Sentinels

Dr. Mattox Beckman

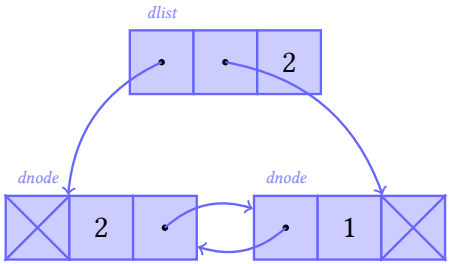
Illinois Institute of Technology
Department of Computer Science

Objectives

- Understand how to form a sentinel.
- Show how to use a sentinel to replace `null`.
- Show how to use a sentinel to remove edge cases.

Our Doubly Linked List So Far

```
1  (struct dlist (front back size) #:mutable)
2  (struct dnode (prev data next) #:mutable)
3  (define (make-dlist) (dlist null null 0))
```



Ordered Insert: Helper Functions

- To insert, we need to find the proper insertion point first.
- Then we can link the node into the doubly linked list.

```
1  (define (is-empty? xx)
2    (null? (dlist-front xx)))
3
4  (define (find-greater dns elt)
5    (cond ((null? dns) null)
6          ((< elt (dnode-data dns)) dns)
7          (else (find-greater (dnode-next dns) elt))
8          ))
```

Insert Front: Outline

```
(define (ordered-insert xx elt)
  (inc-dlist-size! xx)
  (let ((node (dnode null elt null))
        (target (find-greater (dlist-front xx) elt)))
    (cond ((is-empty? xx)  ; nothing in the list
           ; ...
          ((null? target)  ; last element
           ; ...
          ((null? (dnode-prev target))   ; first element
           ; ...
          (else (begin
           ; ...
          )))
```


Insert Front: Empty Case

```
(define (ordered-insert xx elt)
  (inc-dlist-size! xx)
  (let ((node (dnode null elt null))
        (target (find-greater (dlist-front xx) elt)))
    (cond ((is-empty? xx)  ; nothing in the list
           (begin (set-dlist-front! xx node)
                  (set-dlist-back! xx node)) )
          ((null? target)  ; last element
           ; ...
          ((null? (dnode-prev target))   ; first element
           ; ...
          (else (begin
           ; ...
          )))
```


Insert Front: Last element

```
(define (ordered-insert xx elt)
  (inc-dlist-size! xx)
  (let ((node (dnode null elt null))
        (target (find-greater (dlist-front xx) elt)))
    (cond ((is-empty? xx)  ; nothing in the list
           ; ...
          ((null? target)  ; last element
           (begin (set-dnode-prev! node (dlist-back xx))
                  (set-dnode-next! (dnode-prev node) node)
                  (set-dlist-back! xx node)))
          ((null? (dnode-prev target))   ; first element
           ; ...
          (else (begin
           ; ...
          )))
```


Insert Front: First Element

```
(define (ordered-insert xx elt)
  (inc-dlist-size! xx)
  (let ((node (dnode null elt null))
        (target (find-greater (dlist-front xx) elt)))
    (cond ((is-empty? xx)  ; nothing in the list
           ; ...
          ((null? target)  ; last element
           ; ...
          ((null? (dnode-prev target))   ; first element
           (begin (set-dnode-next! node target)
                  (set-dnode-prev! target node)
                  (set-dlist-front! xx node)))
          (else (begin
           ; ...
          )))
```

Insert Front: Middle Element

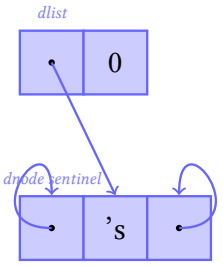
```
(define (ordered-insert xx elt)
  (inc-dlist-size! xx)
  (let ((node (dnode null elt null))
        (target (find-greater (dlist-front xx) elt)))
    (cond ((is-empty? xx)  ; nothing in the list
           ; ...
          ((null? target)  ; last element
           ; ...
          ((null? (dnode-prev target))   ; first element
           ; ...
          (else (begin
                  (set-dnode-next! node target)
                  (set-dnode-prev! node (dnode-prev target))
                  (set-dnode-next! (dnode-prev node) node)
                  (set-dnode-prev! (dnode-next node) node)))
          )))
```


Old Insert Front

```
(define (insert-front xx elt)
  (let ((node (dnode null elt (dlist-front xx))))
    (cond ((null? (dlist-front xx))
           (begin (set-dlist-front! xx node)
                  (set-dlist-back! xx node)
                  (inc-dlist-size! xx)))
          (else (begin (set-dnode-prev! (dlist-front xx) node)
                       (set-dlist-front! xx node)
                       (inc-dlist-size! xx))))
    ))
```

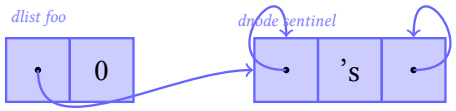

Introducting Sentinels!

```
(struct dlist (sentinel size) #:mutable)
(struct dnode (prev data next) #:mutable)
(define (make-dlist)
  (let ((sentinel (dnode null 'sentinel null)))
    (begin
      (set-dnode-prev! sentinel sentinel)
      (set-dnode-next! sentinel sentinel)
      (dlist sentinel 0))))
```



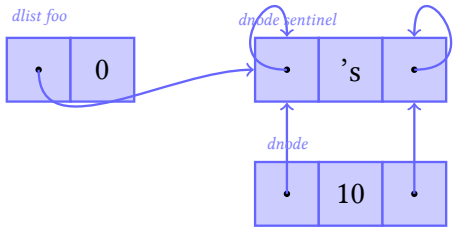
Insert Front

```
(define (insert-front xx elt)
  (let ((node (dnode (dlist-sentinel xx) elt (dnode-next (dlist-sentinel x
    (begin
      (set-dnode-prev! (dnode-next node) node)
      (set-dnode-next! (dlist-sentinel xx) node)
      (inc-dlist-size! dlist))))
(define foo (make-dlist))
(insert foo 10)
```



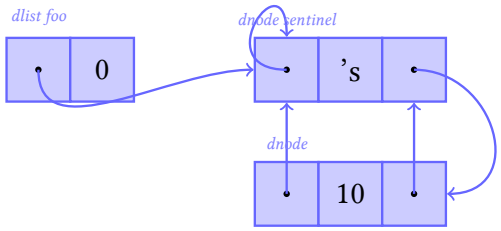
Insert Front

```
(define (insert-front xx elt)
  (let ((node (dnode (dlist-sentinel xx) elt (dnode-next (dlist-sentinel xx)))))
    (begin
      (set-dnode-prev! (dnode-next node) node)
      (set-dnode-next! (dlist-sentinel xx) node)
      (inc-dlist-size! dlist))))
(define foo (make-dlist))
(insert foo 10)
```



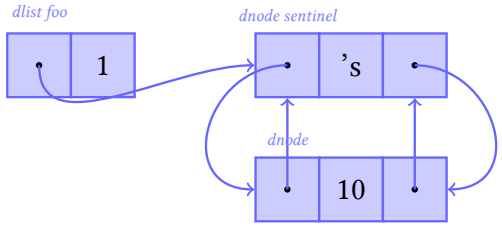
Insert Front

```
(define (insert-front xx elt)
  (let ((node (dnode (dlist-sentinel xx) elt (dnode-next (dlist-sentinel xx)))))
    (begin
      (set-dnode-prev! (dnode-next node) node)
      (set-dnode-next! (dlist-sentinel xx) node)
      (inc-dlist-size! dlist))))
(define foo (make-dlist))
(insert foo 10)
```



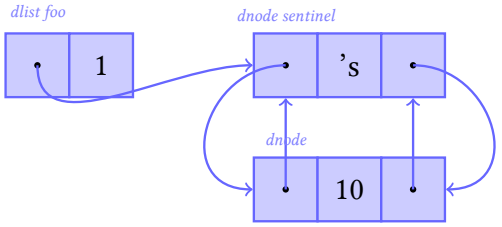
Insert Front

```
(define (insert-front xx elt)
  (let ((node (dnode (dlist-sentinel xx) elt (dnode-next (dlist-sentinel xx)))))
    (begin
      (set-dnode-prev! (dnode-next node) node)
      (set-dnode-next! (dlist-sentinel xx) node)
      (inc-dlist-size! dlist))))
(define foo (make-dlist))
(insert foo 10)
```



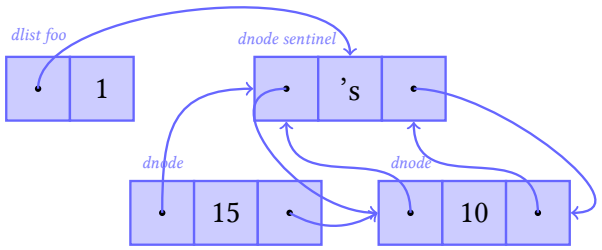
Insert 15

```
(define (insert-front xx elt)
  (let ((node (dnode (dlist-sentinel xx) elt (dnode-next (dlist-sentinel xx)))))
    (begin
      (set-dnode-prev! (dnode-next node) node)
      (set-dnode-next! (dlist-sentinel xx) node)
      (inc-dlist-size! dlist))))
; ..
(insert foo 15)
```



Insert 15

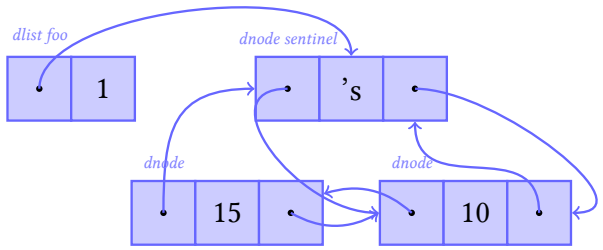
```
(define (insert-front xx elt)
  (let ((node (dnode (dlist-sentinel xx) elt (dnode-next (dlist-sentinel x
    (begin
      (set-dnode-prev! (dnode-next node) node)
      (set-dnode-next! (dlist-sentinel xx) node)
      (inc-dlist-size! dlist))))
; ..
(insert foo 15)
```



Sentinels!

Insert 15


```
(define (insert-front xx elt)
  (let ((node (dnode (dlist-sentinel xx) elt (dnode-next (dlist-sentinel x
    (begin
      (set-dnode-prev! (dnode-next node) node)
      (set-dnode-next! (dlist-sentinel xx) node)
      (inc-dlist-size! dlist))))
; ..
(insert foo 15)
```

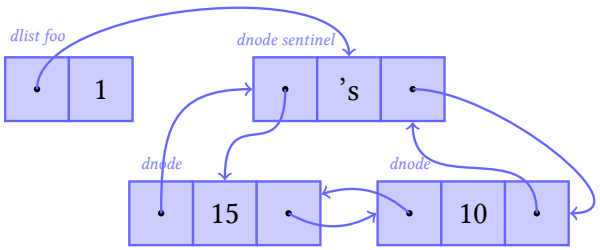



Dr. Mattox Beckman (IIT) Sentinels 13 / 13

Sentinels!

Insert 15


```
(define (insert-front xx elt)
  (let ((node (dnode (dlist-sentinel xx) elt (dnode-next (dlist-sentinel x
    (begin
      (set-dnode-prev! (dnode-next node) node)
      (set-dnode-next! (dlist-sentinel xx) node)
      (inc-dlist-size! dlist))))
; ..
(insert foo 15)
```

Dr. Mattox Beckman (IIT) Sentinels 13 / 13