Lecture 7: Sorting

[4, 1, 3, 2] $\xrightarrow{\text{sort}}$ [1, 2, 3, 4]

$\leq$

Specification

A sorted list is ... (in increasing order)

① $[x_1, x_2, x_3, \ldots x_n]$ is sorted

iff for all i s.t. $1 \le i \le n$

for all j s.t. $i \le j \le n$

$x_i \le x_j$

(2) A list is sorted iff

(a) it is []

(b) it is x :: xs

where xs is sorted

x ≤ everything in xs

[1, 2, 3, 4] is sorted

① [2, 3, 4] is sorted → [3, 4] sorted
⇒ 2 ≤ [3, 4] ✓

② 1 ≤ everything in [2, 3, 4]

3 A list is sorted iff

(a) it's [], or

(b) it's x::xs where

(1) xs sorted

(2) x ≤ the first element of xs (if it has one)

(4) A list is sorted, iff

(a) it is []

(b) it's [x]

(c) It's x :: y :: ys, where y :: ys is sorted and $x \leq y$

```
(* Purpose/specification:
     output a sorted permutation of the input list *)

fun isort(l: int list) : int list =
  case l of
    [] => []
  | x::xs => insert(x, isort(xs))
```

"insertion sort"

a sorted perm. of xs

Permutations

l_1 is a permutation of l_2

means

they have the same elts.

occuring the same nombe of times,

but possibly in a different

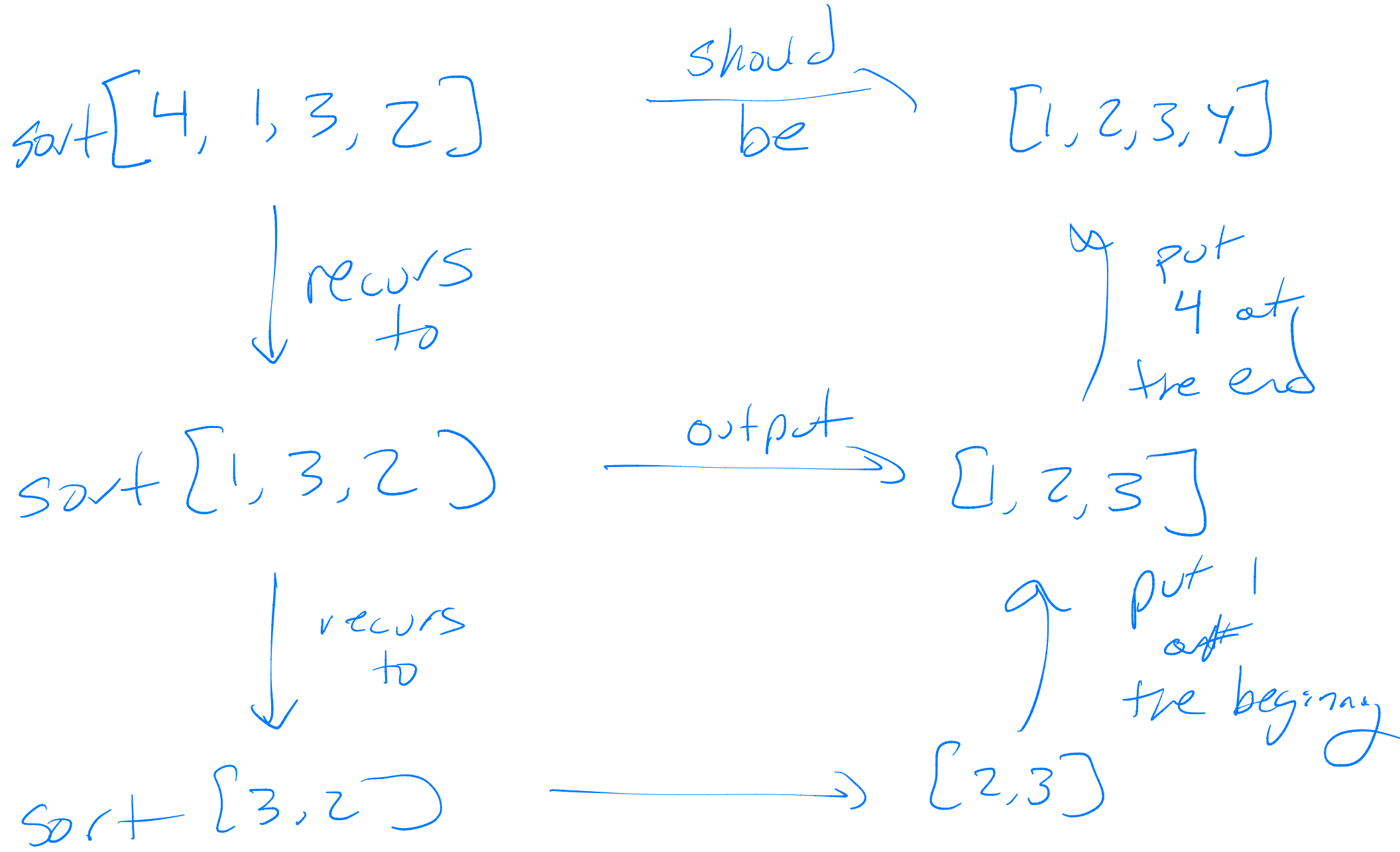
order

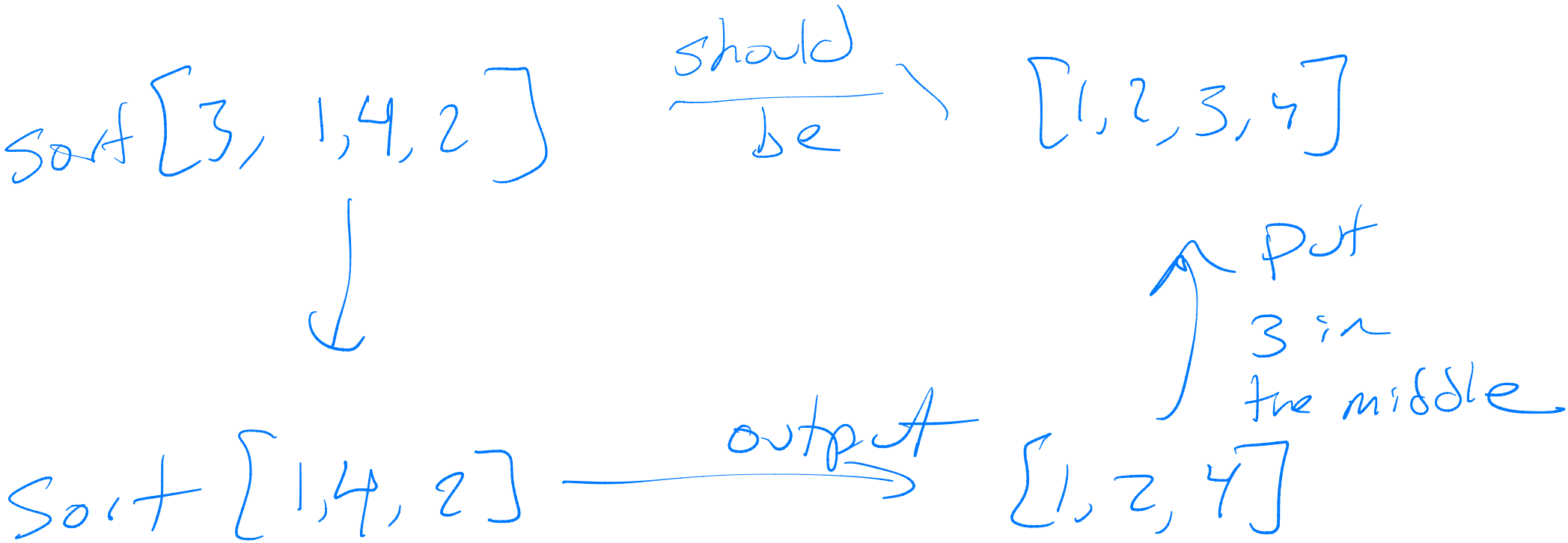
[4, 1, 3, 2] is a perm of [1, 2, 3, 4]

[4, 1, 3, 2, 5] is not

[4, 1, 3, 2, 2] is not

[4, 1, 3] is not


sort[4, 1, 3, 2]
should be
[1, 2, 3, 4]
recurs to
put 4 at the end
sort [1, 3, 2]
output
[1, 2, 3]
recurs to
put 1 at the beginning
sort [3, 2]
[2, 3]


sort [3, 1, 4, 2]
should be
[1, 2, 3, 4]
put 3 in the middle
sort [1, 4, 2]
output
[1, 2, 4]

```
(* Spec: insert(x, l) outputs
        a sorted permutation of
          x :: l  if l is sorted *)

fun insert(x: int, l: int list) : int list =
    case l of
        [] => [x]
      | y::ys => case x ≤ y of
                   true => x :: y::ys
                 | false =>
                     y :: insert(x, ys)
```

l

Thm: for all int list values l and int x
if l is sorted, then
insert(x, l) is a sorted perm of $x :: l$

Case for $[]$:

insert$(x, [])$ $=$ $[x]$ sorted ✓
perm of $x :: []$ ✓

Case for $y :: ys$:

Case for $x \leq y \cong$ true:

insert$(x, y :: ys)$ $\cong$ $x :: y :: ys$

- sorted $x \leq y$ ✓ $y :: ys$ is sorted by assumption
- perm of $x :: y :: ys$ ✓

Case $x \leq y$ is false

$\therefore y < x \rightarrow y \leq x$

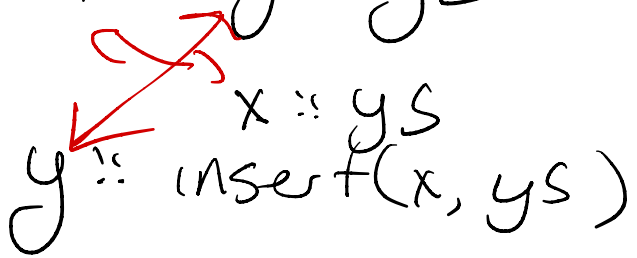
$insert(x, y :: ys) \cong y :: insert(x, ys)$

By IH, $insert(x, ys)$ is a sorted perm of $x :: ys$

[ys is sorted]

① sorted

② permutation of $x :: y :: ys$



Thm for all l:int list,

isort(l) is a sorted perm. of l.

Case for []:

isort([]) ≅ [] is a sorted perm of []

Case for x::xs:

To show: isort(x::xs) ≅ insert(x, isort xs) is a sorted perm of x::xs

by insert spec using isort xs is a sorted list (IH)

insert(x, isort xs) is a sorted perm of x :: isort(xs)

by IH: isort(xs) is a perm of xs

↳ is a perm of x :: xs as well

$$W_{isort}(n) = 1 + W_{isort}(n-1) + W_{insert}(n-1)$$

length of list

length of input to insert

$$W_{insert}(n) = \begin{cases} \cancel{k_0}\ 1 & \text{if it stops} \\ \cancel{k_1}\ 1 + W_{insert}(n-1) & \text{if it continues} \end{cases}$$

length of list input

"worst case analysis"

$$\leq \cancel{k_1}\ 1 + W_{insert}(n-1)$$

best case isort [1, 2, 3, 4]

is O(n)

worst case isort [4, 3, 2, 1]

$W_{insert}(n)$ is $O(n)$

$$W_{isort}(n) = 1 + W_{isort}(n-1) + W_{insert}(n-1)$$

$$\leq 1 + W_{isort}(n-1) + n$$

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

$\therefore$ $W_{isort}(n)$ is $O(n^2)$

$$n + (n-1) + (n-2) + \cdots$$

n Sommorial

$$\frac{n(n+1)}{2} \quad \text{is} \quad \frac{n^2}{2} + \frac{n}{2}$$