

# CS 135 – L14: General Recursion

Luke Lu • 2025-12-17

---

## Counting Up

### Count Up Template

```
(define (count-up-template n limit)
  [cond
    [(>= n limit) ...]
    [else (... (count-up-template (add1 n) limit))]])
```

## Mutual Recursion

```
(define (keep-next lst)
  (cond
    [(empty? lst) empty]
    [else (cons (first lst) (skip-next (rest lst))))])

(define (skip-next lst)
  (cond
    [(empty? lst) empty]
    [else (keep-next (rest lst))]))
```

## Merge Sort

Merge Sort has the time complexity of  $O(n \log n)$

```
; merge: (listof Num) (listof Num) -> (listof Num)
; Requires: lists must be sorted in increasing order
(define (merge lst1 lst2)
  (cond
    [(empty? lst1) lst2]
    [(empty? lst2) lst1]
    [(< (first lst1) (first lst2)) (cons (first lst1) (merge (rest lst1) lst2))]
    [(> (first lst1) (first lst2)) (cons (first lst2) (merge lst1 (rest lst2)))]
    [else (cons (first lst1) (cons (first lst2) (merge (rest lst1) (rest lst2))))])

(define (mergesort lst)
  (cond
    [(or (empty? lst) (empty? (rest lst))) lst]
    [else (merge (mergesort (keep-next lst)) (mergesort (skip-next lst))))])
```