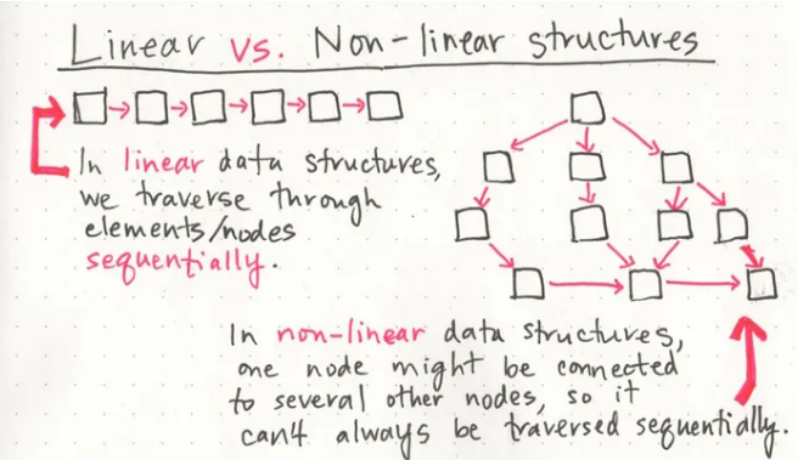
Data Structures Fall 2023
Lab 06: Pointer-based Linked List

What is a Linked List

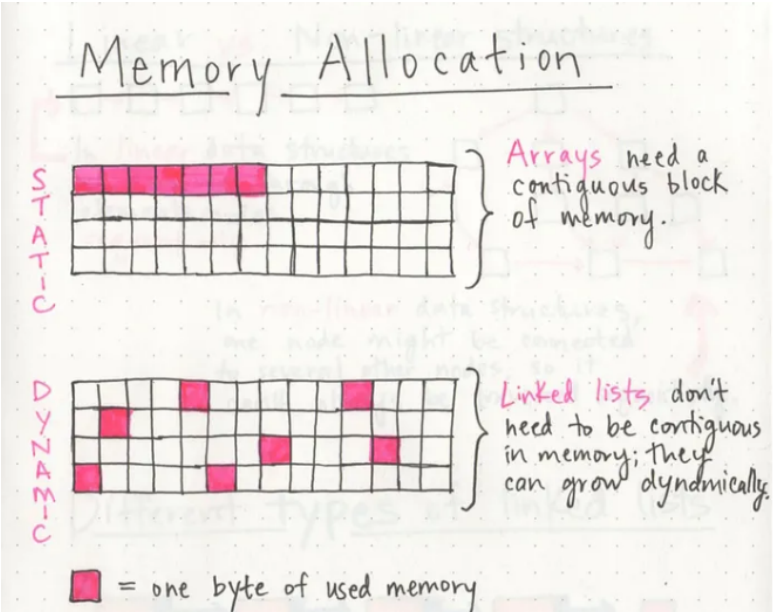
linked lists are linear data structures, meaning there is a sequence and an order to how they are constructed and traversed. In order to get to the end of the list, we have to go through all of the items in the list in order, or sequentially. Linear structures, however, are the opposite of non-linear structures. In non-linear data structures, items don't have to be arranged in order, which means that we could traverse the data structure non-sequentially.



How is a Linked List different from an Array?

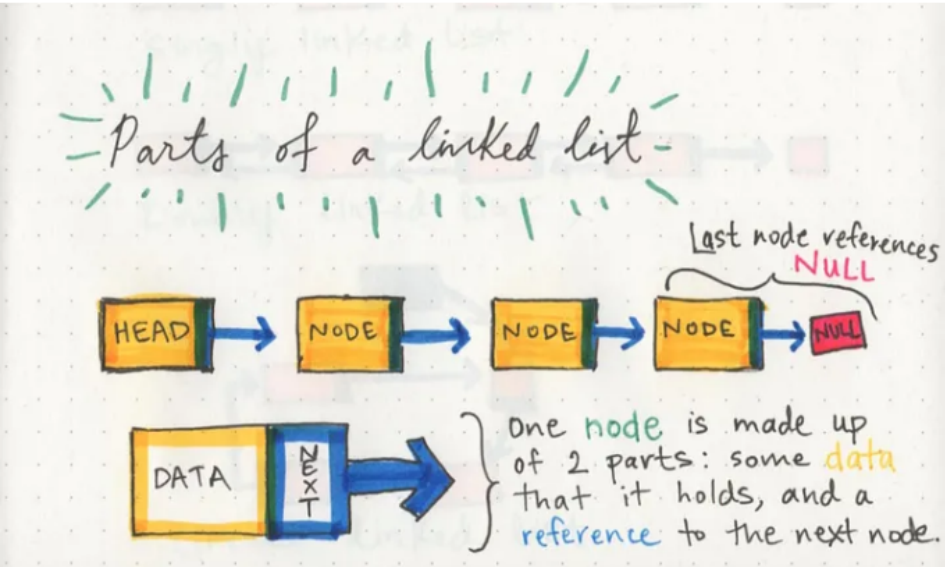
The biggest differentiator between arrays and linked lists is the way that they use memory in our machines.When an array is created, it needs a certain amount of memory. If we had 7 letters that we needed to store in an array, we would need 7 bytes of memory to represent that array. But, we'd need all of that memory in one contiguous block. That is to say, our computer would need to locate 7 bytes of memory that was free, one byte next to the another, all together, in one place.

On the other hand, when a linked list is born, it doesn't need 7 bytes of memory all in one place. One byte could live somewhere, while the next byte could be stored in another place in memory altogether! Linked lists don't need to take up a single block of memory; instead, the memory that they use can be scattered throughout.



Parts of a Linked List:

A linked list is made up of a series of nodes, which are the elements of the list.The starting point of the list is a reference to the first node, which is referred to as the head. Nearly all linked lists must have a head, because this is effectively the only entry point to the list and all of its elements, and without it, you wouldn't know where to start! The end of the list isn't a node, but rather a node that points to null, or an empty value



A single node is also pretty simple. It has just two parts: data, or the information that the node contains, and a reference to the next node.

Task 01

Design a class linked List with data members Data and next pointer. Your class must provide implementation for the following

- A default constructor for Node
- A parameterized constructor
- Getters for data and next
- Setters for data and next

Design a linkedList class with a pointer of type Node. This will carry the address of **head** initially, later it will be required for traversal.

Your class must provide implementation for the following

1. A default constructor
2. **Insert** function to insert data item. For every new data element to be inserted, you will have to reach up to the end point of LL and then perform insertion
3. **Insert to head** function to insert data element at the start of your LL
4. Design **isEmpty** to check if there is no data element in the LL
5. Why don't you need to design isFull
6. A **Search** function to search for any element provided by user
7. **Update** function to update/replace one data element with another
8. A function **Insert at index** that will count data elements in the LL and then place a new data element in the specified location. To know exactly after which node we have to make insertion, we need count the elements and that will help us to identify index (index doesn't exist actually, but we are using this term to get to know the location for insertion)
9. A function **Delete** to delete the specified data element, for this you will have to save the address of the next node that appears after the one to be deleted. Saving the address will help you after deleting the element to concatenate the list with the elements appearing after the element that is supposed to be deleted.
10. A function to **print** the data of LL

Task 2

Imagine you are a bioinformatician specializing in DNA sequence analysis. In the field of bioinformatics, it's common to work with numerical representations of DNA sequences, where each DNA nucleotide is encoded as a number. Specifically, Adenine (A) is represented as 0, Cytosine (C) as 1, Guanine (G) as 2, and Thymine (T) as 3. You are tasked with developing an algorithm that operates on a linked list, where each node contains a numerical representation of a nucleotide.

Your goal is to create a program that swaps every two adjacent nodes in the linked list and returns the head of the modified list. It's important to note that in this context, you should preserve the numerical values representing the nucleotides, and only the arrangement of these numerical representations in the linked list should be altered.

For example, if you are provided with a linked list representing a DNA sequence like this: [0, 3, 1, 2, 2, 0], your algorithm should return a modified linked list: [3, 0, 2, 1, 0, 2].