CSCI 204: Data Structures & Algorithms

Singly Linked lists
Part 2

Revised based on textbook author's notes.

A quick review what we did last Friday

- Discussed the basic idea of singly linked list
- Implemented some functions, insert_before(), insert_after()
- Run a demo from the code on the course website (11-code)

Removing nodes

- How to remove a node from a singly linked list

```
def remove_node(self, target):
    """ Remove the node containing the target
    """
    prev = None
    cur  = self.head
    while cur != None and cur.data != target:
        prev = cur
        cur  = cur.next

    if cur != None:    # found it and remove it
        if cur == self.head:
            self.head = cur.next  # head is removed, reset head
        else:
            prev.next = cur.next  # remove a middle node
```

- In general, removing a node involves a search first.

List search

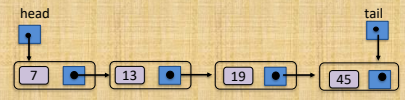
```
def search( self, target ):
  cur_node = self.head
  while cur_node is not None and \
        cur_node.data != target :
    cur_node= cur_node.next

  return cur_node
```

What are the possible return values?

How to insert a node into a sorted list?

Insert a node to sorted list



How to insert a value 14 into this list?

Two steps:

1. Find where it should be inserted
2. Put the node in place

Be careful in the cases of first and last node.

Find the correct place

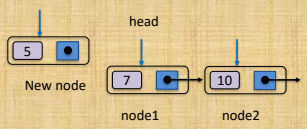
Assume sorting the list in ascending order, how to find the proper place for a new node?

```
def find_place(self, node):
    """ Find where the right place for the node in a sorted list
    """
    prev = None
    cur  = self.head
    while cur != None and cur.data < node.data:
        prev = cur
        cur = cur.next
    return prev, cur
```

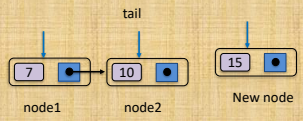
Insert the node

Need to consider three cases:
-- Insert in before the head
-- Insert after the tail
-- Insert in the middle

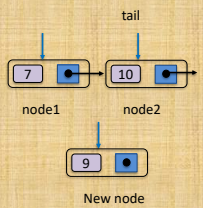
Insert before the head



Insert after the tail



Insert in the middle



Work as an activity, report it back to the class.