

CS 2100


---

Linked Lists  
(remove &  
housekeeping)

---

---

---



# Recap

- HWS posted - Zybook
- We do have lab next week - but
- No class next Friday
- Next week:  
readings on both  
Monday + Wednesday

# Today : end of Lists - finally!

## erase:

```
/** Function to erase a node
 *
 * Parameter: an iterator to the node we wish to delete
 * Returns an iterator to the next node in the list
 */
iterator erase(iterator position) {

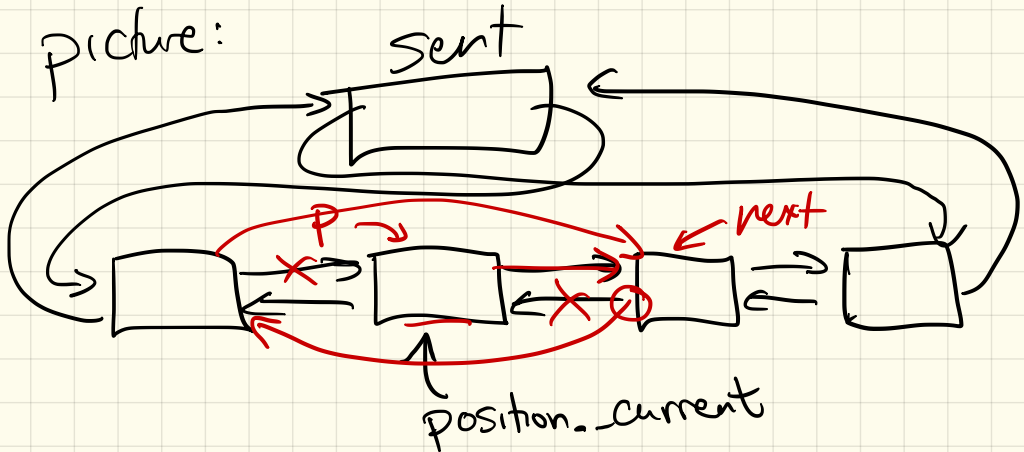
    //make some temporaries - temp stores return value, p is just readability
    Node* next = position._current->_next;
    Node* p = position._current;

    //error check
    if (empty())
        throw domain_error("can't remove from an empty list");
    if ((p == NULL) || (p == _sent))
        throw runtime_error("not a valid iterator to remove");
position._current->_next->_previous = position._current->_previous;
position._current->_previous->_next = position._current->_next;
    delete position._current;

    _size--;

    return iterator(next);
}
```

picture:



Housekeeping:

Destructor:

while (!empty())

# Runtimes

insert + delete:

$O(1)$

operator  $[]$  :  $O(n)$   
(on HW!)

Code:

↳ go compile!

# Next: Searching!

Given a value  $x$  & data structure  $S$ , output true if  $x$  is in  $S$ .

Often also want an iterator to the value, or an index (if array-based).

Two ways:

- Linear search
- Binary search
  - ↳ sorted list!

# Coding + runtimes: Linear Search:

- You've actually done the code for this (or nearly have) in both `LinkedList` + `Vector`!

A simple loop to run through the data!

- return true if ever found  
(or iterator/location)

- return false if not found

Vectors:  $O(n)$

Lists:  $O(n)$



# Binary Search:

→ Compare  $x$  to middle value in data structure.

$x > \text{middle}$

[recurse on right half

$x < \text{middle}$

recurse on left

else return true

$$B(n) = 1 + B\left(\frac{n}{2}\right)$$

$$= 1 + 1 + B\left(\frac{n}{4}\right)$$

$$= 1 + 1 + 1 + B\left(\frac{n}{8}\right)$$

$$= O(\log_2 n)$$

Can't do efficient binary  
Search on linked  
Structure:

$$B(n) = O(n) + B\left(\frac{n}{2}\right)$$

⋮

$$= O(n)$$