

# Scheme Notes 04

Geoffrey Matthews

Department of Computer Science  
Western Washington University

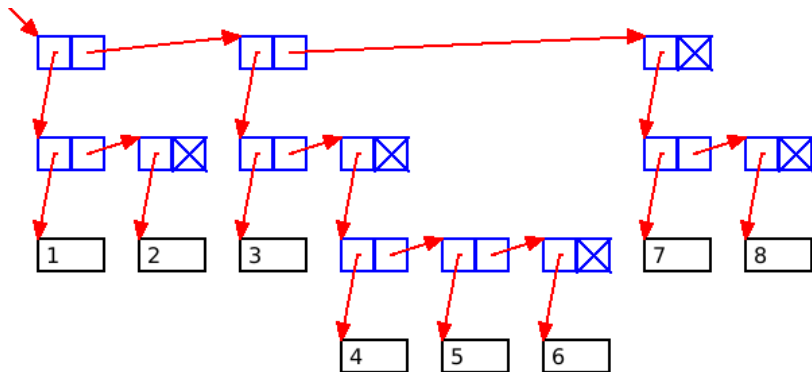
February 2, 2018

# Trees

```
(define tree1 (cons (list 1 2) (list 3 4)))
```

```
(define tree2 (list (list 1 2)
                    (list 3
                          (list 4 5 6))
                    (list 7 8)))
```

- Run boxarrow.rkt for pictures.



# count-leaves

## count-leaves

```
(define (count-leaves tree)
  (cond ((empty? tree) 0)
        ((not (pair? tree)) 1)
        (else (+ (count-leaves (car tree))
                   (count-leaves (cdr tree))))))
```

fringe

## fringe

```
(define (fringe tree)
  (cond ((empty? tree) '())
        ((not (pair? tree)) (list tree))
        (else (append (fringe (car tree))
                        (fringe (cdr tree))))))
```

# sum-fringe

## sum-fringe

```
(define (sum-fringe tree)
  (cond ((empty? tree) 0)
        ((number? tree) tree)
        (else (+ (sum-fringe (car tree))
                   (sum-fringe (cdr tree))))))
```



# map-tree

## map-tree

```
(define (map-tree tree op)
  (cond ((empty? tree) '())
        ((not (pair? tree)) (op tree))
        (else (cons (map-tree (car tree) op)
                      (map-tree (cdr tree) op)))))
```

# scale-tree using map-tree

## scale-tree using map-tree

```
(define (scale-tree tree factor)
  (map-tree tree (lambda (x) (* x factor))))
```

# increment-tree using map-tree

## increment-tree using map-tree

```
(define (increment-tree tree)
  (map-tree tree inc))
```

# Apply

```
(+ 1 2 3 4 5)           => 15  
(+ (list 1 2 3 4 5))   => error  
(apply + (list 1 2 3 4 5)) => 15
```