Chapter 3

**6) Give an algorithm for finding the second-to-last node in a singly linked list in which the last node is indicated by a null next reference.**

Two potential options:

1. Option 1: Assuming Linked List has size variable
   1. For i equal to size of the linked list – 2, traverse the linked list until size is equal to 0
2. Option 2:
   1. While current node.next.next is not null
      1. Current = node.next

**8) Describe a method for finding the middle node of a doubly linked list with header and trailer sentinels by "link hopping," and without relying on explicit knowledge of the size of the list. In the case of an even number of nodes, report the node slightly left of center as the "middle." What is the running time of this method?**

Doubly linked list will be referred to as list:

* Function findMid
  + Node start = head of list
  + Node end = end of list
  + Continue while loop until broken
    - If start equals end
      * Break
    - If start.next equals end
      * Break
    - Start = forward one node
    - End = back one node
  + Return start

Because we are traversing the list from both sides simultaneously, this should be O(n).

**17) Let be A an array of size n >= 2 containing integers from 1 to n-1 inclusive, one of which is repeated. Describe an algorithm for finding the integer in A that is repeated.**

Brute force approach:

* For (int index = 0; i < array.length; index++)
  + For (int j = index + 1; j < array.length; j++)
    - If array[index] == array[j]
      * Return array[index]
* Return null if no duplicate found

This runs O(n^2)

Not brute force:

* For items in the array
  + Insert items into a HashSet
  + If entering into a HashSet fails, return the element
* Return null if no duplicate found

This should run O(n)

**18) Let B be an array of size n >= 2 containing integers from 1 to n – 5 inclusive, exactly five of which are repeated. Describe an algorithm for finding the five integers in B that are repeated. As an example, with n = 12, we are given 12 integers, each from 1 to 7, and if B = [5,2,6,3,2,7,1,5,4,4,7,1] the goal is to identify that numbers [1,2,4,5,7] were the repeated ones.**

* For element in array
  + If inputting item into a HashSet fails
    - Add item to ArrayList of duplicate values found
* Return the ArrayList

**23a) Suppose you are designing a multiplayer game that has n >= 1000 players, numbered 1 to n, interacting in an enchanted forest. The winner of this game is the first player who can meet all the other players at least once (ties are allowed). Assuming that there is a method meet (i,j), which is called each time a player i meets a player j (with i =/= j), describe a way to keep track of the pairs of meeting players and who is the winner.**

Create object player that holds a HashSet of players met, player number, player score

* Meet(i,j)
  + Player i tries to input player j’s player number into their HashSet
  + If player input is successful and thus not in HashSet
    - Increment player I’s score
  + Player j tries to input player I’s player number into their HashSet
  + If player input is successful and thus not in HashSet
    - Increment player j’s score
  + If both player I and J are at score n – 1, met everyone except themself
    - Announce a tie
  + If Player I is at score n-1
    - Announce Player I’s win
  + If Player j is at score n-1
    - Announce Player j’s win

**25) Describe an algorithm for concatenating two singly linked lists L and M, into a single list L that contains all the nodes of L followed by all the nodes of M.**

* Traverse linked list L until pointer next node is null
  + Save next node in a temp node
* While M node’s pointer to next node does not equal null
  + tempNode’s pointer to next node is equal current M Node
  + tempNode is set to tempNode.next
* tempNode’s pointer to next node is equal current M Node (There will be one last node left)
* tempNode is set to tempNode.next

**26) Give an algorithm for concatenating two doubly linked lists L and M, with header and trailer sentinel nodes, into a single list L.**

Set M current node equal to M head’s next Node

* While currentNode does not equal the tail node
  + tempNode is set equal to currentNode
  + currentNode now equal to currentNode next
    - M head next Node is set to currentNode
    - M head is set to currentNode’s previous
  + tempNode pointer to previous is equal to L tail previous
  + tempNode next is set to L’s tailNode
  + tempNode’s previous Node’s next Node is set to the tempNode
  + L’s tail previous Node is set to tempNode

**27) Describe in detail how to swap two nodes x and y (and not just their contents) in a singly linked list L given references only to x and y. Repeat this exercise for the case when L is a doubly linked list. Which algorithm takes more time?**

Singly Linked List:

* Node currX is grabbed from linked list via traversal of the linked list
* Node currY is grabbed from the linked list via traversal of the linked list
* If nodeX is head
  + Grab node prevY by traversing linked list until next node is equal to Y
  + Set prevY’s next node to nodeX
  + Make prevY the head of the linked list
* If nodeY is head
  + Grab node prevX by traversing linked list until next node is equal to Y
  + Set prevX’s next node to nodeY
  + Make prevX the head of the linked list
* Else
  + Grab prevX by traversing the linked list until next node is equal to X
  + Grab prevY by traversing the linked list until next node is equal to Y
  + Set prevX’s next node to node Y
  + Set prevY’s next node equal to nodeX
* Set tempNode equal to nodeX’s next node
* Set nodeX’s next node equal to nodeY’s next node
* Set nodeY’s next node equal to tempNode

Doubly Linked List

* Get nodeY via traversal of the linked list
* Set prevY equal to nodeY’s previous node
* Get nodeX via traversal of the linked list
* Set prevX equal to nodeX’s previous node
* Set prevX next node equal to nodeY
* Set nodeY previous to prevX
* Set prevY’s next node equal to nodeX
* Set nodeX previous node equal to prevY
* Create a tempNode which holds nodeX’s next Node
* Set nodeX’s next node equal to NodeY’s next Node
* Set nodeY’s next node equal to temp node
* Set nodeX’s previous’s next node equal to nodeX
* Set nodeY’s previous’s next node equal to nodeY

Between these two, swapping a doubly linked list is much faster because you already have pointers on the node towards the previous node and the next node.

**28) Describe in detail an algorithm for reversing a singly linked list using only a constant amount of additional space.**

* Initialize prevNode equal to null
* Set current equal to the head of the linked list
* Initialize nextNode
* while the current’s next node does not equal null
  + nextNode is set equal to current’s next node
  + current’s next node is equal to prevNode
  + prevNode is set equal to current
  + current is equal to nextNode
* Set prevNode equal to the head

**29) Suppose you are given two circularly linked lists, L and M. Describe an algorithm for telling if L and M store the same sequence of elements (but perhaps with different starting points).**

* If size of L does not equal size of M
  + return false
* Initialize int variable rotations as 0
* Intialize boolean variable as false
* While rotations is less than the size of the linked lists AND bool is false
  + Check if current head of M is NOT equal to current head of L
    - Rotate M
    - Increment rotations
  + Else (Heads are equal)
    - Set bool equal to true
    - For int index = 0; index < size of lists; index++
      * If index-th value in M and L are NOT equal
        + Rotate M
        + Increment rotations
        + Bool is set to false
        + Break loop
* Return bool

**30) Given a circularly linked list L containing an even number of nodes, describe how to split L into two circularly linked lists of half the size in interleaved fashion, such that list A,B,C,D,E,F is split into lists A,C,E and B,D,F.**

* Set int listSize equal to size of circularly linked list;
* Initialize new targetList for every other element
* For int index = 0; index < (listSize / 2) - 1; index++;
  + tempNode is equal to originalList.tail.next
  + originalList.tail.next = originalList.tail.next.next
  + Add tempNode to targetList (this will go through normal process of setting to tail, of circularly linked list if it is the first node)
  + Rotate the original list

We skip every other node because we are popping one and then rotating past one.