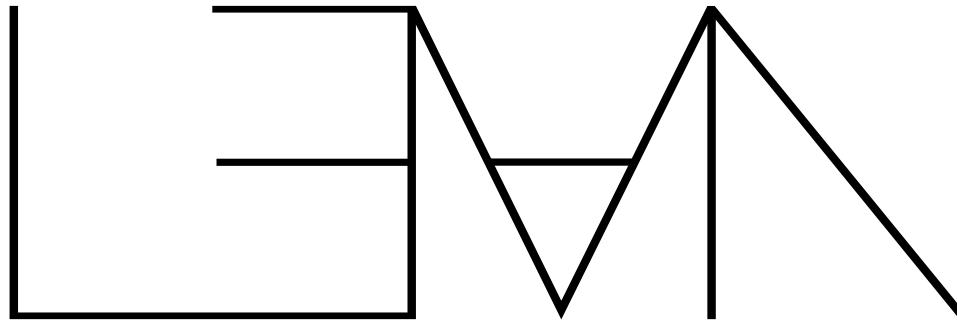


# Programs and Proofs



Anthony Wang (xy)

2026-01-01

# Contents

# Insertion sort

## Contents

```
variable [LE α] [DecidableLE α]
[Std.IsLinearOrder α] [BEq α] [LawfulBEq α]
(xs : List α)
```

```
@[grind]
def insert (a : α)
| [] => [a]
| x :: xs =>
  if a ≤ x then
    a :: x :: xs
  else
    x :: insert a xs
```

```
@[grind]
def insertionSort : List α → List α
| [] => []
| x :: xs => insert x (insertionSort xs)
```

```
@[grind]
def Sorted : List α → Prop
```

```
| [] | [_] => True
| x :: x' :: xs => x ≤ x' ∧ Sorted (x' :: xs)
```

```
theorem insertCorrect x : (Sorted xs →
Sorted (insert x xs)) ∧ (x :: xs).Perm
(insert x xs) := by
  induction xs with
  | nil => grind
  | cons _ t => cases t <;> grind
```

```
theorem insertionSortCorrect : Sorted
(insertionSort xs) ∧ xs.Perm (insertionSort
xs) := by
  induction xs with
  | nil => grind
  | cons h t => grind [insertCorrect
(insertionSort t) h]
```