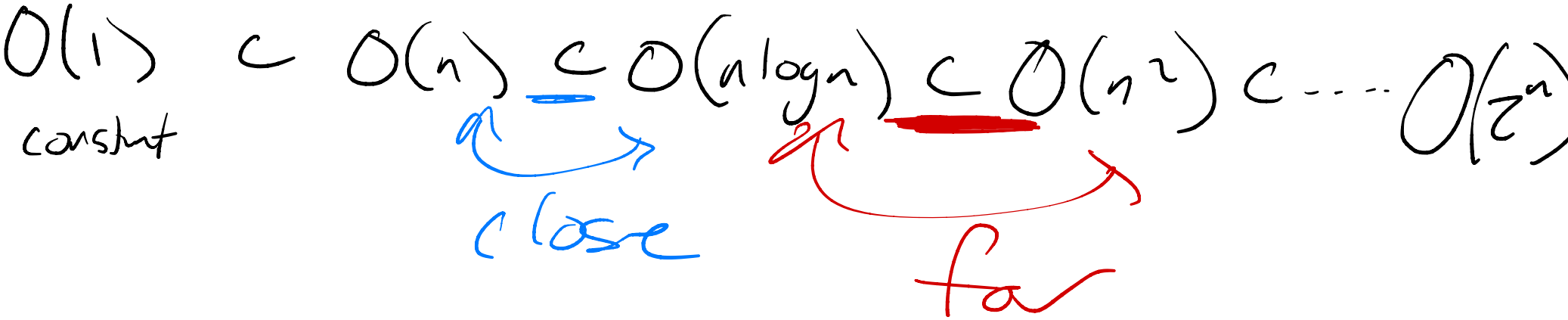
$\log_2(n)$ is the number such that

$$2^{\log_2(n)} = n$$

$\log_2(1 \text{ billion}) = 30$

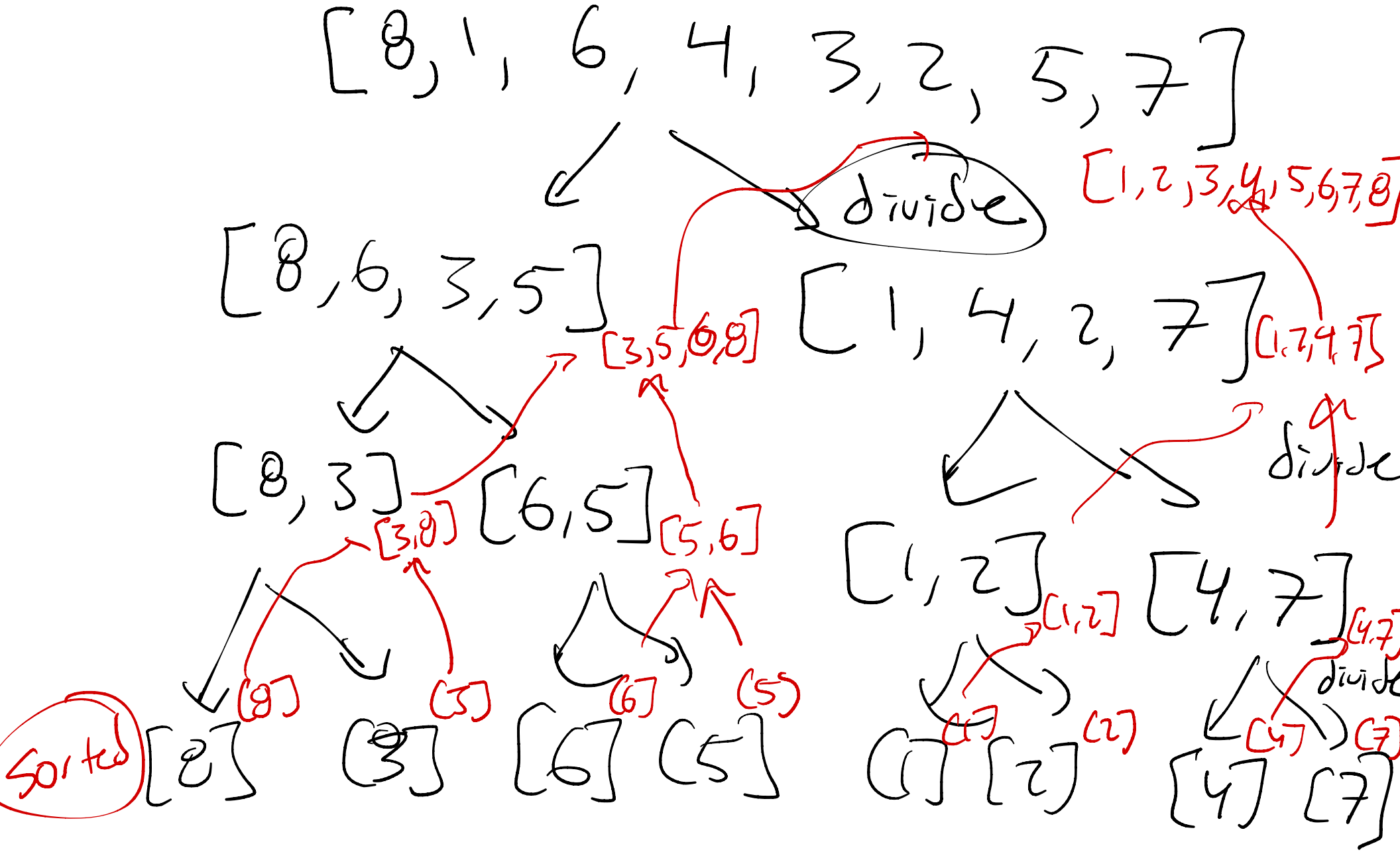
$\log_2(10^{80}) \approx 265$

↳ # of atoms in the universe


O(1) ⊂ O(n) ⊂ O(nlogn) ⊂ O(n²) ⊂ ... O(2ⁿ)
constant
close
far

Divide and Conquer Combine

1. Divide problem into 2 or more subproblems (of equal size) ✓ — helper
2. Recur to solve those subproblems
3. Combine results of that into the overall result — helper


[8, 1, 6, 4, 3, 2, 5, 7]
divide
[8, 6, 3, 5]
[1, 4, 2, 7]
[1,2,3,4,5,6,7,8]
[3,5,6,8]
[1,2,4,7]
[8, 3]
[6, 5]
[1, 2]
[4, 7]
divide
[3,8]
[5,6]
[1,2]
[4,7]
divide
[8]
[3]
[6]
[5]
[1]
[2]
[4]
[7]
sorted
[2]
[3]
[6]
[5]
[1]
[2]
[4]
[7]

(* Spec: mergesort(l) is a sorted perm of l *)

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

divide

combine

sorted

sorted

(* Spec split(l) = (l1, l2) where length(l1) = length(l2)[+1].
append(l1, l2) is a perm. of l *)

```
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
```

Case n of
0 => (0,0)
1 => (1,0)
_ => let val (x,y) = halves(n-2) in (x+1, y+1) end

append(p1, p2) is a perm of xs
length p1 = length p2 [+1]

append(x::p1, y::p2) is a perm of l

(* Specification: For any l_1 and l_2 that are sorted, merge(l_1, l_2) is a sorted permutation of append(l_1, l_2) *)

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

$x \leq y$

$x \leq xs$

$x \leq ys$

Work for split

```
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
```

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

n: length of l

Size of input to split

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

$$W(0) = 1$$
$$W(n) = 1 + W(n-2)$$

$O(n)$

Work for merge

length(l_1) + length(l_2)

$W_{merge}(s) =$

$1 + W_{merge}(s-1)$

k_0

$O(s)$

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

Work for mergesort

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

divide

combine

sorted

sorted

length of l

$W_{ms}(n) =$

① $W_{split}(n)$

② $+ W_{ms}(n/2)$
$+ W_{ms}(n/2)$

size of inputs to the recursive call

③ $+ W_{merge}(n)$

↳ size of inputs to merge

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

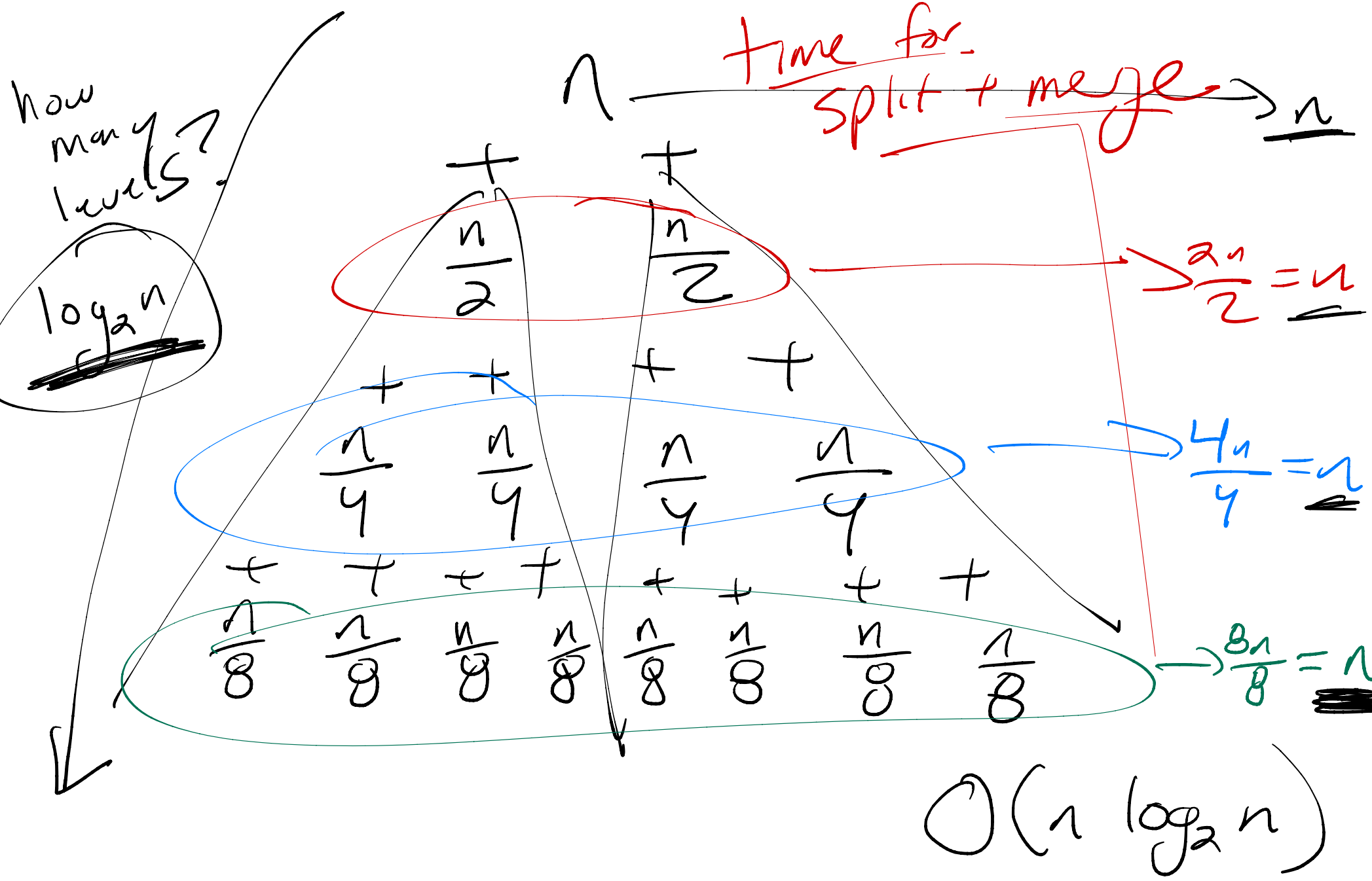
$$\text{is } k_0 n + 2W_{ms}\left(\frac{n}{2}\right) + k_1 n$$

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

$$\text{is } O(n(\log_2 n))$$

Tree method

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



$$O(n \log_2 n)$$