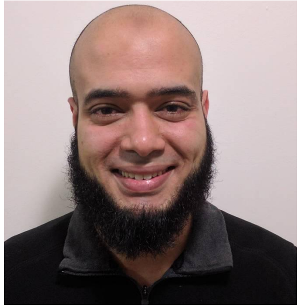
Data Structures

SLL Homework 4

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: Arrange odd & even nodes

- `def odd_pos_even_pos(self)`
- This problem is not about node values, but their positions (odd & even)
  - Rearrange the nodes so that all odd nodes are arranged in order, and they precede all of the even nodes.
  - All odd nodes must remain in the same order
  - All even nodes need to remain in the same order too
- E.g. if the list is 10, 20, 3, 7, 15: Nodes (10, 3, 15) are in the odd positions
- 1, 2, 3, 4 ⇒ 1, 3, 2, 4
- 1, 2, 3 ⇒ 1, 3, 2
- 1, 2, 3, 4, 5, 6, 7 ⇒ 1 3 5 7 2 4 6
- 11, 33, 55, 4, 50, 17, 8 ⇒ 11, 55, 50, 8, 33, 4, 17

Problem #2: Insert alternating

- **def** insert_alternate(self, another_lst)
    - The state of another_lst after the operation **doesn’t matter**
- The function inserts the values from **another** linked list in an **alternating** way with the original list
    - You may think about it as being like a **zig-zag pattern**!
- E.g. if list1 = 1, 2, 3 and list2 = 4,5,6
    - ⇒ 1, 4, 2, 5, 3, 6 [1st from L1, 1st from L2, 2nd from L1, 2nd from L2, 3rd from L1, ….]
- {1, 2, 3}, {4} ⇒ {1, 4, 2, 3}
- {1, 2, 3} {4, 5, 6, 7, 8}⇒ 1, 4, 2, 5, 3, 6, 7, 8
- {}, {1, 2, 3} ⇒ {1, 2, 3}

Problem #3: Adding 2 HUGE integers

- Assume we want to represent number 157 using a linked list
  - It is helpful to have the list as 7 -> 5 -> 1
  - This makes it easy to build and use in mathematical operations
- **Implement: def** add_num(self, another_lst)
  - It adds another number to its **current** values
  - Let’s say the current list is {1, 2, 3} representing 321
  - Another object is: {4, 5, 3} representing 354
  - After the addition, the list will become: 5 7 6 {representing 675}
  - After the result, another_lst **must not** be changed
- {9, 6, 5} + {8, 7, 6, 4, 5, 7, 8, 9} ⇒ {7, 4, 2, 5, 5, 7, 8, 9}
- Don’t convert to integer data type. Use linked lists

Problem #4: Remove repeated values except one

- **def** delete_all_repeated_from_sorted_except_one(self)
- Given a linked list of **sorted** integers. Some of the elements are repeated. Remove all of them except one of them
- Input: 1, 1, 2, 2, 2, 3, 5 ⇒ {1, 2, 3 ,5}
- Input: 1, 1 ⇒ {1}
- Input: 1, 1, 2, 2, 2 ⇒ {1, 2}
- Input: 1, 1, 2, 2, 2, 5 ⇒ {1, 2, 5}
- Input: 1, 2, 2, 2, 3 ⇒ {1, 2, 3}

Problem #5: Remove all repeated

- **def** delete_all_repeated_from_sorted(self)
- Given a linked list of **sorted** integers, keep only nodes that **never repeat** and remove any nodes with values that appear in duplicate
- Input: 1, 1, 2, 2, 2, 3, 5 ⇒ {3, 5}   both 1 and 2 are repeated
- Input: 1, 1 ⇒ {}
- Input: 1, 1, 2, 2, 2 ⇒ {}
- Input: 1, 1, 2, 2, 2, 5 ⇒ {5}
- Input: 1, 2, 2, 2, 3 ⇒ {1, 3}

Problem #6: Reverse Chains

- `def reverse_chains(self, k)`
- Instead of reversing the whole list, you only reverse sub-lists of K nodes
- {1,2,3,4,5,6},   k = 6 ⇒ 6 5 4 3 2 1 [normal reverse]
- {1,2,3,4,5,6},   k = 3 ⇒ 3 2 1 **6 5 4**
    - Reverse the first 3 numbers
    - Reverse the second 3 numbers
- {1,2,3,4,5,6,7}, k = 2 ⇒ 2 1 4 3 6 5 7

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”