Data Structures

SLL Homework 2

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: Delete with key

- Given a list, delete the **first** node with the given key value
- **def** delete_node_with_key(self, n):

```
lst = LinkedList([10, 20, 30, 40])
lst.delete_node_with_key(30)

lst.debug_print_existing_nodes()
result = str(lst)
expected = '10, 20, 40'
```

Problem #2: Swap each pair of consecutive values

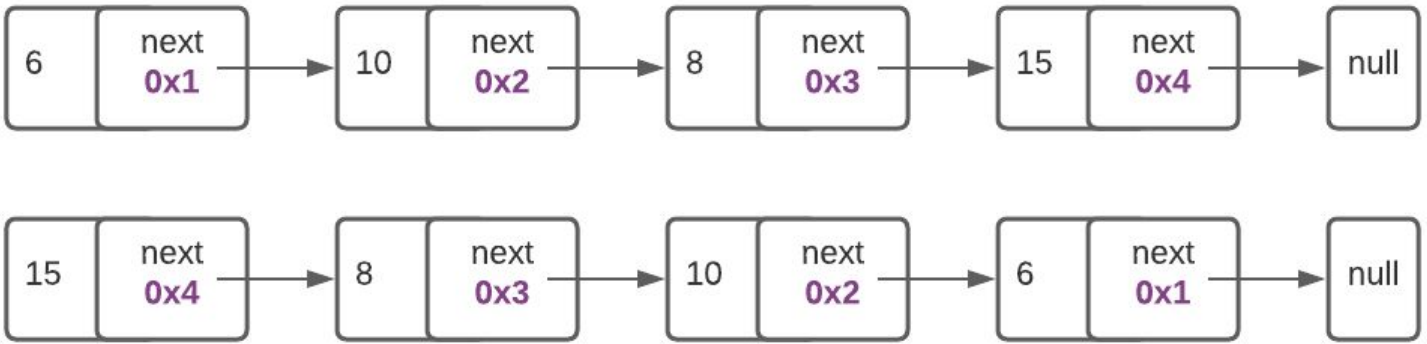
- Given a list, Swap each pair of consecutive values
- E.g. {1, 2, 3, 4} ⇒ {2, 1, 4, 3}
- E.g. {1, 2, 3, 4, 5} ⇒ {2, 1, 4, 3, 5}
- def swap_pairs(self)

```
lst = LinkedList([10, 20, 30])
lst.swap_pairs()

lst.debug_print_existing_nodes()
result = str(lst)
expected = '20, 10, 30'
```

Problem #3: Reverse list nodes

- Given a list, reverse its nodes (addresses)
- E.g. {1, 2, 3, 4, 5} ⇒ {5, 4, 3, 2, 1}
- def reverse(self)
- In testing, verify the addresses order is changed
- Your code should be O(1) memory



Problem #4: Delete even positions

- Given a list, delete all nodes at even positions (2, 4, 6, etc)
- E.g. {1, 2, 3, 4, 10} ⇒ {1, 3, 10}
- E.g. {1, 2, 3, 4, 5, 6} ⇒ {1, 3, 5}
- Note: positions NOT values
- def delete_even_positions(self)

Problem #5: Insert to be sorted

- Implement: def insert_sorted(value)
  - Value is something comparable, like integers and strings
- It inserts the value so that the list always remains sorted
- Let’s insert the following values: 10 2 30 4 1
- insert_sorted(10) ⇒ {10}
- insert_sorted(2) ⇒ {2, 10}
- insert_sorted(30) ⇒ {2, 10, 30}
- insert_sorted(4) ⇒ {2, 4, 10, 30}
- insert_sorted(1) ⇒ {1, 2, 4, 10, 30}

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."